Integration of performance aspects into formal methods for concurrency*

Markus Siegle
Universität Erlangen-Nürnberg, IMMD VII
Martensstraße 3, 91058 Erlangen

1 Introduction

Parallel and distributed systems have to fulfill functional requirements (they should function as expected) and temporal requirements (they should meet performance requirements, e.g. offer a certain throughput or assure real-time limits). To meet these goals, designers need techniques and tools for both functional and temporal description and analysis at an early stage of the system lifecycle.

Description of complex systems, as well as their analysis, can be extremely difficult and costly. In this note, after a brief survey of classical methods, we discuss stochastic process algebras, a new constructive approach which promises to overcome some of the tractability problems.

2 Classical methods for functional description and analysis

The following is a list of formal methods for describing concurrency:

- Finite state automata (FSA) which communicate via messages (assuming either synchronous or asynchronous communication). Languages built on top of extended FSA (e.g. SDL [OFM$^+$94] or Estelle [BD87]) are practice-oriented, with existing tool support for many aspects of system design, from functional analysis to code generation.

- Petri nets (PN), comprising a large number of PN classes, ranging from basic place/transition nets to extended high-level nets.

*for *Colloquium on Formal Methods for Concurrency*, LMU München, July 5, 1996.

- Process algebras (PA). In addition to the "classical" process algebras such as CSP [Hoa85] and CCS [Mil89] there are now languages such as LOTOS [BB89], which come with tool support, aiming at practical system design.
- Event structures (ES) [Win86].
- Graph grammars (GG) [Nag79].

The various formalisms employ algebraic, textual or graphical methods for system description. Some methods, for instance Petri nets and graph grammars, provide a combination thereof.

The notion of *event* is common to all of the above methods. An event is an atomic, timeless "action" which can describe any aspect of system behaviour, such as the sending or receipt of a signal, the creation or termination of a process, etc.. While events describe actions, there is a dual view, namely that of *state*. The two views are interchangeable: While an event causes the transition from one state to another, a state is characterized by the set of events which can take place in that state.

Since behaviour is abstracted to discrete events, in concurrent systems events may occur either sequentially or in parallel. A system is completely characterized by all possible sequences of events it can exhibit, or – the dual view – by all possible sequences of states it can go through. Therefore, the purpose of system description is to unambiguously specify all possible sequences of events. This is achieved by describing causal relations between events, with the help of partial orders of events or pre- and post-conditions of events (graphically, causality relations are often depicted by directed arcs). In addition, it is important to define the initial state of a system, i.e. the starting point of its behaviour.

What is special about *concurrent* systems as opposed to sequential systems? In concurrent systems, we can observe truly concurrent behaviour, i.e. several activities which proceed in parallel. There is some interaction between components, but some parts may behave completely independently of each other. One can look at a concurrent system from the local point of view of one selected component (e.g. a process), or from a global point of view (that of an (ideal) external observer who sees the system as a whole). In concurrent systems, there is the phenomenon of non-determinism, i.e. under the same conditions (the same stimulus from the environment) different system behaviour is possible. This can be caused, for instance, by unpredictable signal delays in systems which communicate via asynchronous communication.

Analysis techniques for formal descriptions of (concurrent) systems can be classified as follows:

- Direct, algebraic techniques (such as the computation of invariants for Petri nets).
- Reduction techniques, which aim at transforming a given system into an equivalent but simpler one (e.g. one whose properties are already known).

- Indirect techniques, usually state-space-based. All states of the system are generated and every single state is checked for the property of interest (e.g. reachability analysis of Petri nets).

3 Classical methods for performance evaluation

We only consider model-based techniques for performance evaluation, disregarding experiments on the real system (e.g. measurement). The following list contains the most well-known modelling methods:

- Markov models [Ste94], a very universal and mathematically well-understood technique. The main disadvantage is their relatively low level, which makes it very difficult to describe complex systems. Therefore, higher-level formalisms have been developed on top of Markov processes, with Markov processes being the underlying "semantic model".
- Queueing networks (QN) [Kle75], a very popular technique.
- Stochastic graph models [ST87, Har93], also known as task graphs or precedence graphs. These are especially suited for modelling parallel programs and for workflow analysis.
- Stochastic Petri nets (SPN, GSPN [BK96]), an extension of Petri nets by stochastic time. Some properties of untimed PNs get lost. SPNs nets are now a well-established technique with many existing computer tools.

Analysis techniques:

- Algebraic, symbolic analysis.
- Specialized (very efficient) algorithms, e.g. product form solutions for QNs.
- Reduction techniques, e.g. series-parallel reduction of stochastic graph models, critical path analysis, bounding techniques.
- State space generation and numerical methods for Markov (semi-Markov, renewal, ...) processes.
- Simulation is always an option, but expensive.

4 A combined method: Stochastic Process Algebras

4.1 Overview

Stochastic Petri nets, in spite of their being very popular, have some disadvantages, which make it difficult to specify and analyze complex models: Petri nets are flat, unstructured, monolithic objects. Due to their lack of structure information, net design can easily become an intractable task even for experts. Furthermore, there is no way to approach the state space explosion problem, which often occurs during net analysis, by exploiting structure information.

Stochastic Process Algebras (SPAs) have a potential to overcome some of these problems through the concept of *constructivity.* The term constructivity describes the combination of the following features:

1. Building large models from small components. This results in multi-layered hierarchical models. The parallel composition and choice operators are used to compose components.

2. Abstraction, i.e. hiding of internal details which are not relevant to higher layers of the hierarchy. This is realized by a hiding operator.

3. There is a calculus which allows to compare two components with respect to some well-defined equivalence relation, such as functional bisimulation or Markov bisimulation.

As a result, PA descriptions are hierarchic, well-structured, modular objects. While being suitable for describing even complex systems down to their intricate details, they are — at each level of the hierarchy — simple enough for humans to comprehend. The third of the above features, that of equivalence, contains a great potential for model simplification, i.e. replacing components by equivalent ones which are easier to handle (e.g. with respect to the size of their state space).

The basic idea behind the extension of process algebras by the concept of time is somewhat similar to the idea which led to the development of SPNs: It is also an extension of a purely functional formalism by stochastic timing information. In SPAs, a subset of the actions is timed, i.e. there is a stochastic delay between the activation and the completion of the action. It is desireable to use generally distributed stochastic delays. From the point of view of model analysis, life is a lot easier if delays are exponential or of phase type. If this is this case, the underlying stochastic process is Markovian and can be easily analyzed using standard numerical techniques.

SPA research does not yet have a very long history. The beginnings date back to the late 1980ies. There is an early paper by Herzog [Her90], and major contributions can be found in the dissertations of Götz [Göt94], Hillston [Hil94] and Rettelbach [Ret96].

4.2 Example: The language TIPP

We briefly introduce the language TIPP, a simple SPA, which is defined by the following grammar:

$$P \;::=\; 0 \;\mid\; X \;\mid\; (a,\lambda).P \;\mid\; P+P \;\mid\; P\parallel_S P \;\mid\; recX : P \;\mid\; P\setminus L \;\mid\; !^n_S P$$

The non-terminal symbol P represents a process. The symbol $a \in Act$ is an action with its associated rate λ. The rate is interpreted as the parameter of an exponential distribution. 0 denotes a stopped process. $X \in Var$ is a process variable. The operators for prefixing, choice, parallel composition, recursion and hiding have the usual meaning (cf., e.g. [GHH+95]). $S \subseteq Act$ is the set of synchronizing actions for parallel composition of two processes. $L \subseteq Act$ is the set of actions which are hidden from the environment. Seen from the outside, actions in L are replaced by the invisible action τ.

The somewhat unusual replication operator is defined as:

$$!^n_S P := \underbrace{P \parallel_S P \parallel_S \ldots \parallel_S P}_{n \text{ times}}$$

Thus, the replication operator $!^n_S$ is a specialisation of the general parallel composition operator $\parallel_S$. It is a shorthand notation intended to ease the description of systems with replicated components. It also helps to automatically recognize symmetries and apply state space reduction techniques.

4.3 Analyzing SPA descriptions: state of the art

Having defined the language, the next step is to formally specify its meaning. The meaning of SPA specifications is usually given by structural operational semantics (SOS). Semantic rules are used for building a labelled transition system (LTS) which consists of states and state transitions. (In finite state models) the LTS covers all possible behaviour of the system. It is therefore a suitable basis for functional analysis.

For performance evaluation, the underlying stochastic process has to be studied. In the easiest case, a continuous time Markov chain (CTMC) is derived from the SPA description. If all actions have an exponential delay, the CTMC is isomorphic to the LTS. In case of both exponential and timeless (immediate) actions, some states of the LTS do not contribute to the performance measures and therefore can be eliminated [Ret96]. This is somewhat similar to the elimination of vanishing states during GSPN analysis.

A set of axioms has been established for the language TIPP which allows to check on a syntactical level, whether two descriptions are equivalent [HR94]. The underlying concept of equivalence is that of Markov bisimulation, i.e. both functional and temporal behaviour are covered. It has been shown that this set of axioms is both sound and complete.

Tool support is provided through the TIPP-tool. It has a LOTOS-like input language and includes algorithms for functional and temporal analysis. Its performance evaluation part offers numerical methods for steady-state and transient CTMC solution, some of which can exploit system properties (e.g. use coupling degree between components during iterative aggregation/disaggregation).

4.4 Current focus of SPA research

In view of the tractability problem of complex systems, there are some important questions to be answered:

- How can minimal (non-redundant) representations of a model be generated efficiently? A first approach towards the generation of a minimal semantic model was made by developing a compositional semantics for SPA [RS94, Boh95]. In this work, equivalent states are identified at the earliest possible stage.
- Which sophisticated, structure-based solution techniques can be borrowed from other domains? Promising candidates are, e.g. tensor techniques for structured Markov analysis or matrix geometric solution for specially structured processes with infinite state space.
- Is hierarchical analysis possible? There are, for instance, hierarchical analysis techniques for stochastic graph models which can be adapted for SPAs. First ideas on graph-based SPAs and their numerical analysis are described in [Her96]. This approach works for generally distributed action delays.
- Can coding techniques be used for storing and handling large state spaces efficiently? Binary decision diagrams (BDDs) [EFT93] are possible candidates. However, coding of temporal information is still an unsolved problem.

5 Conclusion

In this note, we briefly surveyed classical methods for concurrency. We then described SPAs, a combined formalism for functional and temporal specification of parallel and distributed systems.

As already stated in the introduction, tractability is the main problem during design and analysis. When specifying complex models using SPAs, the concept of constructivity is extremely helpful, because the designer can benefit from modularity, concentrating on subsystems with well-defined interfaces. During analysis — both functional and temporal — the main question is how to deal with the state space problem? Here again, constructivity of SPAs can be exploited. Instead of blindly generating a (flat) state space, thereby losing all structure information, it is possible to use structure information in order to build structured semantic models and structured performance models.

References

[BB89] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In P.H.J. van Eijk, C.A. Vissers, and M. Diaz, editors, *The Formal Description Technique LOTOS*, pages 23–73. North-Holland, Amsterdam, 1989.

[BD87] S. Budkowski and P. Dembinski. An Introduction to Estelle. *Computer Networks*, 14(1), 1987.

[BK96] F. Bause and P. Kritzinger. *Stochastic Petri Nets*. Vieweg, Braunschweig, 1996.

[Boh95] H. Bohnenkamp. Kompositionelle Semantiken stochastischer Prozeßalgebren zur Erzeugung reduzierter Transitionssysteme. Master's thesis, Universität Erlangen–Nürnberg, IMMD VII, 1995.

[EFT93] R. Enders, T. Filkorn, and D. Taubner. Generating BDDs for symbolic model checking in CCS. *Distributed Computing*, (6):155–164, 1993.

[GHH+95] N. Götz, H. Hermanns, U. Herzog, V. Mertsiotakis, and M. Rettelbach. *Quantitative Methods in Parallel Systems*, chapter Constructive Specification Techniques – Integrating Functional, Performance and Dependability Aspects. Springer, 1995.

[Göt94] N. Götz. *Stochastische Prozeßalgebren – Integration von funktionalem Entwurf und Leistungsbewertung Verteilter Systeme*. Dissertation, Universität Erlangen–Nürnberg, Martensstraße 3, 91058 Erlangen, April 1994.

[Har93] F. Hartleb. Graph Models for Performance Evaluation of Parallel Programs. In A. Bode and M. Dal Cin, editors, *Paralle Computer Architectures: Theory, Hardware, Software, Applications*, pages 80–86. Springer, 1993. LNCS 732.

[Her90] U. Herzog. Formal Description, Time and Performance Analysis. A Framework. In T. Härder, H. Wedekind, and G. Zimmermann, editors, *Entwurf und Betrieb Verteilter Systeme*, pages 172–190. Springer Verlag, Berlin, IFB 264, 1990.

[Her96] U. Herzog. A Concept for Graph-Based Stochastic Process Algebras, Generally Distributed Activity Times and Hierarchical Modelling. Technical report IMMD VII 4/96, Universität Erlangen–Nürnberg, Martensstr. 3, 91058 Erlangen, May 1996.

[Hil94] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[HR94] H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, pages 71–88, Erlangen-Regensberg, July 1994. IMMD, Universität Erlangen-Nürnberg.

[Kle75] L. Kleinrock. *Queueing Systems*, volume 1: Theory. John Wiley & Sons, 1975.

[Mil89] R. Milner. *A Calculus of Communicating Systems.* Prentice Hall, London, 1989.

[Nag79] M. Nagl. *Graphgrammatiken: Theorie, Anwendungen, Implementierung.* Vieweg, 1979.

[OFM$^+$94] A. Olsen, O. Færgemand, B. Møller-Pedersen, R. Reed, and J.R.W. Smith. *Systems Engineering Using SDL-92.* Elsevier, 1994.

[Ret96] M. Rettelbach. *Stochastische Prozeßalgebren mit zeitlosen Aktivitäten und probabilistischen Verzweigungen.* Dissertation, Universität Erlangen–Nürnberg, 1996.

[RS94] M. Rettelbach and M. Siegle. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. In U. Herzog and M. Rettelbach, editors, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling*, pages 89–106, Regensberg/Erlangen, July 1994. Arbeitsberichte des IMMD, Universität Erlangen-Nürnberg.

[ST87] R. Sahner and K. Trivedi. Performance Analysis and Reliability Analysis Using Directed Acyclic Graphs. *IEEE Transactions on Software Engineering*, SE-13(10), October 1987.

[Ste94] W.J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.

[Win86] G. Winskel. Event Structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets, Part II*, chapter 325-392. Springer, LNCS 255, 1986.