Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Realizability of schedules by stochastic time Petri nets with blocking semantics

Loïc Hélouët [a,*], Karim Kecir [b]

[a] INRIA Rennes, France
[b] INRIA Rennes & Alstom, France

A R T I C L E   I N F O

Article history:
Received 21 December 2016
Received in revised form 13 October 2017
Accepted 4 December 2017
Available online xxxx

Keywords:
Petri nets
Unfolding
Scheduling
Realizability

A B S T R A C T

This paper considers realizability of expected schedules by production systems with concurrent tasks, bounded resources that have to be shared among tasks, and random behaviors and durations. Schedules are high level views of desired executions of systems represented as partial orders decorated with timing constraints. Production systems (production cells, train networks...) are modeled as stochastic time Petri nets STPNs with an elementary (1-bounded) semantics. We detail their interleaved operational semantics, and then propose a non-interleaved semantics through the notion of time processes. We then consider *boolean realizability*: a schedule S is realizable by a net $\mathcal{N}$ if S embeds in a time process of $\mathcal{N}$ that satisfies all its constraints. However, with continuous time domains, the probability of a time process with exact dates is null. We hence consider *probabilistic realizability* up to α time units, that holds if the probability that $\mathcal{N}$ realizes S with constraints enlarged by α is strictly positive. Upon a sensible restriction guaranteeing time progress, boolean and probabilistic realizability of a schedule can be checked on the finite set of symbolic prefixes extracted from a bounded unfolding of the net. We give a construction technique for these prefixes and show that they represent all time processes of a net occurring up to a given maximal date. We then show how to verify existence of an embedding and compute the probability of its realization.

1. Introduction

Scheduling basic operations a priori in automated systems (e.g., manufacturing or transport systems, etc.) is a way to manage at best available resources, avoid undesired configurations, or achieve an objective within a bounded delay. Following a predetermined schedule is also a way to meet QoS objectives. For instance, operating a metro network or a fleet of buses usually amounts to implementing at best a predetermined timetable to meet users needs. In many cases, schedules are designed to avoid deadlocks, races or congestion problems, and ease up recovery from minor delays that are part of the normal behavior of the system. Failing to implement a chosen schedule may then result in a loss of QoS, lead to congestion when delays accumulate, cause conflicting accesses to shared resources and, in the worst cases, cause deadlocks. Schedules are high-level views for correct ordering of important or critical operations (i.e., that use shared resources) in a system. They consider time issues such as delays between tasks, optimal dates, durations...in a production plan. They can be seen as partial orders among basic tasks, that abstract low-level implementation details, and are decorated with dates and timing constraints.

* Corresponding author.
E-mail addresses: loic.helouet@inria.fr (L. Hélouët), karim.kecir@alstom.com (K. Kecir).

https://doi.org/10.1016/j.scico.2017.12.004
0167-6423/© 2017 Elsevier B.V. All rights reserved.

Designing a correct and optimal schedule for a system is a complex problem. On one hand, occurrence dates of events can be seen as variables, and correct and optimal schedules as optimal solutions (w.r.t. some criteria) for a set of constraints over these variables. Linear programming solutions have been proposed to optimize scheduling in train networks [1,2]. On the other hand, optimal solutions (for instance, w.r.t. completion date) are not necessarily the most probable nor the most robust ones: indeed, systems such as metro networks are subject to small delays (variations in trips durations from a station to another, passengers misbehavior...). Delays are hence expected and considered as part of the normal behavior of the system. To overcome this problem, metro schedules integrate small recovery margins that avoid the network performance to collapse as soon as a train is late. Consequently, optimal and realizable schedules are not necessarily robust enough if they impose tight realization dates to systems that are subject to random variations (delays in productions, faults...). Note also that the size of timetabling problems for real systems running for hours cannot be handled in a completely automated way by tools, that usually have to be guided by experts to return quasi-optimal solutions. Hence, timetables design involves expert competences that differ from those needed to design systems.

When a schedule and a low-level system description are designed separately, nothing guarantees that the system is able to realize the expected schedule. This calls for tools to check realizability of a schedule. One can expect systems designers and timetable experts to share a common understanding of the behavior of their system, so in general, the answer to a boolean realizability question should be positive. However, being able to realize a schedule does not mean that the probability to meet optimal objectives is high enough. Obviously, in systems with random delays, the probability to realize a schedule with precise dates is null. Beyond boolean realizability, a schedule shall hence be considered as *realizable* if it can be realized up to some bounded imprecision, and with a significant probability.

This paper addresses realizability of schedules by timed systems with concurrently running tasks, working in a stochastic environment. Such systems range from automated production cells, to transport networks and human organizations. The nature of considered systems calls for the use of concurrency models, for which one can expect efficient decision procedures as long as global states need not be computed. Addressing realizability also forces to consider time: one cannot consider that a schedule is realized if it cannot be implemented in a finite and reasonable amount of time. Last, realization of a schedule can not be addressed in a purely boolean setting: one can not consider that a system realizes a schedule if the probability of executions that implement a schedule is null. This calls for models with probabilities, and for a quantitative notion of realizability.

The first contribution of the paper is the formalization of the implementation model with stochastic time Petri nets (STPN for short) and of its partial order semantics. STPNs are inspired from [3], but with a *blocking semantics*. In this semantics, a transition cannot fire if its postset is not empty. Considering blocking is essential to model systems with shared resources. This semantics guarantees 1-boundedness of places along all runs. We show that this blocking semantics can be captured in a concurrent setting by *time processes*. However, in a timed setting, blocking has non-trivial consequences on both interleaved and partial-order semantics. First, it forces to adapt the notion of urgency, as an urgent transition may have to wait for the postset of a transition to be free to fire. A second difficulty is related to the partial order semantics. Processes are obtained by unfolding STPNs with standard algorithms [4,5], and then adding dates to events of these untimed processes in a way that is consistent with the time constraints attached to transitions of the net. A major difficulty is that under blocking semantics, process construction is not monotonic anymore. Indeed, due to blocking, one cannot decide to stop unfolding as soon as a set of constraints is unsatisfiable.

The second contribution of this paper is a boolean framework to address realizability of schedules by a STPN. Schedules are specified as partial orders decorated with time constraints. Boolean realizability of a schedule S by a STPN $\mathcal{N}$ is defined as the existence of a process of $\mathcal{N}$ that *embeds* S, and whose dates are compatible with the time constraint specified in S. We prove that upon some reasonable time progress assumption that guarantees time progress, realizability can be checked on a finite set of *symbolic processes*, obtained from a bounded untimed unfolding of $\mathcal{N}$. Symbolic processes are processes of the unfolding with satisfiable constraints on occurrence dates of events. A symbolic framework to unfold time Petri nets was already proposed in [6,7] but blocking semantics brings additional constraints on firing dates of transitions.

Embedding a schedule S in a process of $\mathcal{N}$ only guarantees *boolean realizability*: the probability of a time process in which at least one event is forced to occur at a precise date can be null. The third contribution of the paper is the study of a quantitative notion of realizability ensuring that a schedule can be realized up to some bounded imprecision, with a non null probability. Uncertainty in the systems we consider originates from random delays with low individual penalty. We hence do not consider major failures that result in hours of delays: such failures call for use of recovery scenarios that lay outside the normal behavior of a system, and realizability addresses normal situations perturbed by small deviations with respect to the expected most probable values. These deviations are defined via continuous probability distributions for the actual value that a duration may take. We use transient analysis techniques [3] for STPNs to compute the probability of occurrence of the set of processes that realize a schedule, and consequently prove its realizability up to an imprecision of α with a strictly positive probability. The last contribution of the paper is the application of boolean and quantitative realizability to a practical case study, namely verification of correct scheduling of train trips in a metro network.

The paper is organized as follows: Section 2 defines the models used in the paper, namely stochastic time Petri nets with a blocking semantics and schedules, and gives a concurrent semantics to STPNs. Section 3 defines a notion of unfolding and symbolic process for STPNs. Section 4 shows how to verify that a schedule is compatible with at least one process of the system and measure the probability of such realization. Section 5 shows how to check realizability on a case study, namely correct realization of trips in a train network. Section 6 compares the contents of this paper with former models and former