Markov Decision Petri Net and Markov Decision Well-formed Net formalisms

M. Beccuti, G. Franceschinis[1⋆] and S. Haddad[2⋆⋆]

[1] Univ. del Piemonte Orientale,
giuliana.franceschinis@mfn.unipmn.it
[2] LAMSADE CNRS, Univ. Paris Dauphine
haddad@lamsade.dauphine.fr

Abstract. In this work, we propose two high-level formalisms, *Markov Decision Petri Nets* (MDPNs) and *Markov Decision Well-formed Nets* (MDWNs), useful for the modeling and analysis of distributed systems with probabilistic and non deterministic features: these formalisms allow a high level representation of Markov Decision Processes. The main advantages of both formalisms are: a macroscopic point of view of the alternation between the probabilistic and the non deterministic behaviour of the system and a syntactical way to define the switch between the two behaviours. Furthermore, MDWNs enable the modeller to specify in a concise way similar components. We have also adapted the technique of the symbolic reachability graph, originally designed for Well-formed Nets, producing a reduced Markov decision process w.r.t. the original one, on which the analysis may be performed more efficiently. Our new formalisms and analysis methods are already implemented and partially integrated in the GreatSPN tool, so we also describe some experimental results.

1 Introduction

Markov Decision Processes (MDP). Since their introduction in the 50's, Markov Decision process models have gained recognition in numerous fields including computer science and telecommunications [13]. Their interest relies on two complementary features. On the one hand, they provide to the modeler a simple mathematical model in order to express optimization problems in random environments. On the other hand, a rich theory has been developed leading to efficient algorithms for most of the practical problems.

Distributed Systems and MDPs. The analysis of distributed systems mainly consists in (1) a modeling phase with some high-level formalism like Petri nets (PN) or process algebra, (2) the verification of properties expressed in some logic (like LTL or CTL) and (3) the computation of performance indices by enlarging

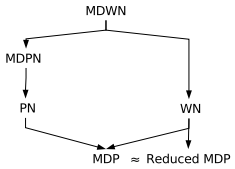
⋆ The work of these authors was supported in part with the Italian MUR "local" research funds.
⋆⋆ The work of this author was supported in part by ANR Setin project CheckBound.

the model with stochastic features and applying either (exact or approximate) analysis methods or simulations. In this framework, a MDP may be viewed as a model of a distributed system where it is possible to perform a non deterministic choice among the enabled actions (*e.g.*, the scheduling of tasks) while the effect of the selected action is probabilistic (*e.g.*, the random duration of a task). Then, with appropriate techniques, one computes the probability that a property is satisfied w.r.t. the "worst" or the "best" behavior [3,7]. The time model that will be considered in this paper is discrete: each non deterministic choice is taken in a given decision epoch, after the probabilistic consequence of the choice has been performed, a new decision epoch starts.

Here the way we model distributed systems by MDPs is rather different. During a phase, the system evolves in a probabilistic manner until periodically a (human or automatic) supervisor takes the control in order to configure, adapt or repair the system depending on its current state before the next phase. In other words, usual approaches consider that the alternation between non deterministic and probabilistic behavior occurs at a microscopic view (*i.e.*, at the transition level) whereas our approach adopts a macroscopic view of this alternation (*i.e.*, at a phase level). It should be emphasized that, depending on the applications, one or the other point of view should be preferred and that the user should have an appropriate formalism and associated tools for both cases. For instance PRISM [11], one of the most used tools in this context, works at the microscopic level whereas the formalism of *stochastic transition systems* is based on a macroscopic view [8]. The latter formalism is a slight semantical variation of generalized stochastic Petri nets [12] where the choice among the enabled immediate transitions is performed non deterministically rather than probabilistically. Despite its simplicity, this formalism has a serious drawback for the design process since the modeler has no mean to syntactically define the switches between the probabilistic behavior and the non deterministic one. Furthermore, the difference between the *distributed* feature of the probabilistic behavior and the *centralized* one of the non deterministic behavior is not taken into account.

Our contribution. In this work, we propose a high-level formalism in order to model distributed systems with non deterministic and probabilistic features. Our formalism is based on *Well-formed Petri Nets* (WN) [4]. First, we introduce *Markov Decision Petri nets* (MDPN): an MDPN is defined by three parts, a set of active components (*e.g.*, processes or machines), a probabilistic net and a non deterministic net. Every transition of the probabilistic net is triggered by a subset of components. When every component has achieved the activities related to the current probabilistic phase, the supervisor triggers the non deterministic transitions in order to take some decisions, either relative to a component or global. Every transition has an attribute (run/stop) which enables the modeler to define when the switches between the nets happen. The semantics of this model is designed in two steps: a single Petri net can be derived from the specification and its reachability graph can be transformed with some additional information, also specified at the MDPN level, into an MDP.

Distributed systems often present symmetries *i.e*, in our framework, many components may have a similar behavior. Thus, both from a modeling and an analysis point of view, it is interesting to look for a formalism expressing and exploiting behavioral symmetries. So we also define *Markov Decision Well-formed nets* (MDWN) similarly as we do for MDPNs. The semantics of a model is then easily obtained by translating a MDWN into a MDPN. Furthermore, we develop an alternative approach: we transform the MDWN into a WN, then we build the symbolic reachability graph of this net [5] and finally we transform this graph into a reduced MDP w.r.t. the original one. We argue that we can compute on this reduced MDP, the results that we are looking for in the original MDP. The different relations between the formalisms are shown in the figure depicted below. Finally we have implemented our analysis method within the GreatSPN tool [6] and performed some experiments.



Organization of the paper. In section 2, we recall basic notions relative to MDPs, then we define and illustrate MDPNs. In section 3, we introduce MDWNs and develop the corresponding theoretical results. In section 4, we present some experimental results. In section 5, we discuss related work. Finally we conclude and give some perspectives in section 6.

2 Markov Decision Petri Net

2.1 Markov Decision Process

A (discrete time and finite) MDP is a dynamic system where the transition between states (*i.e.*, items of S a finite set) are obtained as follows. First, given s the current state, one non deterministically selects an action among the subset of actions currently enabled (*i.e.*, A_s). Then one samples the new state w.r.t. to a probability distribution depending on s and $a \in A_s$ (*i.e.*, $p(\cdot|s,a)$). An MDP includes rewards associated with state transitions; here, we choose a slightly restricted version of the rewards that do not depend on the destination state (*i.e.*, $r(s,a)$). Starting from such elementary rewards, different kinds of global rewards may be associated with a finite or infinite execution thus raising the problem to find an optimal strategy w.r.t. a global reward. For sake of simplicity, we restrict the global rewards to be either the *expected total reward* or the *average reward*. The next definitions formalize these concepts.

Definition 1 (MDP). *An MDP $\mathcal{M}$ is a tuple $\mathcal{M} = \langle S, A, p, r\rangle$ where:*

- *S is a finite set of states,*

- *A is a finite set of actions defined as $\bigcup_{s\in S} A_s$ where A_s is the set of enabled actions in state s,*
- *$\forall s \in S, \forall a \in A_s, p(\cdot|s,a)$ is a (transition) probability distribution over S such that $p(s'|s,a)$ is the probability to reach s' from s by triggering action a,*
- *$\forall s \in S, \forall a \in A_s, r(s,a) \in \mathbb{R}$ is the reward associated with state s and action a.*

A finite (resp. infinite) execution of an MDP is a finite (resp. infinite) sequence $\sigma = s_0 a_0 \dots s_n$ (resp. $\sigma = s_0 a_0 \dots$) of alternating states and actions, s.t. $\forall i, s_i \in S \wedge a_i \in A_{s_i}$ and $p(s_{i+1}|s_i,a_i) > 0$.

The total reward of such an execution is defined by $trw(\sigma) = \sum_{i=0}^{n-1} r(s_i,a_i)$ (resp. $trw(\sigma) = \lim_{n\to\infty} \sum_{i=0}^{n-1} r(s_i,a_i)$ provided the limit exists) and its average reward is $arw(\sigma) = \frac{1}{n}\sum_{i=0}^{n-1} r(s_i,a_i)$ (resp. $arw(\sigma) = \lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} r(s_i,a_i)$ provided the limit exists).

We denote $SEQ^*(\mathcal{M})$ (resp. $SEQ^\infty(\mathcal{M})$) the set of finite (resp. infinite) sequences. A strategy st is a mapping from $SEQ^*(\mathcal{M})$ to A such that $st(s_0a_0\dots s_n)$ belongs to A_{s_n}. Since a strategy discards non determinism, the behavior of $\mathcal{M}$ w.r.t. st is a stochastic process $\mathcal{M}^{st}$ defined as follows. Assume that the current execution is some $s_0a_0\dots s_n$ then $a_n = st(s_0a_0\dots s_n)$ and the next state s_{n+1} is randomly chosen w.r.t. distribution $p(\cdot|s_n,a_n)$. Consequently, the reward of a random sequence of $\mathcal{M}^{st}$ is a random variable and the main problem in the MDP framework is to maximize or minimize the mean of this random variable and to compute the associated strategy when it exists. In finite MDPs, efficient solution techniques have been developed to this purpose [13].

Here we want to model systems composed by multiple active components whose behavior during a period is described in a probabilistic way and a centralized decision maker taking some decisions between execution periods (*e.g.*, assigning available resources to components). Let us illustrate this kind of systems by a toy example. Imagine an information system based on two redundant computers: this system is available as long as one computer is in service. A computer may fail during a period. At the end of a period, the decision maker can choose to send a single repairman to repair a faulty computer when he is not yet busy. There is a fixed probability that the repairing ends inside the period. In this framework, the rewards denote costs (for unavailability and repairs) and the analysis aims at minimizing them. The MDP corresponding to this system is shown in Fig. 1, where the states description does not maintain the distinction between the components; this is only possible when the computers are identical.

The design of this MDP is rather easy. However when the system has more computers and repairmen with different behaviors, then modeling it at the MDP level becomes unfeasible.

2.2 Markov Decision Petri Net

A Markov Decision Petri Net $\mathcal{MN}$ is composed by two different parts (i.e. two extended Petri nets): the probabilistic one N^{pr} and the non deterministic one

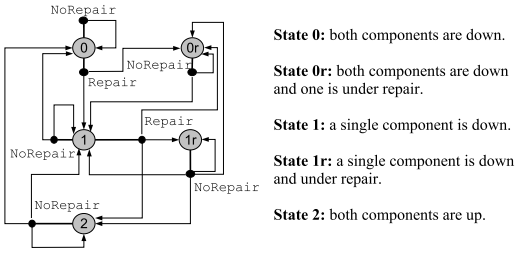


Fig. 1. Symbolic representation of the MDP modeling in this section

N^{nd} called the *decision maker*; it is thus possible to clearly distinguish and design the probabilistic behavior of the system and the non deterministic one. The probabilistic part models the probabilistic behavior of the system and can be seen as composition of a set of n components ($Comp^{pr}$) that can interact; instead the non deterministic part models the non deterministic behavior of the system where the decisions must be taken (we shall call this part *the decision maker*). Hence the global system behavior can be described as an alternating sequence of probabilistic and non deterministic phases.

The probabilistic behavior of a component is characterized by two different types of transitions $Trun^{pr}$ and $Tstop^{pr}$. The $Trun^{pr}$ transitions represent intermediate steps in a probabilistic behavior phase and can involve several components (synchronized through that transition), while the $Tstop^{pr}$ ones always represent the final step of the probabilistic phase of at least one component.

In the non deterministic part, the decisions can be defined at the system level (transitions of T_g^{nd}) or at the component level (transitions of T_l^{nd}). The sets T_g^{nd} and T_l^{nd} are again partitioned in $Trun_g^{nd}$ and $Tstop_g^{nd}$, and $Trun_l^{nd}$ and $Tstop_l^{nd}$ with the same meaning. The decision maker does not necessarily control every component and may not take global decisions. Thus the set of controllable "components" $Comp^{nd}$ is a subset of $Comp^{pr} \uplus \{id_s\}$ where id_s denotes the whole system.

The probabilistic net is enlarged with a mapping *weight* associating a weight with every transition in order to compute the probabilistic choice between transitions enabled in a marking. Furthermore it includes a mapping *act* which associates to every transition the subset of components that (synchronously) trigger the transition. The non deterministic net is enlarged with a mapping *obj* which associates with every transition the component which is involved by the transition. The following definition summarizes and formalizes this presentation.

Definition 2 (Markov Decision Petri Net (MDPN)). *A Markov Decision Petri Net (MDPN) is a tuple* $\mathcal{MN} = \langle Comp^{pr}, Comp^{nd}, N^{pr}, N^{nd} \rangle$ *where:*

- $Comp^{pr}$ *is a finite non empty set of components;*
- $Comp^{nd} \subseteq Comp^{pr} \uplus \{id_s\}$ *is the non empty set of controllable components;*

- *N^{pr} is defined by a PN with priorities [12] $\langle P, T^{pr}, I^{pr}, O^{pr}, H^{pr}, prio^{pr}, m_0 \rangle$, a mapping weight: $T^{pr} \to \mathbb{R}$ and a mapping act: $T^{pr} \to 2^{Comp^{pr}}$. Moreover $T^{pr} = Trun^{pr} \uplus Tstop^{pr}$*
- *N^{nd} is defined by a PN with priorities $\langle P, T^{nd}, I^{nd}, O^{nd}, H^{nd}, prio^{nd}, m_0 \rangle$ and a mapping obj: $T^{nd} \to Comp^{nd}$. Moreover $T^{nd} = Trun^{nd} \uplus Tstop^{nd}$.*

Furthermore, the following constraints must be fulfilled:

- *$T^{pr} \cap T^{nd} = \emptyset$. A transition cannot be non deterministic and probabilistic.*
- *$\forall id \in Comp^{pr}, \exists C \subseteq Comp^{pr}$, s.t. $id \in C$ and $act^{-1}(\{C\}) \cap Tstop^{pr} \neq \emptyset$. Every component must trigger at least one* final *probabilistic transition.*
- *$\forall id \in Comp^{nd}, obj^{-1}(\{id\}) \cap Tstop^{nd} \neq \emptyset$. Every controllable component must be the object of at least one* final *non deterministic transition.*

Note that the probabilistic part and the decision maker share the same set of places and the same initial marking. Let us now introduce the rewards associated with the MDPN net. As will be developed later, an action of the decision maker corresponds to a sequence of transition firings starting from some marking. We choose to specify a reward by first associating with every marking m a reward $rs(m)$, with every transition t a reward $rt(t)$ and then by combining them with an additional function rg (whose first parameter is a state reward and the second one is a reward associated with a sequence of transition firings). The requirement on its behavior given in the next definition will be explained when presenting the semantics of a MDPN.

Definition 3 (MDPN reward functions). *Let $\mathcal{MN}$ be a MDPN. Then its reward specification is given by:*

- *$rs : \mathbb{N}^P \to \mathbb{R}$ which defines for every marking its reward value.*
- *$rt : T^{nd} \to \mathbb{R}$ which defines for every transition its reward value.*
- *$rg : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, not decreasing w.r.t its second parameter.*

An example of (a portion of) probabilistic and non deterministic subnets is shown in Fig. 3: in the framework of the MDPN formalism, the annotations on arcs and next to the places and transitions should be ignored. The decision maker implements the possible ways of assigning resources (e.g. for component repair) to those components that need them (e.g. failed components): for each component needing a resource two possibilities are included, namely assign or not assign resource. The probabilistic part shows 3 system components: two of them are controllable (let's call them Proc and Mem), one is not controllable (let's call it ResCtr). The Proc and Mem components can work fine or fail, the third one supervises the repair process (when the resource is available) and the Proc and/or Mem resume phase. $Tstop$ transitions are e.g. *WorkFineProc*, *WaitRepProc*, *ResumeProc*, *ResumeMemProc*, the first two involving only Proc, the third involving Proc and ResCtr, the last one involving all three components; the firing of these transitions mean that the involved components have reached a stable state in the current decision epoch. $Trun$ transitions are e.g. *FailProc*, *FailMem* involving respectively Proc and Mem, and *EndRep* involving

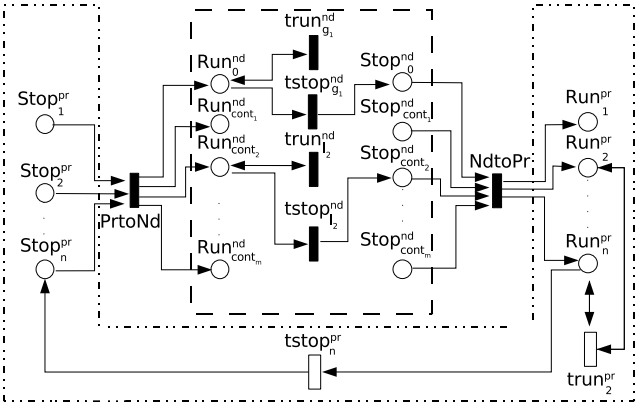


Fig. 2. Arcs connecting the places $Stop_i^{pr}$, Run_i^{nd} and the transition $PrtoNd$; arcs connecting the places $Stop_i^{nd}$, Run_i^{nd} and the transition $NdtoPr$

only ResCtr. These transitions represent intermediate steps in the components evolution (e.g. a failure can be followed by a wait for repair resource or a resume final step).

MDPN semantics. The MDPN semantics is given in three steps. First, one composes the probabilistic part and the decision maker in order to derive a unique PN. Then one generates the (finite) reachability graph (RG) of the PN. At last, one produces an MDP from it.

From MDPN to PN. First we explain the semantics of additional places $Stop_i^{pr}$, Run_i^{pr}, $Stop_i^{nd}$, Run_i^{nd}, $Stop_0^{nd}$ and $Run_0^{nd}$and additional non deterministic transitions $PrtoNd$ and $NdtoPr$. Places $Stop_i^{pr}$, Run_i^{pr}, $Stop_i^{nd}$, Run_i^{nd}, $Stop_0^{nd}$ and Run_0^{nd} regulate the interaction among the components, the global system and the decision maker. There are places Run_i^{pr}, $Stop_i^{pr}$ for every component i, while we insert the places Run_0^{nd} and $Stop_0^{nd}$ if the decision maker takes same global decision and the pair of places Run_i^{nd} and $Stop_i^{nd}$ for every controllable component $i \in Comp^{nd}$. Non deterministic transitions $PrtoNd$ and $NdtoPr$ ensure that the decision maker takes a decision for every component in every time unit: the former triggers a non deterministic phase when all the components have finished their probabilistic phase whereas the latter triggers a probabilistic phase when the decision maker has taken final decisions for every controllable component.

The scheme describing how these additional items are connected together and with the nets of the MDPN is shown in Fig. 2. The whole PN $N^{comp} = \langle P^{comp}, T^{comp}, I^{comp}, O^{comp}, H^{comp}, prio^{comp}, m_0^{comp} \rangle$ related to a MDPN $\mathcal{MN}$ is defined below.

– $P^{comp} = P \uplus_{i \in Comp_{pr}} \{Run_i^{pr}, Stop_i^{pr}\} \uplus_{i \in Comp_{nd}} \{Run_i^{nd}, Stop_i^{nd}\}$

- $T^{comp} = T^{pr} \uplus T^{nd} \uplus \{PrtoNd, NdtoPr\}$
- The incidence matrices of N^{comp} are defined by:
  - $\forall p \in P, t \in T^{nd}$,
    $I^{comp}(p,t) = I^{nd}(p,t), O^{comp}(p,t) = O^{nd}(p,t), H^{comp}(p,t) = H^{nd}(p,t)$
  - $\forall p \in P, t \in T^{pr}$,
    $I^{comp}(p,t) = I^{pr}(p,t), O^{comp}(p,t) = O^{pr}(p,t), H^{comp}(p,t) = H^{pr}(p,t)$
  - $\forall t \in Tstop^{pr}$ *s.t.* $i \in act(t) : I^{comp}(Run_i^{pr}, t) = O^{comp}(Stop_i^{pr}, t) = 1$
  - $\forall t \in Trun^{pr}$ *s.t.* $i \in act(t) : I^{comp}(Run_i^{pr}, t) = O^{comp}(Run_i^{pr}, t) = 1$
  - $\forall t \in Tstop^{nd}$ *s.t.* $i \in act(t) : I^{comp}(Run_i^{nd}, t) = O^{comp}(Stop_i^{nd}, t) = 1$
  - $\forall t \in Trun^{nd}$ *s.t.* $i \in act(t) : I^{comp}(Run_i^{nd}, t) = O^{comp}(Run_i^{nd}, t) = 1$
  - $\forall i \in Comp_{pr} : I^{comp}(Stop_i^{pr}, PrtoNd) = O^{comp}(Run_i^{pr}, NdtoPr) = 1$
  - $\forall i \in Comp_{nd} : I^{comp}(Stop_i^{nd}, NdtoPr,) = O^{comp}(Run_i^{nd}, PrtoNd) = 1$
  - for all $I(p,t), O(p,t), H(p,t)$ not previously defined,
    $I^{comp}(p,t) = O^{comp}(p,t) = 0, H^{comp}(p,t) = \infty$;
- $\forall t \in T^{nd}, prio(t) = prio^{nd}(t)$, $\forall t \in T^{pr}, prio(t) = prio^{pr}(t)$,
  $prio(PrtoNd) = prio(NdtoPr) = 1$, (actually these values are irrelevant)
- $\forall p \in P, m_0^{Comp}(p) = m_0(p)$, $m_0^{Comp}(Run_i^{nd}) = 1$,
  $m_0^{Comp}(Stop_i^{nd}) = m_0^{Comp}(Run_i^{pr}) = m_0^{Comp}(Stop_i^{pr}) = 0$.

RG semantics and transitions sequence reward. Considering the RG obtained from the PN we observe that the reachability set (RS) can be partitioned into two subsets: the non deterministic states (RS_{nd}), in which only non deterministic transitions are enabled, and the probabilistic states (RS_{pr}), in which only probabilistic transitions are enabled. By construction, the PN obtained from a MDPN can never reach a state enabling both nondeterministic and probabilistic transitions. A probabilistic transition can be enabled only if there is at least one place Run_i^{pr} with $m(Run_i^{pr}) > 0$, while a non deterministic transition can be enabled only if there is at least one place Run_i^{nd} with $m(Run_i^{nd}) > 0$. Initially only Run_i^{nd} places are marked. Then only when all the tokens in the Run_i^{nd} places have moved to the $Stop_i^{nd}$ places (through the firing of some transition in $Tstop^{nd}$), the transition $NdtoPr$ can fire, removing all tokens from the $Stop_i^{nd}$ places and putting one token in every Run_i^{pr} place. Similarly, transition $PrtoNd$ is enabled only when all tokens have moved from the Run_i^{pr} to the $Stop_i^{pr}$ places; the firing of $PrtoNd$ brings the tokens back in each Run_i^{nd} place. Thus places Run_i^{pr} and places Run_i^{nd} cannot be simultaneously marked.

Observe that any *path* in the RG can be partitioned into (maximal) sub-paths leaving only states of the same type, so that each path can be described as an alternating sequence of non deterministic and probabilistic sub-paths. Each probabilistic sub-path can be substituted by a single "complex" probabilistic step and assigned a probability based on the weights of the transitions firing along the path. The non deterministic sub-paths can be interpreted according to different semantics (see [2] for a detailed discussion). Here we select the following semantics: a path through non deterministic states is considered as a single complex action and the only state where time is spent is the first one in the sequence (that is the state that triggers the "complex" decision multi-step). So only the first state in each path will appear as a state in the MDP (the other

states in the path are *vanishing*, borrowing the terminology from the literature on generalized stochastic Petri nets).

Let us now define the reward function for a sequence of non deterministic transitions, $\sigma \in (T^{nd})^*$; abusing notation we use the same name $rt()$ for the reward function for single transitions and for transition sequences. The following definition $rt(\sigma)$ assumes that the firing order in such a sequence is irrelevant w.r.t. the reward which is consistent with an additive interpretation when several decisions are taken in one step.

Definition 4 (Transition sequence reward $rt(\sigma)$). *The reward for a non deterministic transition sequence is defined as follows:*

$$rt(\sigma) = \sum_{t \in T^{nd}} rt(t)|\sigma|_t$$

where $|\sigma|_t$ is the number of occurrences of non deterministic transition t in σ.

Generation of an MDP given a RG of a MDPN and the reward structure. The MDP can be obtained from the RG of the PN model in two steps: (1) build from the RG the RG_{nd} such that given any non deterministic state nd and any probabilistic state pr all maximal non deterministic sub-paths from nd to pr are reduced to a single non deterministic step; (2) build the RG_{MDP} (*i.e.*, a MDP) from the RG_{nd} such that given any non deterministic state nd and any probabilistic state pr, all maximal probabilistic sub-paths from pr to nd are substituted by a single probabilistic step. Finally derive the MDP reword from rs,rt and rg functions.

Let nd be a non deterministic state reached by a probabilistic transition (such states will be the non deterministic states of RG_{nd}). We focus on the subgraph "rooted" in nd and obtained by the maximal non deterministic paths starting from nd. Note that the probabilistic states occurring in this subgraph are terminal states. If there is no finite maximal non deterministic sub-paths starting from nd then no probabilistic phase can follow. So the construction is aborted. Otherwise, given every probabilistic state pr of the subgraph, one wants to obtain the optimal path $\sigma_{nd,pr}$ from nd to pr w.r.t. the reward. Once for every such pr, this path is computed, in RG_{nd} an arc is added from nd to pr labeled by $\sigma_{nd,pr}$. The arcs starting from probabilistic states are unchanged in RG_{nd}.

Thus the building of RG_{nd} depends on whether the optimization problem is a maximization or a minimization of the reward. We only explain the minimization case (the other case is similarly handled). We compute such a sequence using the Bellman and Ford (BF) algorithm for a single-source shortest paths in a weighted digraph where the transition reward is the cost function associated with the arcs. This algorithm is sound due to our (cumulative) definition for rewards of transition sequences. Note that if the BF algorithm finds a negative loop (*i.e.*, where the reward function decreases), the translation is aborted. Indeed the optimal value is then $-\infty$ and there is no optimal sequence: this problem must be solved at the design level.

We now explain how to transform RG_{nd} into the MDP RG_{MDP}. Given a probabilistic state pr and a non deterministic state nd we want to compute the probability to reach nd along probabilistic sub-paths. Furthermore, the sum of

these transition probabilities over non deterministic states must be 1. So if in RG_{nd}, there is a terminal strongly connected component composed by only probabilistic states, we abort the construction. The checked condition is necessary and sufficient according to Markov chain theory. Otherwise, we obtain the transition probabilities using two auxiliary matrices. $\mathbf{P}^{(pr,pr)}$, a square matrix indexed by the probabilistic states, denotes the one-step probability transitions between these states and $\mathbf{P}^{(pr,nd)}$, a matrix whose rows are indexed by the probabilistic states and columns are indexed by non deterministic states, denotes the one-step probability transitions from probabilistic states to non deterministic ones. Let us describe how these transition probabilities are obtained. These probabilities are obtained by normalizing the weights of the transitions enabled in pr. Now again, according to Markov chain theory, matrix $\mathbf{P} = (\mathbf{Id} - \mathbf{P}^{(pr,pr)})^{-1} \circ \mathbf{P}^{(pr,nd)}$, where $\mathbf{Id}$ is the identity matrix represents the searched probabilities. A similar transformation is performed in the framework of stochastic Petri nets with immediate transitions (see [12] for the details).

Finally in the MDP, the probability distribution $p(\cdot|nd, \sigma)$ associated with state nd and (complex) action σ, assuming $nd \xrightarrow{\sigma} pr$, is given by the row vector $\mathbf{P}[pr, \cdot]$ and the reward function for every pair of state and action is defined by the following formula: $r(nd, \sigma) = rg(rs(nd), rt(\sigma))$. Since rg is not decreasing w.r.t. its second parameter, the optimal path w.r.t. rt found applying the Bellman and Ford algorithm is also optimal w.r.t. $rg(rs(nd), rt(\cdot))$.

Discussion The MDPN is a high-level formalism for specifying MDPs. However this formalism suffers a drawback: by definition, the components are identified and always distinguished in the state representation, even if they have similar behavior (*i.e.*, even if one component is an exact copy of another component). This can have an impact both at the level of the model description (which could become difficult to read when several components are present), and at the level of the state space size. In the next section, we cope with these problems by introducing a higher-level formalism.

3 Markov Decision Well-formed Net

3.1 WN informal introduction

WNs are an high-level Petri net formalism whose syntax has been the starting point of numerous efficient analysis methods. Below, we describe the main features of WNs. The reader can refer to [4] for a formal definition.

In a WN (and more generally in high-level nets) a color domain is associated with places and transitions. The colors of a place label the tokens contained in this place, whereas the colors of a transition define different ways of firing it. In order to specify these firings, a color function is attached to every arc which, given a color of the transition connected to the arc, determines the number of colored tokens that will be added to or removed from the corresponding place. The initial marking is defined by a multi-set of colored tokens in each place.

A color domain is a Cartesian product of color classes which may be viewed as primitive domains. Classes can have an associated (circular) order expressed by means of a successor function. The Cartesian product defining a color domain is possibly empty (e.g., for a place which contains neutral tokens) and may include repetitions (e.g., a transition which synchronizes two colors inside a class). A class can be divided into static subclasses. The colors of a class have the same nature (processes, resources, etc.), whereas the colors inside a static subclass have the same potential behavior (batch processes, interactive processes, etc.).

A color function is built by standard operations (linear combination, composition, etc.) on basic functions. There are three basic functions: a projection which selects an item of a tuple and is denoted by a typed variable (e.g., p, q); a synchronization/diffusion that is a constant function which returns the multiset composed by all the colors of a class or a subclass and is denoted S_{C_i} ($S_{C_{i,k}}$) where C_i ($C_{i,k}$) is the corresponding (sub)class; and a successor function which applies on an *ordered* class and returns the color following a given color.

Transitions and color functions can be guarded by expressions. An expression is a boolean combination of atomic predicates. An atomic predicate either identifies two variables $[p = q]$ or restricts the domain of a variable to a static subclass.

Examples of arc functions, transition guards, color domains can be seen in the model portions of Fig. 3 and Fig. 4. The details about the WN notation can be found in [4].

The constraints on the syntax of WN allow the automatic exploitation of the behavioral symmetries of the model and the performance of the state-space based analysis on a more compact RG: the symbolic reachability graph (SRG). The SRG construction lies on the *symbolic marking* concept, namely a compact representation for a set of equivalent ordinary markings. A symbolic marking is a symbolic representation, where the actual identity of tokens is forgotten and only their distributions among places are stored. Tokens with the same distribution and belonging to the same static subclass are grouped into a so-called dynamic subclass. Starting from an initial symbolic marking, the SRG can be constructed automatically using a symbolic firing rule [4].

Various behavioral properties may be directly checked on the SRG. Furthermore, this construction leads also to efficient quantitative analysis, e.g. the performance evaluation of Stochastic WNs (SWNs) [4] (a SWN is obtained from a WN by associating an exponentially distributed delay with every transition, which may depend only on the static subclasses to which the firing colors belong).

3.2 Markov Decision Well-formed Net definition

A Markov Decision Well-formed Net, like an MDPN, is composed by two distinct parts: the probabilistic one and the non deterministic one, and also in this case the set of transitions in each part is partitioned into $Trun$ and $Tstop$. Each part of a MDWN is a WN model: the two parts share the same set of color classes. A MDWN comprises a special color class, say C_0, representing

the system components: its cardinality $|C_0|$ gives the total number of components in the system. This class can be partitioned into several static subclasses $C_0 = (\biguplus_{k=1}^{m} C_{0,k}) \uplus (\biguplus_{k=m+1}^{n_0} C_{0,k})$ such that colors belonging to different static subclasses represent components with different behavior and the first m static subclasses represent the controllable components while the others represent the non-controllable components. Observe that the model is parametric in the number of components of each.

Let us describe the specification of transition triggering by components in an MDWN. First, remember that the firing of a transition t involves the selection a color $c = (c_{i,j})_{i\in 0..n, j\in 1..e_i} \in cd(t) = \bigotimes_{i\in 0..n} C_i^{e_i}$. Thus the subset of components $\{c_{0,j}\}_{j\in 1..e_0}$ defines which components trigger the firing of $t(c)$.

- When the *type* ($synctype(t)$) of t is $Some$ then the subset of components that trigger this firing is $Comp(t,c) = \{c_{0,j}\}_{j\in dyn(t)}$, where $dyn(t) \subseteq \{1,\ldots,e_0\}$. Note that when t is a probabilistic transition, this requires that $dyn(t) \neq \emptyset$ whereas when t is a non deterministic one, this requires that $|dyn(t)| \leq 1$ (with the convention that $dyn(t) = \emptyset$ means that t is a decision relative to the system). Furthermore in the latter case, we assume that the guard of t entails that when $dyn(t) = \{c_{0,j}\}$, $c_{0,j} \in \biguplus_{k=1}^{m} C_{0,k}$, *i.e.* $c_{0,j}$ is a controllable component.
- When the *type* of t is *Allbut* then the subset of components that trigger this firing is $Comp(t,c) = \biguplus_{k\in static(t)} C_{0,k} \setminus \{c_{0,j}\}_{j\in dyn(t)}$ where $static(t) \subseteq \{1,\ldots,n_0\}$. Note that this type requires that t is a probabilistic transition. Additional conditions in the following definition ensure that this set of components is not empty.

Definition 5 (Markov Decision Well-formed Net (MDWN)). *A Markov Decision Well-formed is a tuple $\mathcal{MDWN} = \langle N^{pr}, N^{nd}, synctype, dyn, static\rangle$ where:*

- *N^{pr} is defined by a WN $\langle P, T^{pr}, \mathcal{C}, cd^{pr}, I^{pr}, O^{pr}, H^{pr}, \phi^{pr}, prio^{pr}, m_0\rangle$,, a mapping weight for each transition t, from $cd^{pr(t)}$ to $\mathbb{R}$*
- *N^{nd} is defined by a WN $\langle P, T^{nd}, \mathcal{C}, cd^{nd}, I^{nd}, O^{nd}, H^{nd}, \phi, prio, m_0\rangle$;*
- *$synctype : T^{pr} \cup T^{nd} \to \{Some, Allbut\}$ is a function which associates with every transition a label, s.t. $\forall t \in T^{nd} \Rightarrow synctype(t) = Some$.*
- *$dyn(t)$, where $t \in T^{pr} \cup T^{nd}$ and $cd(t) = \bigotimes_{i\in\{0,\ldots,n\}} C_i^{e_i}$, is a subset of $\{1,\ldots,e_0\}$ (cd is either cd^{pr} or cd^{nd});*
- *$static(t)$, defined when $synctype(t) = Allbut$, is a subset of $\{1,\ldots,n_0\}$ where n_0 represents the number of static subclasses in C_0.*

Furthermore, the following constraints must be fulfilled:

- *$T^{pr} \cap T^{nd} = \emptyset$;*
- *$T^{pr} = Trun^{pr} \uplus Tstop^{pr} \wedge T^{nd} = Trun^{nd} \uplus Tstop^{nd}$;*
- *$\forall t \in T^{pr} \wedge synctype(t) = Some \Rightarrow dyn(t) \neq \emptyset$;*
- *$\forall t$ s.t. $synctype(t) = Allbut$, $\sum_{j\in static(t)} |C_{0,j}| > |dyn(t)|$ (see discussion above);*

- $\forall t \in T^{nd} \Rightarrow 0 \leq |dyn(t)| \leq 1$; *moreover the transition guard* $\phi(t)$ *should enforce that when* $t(c)$ *is fireable with* $c = (c_{i,k})_{i \in 0..n, k \in 1..e_i} \in cd(t)$ *and* $j \in dyn(t)$ *then* $c_{0,j} \in \biguplus_{k=1}^{m} C_{0,k}$;
- $\forall c_0 \in C_0, \exists t \in Tstop^{pr}, \exists c \in cd(t),$ *s.t.* $\phi(t)(c) \wedge c_0 \in Comp(t,c)$ *and* $\forall c_0 \in \biguplus_{k=1}^{m} C_{0,k}, \exists t \in Tstop^{nd}, \exists c \in cd(t),$ *s.t.* $\phi(t)(c) \wedge c_0 \in Comp(t,c)$. *These conditions can be ensured by appropriate syntactical sufficient conditions.*
- $\forall \{j, j'\} \subseteq dyn(t) \wedge j \neq j', \forall c = (c_{i,k})_{i \in 0..n, k \in 1..e_i} \in cd(t)$ *s.t.* $t(c)$ *is possibly fireable one must have* $c_{0,j} \neq c_{0,j'}$. *This should be enforced by the transition guard.*

Now we introduce the rewards associated to the MDWN. Two types of reward functions are possible: the place reward and the transition reward. Before introducing the place reward we must define the set $\widetilde{C}$.

Definition 6 ($\widetilde{\mathcal{C}}$). *Let* $i \in \{1, \ldots, n\}$, $\widetilde{C}_i$ *is the set* $\{1, \ldots, n_i\}$ *where* n_i *is the number of static subclasses in* C_i. $\widetilde{\mathcal{C}}$ *is the set of sets* $\{\widetilde{C}_i\}_{i \in I}$ *with* $I = \{0, \ldots, n\}$.

We can always map the color class $\mathcal{C}$ on the set $\widetilde{\mathcal{C}}$ such that the definition of the $\widetilde{cd}$ function immediately follows.

Definition 7 ($\widetilde{cd}$). *The function* $\widetilde{cd}(p)$ *is defined as follows:*

$$\widetilde{cd} \stackrel{def}{=} \widetilde{(\bigotimes_{i \in I} C_i^{e_i})} = \bigotimes_{i \in I} \widetilde{C}_i^{e_i}$$

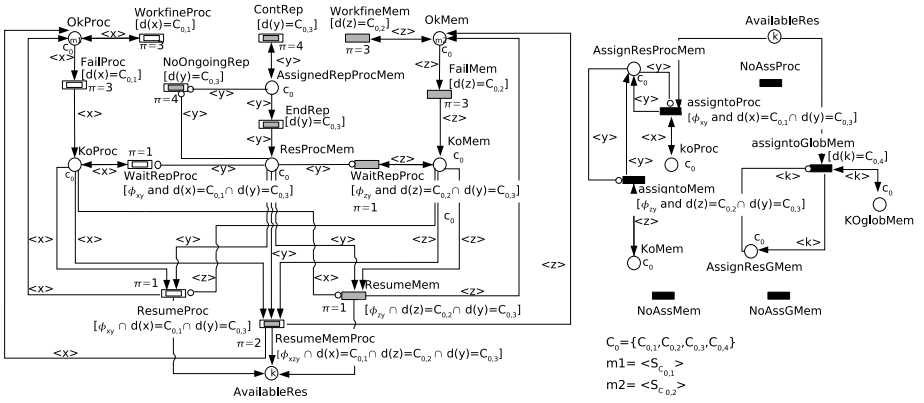
For instance if $C_0 = C_{0,1} \cup C_{0,1} \cup C_{0,3}$ where $C_{0,1} = \{comp_1, comp_2\}, C_{0,2} = \{comp_3\}, C_{0,3} = \{comp_4\}$, hence $\widetilde{C}_0 = \{C_{0,1}, C_{0,1}, C_{0,3}\}$, and $p \in P$ with $cd(p) = C_0 \times C_0 \times C_0$ then $\widetilde{cd}(p) = \widetilde{C}_0 \times \widetilde{C}_0 \times \widetilde{C}_0$, $c = \langle comp_1, comp_2, comp_3 \rangle \in cd(p)$ and $\widetilde{c} = \langle 1, 1, 2 \rangle \in \widetilde{cd}(p)$.

It is important to observe that a unique $\widetilde{c}$ corresponds to every c.

Definition 8 (MDWN reward functions).

- $rs : \bigotimes_{p \in P} \mathbb{N}^{\widetilde{cd}(p)} \to \mathbb{R}$ *is a function which returns for every colored marking a reward value.*
- $\forall t \in T^{nd}, rt[t] : cd(t) \to \mathbb{R}$ *is a vector which associates with every transition a function defining the reward value of its instances; two instances may be assigned a different reward value only if there exists a standard predicate capable to distinguish the two.*
- $rg : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ *is defined as in MDPN.*

An example of MDWN is shown in Fig. 3, the same already used to illustrate MDPNs, but this time color annotations on arcs, transitions and places are relevant. In this model we are assuming that there are several instances of Proc, Mem and ResCtr components (grouped in banks, each with one instance of each component): rather than replicating the same subnet several times, we use colored tokens to represent several instances on the same net structure (there is also another controllable component not shown in the probabilistic subnet, but visible in the decision maker). Class C_0 comprises four static subclasses, one for



$Trun^{pr} = \{FailProc, FailMem, EndRep\}$ all other transitions belong to $Tstop^{pr}$; all variables are component parameters. All transitions in the decision maker are in $Tstop^{nd}$. Transition priorities are denoted $\pi = prio(t)$ in the figure. $C_{0,1}$, $C_{0,2}$ and $C_{0,3}$ are the Proc, Mem and ResCtr subclasses respectively. Here we have represented the probabilistic transitions with different simbols (double rectangle, gray rectangle and double gray rectangle) depending on the involved components

Fig. 3. MDWN example. On the left: a portion of probabilistic part, on the right: the decision maker.

each component type. The cardinality of the Proc, Mem and ResCtr subclasses corresponds to the number of banks in the system. Arcs in Fig. 3 are annotated with very functions (tuples of projections) and all the variables appearing in the functions in this example are component parameters. The guards on the arcs include a term in the form $d(x) = CompType$ to force parameter x to range within static subclass $CompType$. The additional terms ϕ_{xyz}, ϕ_{xz}, ϕ_{yz} are not detailed here, but are used to associate components in the same bank: in fact the probabilistic part of the model must correctly synchronize components of type Proc, Mem and ResCtr belonging to the same bank (the model represents a situation where only one resource is assigned to each bank at a time, and it can be used to resume all failed components in the bank).

3.3 MDWN semantics

In this section we are going to describe how it is possible to obtain from an MDWN model the corresponding MDP model. The two possible methods are shown in the figure of the introduction.

The first method requires the unfolding of the MDWN in order to obtain an equivalent MDPN and to derive from this an MDP, but this is not very efficient in fact it will multiply the number of places, transitions and arcs, moreover if the number of components is high the cost for computing the results will be high. In [2] it is possible to find the details of this method.

Instead the second method derives directly from an MDWN model an MDP. This second method can be decomposed in two steps: the first step defines how to compose the probabilistic part and the decision maker and to derive from such composition a unique WN. The second step consists in generating the (finite)

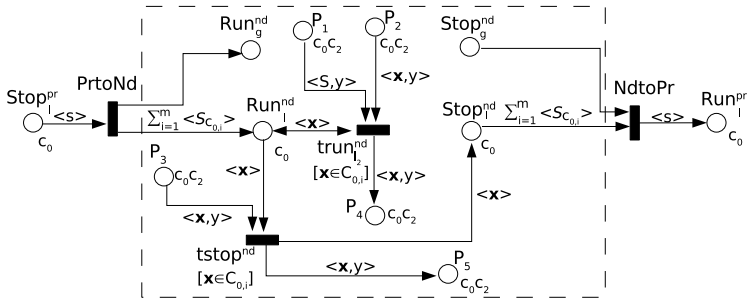


Fig. 4. arcs connecting places $Stop^{pr}$, Run_l^{nd}, Run_g^{nd}, and transition $PrtoNd$ and their functions; arcs connecting places $Stop_l^{nd}$, $Stop_g^{nd}$, Run_l^{nd} and transition $NdtoPr$ and their function; example of connection of the decision maker to places Run^nd and $Stop^nd$: component parameters are highlighted in boldface in the arc functions.

RG of the WN obtained in the first step and then in deriving an MDP from it. In this way there is no need to produce the intermediate MDPN.

Before describing the second method we must explain the use of the places $Stop_l^{pr}$, Run_l^{pr}, $Stop_l^{nd}$, Run_l^{nd}, $Stop_g^{nd}$, Run_g^{nd} and the non deterministic transitions $PrtoNd$ and $NdtoPr$, that are introduced during the composition phase.

The places $Stop^{pr}$, Run^{pr}, $Stop_l^{nd}$, Run_l^{nd}, $Stop_g^{nd}$ and Run_g^{nd} are used in order to regulate the interaction among the components, the global system and the decision maker like the similar places in the semantics for MDPN. The color domain of the places $Stop^{pr}$, Run^{pr}, $Stop_l^{nd}$ is C_0, that is they will contain colored tokens representing the components; while Run_g^{nd}, $Stop_g^{nd}$ are neutral. The non deterministic transitions $PrtoNd$ and $NdtoPr$ are used to assure that the decision maker takes a decision for every component in every time unit.

The schema describing how the places $Stop^{pr}$, Run^{pr}, $Stop_l^{nd}$, Run_l^{nd}, $Stop_g^{nd}$ and Run_g^{nd} and the transitions $PrtoNd$ and $NdtoPr$ are connected, is shown in Fig.4. Observe that the basic schema is the same already defined for MDPN but now the arcs are annotated with function $< S >$ meaning that all components must synchronize at that point.

Let us describe how to derive a unique WN composing the probabilistic part with the non deterministic part. Places Run^{pr} and $Stop^{pr}$, introduced above, are connected with its run/stop transitions of N^{pr} in the same way as for MDPNs, similarly places Run_l^{nd} and $Stop_l^{nd}$ Run_g^{nd} and $Stop_g^{nd}$ introduced above are connected to the run/stop transitions of N^{nd} as for MDPNs, but now the arcs must be annotated with the following functions.

- $\forall t \in T^{pr} \cup T_l^{nd}$, if $synctype(t) = Some$ then the function is $\langle \sum_{i \in dyn(t)} x_{0,i} \rangle$, where variable $x_{0,i}$ denotes the i-th component of type C_0 in the color domain of t. This function selects the colors of the component that trigger the transition, thus checking that all of them are still active.
- $\forall t \in Trun^{pr}$, if $synctype(t) = Allbut$ then the function is $\langle \sum_{j \in static(t)} S_{0,j} - \sum_{i \in dyn(t)} x_{0,i} \rangle$ with the same interpretation.

Observe that the arcs connecting transitions T_g^{nd} and places Run_g^{nd}, $Stop_g^{nd}$ are not annotated with any function because these places have *neutral* color (i.e. they contain plain black tokens) since they are related to the decision w.r.t. the whole system.

Once the composed WN is built, its RG can be constructed and transformed into a MDP following the same two steps already explained for MDPN. Here, since we start from a high-level net, the resulting reachability graph may be huge. So the following subsection describe how the properties of WN can be extended to MDWN so that a smaller MDP can be directly derived from the Symbolic Reachability Graph (SRG) of the corresponding WN.

3.4 Theoretical results on symmetry exploitation

In this section, we informally describe how we exploit the symbolic reachability graph in order to obtain a reduced MDP on which the solution to the original problem can be computed (see [2] for a complete theoretical development).

First, let us pick a symbolic reachable marking which only enables non deterministic transitions and an ordinary marking belonging to this symbolic marking. Now let us pick two ordinary firings from this marking corresponding to the same symbolic firing. Suppose that, at some instant of an execution, a strategy selects one of these firings. Then, after selecting the other firing, one mimics the original strategy by applying one of the permutations which lead from the former firing to the latter one to any subsequent (probabilistic or non deterministic) firing and let invariant the ordinary marking. Due to our assumptions about the rewards, the two executions yield the same (total or average) reward. It means that the choice of the second firing is at least as good as the selection of the first firing. Since the argument is symmetric, one concludes that the selection of any non deterministic firing inside a symbolic arc is irrelevant.

Then the reduced MDP obtained from the SRG by considering that a symbolic firing of a non deterministic transition corresponds to a *single* decision and that the weight of probabilistic symbolic firing is the weight of any ordinary firing inside it (any choice leads to the same weight due to our assumptions) multiplied by the number of such firings provides an MDP *equivalent* to the original one w.r.t. the considered optimization problem. Indeed the rewards do not depend on the choice of an ordinary marking inside a symbolic marking and the choice of an ordinary firing inside a symbolic firing. We will call SRG_{nd} the SRG where all the transition instances passing only through non deterministic states are reduced to one non deterministic step and SRG_{MDP} the SRG_{nd} where all probabilistic paths are substituted by single probabilistic arcs.

4 Experiments discussion

In this section we will present an example modeling a multiprocessor system where each processor has a local memory, but with also a global shared memory that can be used by any processor when its local memory fails. Each processor,

Table 1. Results for the example modeling a multiprocessor system. The RG for Proc=4 and Mem=4 is not computed because it requires a lot of time; its size is computed indirectly by the SRG

	Proc=2,Mem=2,Res=2			Proc=3,Mem=3,Res=2			Proc=4,Mem=4,Res=2		
	Prob.	Non det.	Time	Prob.	Non det.	Time	Prob.	Non det.	Time
RG	19057	21031	13s	755506	863886	1363s	26845912	31895848	>13h
RG_{nd}	19057	441	9s	755506	4078	2833s			
RG_{MDP}	0	441	2s	0	4078	250s			
SRG	9651	10665	9s	132349	150779	284s	1256220	1478606	5032s
SRG_{nd}	9651	219	3s	132349	831	222s	1256220	2368	12795s
SRG_{MDP}	0	219	1s	0	831	28s	0	2360	518s
RG **prio**	19057	5235	9s	755506	103172	983s	26845912	1863024	>13h
RG_{nd} **prio**	19057	411	4s	755506	4078	1830s			
RG_{MDP} **prio**	0	411	2s	0	4078	246s			
SRG **prio**	9651	2697	6s	132349	18904	187s	1256220	96044	3270s
SRG_{nd} **prio**	9651	219	2s	132349	831	75s	1256220	2368	1560s
SRG_{MDP} **prio**	0	219	1s	0	831	26s	0	2360	234s

local memory and global shared memory can fail independently; however we consider recoverable failures, that can be solved by restarting/reconfiguring the failed component. The system includes an automatic failure detection system that is able to detect and perform a reconfiguration of the failed component (e.g. by resetting it). The failure detection and recovery system can handle a limited number k of failures in parallel.

Notice that if a local memory M_i and the global shared memory Mg are both failed at the same time, the processor P_i cannot perform any useful work, even if it is not failed and that if the processor P_i and its local memory M_i are simultaneously failed, they are reset together (this is considered as a single reset operation). The components in this system are: n processors, n local memories and one global shared memory. A portion of MDWN representation of this system is depicted in Fig. 3

The decision maker corresponds to the automatic failure detection and recovery system. Several different recovery strategies can be conceived, and we are interested in evaluating the most promising ones with respect to some metrics.

An MDPN (or MDWN) model of this system is composed of a submodel representing all the components of the system (which in turn can be seen as a combination of several submodels of the single components), and a submodel representing the failure detection and recovery system, which in this context corresponds to the decision maker.

The decision maker model may represent any possible recovery strategy, in this case it should be modeled in such a way that any association of up to k recovery resources to any subset of failed components at a given time can be realized by the model.The system must pay a penalty depending of the number of running processors when the number of running processors is less than a given threshold and a repair cost for every recovery. More details about this example are shown in [2]. The table 1 shows the number of states and the computation time respectively of the $RG, RG_{nd}, RG_{MDP}, SRG, SRG_{nd}$ and SRG_{MDP} for different numbers of processors and memories performed with an AMD Athlon

64 2.4Ghz of 4Gb memory capacity. In particular the first, the second and the third line report the number of states and computation time of the RG, the RG_{nd} and the RG_{mdp}, while the following three lines show the number of states and the computation time obtained using the SRG technique. It is easy to observe how the SRG technique wins in terms of memory and time gain with respect to the RG technique.

A further reduction of the number of states for this model can be achieved associating different priorities to the system transitions such that the interleavings between the non deterministic/probabilistic actions are reduced. For instance the last six lines in table 1 show the reduction in terms of non deterministic states and time computation obtained imposing an order on the decision maker choices. (First the decision maker must take all the decisions for the processors then for the memories and in the end for the global memory).

It is not always possible to use this trick since the actions must be independent; the priorities in practice must not reduce the set of possible strategies. Our tool solves the MDPs using the library *graphMDP* developed at the Ecole Nationale Suprieure de l'Aronautique et de l'Espace Toulouse. The optimal strategy is expressed as a set of optimal actions, such that for every system state an optimal action is given.

For example if we consider a model with two processors, two memories and two recovery resources, and with reasonable fault probability, and repair and penalty costs then we observe that if a fault happens and there is a free recovery resource then the recovery of this fault starts immediately and the global memory recovery is preferred with respect the processor recovery and the memory recovery.This is not always true, e.g. if global memory recovery cost is more than four times of the memory repair cost.

After having obtained the optimal strategy we would like to synthesize a new model without non determinism implementing it (this could be achieved by substituting the decision maker part with a new probabilistic part implementing the decisions of the optimal strategy): classical Markov chain analysis techniques could be applied to this model, moreover the new net would constitute a higher level (hopefully easier to interpret) description for the optimal strategy. Unfortunately this is not always easy (especially when the number of states is large), but this is an interesting direction of future research.

5 Related work

In this section we are going to compare our formalism with two other high level formalisms for MDP: the PRISM language and the Stochastic Transition System (STS) proposed in [8].

The PRISM language [11] is a state-based language based on the Reactive Modules formalism of Alur and Henzinger [1]. A system is modeled by PRISM language as composition of modules(components) which can interact with each other. Every model contains a number of local variables used to define it state in every time unit, and the local state of all modules determines the global state.

The behavior of each module is described by a set of commands; such that a command is composed by a guard and a transition. The guard is a predicate over all the (local/nonlocal) variables while a transition describes how the local variable will be update if the its guard is true.

The composition of the modules is defined by a process-algebraic expression: parallel composition of modules, action hiding and action renaming.

Comparing the MDPN formalism with the PRISM language we can observe that they have the same expressive power: we can define local or global non-deterministic actions and the reward function on the states and/or on the actions in both formalisms; such that it is possible to translate MDPN model directly in a PRISM model. The main difference is that by using the MDPN formalism one can define complex probabilistic behaviors and complex non-deterministic actions as a composition of simpler behaviors or actions.

If we compare the PRISM language with the MDWN then we can see that the MDWN has two other advantages: a parametric description of the model and an efficient analysis technique making it possible to automatically take advantage of intrinsic symmetries of the system. In fact the PRISM language has a limited possibility for parametrization. In order to cope with this problem in [9] it was presented a syntactic pre-processor called eXtended Reactive Modules (XRM) which can generate RM models giving to the users the possibility of describing the system using for instance: *for* loops, *if* statements.

Instead several techniques proposed in order to reduce the states explosion problem in PRISM i.e. in [10] were based on the minimization of the RG with respect to bisimulation; but this requires the building of all the state space and then to reduce it; hence our method gives the possibility of managing models with a bigger number of states. It generates directly the Lumped MDP without building all the state space.

A direct comparison between our formalisms and the STS is not possible, because the STSs are an high level formalism for modeling the continuous time MDPs. It extends the Generalized Stochastic Petri Net by introducing transitions with an unspecified delay distributions and by the introducing the possibility of non-deterministic choice among enabled immediate transitions. In every way we can observe that the STS has the same problems of GSPN formalism; that make its utilization less advantageous with respect to the WN. It is also important to observe that there are no tools supporting this formalism.

6 Conclusion

We have introduced MDPNs, based on Petri nets, and MDWNs, based on Well-formed nets, in order to model and analyze distributed systems with probabilistic and non deterministic features. From a modeling point of view, these models support a macroscopic point of view of alternation between the non probabilistic behavior and the non deterministic one of the system and a syntactical way to define the switch between the two behaviors. Furthermore, MDWNs enable the modeler to specify in a concise way similar components. From an analy-

sis point of view, we have adapted the technique of the symbolic reachability graph producing a reduced Markov decision process w.r.t. the original one, on which the analysis may be performed. Our methods are already implemented and integrated in the GreatSPN tool and we have described some experimental results.

References

1. R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
2. M. Beccuti, G. Franceschinis, and S. Haddad. Markov Decision Petri Net and Markov Decision Well-formed Net formalisms. Technical Report TR-INF-2007-02-01, Dipartimento di Informatica, Università del Piemonte Orientale, 2007. http://www.di.unipmn.it/Tecnical-R.
3. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *15th Int. Conf. of Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *LNCS*, pages 499–513, Bangalore, India, 1995. Springer.
4. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42(11):1343–1360, nov 1993.
5. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. A symbolic reachability graph for coloured Petri nets. *Theoretical Computer Science*, 176(1–2):39–65, 1997.
6. G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic petri nets. *Performance Evaluation, special issue on Performance Modeling Tools*, 24(1-2):47–68, November 1995.
7. L. de Alfaro. Temporal logics for the specification of performance and reliability. In *14th Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *LNCS*, pages 165–176, Hansestadt Lbeck, Germany, 1997. Springer.
8. L. de Alfaro. Stochastic transition systems. In *9th International Conference on Concurrency Theory*, volume 1466 of *LNCS*, pages 423–438, Nice, France, 1998. Springer.
9. K. Demaille, S. Peyronnet, and B. Sigoure. Modeling of sensor networks using XRM. In *2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, Paphos, Cyprus, 2006.
10. H. Garavel and H. Hermanns. On combining functional verification and performance evaluation using CADP. In *In FME 2002: International Symposium of Formal Methods Europe,*, pages 10–429, Copenhagen, Denmark, 2000.
11. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *LNCS*, pages 441–444, Vienna, Austria, 2006. Springer.
12. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, John Wiley and Sons, 1995. Download http://www.di.unito.it/∼greatspn.
13. M. L. Puterman. *Markov Decision Processes. Discrete Stochastic Dynamic Programming*. Wiley, 2005.