MODEL CHECKING PLANS FOR FLEXIBLE MANUFACTURING SYSTEMS

Leandro Dias da Silva * **Hyggo Almeida** *
Angelo Perkusich * **Péricles Rezende Barros** *

* *Electrical Engineering Department*
Federal University of Campina Grande
58109-970, Campina Grande, PB, Brazil
{*leandro,hyggo,perkusic,prbarros*}*@dee.ufcg.edu.br*

Abstract: In this work an approach to analyze non-deterministic execution plans for flexible manufacturing systems is presented. This approach consists of modeling the system using coloured Petri nets and then, by simulation, discover the possible executions for the system. After, model checking is used to prove that the model behavior do not deviate from the planning. Based on this approach it is possible to model a production system based on cells and to predict and to prove its behavior before the system is developed. Therefore, the analyzed model can be used to develop flexible and dependable production systems to meet the market requirements of quantity, diversity, and quality.

Keywords: Flexible Manufacturing Systems, Planning, Model Checking, Coloured Petri Nets.

1. INTRODUCTION

Flexible Manufacturing Systems (FMS) (Zhou and Venkatesh, 1999) have been used as an approach to deal with the complexities of modern production system. The motivation for this is that these systems, if properly designed and analyzed, can be very flexible, scalable, and dependable. The flexibility arises from the fact that they are organized on cells. The cell consists of production resources, such as machines. Therefore, changing part of the production means, in most cases, changing a cell without modifications on other cells or the rest of the system. The scalability is related to the flexibility explained before, and to add a new feature to the production system usually means to add new cells. The dependability is also related to the flexibility because changing portions of the system reduces the possibility of problems with other portions that are working properly and were not modified.

Due to the technological advances more sophisticated production systems have been developed. These new systems must be carefully designed and analyzed beforehand the actual system is adopted in the production because they are expensive, complex, and will probably change over a short amount of time. Moreover the demand is also increasing not only in quantity and quality but also in diversity of products.

An execution plan is a sequence of nondeterministic actions that the entities of an FMS must take in order to produce a final product. The process for generating a plan is called planning. Due to the inherent decentralized nature of any FMS it is difficult to define deterministic choices based on global information. Therefore, there is no single planning algorithm than can be applied for all systems. Many plan verification methods are described in the literature but they are efficient for some systems and inefficient for others. A possible approach to deal with such situation is to adopt

efficient techniques for either plan generation or verification (Pistore and Traverso, 2001).

Petri nets have been widely used for FMS modeling and analysis (Zhou and Venkatesh, 1999; Proth and Xie, 1997; Desrochers and Al-Jaar, 1994). In this work Hierarchical Coloured Petri Nets (HCPN) (Jensen, 1992) is used. This extension to the Petri nets formalism incorporates the concepts of data types and hierarchy that promote more compact and organized models of complex systems. The use of formal methods in the design of systems aggregates the advantages of automatic simulation, and proof of properties with model checking (Clarke *et al.*, 1999), for example. Therefore, HCPN and model checking are used to reason about the complexity associated with the generation and verification of nondeterministic plans for FMS (Cimatti and Roveri, 2003). A similar approach has been applied to the multiagent system domain as described in (Silva *et al.*, 2004). The Design/CPN tool set is used for drawing and analyze the HCPN models (Jensen, 1999).

The remaining of this paper is organized as follows. In Section 2 the model for a FMS is described. In Section 3, the plan generation strategy is introduced. In Section 4 the verification of plans is presented. In Section 5, some related work are discussed. Finally, in Section 6, final remarks are presented.

2. A FLEXIBLE MANUFACTURING SYSTEMS MODEL

A Flexible Manufacturing System (FMS) is a system organized in cells. Each cell has one or more related machines and a transport system. Two kinds of transport system are considered, inside cells and outside cells. Some other issues could be part of an FMS model such as, for example, scheduling, and information flow, among others. But in this work the main interest is on production and transport issues of FMS in order to reason about the execution plans.

In Figure 1 a diagram illustrating a simple FMS is shown. The modeling and characterization of such systems has been previously done in (Silva and Perkusich, 2004). As the focus is production and transport issues, a framework is used to develop FMS models related to these systems. Using this framework, any FMS with any number of cells and machines can be modeled and investigated. For any cell,machine or transport system to be considered the designer just plugs its model on the framework and the rest of the system does not change.

The framework consists of the specification of the location for transport, production entities, and the definition of possible restrictions and the relationship between the entities. For Figure 2 the

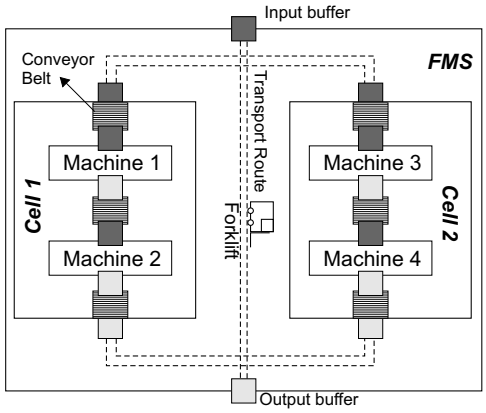

Fig. 1. Flexible manufacturing system.

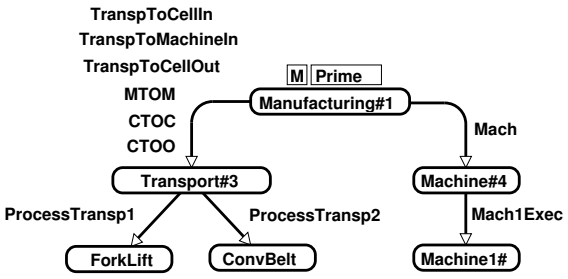

Fig. 2. HCPN hierarchy for the FMS.

hierarchy of the model is shown. In this figure, the Manufacturing page is the main level of the model. In this page there are several substitution transitions to Transport page such as, for example, CTOO, CTCO, TranspToCellIn, and one to Machine page, Mach. The hierarchy mechanisms of colored Petri nets make possible to define substitution transitions that are replaced by another CPN model. It allows the definition of hierarchies in the framework modeling specific types of resources, or transport entities, keeping the structure of the hierarchical upper level unchanged and characterizing a top-down modeling. The realization of the framework as a HCPN model is shown in Figure 3, and the Manufacturing page model shown in Figure 2.

Using this generic model, it is transparent the way how the entities are modeled. Then, an FMS system is modeled as a cell structure that has one or more machines, a production sequence and a transport sequence. The production sequence defines the resources needed to have a final product. The transport sequence defines the requests of transport entities to execute an specific production sequence to produce a product. The system can process several different parts or several identical parts at the same time.

The description of the places and the transitions in Figure 3 are defined as follows. The meaning of the places are: SystemIn, System input buffer; CellIn, cell input buffer; MachineIn, machine input buffer; MachineOut, machine output buffer; CellOut, cell output buffer; SystemOut, system output buffer; MachToMach, request of transport from the machine output to the machine input, in the same cell;

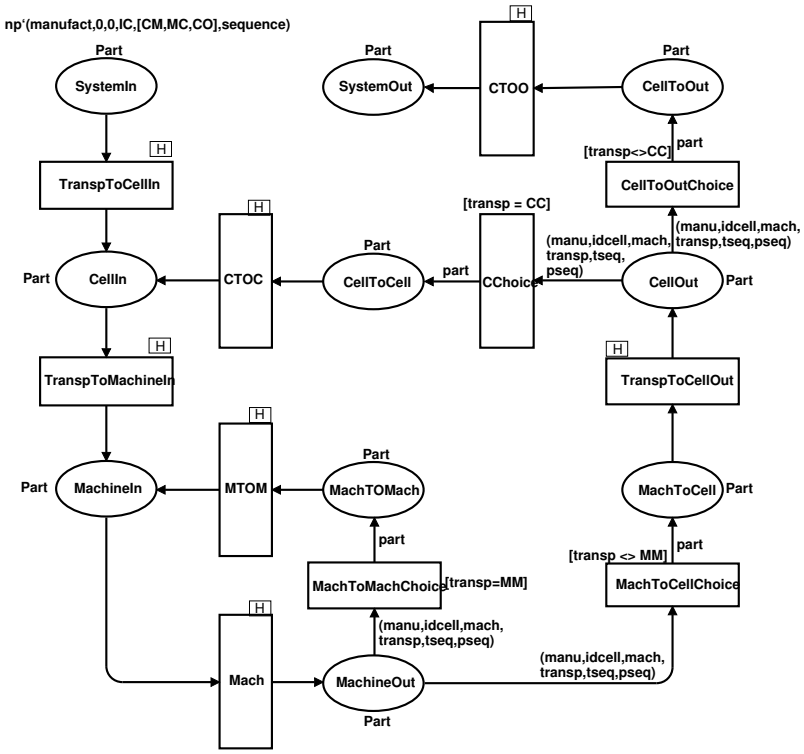

Fig. 3. HCPN framework for FMS.

MachToCell, request of transport from the machine output to the cell output; CellToCell, request of transport from the cell output to the cell input; CellToOut, request of transport from the cell output to the system output.

The meaning of the transitions are:TranspToCellIn, transport from the system input to the cell input; TranspToMachineIn, transport from the cell input to the machine input; Mach, machines; MTOM, transport from the machine output to the machine input, in the same cell; TranspToCellOut, transport from the machine output to the cell output; CTOC, transport from the cell output to the cell input; CTOO, transport from the cell output to the system output; MachToMachChoice, choice of transport from the machine output to the machine input, in the same cell; MachToCellChoice, choice of transport from the machine output to the cell output; CChoice, choice of transport from the cell output to the cell input; CellToOutChoice, choice of transport from the cell output to the system output.

The transition Mach is a substitution transition to the interface of the framework with the models of machines. The transitions TranspToCellIn, TranspToMachineIn, MTOM, TranspToCellOut, CTOC, and CTOO, are substitution transitions to the interface of the framework with the models of transport entities. There are different transitions for the transport system because it is possible to have different transport entities in an FMS.

The production sequence, as well as the transport sequence are specified in the initial marking in place SystemIn. The transitions MachToMachChoice, MachToCellChoice, CChoice, and CellToOutChoice, means a choice to determine what type of transport is required at each point based on the transport sequence defined in the initial marking

3. MODEL BASED PLANNING

The approach adopted on this work is to automatically generate plans based on automatic simulation of the model and model checking. Using the model described on Section 2 and the Design/CPN tool it is possible to carry several different simulations. That is because each simulation covers one possible path for the model. Therefore, after one or more simulations the designer can realize a possible execution plan. A Message Sequence Chart (MSC) for each simulation can be automatically generated. The MSC is useful to see in a more intuitive way the actual path of the execution.

Once the MSC is generated the designer can easily discover the path followed by the part on the system (the token on the model). This path can be a possible execution plan for the model. If it does not make sense then the model is wrong and must be fixed. Otherwise, the path can be considered a correct execution plan for the model. In the last case this plan should be verified in order to guarantee that it holds for every possible behavior of the model, and not only for the specific one followed at simulation time. The verification will be considered on Section 4.

The MSC is generated using a library for Design/CPN. Thus, the designer can call functions from a transition code region. Using HCPN and Design/CPN it is possible to associate code to a transition, and this code is executed each time the transition fires. Therefore, for each plan that the designer wants to generate it is possible to execute functions that creates the MSC at any specific transition or set of transitions to create a customized MSC for each purpose.

Depending on the type of a part to be processed, and the final desired product, a specific production and transport sequence must be followed. This information is specified on the initial marking of the model at place SystemIn, see Figure reffig:framework. In this example, the transport sequence is from the system input to cell input (IC), from the cell input to the machine input (CM), from machine output to cell output (MC), and from cell output to system output.

After simulation the MSC generated for the initial marking described above is shown in Figure 4. This example shows a complete execution plan, but more specific plans can be generated also. For example, suppose that the designer wants to see the execution plan for a specific transport system. In this case the MSC show in Figure 5 represents the execution plan for the conveyor belt used for the transport from the cell input to the machine input (CM).

Using this strategy all the MSC for execution plans can be automatically generated. Moreover, several MSC can be generated for the same plan. Therefore, initial design errors can be found before a more deep, costly, and time consuming verification strategy is performed. The main advantage is that using formal methods one can perform automatic simulation, in order to generate the MSC, and also to perform model checking. The last one is used after the designer has confidence on the design and all generated MSC represents correct execution plans.

4. MODEL CHECKING PLANS

In this work the verification of execution plans is done by performing model checking to verify whether the model models the properties specified or not. The model checking step is performed using the Design/CPN and the ASK/CTL library (Christensen and Mortensen, 1996). First, the state space, that is called occurrence graph, is generated for the model. The occurrence graph is a directed graph that represents all the possible behaviors of a Petri net model. After that, the model checking itself is executed to verify if the specification in temporal logic (Emerson, 1990) is satisfied in this state space. If the specification is satisfied, the plan is proved correct. Otherwise there can be an error in the model.

The model checking technique is used to verify if a model $\mathcal{M}$ satisfies a given specification f: $\mathcal{M} \models f$ (Clarke *et al.*, 1999). In the context of this work the model is an HCPN, and the specification is a Computation Tree Logic (CTL) formula (Christensen and Mortensen, 1996).

Considering the examples illustrated in Section 3 the temporal logic formulas can be specified as follows. Each CTL formula is constructed using atomic propositions and modal and temporal connectors. The atomic propositions are abstractions about the marking, for example, of the model. Suppose, for the complete execution plan, that proposition $P1$ is evaluated to true when there is a part (token) in the system input and a request for a transport from the system input to the cell input (IC) in place SystemIn of the model shown in Figure 1. Suppose also that the proposition $P2$ is evaluated to true when the part (token) is in place CellIn. Therefore, the CTL formula to verify if it holds for every possible behavior of the model is: $\mathbf{AG}(P1 \rightarrow \mathbf{AF}(P2))$. The meaning of this formula is that for all paths, in all states, if $P1$ is true, then $P2$ will be eventually true at some state for all paths.

By following the same reasoning it is possible to define atomic propositions for all intermediate state of the execution plan shown in Figure 4, and to construct the formula to verify it. Suppose that $P3$ is true when a transport request (CM) is in place CellIn and $P4$ when the token is in place MachineIn. Then, $\mathbf{AG}(P3 \rightarrow \mathbf{AF}(P4))$ can be used to prove that all requests in the input of the cell will be treated and the result is that the token will be in the right place, the machine input. The following formula is used to verify the overall execution plan:

$$AND((\mathbf{AG}(P1 \rightarrow \mathbf{AF}(P2))), (\mathbf{AG}(P3 \rightarrow \mathbf{AF}(P4))), ...)(1)$$

It is important to note that the complexity of the formula to prove the plan depends on the level of detail need at a given point of the design. For example, the designer can prove first the initial and final point, that is, $P1$ is a request at system input, and $P2$ is a token at system output. Then, it is possible to refine the proof to include intermediary points, as in Equation 1. After, another possibility is to prove also that not only the path is correct but also that it is followed in the correct way. That is, the execution plan is not only the path, but also how the path is executed. Therefore, the designer is now in the point to prove that the transport requests are handled in a correct way.

In Section 3 the execution plan for the conveyor belt request is illustrated in Figure 5. Based on this plan, and on the model shown in Figure 3 it is possible to identify and to specify the atomic propositions to prove the plan, as discussed above for the overall plan. Therefore, it is also necessary

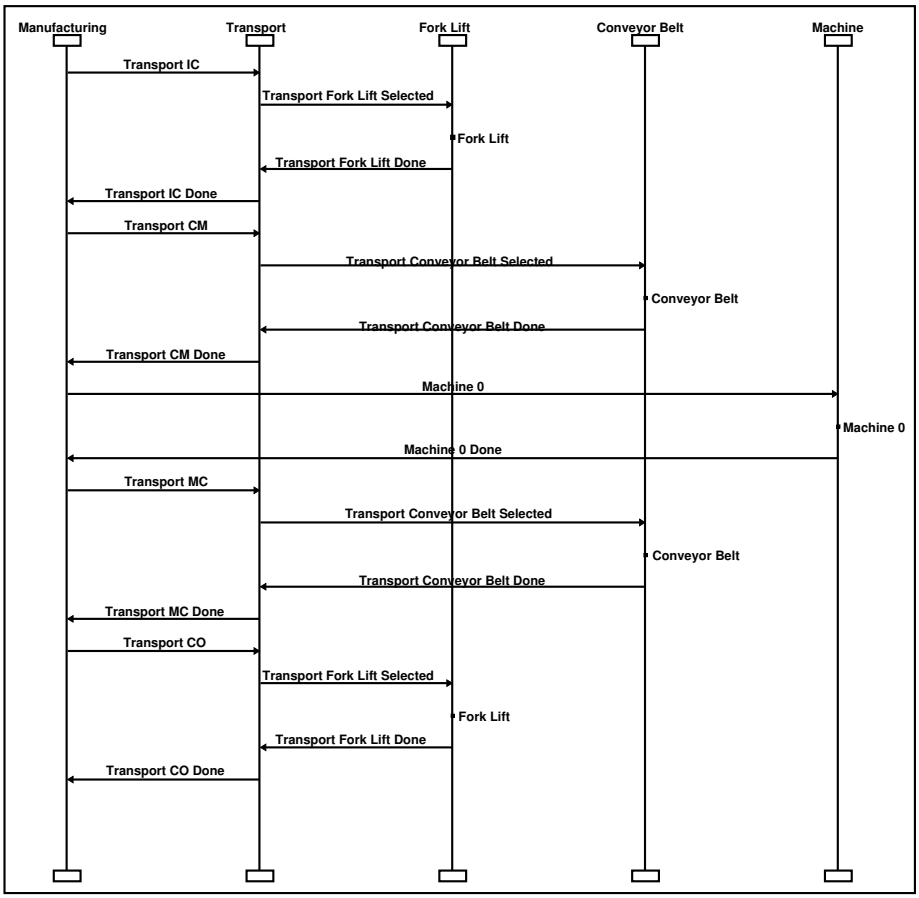

Fig. 4. A transport execution plan (MSC).

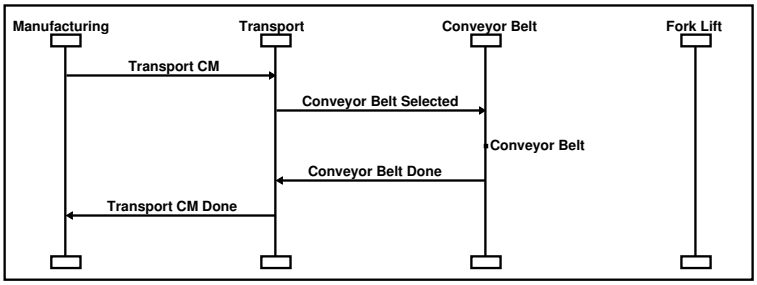

Fig. 5. Conveyor belt execution plan (MSC).

to specify atomic propositions for the Transport page in Figure 2. This is not shown in this paper for the sake of space. But it consists of the decision mechanism to select a proper transport entity based on the request. Finally, Equation 1 can be expanded to include these new propositions.

5. RELATED WORK

Petri nets have been widely used in the domain of FMS domain. (Proth and Xie, 1997; Dicesare *et al.*, 1993; Desrochers and Al-Jaar, 1994; Zhou and Venkatesh, 1999). In most cases low level Petri Nets are used, and thus resulting in models that are either too simple or to complex in terms of description. In the context of this work HCPN is used because it is possible to have more organized and compact models based on this kind of Petri Net. These factors favor the modeling and analyze of complex systems with complex features, such as FMS, in an easier way. Moreover, they do not emphasize execution plans, neither model checking. The first has a critical impact in FMS design and control, and the last is very important for dependability.

In the context of planning, the model checking approach has been used to verify plans for agents in the multiagent domain based on various formalisms. In (Jensen and Veloso, 2000) it is proposed an Ordered Binary Decision Diagram based planning framework for multiagent and nondeterministic domains. In (Xu *et al.*, 2002) it is presented an approach for modelling and

verification of multiagent behaviors using Predicate/Transition Nets. In (Xu and Shatz, 2001) G-net is extended for modelling inheritance of agent classes in multiagent systems, which provides a clean interface between agents with asynchronous communication ability and supports formal reasoning.

6. FINAL REMARKS

In this paper an approach for automatically generate and verify plans for Flexible Manufacturing Systems (FMS) is introduced. Hierarchical Coloured Petri Nets (HCPN) are used for formal specification. The HCPN models are automatically simulated and Message Sequence Charts (MSC) are generated for each simulation. These MSC can be viewed as the execution plan for the MSC. After the designer is confident on the plans generated by simulation, the verification can be performed to proof that the plans hold for every possible behavior of the model. The verification is done using model checking. The Design/CPN tool set is use for edition and simulation of the HCPN models, and the ASK/CTL library is used for model checking.

The approach introduced in this paper is important for the FMS domain because it introduces a formal analysis that promotes dependability. Using HCPN complex models can be developed to match the complexity of nowadays FMS. Moreover, the execution plan verification based on a formal, and automatic approach is a contribution to this domain because the constant changes to systems can be captured by the model, and the execution plans can be proved to be correct.

ACKNOWLEDGEMENTS

The first two authors are with the Graduate Program in Electrical Engineering, Federal University of Campina Grande (COPELE/UFCG). The research reported in this paper is partially supported by grants 305110/2002-0 and 200365/2004-5 from the Brazilian National Research Council (CNPq), a scholarship from CAPES for the first author and a scholarship from CNPq for the second author.

REFERENCES

Christensen, S. and K. Mortensen (1996). *Design/CPN ASK-CTL Manual*. Univ. of Aarhus.

Cimatti, A. and M. Roveri (2003). Conformant Planning via Symbolic Model Checking. *In JAIR* pp. 305–338.

Clarke, E., O. Grumberg and D. Peled (1999). *Model Checking*. The MIT Press.

Desrochers, A. and R. Al-Jaar (1994). *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis*. IEEE Press.

Dicesare, F., Harhalakis, G., Proth, J.M., Silva, M. and Vernadat, F.B., Eds. (1993). *Practice of Petri Nets in Manufacturing*. Kluwer Academic Pub.. London.

Emerson, E. Allen (1990). Temporal and modal logic. In: *Handbook of Theoretical Computer Science* (jan Van Leeuwen, Ed.). Vol. B: Formal Models And Semantics. Chap. 16, pp. 995–1072. Elsevier Science.

Jensen, K. (1992). *Coloured Petri Nets: Basic Concepts, Analysis, Methods and Practical Use*. EACTS – Monographs on Theoretical Computer Science. Springer-Verlag.

Jensen, K. (1999). *Design/CPN 4.0*. University of Aarhus, Denmark, http://www.daimi.aau.dk/designCPN/.

Jensen, Rune and Manuela Veloso (2000). OBDD-based Universal Planning for Multiple Synchronized Agents in No n-Deterministic Domains. In: *Proceedings of the Fifth International Conference on Artificial I ntelligence Planning Systems*. Breckenridge, CO. pp. 167–176.

Pistore, M. and P. Traverso (2001). Planning as Model Checking for Extended Goals in Non-deterministic Domains. In: *Proceedings of IJCAI'01*. pp. 479–486.

Proth, J. and X. Xie (1997). *Petri Nets : A Tool for Design and Management of Manufacturing Systems*. John Wiley & Sons. New York.

Silva, L., A. Perkusich, H. Almeida and E. Costa (2004). A Formal Approach for the Verification of Multiagent Plans based on Model Checking and Petri Nets. In: *Proceedings of SELMAS/ICSE'04*. Edinburgh, UK. pp. 145–151.

Silva, L. and A. Perkusich (2004). A Systematic And Formal Approach to the Specification of Flexible Manufacturing Systems Reusing Coloured Petri Nets Models. In: *Proceedings of INCOM/IFAC'04*. Salvador, Brazil.

Xu, Dianxiang, Richard Volz, Thomas Ioerger and John Yen (2002). Modeling and Verifying Multi-agent Behaviors using Predicate/Transit ion Nets. In: *Proceedings of the 14th international conference on Software Engineering and knowledge engineering*. ACM Press. pp. 193–200.

Xu, H. and S. M. Shatz (2001). A Framework for Modeling Agent-Oriented Software. In: *Proceedings of the 21st International Conference on Distributed C omputing Systems (ICDCS)*.

Zhou, M. and K. Venkatesh (1999). *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Wourd Scientific.