Constraint-Oriented Specification of Performance Aspects

Joost-Pieter Katoen[1,2]

[1] Software Modeling and Verification, RWTH Aachen, Germany
[2] Formal Methods and Tools, University of Twente, The Netherlands

Abstract. This note sketches how to extend (distributed) system specifications with performance constraints. The emphasis is on how to include performance aspects in a modular way. The key of the approach is to specify random delays as separated processes that are composed in parallel with an untimed, functional system specification. The use of parallel processes as separate constraints is in accordance with the *constraint-oriented specification style* as originally proposed by Vissers *et al.*.

Constraint-oriented specification. The paradigm of "separation of concerns" has (and still is) of major importance in computer science. The constraint-oriented specification style [4] is a format par excellence to support this principle when specifying the observable characteristics of complex distributed systems. It has been originally developed to support the early phases of the design trajectory. Put in a nutshell, constraints such as local and end-to-end service constraints are viewed as separate processes. Parallel composition is used to combine these constraints much in the same vein as logical conjunction; see [5]. The constraint-oriented specification style has been quite successful in modeling the functional behaviour of complex protocol standards.

Performance. This note shows that not only functional but also *performance aspects*—that typically are considered a posteriori but should be considered in early design phases—can be specified conveniently by considering them as separate constraints. Besides being able to really view performance as a separate concern, the main advantage of this approach is that existing system specifications can easily be extended with quality-of-service aspects without altering the present processes (i.e., the constraints). As this note is intended to honour the constraint-oriented specification style as originally proposed by Vissers about twenty years ago [4], we do not treat all details (see [1]) but rather focus on the essential ingredients. We use Basic LOTOS, the process algebra that (due to the work by Vissers and his co-workers) has been standardised by ISO in 1989, as a framework to illustrate the approach and show how random time constraints specified as phase-type distributions can be imposed. The use of LOTOS is for illustration purpose only. By no means, the approach is restricted to LOTOS. The same principle applies to other specification languages as well.

Phase-type distributions. Phase-type distributions can be considered as matrix generalizations of exponential distributions, and include frequently used distributions such as Erlang, Cox, hyper- and hypo-exponential distributions. Intuitively, a phase-type distribution can be considered as a CTMC with a single absorbing state (a state that is never left once reached). The time until absorption of this absorbing CTMC determines the phase-type distribution [3]. Any random delay can be approximated arbitarily closely by a phase-type distribution. Fitting algorithms can be used to efficiently generate phase-type distributions from measurements in an accurate manner. Existing untimed specifications can thus be extended with rather general random timing constraints by just parallel composition.

The elapse operator. For specification convenience, an *elapse* operator is used to impose phase-type distributed time constraints on specific actions. The semantics of this operator is defined by means of a translation into the basic operators of LOTOS—it is, in fact, just "syntactic sugar". Due to the compositional properties of LOTOS, important properties such as congruence results carry directly over to this operator. Delays are imposed as time constraints between two actions, and a delay may be "interrupted" if some action of some kind occurs in the meanwhile. That is, the elapse operator is an operator with four parameters, syntactically denoted by [**on** S **delay** D **by** Q **unless** B]:

- a phase-type distribution Q that determines the duration of the time constraint,
- a set of actions S (start) that determines when the delay (governed by Q) starts,
- a set of actions D (delay) which have to be delayed, and
- a set of actions B (break) which may interrupt the delay.

Thus, for instance, [**on** $\{a\}$ **delay** $\{b\}$ **by** $\widehat{Q}$ **unless** $\varnothing$] imposes the delay of $\widehat{Q}$ (modeling a phase-type distribution) between a and b. Semantically, the intuition behind this operator is that it enriches the CTMC Q with some synchronization potential (yielding $\widehat{Q}$) that is used to initialize and reset the time constraint in an appropriate way. The time constraint is imposed on a process P by means of parallel composition, such as in:

$$P \,||_{S\cup D\cup B}\, [\textbf{on } S \textbf{ delay } D \textbf{ by } Q \textbf{ unless } B] \quad .$$

The elapse operator is in fact nothing special and can be defined in terms of the existing Basic LOTOS operators such as choice, disruption and enabling. Let Q be a an absorbing CTMC—in the sequel we will show how to specify such process in LOTOS. Then the process:

$$\widehat{Q} \;=\; [\textbf{on } s \textbf{ delay } d \textbf{ by } Q \textbf{ unless } b]$$

is defined as:

$$\widehat{Q} := (d \,;\, \textbf{exit}[]s;\, (Q >> d \,;\, \textbf{exit})\, [> b \,;\, \textbf{exit}) >> \widehat{Q}$$

How to specify phase-type distributions? It remains to be explained how phase-type distributions can be specified. As stated before, a phase-type distribution is just a CTMC, i.e., a transition system in which each transition is labeled with a non-negative real number that indicates the average speed of moving from one state to another. So, in principle, any formalism in which such processes can be described would do. To stay within the setting of LOTOS, just a small—though essential—add-on needs to be incorporated, viz. the delay prefix. In this way we explicitly separate between the advance of time and the occurrence of actions. That is to say, actions are not just considered to be atomic, but also are assumed to take no time. This distinction leads to a behaviour where two distinct phases can be distinguished: phases in which actions occur (and possibly, state changes) and phases in which time elapses (but no state change).

So, besides action-prefix (denoted $a\ ;\ P$) we now have $(\lambda)\ ;\ P$, where λ is a parameter of a negative exponential distribution. Intuitively, $(\lambda)\ ;\ P$ delays for a time which is exponentially distributed with rate λ prior to exhibiting the behavior of P. Stated differently, the probability to behave like P within t time units is $1 - e^{-\lambda \cdot t}$, or simpler: it takes on average $\frac{1}{\lambda}$ time units to evolve into P.

That's all we need. All other operators are unaffacted, and the semantic principles of LOTOS remain the same. The interpretation of the new delay prefix can be captured by the following laws:

$$\text{(B1) } P \,[]\, Q = Q \,[]\, P \quad \text{(B2) } (P \,[]\, Q) \,[]\, R = P \,[]\, (Q \,[]\, R) \quad \text{(B3) } P \,[]\, \textbf{stop} = P$$

$$\text{(B4) } (\lambda + \mu)\ ;\ P = (\lambda)\ ;\ P \,[]\, (\mu)\ ;\ P$$

The axioms (B1) through (B3) are well known and standard for process algebra. Axiom (B4) is a distinguishing law and can be regarded as a replacement in the Markovian setting of the traditional idempotency axiom for choice ($P \,[]\, P = P$). It reflects that the resolution of choice is modeled by the minimum of (statistically independent) exponential distributions. Together with standard laws for handling recursion on classical process calculi, these axioms can be shown to form a sound and complete axiomatization.

Using choice, sequential composition, and recursion we are now in a position to specify any CTMC (and thus any phase-type distribution and random delay) needed. For example, the process:

$$PH := (\lambda)\ ;\ \Big((\mu)\ ;\ \textbf{exit} \,[]\, (\kappa)\ ;\ (\nu)\ ;\ (\nu)\ ;\ (\nu)\ ;\ \textbf{exit}\Big)$$

describes an absorbing Markov chain of six states whose distribution of the time until absorption is plotted in Figure 1 (for a specific choice of the parameters λ, μ, κ, and ν). In fact, due to the co-existence of action- and delay-prefix as two separate constructs—separation of concerns!—the obtained language is expressive enough to even describe more general models, viz. continuous-time Markov decision processes.

Constraint-oriented performance aspects. Finally, let us show that existing constraint-oriented specifications can be simply enriched with performance constraints. Suppose we are given a specification of the form $P \,||_A\, Q$ where A is a

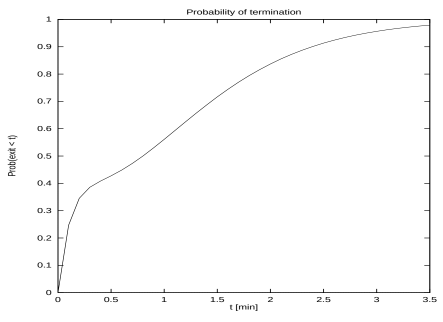

Fig. 1. Probability distribution described by PH ($\lambda = 10$, $\mu = 100$, $\kappa = 150$, $\nu = 2$)

set of actions. Now assume that we want to impose random delays on some of the observable actions from P and Q. This yields

$$(P \,||_A\, Q) \,||_{B_p \cup B_q}\, (\widehat{D_p} \,|||\, \widehat{D_q})$$

where B_p agrees with the alphabet of "delay" process D_p and B_q with the alphabet of delay process D_q. Note that the time constraints are added "on top" of the entire specification. As it suffices to impose a single delay on each action, the processes (actually, phase-type distributions) $\widehat{D_p}$ and $\widehat{D_q}$ are independent, and thus need not to synchronize. In case $\widehat{D_p}$ delays some local actions from P, and $\widehat{D_q}$ delays local actions from Q, the above specification can be rewritten into the equal (modulo weak bisimulation) specification by using parallel composition reshuffling (see [5]). This yields:

$$\underbrace{(P \,||_{B_p}\, \widehat{D_p})}_{\text{local constraints of } P} \qquad ||_A \qquad \underbrace{(Q \,||_{B_q}\, \widehat{D_q})}_{\text{local constraints of } Q}$$

Note that in this system specification, the functional and performance aspects of each individual component are separated, as well as the specifications of the components themselves.

Concluding remarks. This note hopefully has convinced you from the fact that performance aspects can be considered completely in line with the constraint-oriented specification style. For me, this is yet another striking example of why this specification style is so effective and useful. The strength of the proposed technique has been shown in [1] where an existing LOTOS specification (developed by others) of a plain-old telephone system has been enriched with performance constraints in a modular way. Using congruence results, this highly modular specification (with $> 10^6$ states) could be simplified in a component-wise fashion, finally yielding a model (of about 10^3 states) that was amenable to numerical analysis. This could not have been achieved without exploiting constraint-oriented specification.

References

1. H. Hermanns and J.-P. Katoen. Automated compositional Markov chain generation for a plain-old telephony system. *Science of Comput. Prog.*, 36(1):97–127, 2000.
2. P.R. D'Argenio, J.-P. Katoen and E. Brinksma. An algebraic approach to the specification of stochastic systems (extended abstract). In *Programming Concepts and Methods*. Chapman & Hall, pp. 126–147, 1998.
3. M.F. Neuts. *Matrix-geometric Solutions in Stochastic Models–An Algorithmic Approach.* The Johns Hopkins University Press, 1981.
4. C.A. Vissers. Architectural requirements for the temporal ordering specification of distributed systems. European Telecommunications Conference (EUTECO), North-Holland, pp. 79–93, 1983.
5. C.A. Vissers, G. Scollo, M. van Sinderen and E. Brinksma. On the use of specification styles in the design of distributed systems. *Theor. Comput. Sci.*, 89(1):179–206, 1991.