Expressing and Computing Passage Time Measures of GSPN Models with HASL

Elvio Amparore, Paolo Ballarini, Marco Beccuti, Susanna Donatelli, Giuliana Franceschinis

Expressing and computing passage time measures of GSPN models with HASL

Elvio Gilberto Amparore[1], Paolo Ballarini[2], Marco Beccuti[1], Susanna Donatelli[1], and Giuliana Franceschinis[3]

[1] Università di Torino, Dipartimento di Informatica
{beccuti, susi, amparore}@di.unito.it
[2] Ecole Centrale Paris, Laboratoire MAS
paolo.ballarini@ecp.fr
[3] Università Piemonte Orientale, Dipartimento di Informatica
giuliana.franceschinis@di.unipmn.it

Abstract. Passage time measures specification and computation for Generalized Stochastic Petri Net models have been faced in the literature from different points of view. In particular three aspects have been developed: (1) how to select a specific token (called the tagged token) and measure the distribution of the time employed from an entry to an exit point in a subnet; (2) how to specify in a flexible way any condition on the paths of interest to be measured, (3) how to efficiently compute the required distribution. In this paper we focus on the last two points: the specification and computation of complex passage time measures in (Tagged) GSPNs using the Hybrid Automata Stochastic Logic (HASL) and the statistical model checker COSMOS. By considering GSPN models of two different systems (a flexible manufacturing system and a workflow), we identify a number of relevant performance measures (mainly passage-time distributions), formally express them in HASL terms and assess them by means of simulation in the COSMOS tool. The interest from the measures specification point of view is provided by the possibility of setting one or more timers along the paths, and setting the conditions for the paths selection, based on the measured values of such timers. With respect to other specification languages allowing to use timers in the specification of performance measures, HASL provides timers suspension, reactivation, and rate change along a path.

1 Introduction

Performance analysis of systems through Generalized Stochastic Petri Net (GSPN) models or similar formalisms has evolved significantly since their introduction, in particular an interesting research direction concerns how to express and compute relevant performance measures. The languages that have been proposed to this purpose span from classical reward based ones to logics like the Continuous Stochastic Logic (CSL) and its extensions (action based, timed-automata based and reward based), to automata based languages. In this paper the focus is on the automata based languages, allowing to analyze a measure of interest on a *selected set of paths* through the model state space. Such paths are executions of a stochastic process (quite often a Continuous Time Markov Chain (CTMC) for which efficient analysis techniques exist) usually described

by means of a high level formalism like GSPN or Stochastic Process Algebra (SPA) of various sorts.

The *passage time distribution* is a specific type of performance index which is particularly useful when reasoning about properties related with Service Level Agreements (SLA) or safety requirements. In these cases classical performance measures based on mean values, like the average response time, are not sufficient and an estimate of the probability distribution for the time to complete a specific model activity (for example a recovery process) is needed instead. Moreover often only specific behaviors should be accounted for in the computation of such probability distributions, which brings us back to the need of estimating the passage time distribution on a subset of model evolution paths.

A passage time measure specification for CTMC is usually based on the definition of entry, goal and forbidden states: the distribution of the time required to reach a goal state from any entry state without hitting any forbidden state can be computed with different methods and tools [17, 12]. This typically requires the (automatic) manipulation of either the CTMC or of the high level model used to generate the CTMC. When specific paths must be isolated however, the CTMC may undergo a transformation, often obtained by synchronizing it with an automata describing the paths of interest. Examples of languages proposed in the literature to express these type of measures are the Extended Stochastic Probes (XSP [11], operating on PEPA models [16]), Path Automata (PA, operating on Stochastic Activity Networks [18]), Probe Automata (PrA [3], operating on GSPN or on Tagged GSPN [7]). More recently the HASL (Hybrid Automata Stochastic Logic) [8] language has been introduced, operating on any Discrete Event Stochastic Process (but up to now experimented only with an extended version of GSPNs): we shall exploit its expressive power in the present paper, with reference precisely to GSPN models.

Another issue when expressing passage time measures on high level models is the possibility to identify entities in the model (customers), and measure the time required for a selected entity to go through a number of steps (activities), corresponding to the movement of the entity through a sequence of components in the high level model (possibly conditioned on some state-based predicate being true). In GSPN terms this often leads to the need to follow a specific token through the net. This issue is not trivial if the token to be followed may go through places containing other tokens, since they are indistinguishable: this aspect has been tackled in the literature by introducing a formalism extension called Tagged GSPNs [11].

This paper introduces the use of the HASL logic for the specification of passage time measures over GSPN models extended with general firing time distributions. The measure computation is performed using the statistical model checker COSMOS: stochastic discrete event simulation of the GSPN stochastic process synchronized with a Linear Hybrid Automaton (LHA) is performed. Cosmos generates a (statistically significant) sample of GSPN paths conforming to the HASL specification, and estimates the measures of interest from such sample (a confidence interval is provided for each measure).

The conclusion is that the HASL expressive power is adequate to specify (passage time) performance measure as can be specified with PrA and with the TGSPN passage time measure specification language, and that the COSMOS statistical model checker

is an appropriate and useful tool to estimate such measures, even in presence of non exponential transition firing times and on models with very large state spaces. In addition more complex paths selection is possible, in particular those characterized by conditions on the duration of specific phases along the path (requiring to set one or more timers during the evolution, with the possibility of suspending, resuming and proceeding with different rates).

The paper is organized as follows: Section 2 describes the stochastic logic HASL, the associated LHA and the statistical model checker COSMOS. The use of LHA for the specification of passage time of paths of a stochastic Petri net model is then illustrated in Section 3, based on a simple workflow model taken from the literature; in this section we also report results for the passage time computation using COSMOS. The literature on the definition of passage time for tagged GSPN, based on entry, exit and forbidden condition for subnet identification or based on Probe Automata, is then recalled in Section 4. The difference and similarities of HASL based specification with respect to Probe Automata specification of passage times is discussed in Section 5, supported by a classical FMS example. Finally some conclusive remarks are given in Section 6.

2 Background: HASL

The Hybrid Automata Stochastic Logic (HASL) [9] is a recently introduced, automata-based, formalism for statistical model checking of discrete event stochastic processes (DESP). It enjoys two main features: generality and expressiveness. HASL is general with respect to modelling capabilities as it addresses a class of models (i.e. DESPs) which includes, but (unlike most stochastic logics) is not limited to, CTMCs. With respect to expressiveness HASL turns out to be a powerful language through which temporal reasoning is naturally blended with elaborate reward-based analysis. In that respect HASL unifies the expressiveness of CSL[4] and its action-based variant [5], timed-automata [14, 10] and reward-based [15] extensions, in a single powerful formalism. The HASL model checking method belongs to the family of statistical model checking approaches (i.e. those that employ stochastic simulation as a means to estimate a model's property) and, more specifically, it employs confidence-interval methods to estimate the expected value of the target measure (i.e. either a measure of probability or a generic real-valued measure). Finally a prototype software tool for HASL model checking, named COSMOS [8], gives the possibility to actually apply HASL to real case studies (see [8] and [13] for a comparison of COSMOSwith other tools implementing statistical model checking). In the following we informally introduce the basic elements of the HASL methodology referring the reader to the literature [9] for formal details.

2.1 HASL models: DESPs as GSPNs

The HASL logic refers to DESP models. Informally a DESP is a stochastic process consisting of a (possibly infinite) set S of states and whose dynamic is triggered by a (finite) set E of (time-consuming) discrete events. For reasons of generality no restrictions are considered on the nature of the delay distributions associated with events, thus any

distribution with non-negative support may be considered. In practice the HASL framework [9] has been formalized referring to a high-level representation of DESP, namely: an HASL model consists of an (extended) GSPN whose timed-transitions may be associated with any delay distribution with non-negative support (e.g. Exponential, Deterministic, Uniform, LogNormal, etc.). The choice of GSPNs as modeling formalisms is due to two factors: (1) they allow a flexible modeling w.r.t. the policies defining the process (choice, service and memory) and (2) they provide an efficient path generation (due the simplicity of the *firing rule* which drives their dynamics). For the sake of space in this paper we omit the formal definition of DESP but we rather introduce the rationale behind it through description of some examples. Also, in the remainder we will simply use the notation term GSPN referring, in facts, to a DESP model in GSPN form.

2.2 HASL formulas: a Linear Hybrid Automaton and a target expression

A HASL formula is a pair $(\mathscr{A}, Z)$ where $\mathscr{A}$ is Linear Hybrid Automaton (i.e. a restriction of hybrid automata [2]) and Z is an expression involving *data variables* of $\mathscr{A}$. The goal of HASL model checking is to estimate the value of Z by synchronization of a GSPN $\mathscr{N}$ with the automaton $\mathscr{A}$. This is achieved through stochastic simulation of the synchronized process ($\mathscr{N} \times \mathscr{A}$), a procedure by means of which infinite timed executions of process $\mathscr{N}$ are selected through automaton $\mathscr{A}$ until some final state is reached or the synchronization fails. During such synchronization, data variables evolve and the values they assume condition the successive evolution of the synchronization. The synchronization stops as soon as either: a final location of $\mathscr{A}$ is reached (in which case the values of the variables are considered in the estimate of Z), or the considered trace of $\mathscr{N}$ is rejected by $\mathscr{A}$ (in which case variables' values are discarded).

LHA: Again here we only provide an informal description of LHA referring the reader to [9] for formal definitions. Simply speaking an LHA is an automaton consisting of the following elements: a finite set of locations L (some of which are *Final* and some other are *Initial*); a finite set of events E (corresponding to the GSPN transition labels); a finite non-empty set X of n real variables; a location labeling function ($\Lambda : L \to Prop$) which associates each location with a (boolean) property that refers to GSPN states (i.e. markings); a flow function ($flow : L \mapsto Ind^n$) which associates with each location an n-tuple of GSPN indicators (conditions referring to GSPN markings) expressing the rate (i.e. the first derivative) at which each data variable in X changes in that location; variable's flow can be simple constants or functions of the current state of the GSPN (hence they are expressed by means of *GSPN state indicators*, denote *Ind*). Finally a transition from a source location l to a target location l' has the following form $l \xrightarrow{E',\gamma,U} l'$, where: I) $E' \subseteq E \cup \{\sharp\}$ is a set of labels of synchronizing events (including the extra label $\sharp$ denoting autonomous edges); II) γ is a constraint (i.e. a boolean combination of inequalities of the form $\sum_{1 \le i \le n} \alpha_i x_i + c \prec 0$ where $\alpha_i, c \in Ind$ are GSPN state indicators and $\prec \in \{=, <, >, \le, \ge\}$ and $x_i \in X$). III) U is a set of *updates* (i.e. an n-tuple of functions $u_1, ..., u_n$ where each u_k is of the form $x_k = \sum_{1 \le i \le n} \alpha_i x_i + c$ where the $\alpha_i, c \in Ind$ are GSPN indicators) by means of which new values are assigned to variables of X on traversing of the edge.

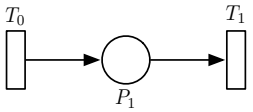
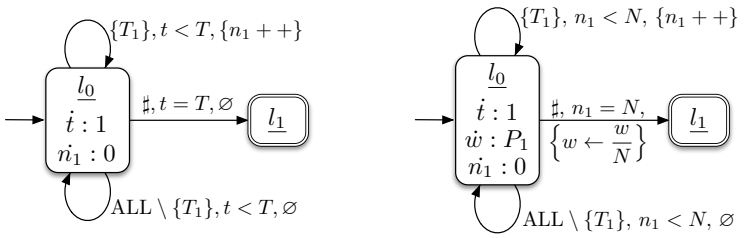


Fig. 1. A toy GSPN model.



(a) Time bounded measure.

(b) Event bounded measure.

Fig. 2. Example of LHA for simple properties of the GSPN model in Fig. 1.

Example of LHA: Figure 2 depicts two variants of a simple two locations LHA characterizing path measures of the toy GSPN model of Figure 1 (such a GSPN may represent a simple unbounded queue with service/arrival represented by T_1 and T_0 respectively, while tokens in place P_1 represent the customers in queue and in service). The locations named l_0 and l_1 are the initial and the final locations for the two automata. Both automata employ two data-variables: an integer variable n_1 (hence with flow $\dot{n}_1 = 0$ in every location), counting the occurrences of transition T_1, and a real-valued variable t which is used as a timer (hence with flow $\dot{t} = 1$) to record the simulation-time along the path; moreover the LHA of Figure 2(b) has a variable w with flow equal to the marking of place P_1 (hence it measures the integral of the number of waiting customers in the queue) along the observed path. Both automata have two synchronizing edges (the self-loops on l_0) and one autonomous edge (from l_0 to l_1). The topmost synchronizing edge synchronizes with occurrences of T_1 and by incrementing the value of n_1 at each synchronization it allows for counting the occurrences of T_1 (stored in n_1). The bottommost synchronizing edge, on the other hand, synchronizes with any transitions (denoted *ALL*) of the GSPN model except T_1, without performing any update. The autonomous edge $l_0 \to l_1$ instead leads to the final location as soon as its constraint is fulfilled. Note that the the condition leading to the final location in the LHA of Figure 2 (b) represents a time-bounded constraint on the simulation time t, i.e.: as soon as $t = T$ the processed path is accepted (and, by that time, n_1 will be equal to the number of firings of T_1 up to time $t = T$). On the other hand, in the LHA of Figure 2 (b) the condition that leads to the final location is an event-bounded constraint, as it accepts paths as soon as T_1 has occurred $n_1 = N$ times. More specifically, variable n_1 is incremented every time T_1 fires in location l_0, until n_1 becomes equal to N; in the same location variable w grows with rate equal to the marking of P_1. Just before taking the transition from l_0 to l_1, w contains therefore a quantity that corresponds to the summation of the residence times

of all tokens observed in $P1$ along the simulation; when the transition is taken, it is updated to $w = w/n_1$, hence, on acceptance, w will be equal to the average residence time of customers in place P_1 along the observed path, that is to say after the first N occurrences of T_1.

HASL expression: The second component of an HASL formula is an expression, denoted as Z and defined by the following grammar:

$$\begin{aligned} Z &::= E(Y) \mid Z+Z \mid Z \times Z \mid CDF_I(Y) \mid PDF_I(Y) \mid PROB() \\ Y &::= c \mid Y+Y \mid Y \times Y \mid Y/Y \mid last(y) \mid min(y) \mid max(y) \mid int(y) \mid avg(y) \\ y &::= c \mid x \mid y+y \mid y \times y \mid y/y \end{aligned} \tag{1}$$

y is an arithmetic expression built on top of LHA data variables (x) and constants (c). Y is a path dependent expression built on top of basic path random variables such as $last(y)$ (i.e. the last value of y along a synchronizing path), $min(y)$/$max(y)$ (the minimum/maximum value of y along a synchronizing path), $int(y)$ (the integral over time along a path) and $avg(y)$ (the average value of y along a path). Finally Z, the target measure of an HASL experiment, is an arithmetic expression built on top of the first moment of Y ($E[Y]$), and thus allowing to consider more complex measures including, e.g. $Var(Y) \equiv E[Y^2] - E[Y]^2$, $Covar(Y_1, Y_2) \equiv E[Y_1 \cdot Y_2] - E[Y_1] \cdot E[Y_2]$.
The expressions $CDF_I(Y)$ and $PDF_I(Y)$ compute a sequence of cumulative/instantaneous values, subdivided in discrete samples of uniform size described by a *sampling interval* $I = \langle t_0, t_{Final}, \Delta t \rangle$, where t_0 and t_{Final} are the extremes of the sampling and Δt is the size of the uniform step at which the samples are taken. Usually, the target data variable of CDF/PDF is a time counter, in order to compute a density or a distribution function. The target expression $PROB()$ measures the mean number of paths accepted by the LHA automaton over the total number of simulated paths. It is the only operator that is influenced by rejected paths, since $E(Y)$, $CDF_I(Y)$ and $PDF_I(Y)$ takes samples from accepted paths only.

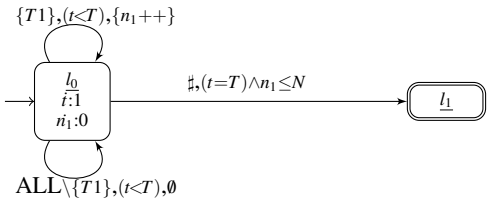


Fig. 3. The LHA used to compute the probability of observing less than N occurrences of $T1$ within time T.

Measures of probability with HASL:

With reference to the two LHAs of Figure 2, and considering the quantities accumulated along the accepted paths in the variables t, n_1 and w, we can define the expressions $E(last(n_1))$ for the LHA (a) to estimate the mean number of $T1$ firing up to time

T, while for the LHA ((b) we could define $E(last(w))$ to estimate the mean average waiting time for behaviors up to the N-th firing of $T1$ and $CDF_I(t)$ for a given interval, to estimate the (normalized) distribution of the time to complete

Note that, in the definition provided by HASL, and its corresponding computation implemented in Cosmos, the CDF is actually normalized, so as to asymptotically reach 1. This is motivated by the fact that the CDF of certain quantities may be "defective" in HASL, since it is computed only on accepted paths: if only a subset of the paths is accepted, the CDF may not tend to 1 but to an asymptotic value which is the probability of accepted paths.

Figure 3 shows another simple example of LHA (a small variant of the LHA in Figure 2(a)) for measuring ϕ_1: *the probability that the number of occurrences of transition T_1 within time T does not exceed N*. This is achieved by using the condition $(t = T) \wedge (n_1 < N)$ on the autonomous edge from l_0 to l_1 and $t < T$ on the other two edges departing from l_0. If a path counts more than N occurrences of $T1$ within time T, the edge $l_0 \rightarrow l_1$ will not be triggered and the LHA will not be able to synchronize with any successive GSPN transitions. Such a path is therefore rejected.The rate of acceptance is measured by defining the target measure $Z = PROB()$.

3 HASL and passage time of selected paths

Given a discrete-state stochastic process with state space S, and sets $E, G, D \subset S$ the *passage-time* $Prob(E, G, D, t)$ is a CDF measure expressing the probability of reaching any goal state in G starting from any enter state in E and avoiding forbidden states in D and with a delay no greater than t [17, 12]. The measure $Prob(E, G, D, t)$ is defined in function of the state-dependent random variable $P_x^{G,D}$, denoting the probability to reach a goal state in G starting from state x and avoiding states in D.
Here we discuss how to take advantage of an expressive property specification formalism, such as HASL, from the point of view of passage-time related measures. More specifically we are going to show how the HASL formalism can be used to express "standard" passage time measures as well as more complex ones. In other words we consider the possibility of an extended characterisation of passage-time measure where the constraining factor does not necessarily consist of state-conditions (i.e. the forbidden states D) but it may involve performance characteristics of the model (i.e. measured during the passage from an entry state to the reaching of a goal state). Such *performance-constrained* extensions of passage time can be formally expressed and measured through the HASL logic.

In this section we consider passage times as the time to traverse a set of selected paths, while we shall consider the use of HASL for representing the passage time of tagged customers in subnets of GSPN models, as defined in [3], in Section 4. The presentation in this section is supported by a specific GSPN model taken from the literature [1], which represents a business workflow. This will allow us to illustrate the richness of HASL in characterizing paths based on the actions performed and on the states visited along the path, as in any stochastic logic, but also based on accumulated discrete and continuous measures, including accumulated performance indices. The generality

of the model considered will also allow us to bring evidence of the usefulness of such a rich specification language.

3.1 Business workflow model

The model considered is a case of an order-handling business process model, taken from [1]. Fig. 4 shows the GSPN that models the processing of a single order. The workflow

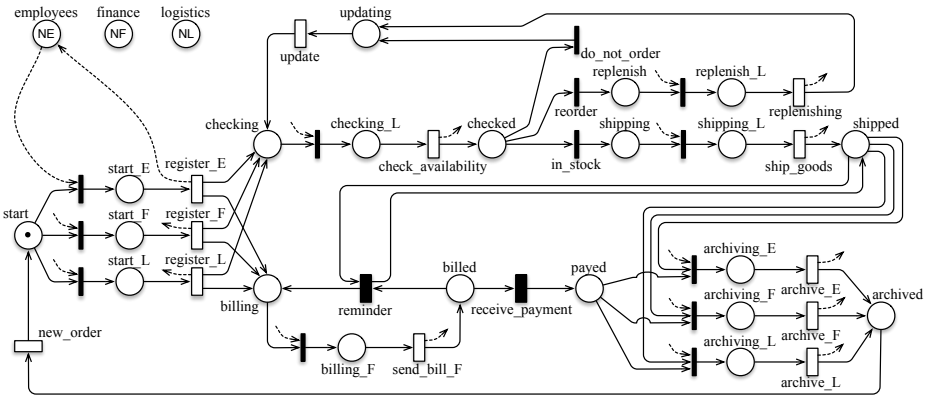


Fig. 4. The workflow model for an order-handling process.

involves two separate tasks of preparing and sending the bill to the client, and to ship the requested goods. The company reckons on three types of employees: those who manage accounting (F), logistics (L) and generic employees (E). Different tasks are carried out by different employees. The Petri net is made of some subnets consisting of an immediate transition, a place and a timed transition. Such subnets first allocate one of these staff resources, execute the specified task and then release the resource. The staff is represented by three places *finance*, *logistics* and *employees*. Arrows from and to these three places are drawn only for the case of the activity represented by the *register_E* transition, done by a generic employee, which is reserved just before the *register_E* transition becomes enabled, and released upon firing. All the other subnets whose timed transitions have labels with suffixes "_E", "_F" and "_L" use a similar schema, acquiring and releasing the appropriate employee resources.

The Petri net represents the lifetime of an order from the *start* place, when it is received, to the *archived* place, when it has been served. Upon receiving of an order, one employee prepares the request to the warehouse and to the accounting department. A logistic personnel checks if the requested item is available: if it is not, a reorder is issued (*replenish*), and the shipping is delayed until the items are available (*update*). In the meanwhile, the bill is sent to the client. If after some time the payment has not been received (*receive_payment*), the billing is resent (*reminder*). When the item has been shipped and payed, the request is archived. The actual model contains several replicas

of the model in Figure 4, all sharing the three resource places. Since replicas are kept separate, we can easily follow the possible paths followed by each single order.

3.2 HASL based passage-time measures

Referring to the GSPN model of the business workflow we illustrate a number of passage-time measures expressed in HASL terms. For each measure we provide first an informal specification then the corresponding formal characterization as an HASL formula (i.e. an LHA paired with an HASL expression as by grammar (1)). Finally for each such measures we provide numerical results obtained by running experiments with the COSMOS tool.

Measure w1: *the CDF of the passage-time for an ordered good to be delivered.*

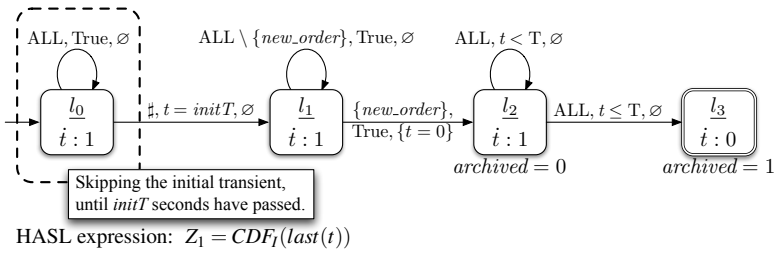


Fig. 5. The LHA automaton for the workflow passage-time w_1.

The probability measure w1 can be encoded by the HASL formula $\phi_{w1} = (\mathscr{A}_1, Z_1)$ where $\mathscr{A}_1$ is the LHA depicted in Figure 5 and Z_1 is the HASL expression $Z_1 = CDF_I(last(t))$. The automaton $\mathscr{A}_1$ employs one data variable: a timer t that records the simulation time of the synchronising paths. $\mathscr{A}_1$ works as follows: the initial location l_0 is used only to emulate a transient window, letting a random trajectory of duration $initT$ being simulated before the actual analysis starts in location l_1. The automaton then remains in l_1 for as long as the first occurrence of (the GSPN) transition *new_order*, whose firing takes $\mathscr{A}_1$ into location l_2: note that on traversing the $l_1 \to l_2$ edge the timer t is reset, corresponding to the beginning of the passage-time measuring. From l_2 a path is accepted (i.e. by reaching final location l_3) as soon as the GSPN place *archived* is filled in with a token (representing the delivering of previous incoming order). The edge from l_2 to the final locations l_3 is taken only when the condition *archived*=1 is met. If at the instant when place *archived* is filled in with one token the passage-time is $t < T$ then the path is accepted. On the contrary, if $t > T$ the LHA becomes unable to synchronize with further transitions of the GSPN, and the path is rejected.

Measure w2: *the CDF of the passage-time for an ordered good to be delivered given that it was out-of-stock.*

The probability measure w2 can be encoded by the HASL formula $\phi_{w2} = (\mathscr{A}_2, Z_2)$ where $\mathscr{A}_2$ is the LHA depicted in Figure 6 and $Z_2 = CDF_I(Last(t))$. Automaton $\mathscr{A}_2$ is

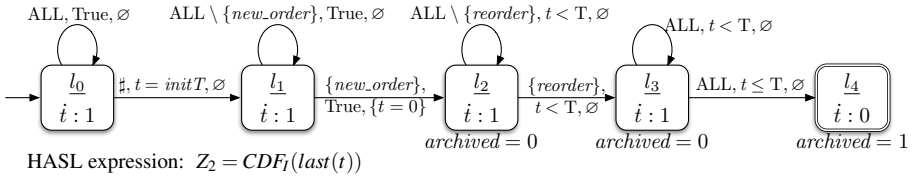


Fig. 6. The LHA automaton for the workflow property w_2.

a variant of $\mathscr{A}_1$ and employs the same data variable t for the elapsed time. As in $\mathscr{A}_1$ the initial location l_0 is used only to let the transient window pass before entering in l_1 where the actual analysis of the synchronised path begins. From l_1 the LHA moves to l_2 on firing of a *new_order* transition (which also trigger the timer t reset). From l_2, l_3 is reached only on occurrence of a *reorder* transition (i.e. the ordered good was out-of-stock). From l_3 the final location l_4 is then reached under exactly the same conditions as of $\mathscr{A}_1$. As a result the GSPN paths leading to the final location are those which contain an occurrence of *new_order* followed by an occurrence of *reorder* and that finally lead to a marking $M(archived) = 1$.

Measure w3: *the CDF of the passage-time for an ordered good to be delivered given that it is not out-of-stock and that the total delay for checking its availability and shipping it does not exceed K .*

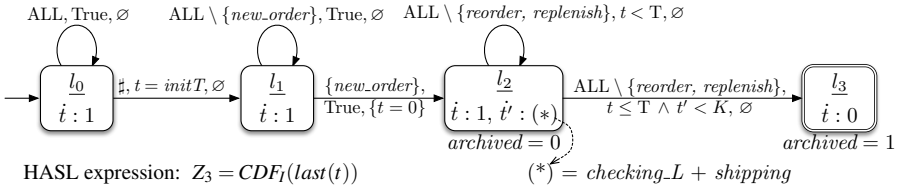


Fig. 7. The LHA automaton for the workflow property w_3.

The probability measure w3 can be encoded by the HASL formula $\phi_{w3} = (\mathscr{A}_3, Z_3)$ where $\mathscr{A}_3$ is the LHA depicted in Figure 7 and $Z_3 = CDF_I(last(t))$. Automaton $\mathscr{A}_3$ employs two data variables: a timer t, (as $\mathscr{A}_1$ and $\mathscr{A}_2$) plus an extra real-valued variable t' which is used to further conditioning the accepted paths. More specifically t' is used to measure the total time that the system spends in either checking the availability of the ordered good or in shipping it while assuming that the ordered good does not require a *reorder* (i.e. it is present in stock). For this the rate of variation of t' (i.e. its flow) corresponds to the sum of the marking of places *checking_L* and *shipping* (location l_2). Furthermore note that the exclusion of paths containing an occurrence of either *reorder* or *replenish* is obtained by using $ALL \setminus \{reorder, replenish\}$ in the on all arcs departing from l_2 as synchronisation (constraint) set: i.e. if the currently simulated GSPN path has lead to l_2 then the occurrence of either *reorder* or *replenish* causes the path to be

rejected (no LHA transition is enabled). Finally, from l_2 a path is accepted as soon as place *archived* is filled in with one token (delivery of the ordered good) and the condition on t' is satisfied (i.e. if $t' < K$, where K is a constant parameter of the $\mathscr{A}_3$).

Measure w4: *the CDF of the passage-time for an ordered good to be delivered given that the total delay for reordering and updating the stock does not exceed K .*

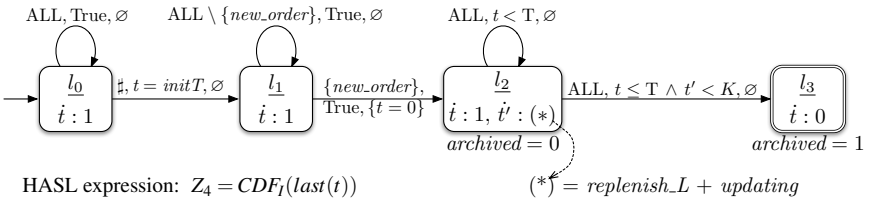


Fig. 8. The LHA automaton for the workflow property w_4.

The probability measure w4 can be encoded by the HASL formula $\phi_{w4} = (\mathscr{A}_4, Z_4)$ where $\mathscr{A}_4$ is the LHA depicted in Figure 8 and $Z_4 = CDF_I(Last(t))$. Automaton $\mathscr{A}_4$ is a variant of $\mathscr{A}_3$ and uses the same two data variables only that now t' is used to measure the total time that the system spends in either replenishing after a re-order or updating the stock. Thus the rate of variation of t' corresponds to the sum of the marking of places *replenish_L* and *updating* (location l_2).

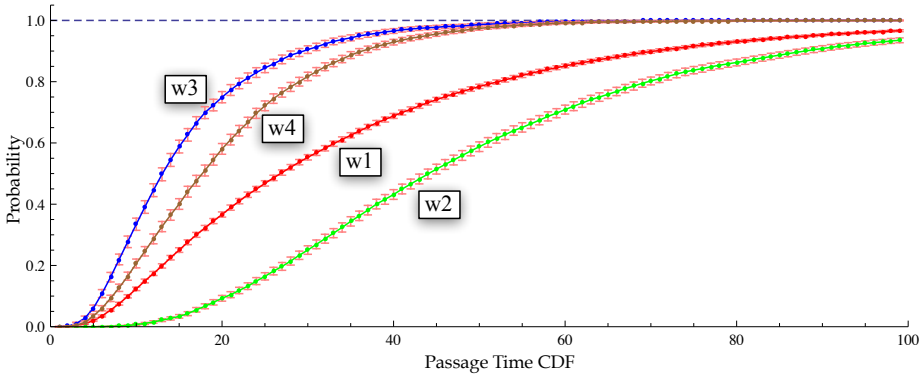


Fig. 9. Passage times for the workflow properties w_1 to w_4, with *sampling interval* $I = \langle 0, 100, 1 \rangle$.

Experiments: to assess the value of w_1, w_2, w_3 and w_4 we have performed a number of experiments using the COSMOS tool: i..e. we encoded the GSPN model of the business workflow and the LHA formulae $\phi_{w1}, \phi_{w2}, \phi_{w3}, \phi_{w4}$ into COSMOS and executed a set of experiments. Figure 9 shows the plots of the results for the passage-time CDF

corresponding to HASL formulae $\phi_{w1}, \phi_{w2}, \phi_{w3}, \phi_{w4}$, including the confidence intervals for each time sample.

The plot of ϕ_{w1} represents the CDF of the average passage time in the workflow net, from the arrival of a new order to its archiving. The workflow has three replicas of the model, and has 1 generic employee, 1 accounting employee and 2 persons at the logistics. The transition names appearing on LHA edges refer to the first of the three replicas (due to the system symmetry the result is independent on the replica chosen for the measure). The other three curves represent the passage time distributions of a selected set of paths, where only paths that respect the additional constraints described in the LHAs are considered. For instance, ϕ_{w3} selects only paths that avoid reorders and also perform checking and shipping within a given time bound, so the average passage time of ϕ_{w3}-accepted paths will be less that that of ϕ_{w1}. On the contrary, ϕ_{w2} consider only paths that do at least one reorder, so their average passage time will be greater than that of ϕ_{w1}. Finally ϕ_{w4} lays in between ϕ_{w1} and ϕ_{w3} because it admits paths that require a reorder, but impose a limit on the total delay for reordering and updating the stock.

The workflow model keeps the status of each ordered good separated, by using a distinct replication of the subnet where the order is circulating. In general, keeping tracks of specific tokens in a net can be derived by a proper tagging of a selected token through the subnets where the tagged token flows. This can be done automatically by using *tagged GSPNs*. In the next sections we shall consider the use of HASL for the specification and computation of passage times, as defined for queueing networks and tagged GSPN.

COSMOS performances						
measure	conf-level	width	num-cores	build-time (s)	runtime (s)	gen-paths
w1	99%	0.01	1	2.91	56.07	6.7 e03
w1			2	2.91	32.86	6.7 e03
w1			4	2.91	25.66	6.7 e03
w1		0.001	1	2.91	5569.42	6.635 e06
w1			2	2.91	3307.99	6.635 e06
w1			4	3.50	2521.05	6.635 e06
w1	95%	0.01	1	2.87	33.37	3.9 e03
w1			2	2.85	20.15	3.9 e03
w1			4	2.84	13.07	3.9 e03
w1		0.001	1	3.21	3262.36	3.842 e06
w1			2	2.82	1917.90	3.842 e06
w1			4	3.33	1365.10	3.842 e06
¡¡¡¡¡¡¡ .mine ======= ¿¿¿¿¿¿¿ .r673						

Table 1. COSMOS runtime in function of confidence-level, interval-width and chosen level of parallelisation.

COSMOS performances. ¡¡¡¡¡¡¡ .mine Table 1 reports about the performances of the COSMOS tool[4]. referring to experiments for assessing measure **w1** (i.e. CDF of passage time for an ordered good to be delivered) of the workflow model. Essentially it show how the simulation-time (i.e. the runtime for sampling trajectories in a quantity

[4] experiments executed on an Apple MacBook Pro, processor Intel dual Core i7 2.8GHz, 8GB 1333 MHZ DDR3 RAM. 256KB L2 cache, 4MB L3 cache.

sufficient to match the required accuracy of estimation) varies in function of the chosen accuracy (i.e. the confidence level and width of the estimated interval) of an experiment. Since Cosmos also allows for the parallelisation of trajectory simulation we also considered the level of parallelisation (expressed in terms of chosen number of cores over which an experiment is distributed) as a parameter of each experiment. Observe that, quite sensibly, the runtime gain is less than linear wrt the level of parallelisation ======= Table 1 reports about the performances of the COSMOS tool[5]. referring to experiments for assessing measure **w1** of the workflow model. Essentially it show how the simulation-time (i.e. the runtime for sampling trajectories in a quantity sufficient to match the required accuracy of estimation) varies in function of the chosen accuracy (i.e. the confidence level and width of the estimated interval) of an experiment. Since Cosmos also allows for the parallelisation of trajectory simulation we also considered the level of parallelisation (expressed in terms of chosen number of cores over which an experiment is distributed) as a parameter of each experiment. Observe that, quite sensibly, the runtime gain is less than linear wrt the level of parallelisation ¿¿¿¿¿¿¿ .r673 and in particular the gain drastically decreases when the chosen number of cores over which the computation is distributed is beyond the actual cores the CPU consists of (in our experiments two)[6]. Finally observe that Cosmos adopts a model-driven code-generation scheme, i.e. a customised C++ instance of HASL simulator is generated for each input (GSPN, HASL-formula) pair and compiled before the actual computation starts. The time to complete this phase, is illustrated in the build-time column of Table 1, whereas the gen-paths column indicates the number of trajectories generated by each experiment.

4 Background: tagged GSPN and Probe Automata

GSPN have been widely used in research and applications to compute classical performance measures, as the mean number of tokens in places or the throughput of transitions. One type of performance index which is not straightforward to define on GSPN models is the distribution of the time required for a (specific) token to pass through a given sub-net , since it requires "token-centric" [6, **?**] view of the system where the tokens represent system entities/customers moving through the net. However, in general, tokens in GSPNs cannot be interpreted as entities which travel throughout the models, but they are indistinguishable quantities consumed and generated by transition firing. Hence, the idea of selecting one token to make it "tagged" and of following it throughout the net is not a trivial task, unless exploiting certain structural properties of the net.

The work in [6] proposes to exploit invariant properties (i.e. p-invariant) to identify places where tokens with indistinguishable behavior are preserved so that such tokens can be treated similarly to the customers of Queuing Network models and can thus be tagged by the modeler to compute the distribution of the time required by one specific token to travel between points of the net. This has led to the introduction of the Tagged

[5] experiments executed on an Apple MacBook Pro, processor Intel dual Core i7 2.8GHz, 8GB 1333 MHZ DDR3 RAM. 256KB L2 cache, 4MB L3 cache.

[6] The runtime for num-cores > 4, not shown in Table 1 for the sake of space, are essentially the same as that for num-cores$= 4$.

Generalized Stochastic Petri Net (TGSPN) formalism which extends classical GSPN with primitives to specify the subnet on which the passage time should be computed. In particular the counterpart of the observed (tagged) customer is the identification of a p-semiflow that leads to a partial unfolding of the subnet identified by the semiflow: transitions and places of the subnet are replicated: each place P_i has a replica P_i^{tag} (same for transitions), and the token in the tagged subnet represents the tagged customer. The condition upon which to start, finish or stop the computation of the passage time are specified by the *Entry*, *Exit* and *Forbid* conditions, identifying respectively the transitions corresponding to the start and stop of passage time count, and those causing the abort of the measurement (i.e. allowing to discard the paths where any forbidden transition fires). Actually the conditions are specified as triplets $\{\langle t, C_{in}, C_{out}\rangle\}$, where C_{in} is the marking condition that has to be satisfied in the tangible marking where t is fired, and C_{out} is the marking condition which has to be satisfied in the tangible marking reached after firing it. Any of the three elements may be omitted. The TGSPN computation of passage time is based on the tangible reachability graph. Therefore, the semantics is slightly more complicated, since arcs in a TRG are labeled with *extended firing sequences*: a timed transition followed by zero or more immediate, leading to a tangible marking. So $Entry = \{\langle t, C_{in}, C_{out}\rangle\}$ means that the computation of the passage time starts when we reach a tangible marking satisfying C_{out} from a tangible marking satisfying C_{in} with an extended firing sequence containing t.

To make this more concrete we introduce another example taken from the literature. This is the model of a Flexible Manufacturing System (FMS), a model that was already used in [6] to explain tagged GSPN, and that we report in Fig. 10 for ease of reference. This FMS comprises four manufacturing stations, from M_1 through M_4, where two of

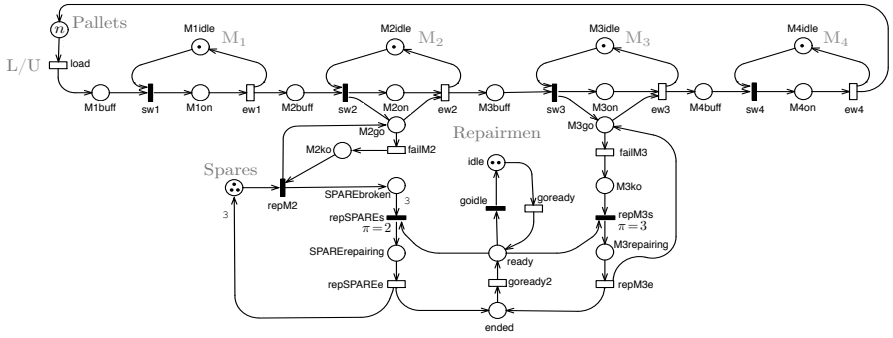


Fig. 10. A FMS with machines breakdown and repair.

them, M_2 and M_3, can fail. Raw parts are loaded on suitable pallets at the Load/Unload (L/U) station represented by the pair – place *Pallets* and single server transition *load* – and are then manufactured, being sequentially brought to the four machines. The model then cycle back (pallet reused on a new raw part). Machine M_2 and M_3 can fail, but

while machine M_3 has no spares, so when it breaks down it has to undergo a reparation by a repairman (initially located in place *REPMANidle*), machine M_2 has a set of spares available in place *Spares*. When all spares have been used, a repairman will intervene to repair all spares, and make them available again. Repairmen are not always available: they cycle between vacation and repair periods. Upon return from a vacation (*goready*) a repairman checks if a machine has failed (*repSPAREs*, *repM*$_3$*s*) and in such case starts a repair activity (*repSPAREe*, *repM*$_3$*e*), otherwise goes back to vacation (*goidle*). After the repairman ends working on a failed machine, he takes a rest (*goready2*) before starting a new cycle. If both M_2 and M_3 require the intervention of a repairman, priority is given to machine M_3.

The modeler could be interested in several passage time measures on such model, for instance the distribution of the waiting time in place M_2ko, that is the time spent by a part in machine M_2 awaiting for the spare parts to be replaced by a repairman. This could be simply expressed as: ($\mathscr{P}_1$) the first passage time from a state where M_2ko becomes marked (due to the firing of transition $failM_2$ when there are no tokens representing spare tools in place *Spares*) to a state where the same place becomes empty (due to the firing of transition $repM_2$). This passage time is specified in TGSPN as $Entry = \{\langle failM_2, Spares = 0, -\rangle\}$ and $Exit = \{\langle repM_2, -, -\rangle\}$. Note that the transition $repM_2$ in the triplet means "an extended firing sequence that contains $repM_2$", which could match, for instance, the firing of the timed transition *repSPAREe* followed by the two immediates $repM_2$ and *repSPAREs*.

However, it has been recognized that the "triplet-based" specification language of TGSPN is not flexible enough to express passage time measures in presence of more elaborate customer behaviour. For example we cannot express the requirement of a passage time distribution of the full cycle in the system (time between two successive firings of transition $load^{tag}$), taking into account only those paths which have experienced at least one "real" breakdown of machine M_2 (a breakdown when no spare is available, which is equivalent to reaching a tangible marking satisfying the condition $M_2ko > 0$), before starting the computation of the passage time. To cope with this limitation Probe Automata (PrA) have been introduced in [3]. PrA uses a path automaton [18], which recognizes paths based on the transitions that fire along the path, enriched with the pre and post conditions on the arcs: an arc labelled t can be taken in marking m only if m satisfies the precondition of the arc, and the state reached through the firing of t satisfies the post condition. A formal definition of PrA can be found in [3], and we only recall

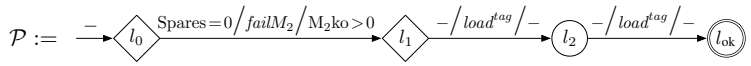


Fig. 11. PrA measuring the tagged token cycle time after a stop for breakdown of machine M_2.

them informally on the example of Fig. 11, that models the complex passage time measure described earlier in words. The PrA in the figure has four locations, and locations can be of two different types: passage locations, where passage time is accumulated, and non-passage location, where time is not accumulated. Non-passage locations (l_0 and l_1)

are drawn as rhombuses and passage (l_2) locations are drawn as circles. As usual, initial locations (only l_0) are identified by an entering edge, while final locations (l_{ok}) have a double border. Finally, edges are labeled with constraints written as $C_{in}/t/C_{out}$.

5 HASL and Probe Automata

When using HASL for the specification of passage time properties of PrA type, there are some peculiarities of PrA that require specific attention. A very general difference is that PrA have been explicitly designed to specify passage time over GSPN subnets, so we cannot expect LHA to be as compact as PrA in representing the same property.

There are indeed two peculiar features of PrA that make them more compact than LHA for passage time specification. The first one is that PrA have implicit loops over locations, accepting all transition firings different from the one present on outgoing arcs. In LHA instead, all events have to be specified, otherwise a path is rejected. The second one is that pre and post conditions over arcs in PrA have no direct counterpart in LHA. An equivalent LHA automaton will have a multiplication of locations to model the same behavior of the PrA, to distinguish locations that do and do not satisfy the conditions. This may lead to a significant increase in the number of locations of the LHA. These differences are not really limitations in the expressiveness, it is only an issue of how easy it is to model a property.

On the other side LHA allow for multiple data variables with flow that can depend on the location. PrA instead has a single, implicit, clock variable, which is equivalent of having a single data variable t with flow of 1 in the counting locations, and 0 otherwise. LHA variables allow to store state informations, that in the PrA would require a multiplication of locations.

Another fundamental difference between PrA and LHA is that PrA are defined over the TRG of the GSPN, while LHA observe the full RG, which includes also vanishing states, so what can be observed with PrA differs in nature from what can be observed by an LHA on the same GSPN. An automatic translation from PrA to LHA would require, for each arc of the PrA labeled with a transition t, the expansion into a subnet that accept all sub-paths from tangible to a tangible marking, passing only through vanishing states and having at least a firing of t along the path.

A difference that requires some caution is that measuring of passage-time through Probe-Automata [3] is based on the idea that a probe may be plugged in (hence becoming operative) at any moment in time of the process described by the considered GSPN, which basically amounts to saying that probes compute passage time assuming that the initial state(s) for the probe have a distribution that is equivalent to the steady state probability. Since HASL does not (inherently) support steady-state measures, then, in order to cope with the *probe plugging in* approach, it is necessary that an LHA for passage-time measures is equipped with a so-called *transient emulator*, i.e. a (unique) initial location whose goal is simply to simulate the GSPN model for a given delay *initT* (assuming that at *initT* the GSPN has reached steady state).

Finally, PrA have been carefully designed to allow a numerical solution, while LHA have instead been designed with simulation in mind. This last difference involves two main aspects: the PrA assumes that the underlying state model is finite and known (this

is necessary if we want to allow for an initial distribution, but especially if we want to use the steady state distribution as the initial distribution, as discussed above) and a PrA has a single timer, so as to allow for a reuse of classical Markov renewal theory results in the computation of the passage time

Using again the FMS model, we consider three PrA passage time specifications $\mathscr{P}_1$ to $\mathscr{P}_3$ (taken from [3]), provide the corresponding LHA (F_1 to F_3) and add a new FMS property expressed through LHA F_4, which cannot be expressed by PrA.

The first example (Figure 12) shows a simple passage time CDF specification that leads to the computation of the distribution of a piece of the cycle (time to load the pallet plus the work time of M_1 and M_2) for the tagged customer. $\mathscr{P}_1$ shows the PrA, while F_1 is the LHA. As expected there are some more arcs, and more annotations over arcs. Note the use of the initial additional location $\underline{l_0}$ to skip the initial transient. The passage location l_1 states that time should be accumulated starting in l_2, which is rendered in the LHA by a reset of the clock t while entering location $\underline{l_2}$.

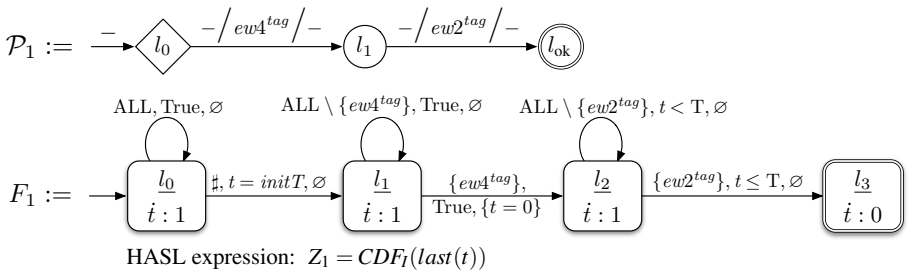


Fig. 12. The Probe Automaton $\mathscr{P}_1$ and the LHA automaton F_1 for the FMS model.

Figure 13 is the cycle time for a tagged token conditioned on at most one breakdown occurring during the cycle (to either the tagged customer itself, or to any other, untagged, customer). The specification of "at most one breakdown" cycle in PrA requires to keep memory of the failures occurred in the past, which leads to the introduction of location l_2 The corresponding LHA appears to be more complex, at least in terms of arc annotations, but it is actually more general: the number of breakdowns is accumulated in the discrete variable n, and paths are accepted if there have been at most N breakdowns.

Property $\mathscr{P}_3$ of Figure 14 shows an example of PrA arcs with pre and post conditions. By requiring that $failM_2$ is pre conditioned on place *Spares* being equal to zero and post conditioned on M_2ko being greater than zero, we are catching the first breakdown of M_2 that finds no spare parts available. In the LHA this is translated expanding the arc from l_0 to l_1 of the PrA into the three locations $\underline{l_1}, \underline{l_2}, \underline{l_3}$ of the LHA.

Finally, Figure 15 shows an FMS property that is not present in [3]. It is an example in which the acceptance of a passage time (in this case the time to go through machines M_1, M_3, and M_3) is conditioned on having a mean number of spare parts under repair (modeled by the sum of tokens in places *SPARESbroken* and *SPARESrepairing*), all along the path, less than a constant parameter *WBS*. This is an interesting case in which

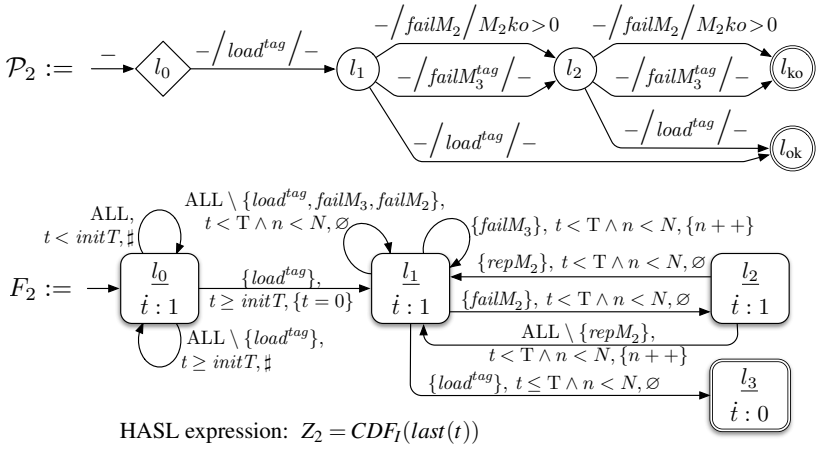


Fig. 13. The Probe Automaton $\mathscr{P}_2$ and the LHA automaton F_2 for the FMS model.

the acceptance of a path is conditioned on a performance measure relative to the path itself.

Figure 16 shows the CDF of passage time for F_4 (Figure 15), comparing the unconditioned passage-time versus the WBS conditioned passage-time (considering two different values of the conditioning bound *WBS*), i.e. the accepted paths conditioned by the fact that the average number of spare parts that are broken or under reparation is less than the specified value (i.e. WBS). Note that, for this plot, we have instructed Cosmos to report "unnormalized", defective, passage time CDF.

All the measures shown in this section refer to the initial marking of 9 pallets in the load/unload station, that can be compared with the results of [3]. Note that the state space of the FMS model may be quite large as the number of pallets grows (for instance, with 20 pallets it has more than 3 million states, or for 300 pallets reaches 150 billion states). With such large state spaces, the simulation approach becomes virtually the only applicable method.

6 Conclusion and future work

In this paper we have considered the problem of defining and computing complex passage time distributions on (tagged) GSPN models, extended with general firing delays, by means of the HASL language. In particular the measure of interest has to be computed on a subset of paths satisfying some requirement on the fired transitions and on the properties of the traversed states. In addition it may be useful to put constraints on the duration of some activity carried on along the path, or on quantitative properties of the path as a whole. Work is ongoing on an extension of Cosmos for *Symmetric Stochastic Nets* (also known as *colored Petri nets*).

Comparing the HASL expressive power with that of PrA and of TGSPN passage time measure definitions it has been shown on an example from the literature that HASL

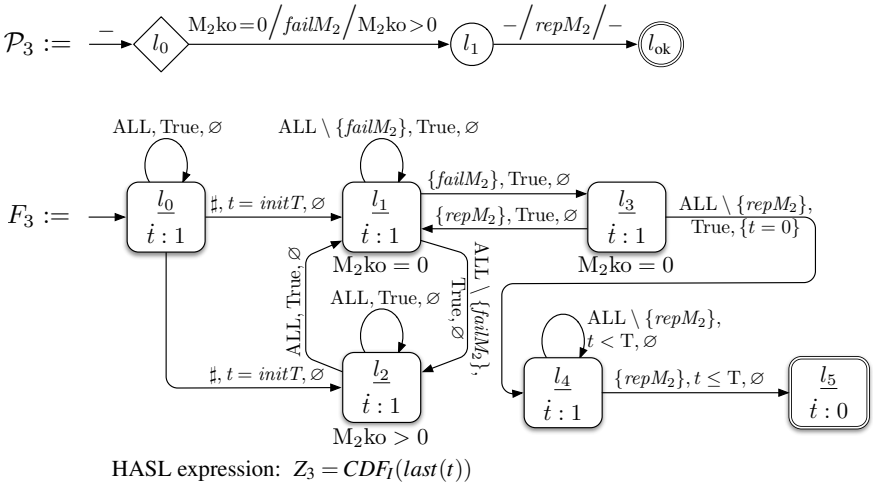


Fig. 14. The Probe Automaton $\mathscr{P}_3$ and the LHA automaton F_3 for the FMS model.

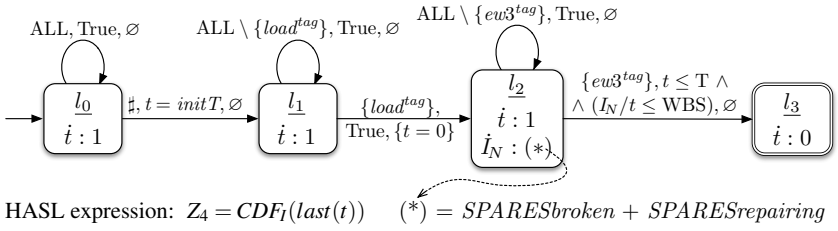


Fig. 15. The LHA automaton for the FMS property F_4.

allows one to define more complex performance measures. In some cases the LHA structure is more complex with respect to the corresponding PrA, since the latter has been specifically designed with passage time specifications in mind.

Finally, the experimental results performed on two different models (i.e. FMS and Business workflow) have shown that the COSMOS statistical model checker is an adequate tool to estimate such measures, even in presence of non exponential transition firing times and on models with very large state spaces.

References

1. Aalst, W.: Business process management demystified: A tutorial on models, systems and standards for workflow management. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets, Lecture Notes in Computer Science, vol. 3098, pp. 1–65. Springer Berlin Heidelberg (2004), `http://dx.doi.org/10.1007/978-3-540-27755-2_1`
2. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid Systems. LNCS 736 (1992)

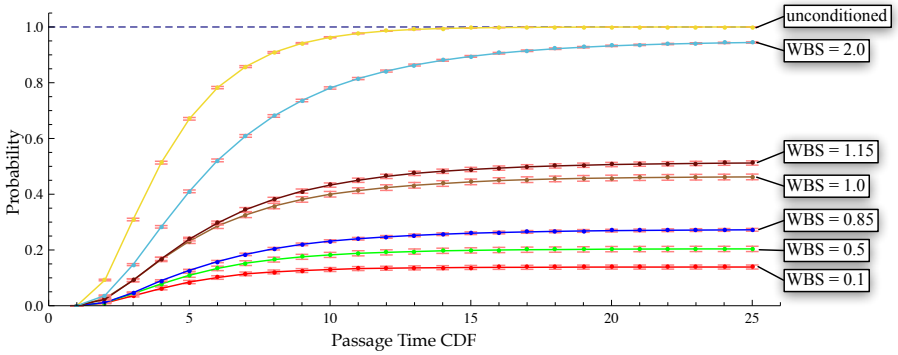


Fig. 16. Passage time property F_4 of the FMS model.

3. Amparore, E., Beccuti, M., Donatelli, S., Franceschinis, G.: Probe automata for passage time specification. In: Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on. pp. 101 –110 (sept 2011)
4. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continuous-time Markov chains. ACM Trans. Comput. Logic 1(1), 162–170 (2000)
5. Baier, C., Cloth, L., Haverkort, B.R., Kuntz, M., Siegle, M.: Model Checking Markov Chains with Actions and State Labels. IEEE Transactions on Software Engineering 33, 209–224 (2007)
6. Balbo, G., Beccuti, M., De Pierro, M., Franceschinis, G.: First Passage Time Computation in Tagged GSPNs with Queue Places. The Computer Journal (2010), first published online July 22, 2010.
7. Balbo, G., De Pierro, M., Franceschinis, G.: Tagged Generalized Stochastic Petri Nets. In: Computer Performance Engineering. vol. LNCS 5652, pp. 1–15. Springer-Verlag, Berlin (2009)
8. Ballarini, P., Djafri, H., Duflot, M., Haddad, S., Pekergin, N.: COSMOS: a statistical model checker for the hybrid automata stochastic logic. In: Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11). pp. 143–144. IEEE Computer Society Press (sep 2011)
9. Ballarini, P., Djafri, H., Duflot, M., Haddad, S., Pekergin, N.: HASL: an expressive language for statistical verification of stochastic models. In: Proc. Valuetools (2011)
10. Chen, T., Han, T., Katoen, J.P., Mereacre, A.: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications. Logic in Computer Science, Symposium on 0, 309–318 (2009)
11. Clark, A., Gilmore, S.: State-aware performance analysis with extended stochastic probes. In: Proceedings of the 5th European Performance Engineering Workshop on Computer Performance Engineering. pp. 125–140. EPEW '08, Springer-Verlag, Berlin, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-87412-6_10
12. Dingle, N.J., Harrison, P.G., Knottenbelt, W.J.: Uniformisation and Hypergraph Partitioning for the Distributed Computation of Response Time Densities in Very Large Markov Models. Journal of Parallel and Distributed Computing 64(8), 309–920 (August 2004)
13. Djafri, H.: Numerical and Statistical Approaches for Model Checking of Stochastic Processes. Ph.D. thesis, ENS Cachan (June 2012)
14. Donatelli, S., Haddad, S., Sproston, J.: Model checking timed and stochastic properties with CSL^{TA}. IEEE Trans. Softw. Eng. 35(2), 224–240 (2009)

15. Haverkort, B., Cloth, L., Hermanns, H., Katoen, J.P., Baier, C.: Model checking performability properties. In: Proc. DSN'02 (2002)
16. Hillston, J.: Process algebras for quantitative analysis. In: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science. pp. 239–248. IEEE Computer Society, Washington, DC, USA (2005), http://portal.acm.org/citation.cfm?id=1078035.1079698
17. Kulkarni, V.: Modeling and Analysis of Stochastic Systems. Chapman & Hall, London, UK (1995)
18. Obal, II, W.D., Sanders, W.H.: State-space support for path-based reward variables. Perform. Eval. 35, 233–251 (May 1999), http://dx.doi.org/10.1016/S0166-5316(99)00010-3