Logistic Process Models Go Petri Nets

Peter Kemper*
Fachbereich Informatik Universität Dortmund
D-44221 Dortmund, Germany
{kemper}@ls4.cs.uni-dortmund.de

Abstract

We briefly illustrate by an example how stochastic Petri nets can bridge the distance between a process oriented modeling language for logistic networks and performance analysis techniques.

1 Introduction

Concurrent processes appear in many application areas, especially in those contexts where a process oriented point of view became popular in recent years. These include among others business processes, processes in production systems and processes in logistic networks. Our interest in logistic networks is empowered by "Sonderforschungsbereich 559" at Dortmund University, it is a major, interdisciplinary research project involving scientists from logistics, mechanical engineering, economics and computer science. The focus point is on modeling and analysis of large logistic networks. Among the many description techniques to describe such networks, we apply a formalized version of so-called process chains [6] which uses a process oriented view of the world.

In this paper we consider generalized stochastic Petri nets (GSPNs) as a mean to bridge the gap between an application specific formalism like process chains and analysis methods developed for concurrent systems, especially Petri net (PN) analysis methods. The problem is closely related to the process-resource allocation problems considered by Ferscha in [4] and in a broader context by Donatelli and Franceschinis [3]. We consider a different application area, and we aim for a translation of process chains to GSPNs, such that structural information about processes and resources is preserved for analysis algorithms that take advantage of it, i.e. Kronecker methods which exploit a compositional structure.

2 Logistic networks: Modelling and Analysis

Logistic networks describe how a delivery of goods and information is performed in order to serve the demand of customers. In a general setting, goods can be material or immaterial, customers can be any kind of person, company or division who provides a task to be done. The application area covers topics like the internal logistics of a single production plant as well as a world-wide network for air cargo. The process chain notation [6] is designed to describe realworld networks with a clear focus on exposing how a single customer demand is satisfied. The idea is to identify redundancies, potentials for an increase of quality of service, cost reduction, increase of productivity and utilization of resources. In logistics, a high quality of service means to deliver the right thing at the right place at the right time in the right quantity and quality. The famous

*This research is partially supported by Deutsche Forschungsgemeinschaft, SFB 559.

just-in-time production in the automobile industry is a good example of an area which relies on excellent logistics. Process chains have been successfully applied by the Fraunhofer Institute, Dortmund, in various projects with industrial partners/customers; within SFB 559 it is applied for modelling freight terminals, air cargo handling at Frankfurt airport and aspects in supply chain management among others.

Modelling such systems for analysis mainly goes for the following results: turn around time of a process, utilization of buffers/storage areas and resources like devices and personnel. The latter is used to set up a cost accounting which assigns each process costs according to the amount and time of resources it uses.

The process chain formalism [6, 2] can be considered from different perspectives: a) it is a graphical notation plus a significant amount of attributes to describe logistic processes. b) It is a parallel programming language with fork/join mechanisms, task creation, synchronization between processes, and communication via shared memory. It has a notion of function with parameters and local variables. c) It provides a classical workload and machine description for performance modeling. A machine/resource description allows for active and passive resources (servers and counters), while the process description with its source and sink nodes gives the workload specification. The whole specification is set up in a hierarchical manner due to so-called functional units which provide services to its environment and which contain resources and other functional units to perform these services.

Process chain models of logistic networks show certain specific patterns: a) process chains which describe customer tasks can result in a variable number of processes as the dynamic behaviour of a model evolves over time, b) process chains which rather describe behaviour of objects which exist permanently, e.g. all kinds of means used in transportation like trucks, trains, ships, but also pallets, containers, reach stackers etc. appear as an invariant number of processes for a model with fixed parameters.

Due to limitations in space we illustrate a mapping from process chains to GSPNs by an example. We consider a small supply chain involving a manufacturer and one of his suppliers who delivers parts on demand by the help of a transportation company. The manufacturer performs a just-in-time production and orders parts when it starts the main processing of his product. This leaves the supplier a certain amount of time to produce the parts, pack them into boxes, load boxes into containers and transport the containers by truck to the manufacturer where container and boxes are unloaded. The last part of this small supply chain is performed by a transportation company, and we assume that it provides a certain amount of trucks, which cycle between manufacturer and supplier. Trucks carry empty boxes on the return route. At the manufacturer side, supplied parts are assembled if they are of the right kind, it is not necessary that the k-th ordered part is assembled into the k-th product.

To give at least an idea of how process chains look like, Fig. 1 shows the simple top level model of the production process of the manufacturer where time proceeds from left to right which implies a causal order of tasks along horizontal lines. Process chains have arrow type nodes to describe subtasks which may be delays, manipulations of counters, service requests for simple servers or other functional units that provide services like the supplier that take orders. In Fig. 1 the task "Car manufacturing" is performed by a functional unit called "manufacturer" whose interface provides "manufacture" and "load_truck" services to an environment. Functional units are visualized at the bottom of a figure, i.e. in our example we find functional units for a manufacturer, a supplier and a carrier. At this level, the model does only show process creation and termination by source and sink nodes, a service call to a functional unit with an integer parameter and a special process to initialize 4 truck processes. The formalism additionally includes connecting bars of different kind in order to allow for fork/join operations, synchronization etc..

Functional units need refinement. Two basic units are predefined: counters and servers.

Counters behave like semaphores, they model limited resources which can be aquired and released. Counters typically describe space in storage areas, in performance analysis counters are considered passive resources. Servers offer only one service, which takes a time that is given by a parameter and due to limited internal resources congestion may happen and scheduling takes place. In performance analysis servers are active resources and well known in queueing networks.

The concept of services imposes a hierarchy of models. At each level one can specify source nodes which either produce processes independently or upon specific conditions like termination of a process in a sink node. The latter possibility is used to describe a cyclic behaviour. Due to space limitations we rather give a corresponding Petri net than the full process chain model for our example.

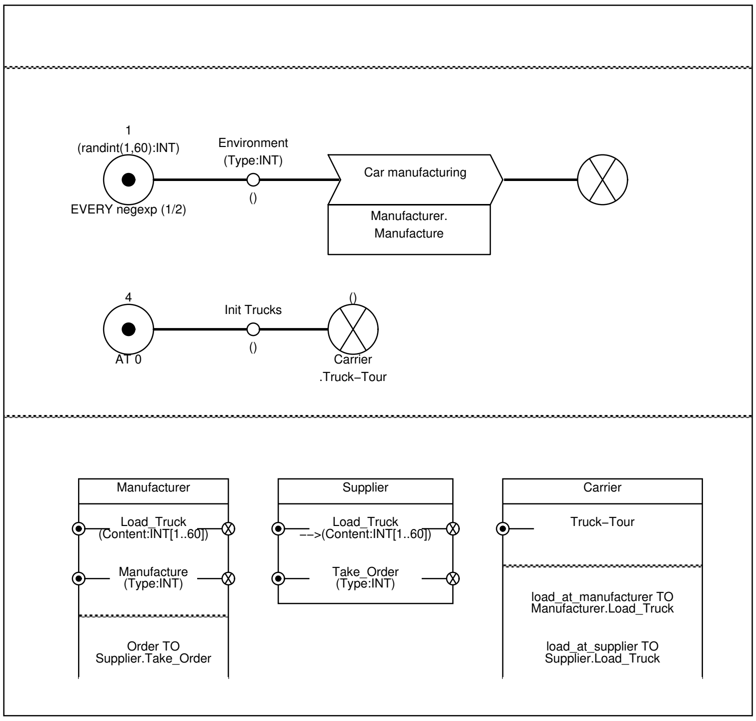

Figure 1: Top level description of manufacturer and supplier model

3 Petri net modeling

The motivation to define a mapping from process chain models to PN models is twofold: a) this way we obtain a clear understanding and semantics of the dynamic behaviour described by a process chain model provided all facettes of the formalism are considered and b) we are able to apply results from PN theory and analysis algorithms on the resulting nets. The latter is more restrictive on what aspects of process chains and how they are treated in the mapping. We focus on the latter approach and go for analysis methods which use the reachability graph and set, resp. an associated Markov chain, since we need performance characteristics like turnaround times and utilizations in the presence of random variables. Such a mapping has to live with certain side conditions mainly implied restrictions of numerical algorithms but by Petri net theory as well:

Data structures and Data dependencies

Process chain models allow for functions with parameters as well as for global and local variables.

The original model[1] distinguishes 60 variants of parts, which are randomly selected by the source of the manufacturing process and the information on which kind of part is carried in a container is stored in local variables/parameters of some process chains. Clearly, for GSPNs one can only encode data by natural variables (places) and constant transformation functions (vector addition/subtraction), which limits what can be described. Being aware of the consequences for state space dimensions, modelling of data and data dependent decisions is only possible in very limited and special cases. So we restrict ourselves to modeling counters which are modified by constant values only, parameters and variables are not modelled, data dependent decisions need to be randomized. In this paper we consider a model where we keep track of the total number of parts being transported, the different variants are only considered where they have an impact on timing, this is the case in the supplier's production and we model a variant specific production time by a random switch to choose a subsequent delay for production. The probabilities of the switch are equivalent to the random switch at the source of the manufacturing process. This construction goes beyond what one can handle automatically in a model.

Finite state space

Numerical analysis for a CTMC requires finiteness of state spaces[2]. For process chains, this is rather unusual since models of logistic systems are typically open models where tasks appear from and disappear into a rather unknown environment. For GSPN modelling we therefore artifically build cyclic systems of finite task populations, i.e. a task leaving the model via a sink reappears in a corresponding source. The impact of this approximation can be estimated by the probability of having an empty environment, i.e. no task can appear due to unavailability of "free" tasks in the environment. If this probability is low, the model captures roughly the behaviour of the unlimited model. The advantage is that these processes yields models of cyclic processes and interaction via synchronization of transitions. This type of models matches the concept of components for Kronecker methods for numerical analysis very well.

Limited size of CTMC and individual processes

A key decision in mapping processes is whether each individual process is mapped on a single safe Petri net or whether all processes of a single process chain is mapped onto the same and token populations correspond to the number of processes. Due to the state space explosion problem and the fact that an analysis method can be effectively applied only for a limited dimension of the CTMC we opt for the latter approach. Consequently we can treat fork/join structures not in a way such that exactly those processes are synchronized (joined) which have been separated (forked) before, but only (sub-)processes of appropriate kind. This problem in combination with timing aspects means that those subprocesses join which perform fastest, it was identified as Min-Match Problem in [3]. For the supplier example this means that parts are assembled if they are available and fit and not if the part is exactly the one ordered for the specific product.

Exponential/Markovian Timing

CTMC analysis is restricted to Markovian timing, i.e. durations must be represented by an exponential distribution or a phase-type distribution whose phases are build on exponential distributions. This is a rather strong limitation, since the constraint of state space dimensions reduces the potential to approximate other distributions arbitrarily well by phase-type distributions.

Figure 2 shows the PN model of our example system. The model is split up into subnets according to subprocesses. Subnets for boxes and containers are duplicated and indexed by "M" for manufacturer and "S" for supplier. Subnets are merged by fusion of transitions with equal labels. Due to an environment place, subnets become strongly connected and describe cyclic processes. The whole net shows one large T-invariant which corresponds to the following sequence of transitions/activities: the source subnet of the manufacturing net produces a token

[1] The original model has been produced by M. Schmitz and M. Völker.

[2] There are few exception like matrix geometric methods which only apply for specific systems.

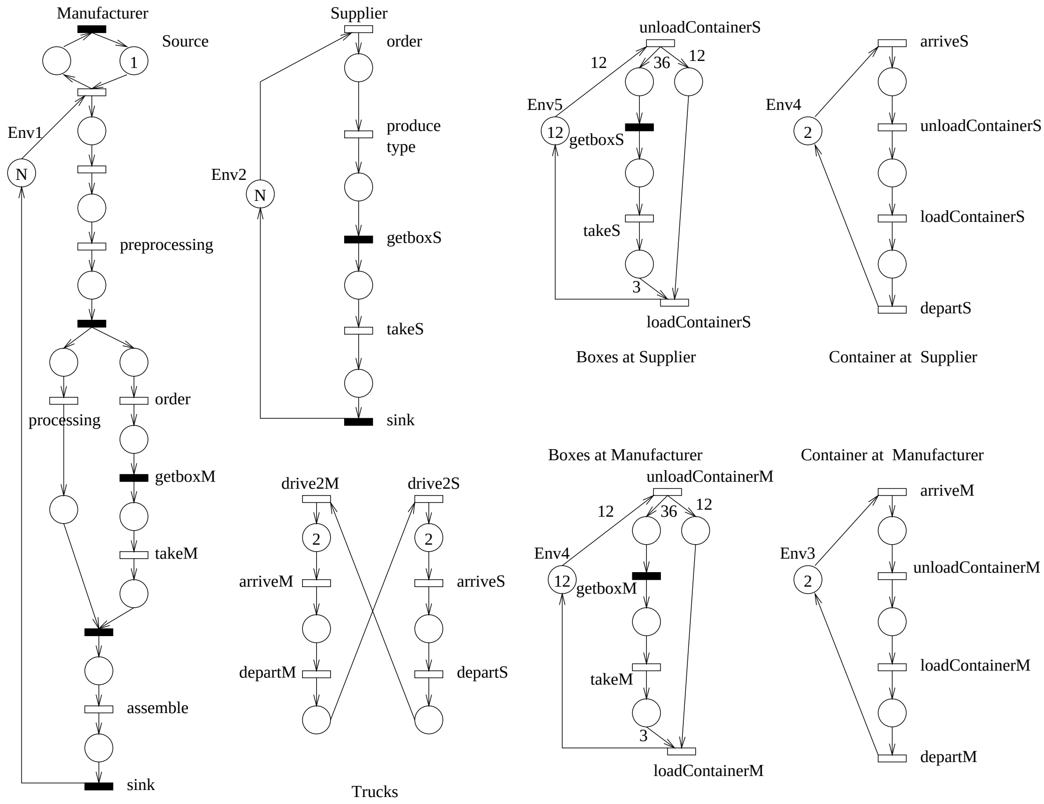

Figure 2: Petri net model

which proceeds downwards in the net and results in firing the order transition, which also takes place in the supplier process. Once the produce transition in the supplier has fired, the supplier needs to get a box and to put the part into it. Filling a box is only possible if a container has been unloaded. Once all 36 parts (equivalent to 12 boxes or 1 container) are filled, boxes are loaded into a container. Loading a container takes place in conjunction with a transition in the truck subnet: once loaded, a truck can depart and drive to the manufacturer. At the manufacturer side, containers and boxes get unloaded such that finally a "getbox" and "take" transition for parts can fire. If at the manufacturer side processing is finished and (concurrently) parts have been taken, a synchronization starts the final assembling step of the production.

The model seems to be of reasonable size, but state based analysis reveals very large state spaces. We assume 4 trucks, space for 1 container at the manufacturer and 1 container at the supplier. For performance analysis, an interesting question is whether the supplier is fast enough to provide a sufficient amount of parts such that no congestion takes place at the synchronization. For this question it is sufficient to analyze the model without the manufacturing process and to check whether the "order" transition of the supplier achieves a sufficient throughput. This model yields a tangible state spaces for N tokens on place "Env2" in the initial marking as given in columns "states"the table below.

N	states	N	states
1	144894	10	3187668
2	289788	20	11156838
3	482980	30	23955808
4	724470	40	41584578
5	1014258	50	64043148

Clearly numerical solution of CTMCs of this size can be achieved only by using appropriate hardware (we use a Sun Enterprise 250, 400 Mhz CPU, 2GB main memory under SunOS 5.7) and Kronecker representations for a space efficient representation. With the current implementation in the APNN toolbox [1] we are able to explore state spaces for $N = 40$ in 210 s, $N = 50$ in 330 s, an iterative solution by Gauss-Seidel takes for $N = 40$, 260 s wall clock time per iteration step. The number of necessary iteration steps depends on the required quality of solutions. For the Kronecker approach we use a partition into two components that follows the partition into subnets given in Fig. 2, i.e. we consider the supplier and the boxes at the supplier side as one component and the subnets for trucks, containers and boxes at the manufacturer side as another component. In this case, we achieve a so-called superposed GSPN where synchronization takes place over timed transitions only, such that an elimination of vanishing states can take place locally during generation of component state spaces. The component including transportation has a constant dimension of 5197 states, while the other component increases with N from 114 to 50388 states for $N = 50$. Analysis takes place in three steps: 1) exploration of component state spaces and generation of the Kronecker representation of the CTMC, 2) exploration of the overall state space which results in a decision diagram type data structure to represent the tangible reachability set (TRS), and 3) numerical analysis by an iterative solution method like Gauss-Seidel using the Kronecker representation of the CTMC and the decision diagram representation of TRS. The representation of TRS is a directed acyclic graph with 2 levels, the top one with one node for the reachable states of the first component and the second with nodes for the conditional reachability sets of the second component. In our case we observe two nodes at the second level, such that the overall structure requires only 624508 bytes for representing TRS for $N = 50$. Due to lack of space we do not give further details on the timing behaviour observed for this model, the interested reader is referred to [5].

Acknowledgements I would like to thank M. Völker and M. Schmitz for providing the process chain model of the considered example.

References

[1] F. Bause, P. Buchholz, and P. Kemper. A toolbox for functional and quantitative analysis of DEDS (extended abstract). In R. Piugjaner, N. Savino, and S. Bartomeu, editors, *Proc. 10th int. Conference Computer Performance Evalutation, Modelling Techniques and Tools*, number 1469 in LNCS, pages 356–359. Springer, 1998.

[2] H. Beilner, F. Bause, H. Tatlitürk, A. van Almsick, and M. Völker. Zum B-Modellformalismus - Version B1 - zur Vorbereitung automatisierter Analysen von Modellen logistischer Systeme (in german). Technical Report Nr. 99002, Sonderforschungsbereich 559, Universität Dortmund, 1999.

[3] S. Donatelli and J. Franceschinis. The psr methodology: integrating hardware and software models. In J. Billington and W. Reisig, editors, *Application and Theory of Petri Nets 1996*, LNCS 1091, pages 133–152. Springer, 1996.

[4] A. Ferscha. A petri net approach for performance oriented parallel program design. *Journal of Parallel and Distributed Computing*, 15(3):188–206, 1992.

[5] P. Kemper. Über die Modellierung von Prozessketten durch generalisierte stochastische Petri Netze (in german). Technical report, Sonderforschungsbereich 559, Universität Dortmund, 2000. in preparation.

[6] A. Kuhn. *Prozessketten in der Logistik, Entwicklungstrends und Umsetzungsstrategien (in German)*. Verlag Praxiswissen, Dortmund, 1995.