Carrying Probabilities to the Infinite World*

Parosh Aziz Abdulla

Uppsala University, Sweden

Abstract. We give en example-guided introduction to a framework that we have developed in recent years in order to extend the applicability of program verification to the context of systems modeled as infinite-state Markov chains. In particular, we describe the class of *decisive Markov chains*, and show how to perform qualitative and quantitative analysis of Markov chains that arise from probabilistic extensions of classical models such as Petri nets and communicating finite-state processes.

1 Introduction

In recent years, several approaches have been proposed for automatic verification of infinite-state systems (see e.g., [2, 1]). In a parallel development, there has been an extensive research effort for the design and analysis of models with stochastic behaviors (e.g., [12, 7, 6, 11]). Recently, several works have considered verification of infinite-state Markov chains that are generated by push-down systems (e.g., [9, 10]). We consider verification of Markov chains with infinite state spaces. We describe a general framework that can handle probabilistic versions of several classical models such as Petri nets and communicating finite-state processes. We do that by defining abstract conditions on infinite Markov chains that give rise to the class of *decisive Markov chains*. For this class, we perform qualitative and quantitative analysis wrt. standard properties such as reachability and repeated reachability of a given set of configurations. This presentation is informal and example-based. For the technical details, we refer to our works in [3–5].

2 Transition Systems

A transition system $\mathcal{T}$ is a pair $(C, \longrightarrow)$ where C is a (potentially infinite) set of *configurations*, and $\longrightarrow \subseteq C \times C$ is the *transition relation*. As usual, we write $c \longrightarrow c'$ to denote that $(c, c') \in \longrightarrow$ and use $\stackrel{*}{\longrightarrow}$ to denote the reflexive transitive closure of $\longrightarrow$. For a configuration c, a *c-run* is a sequence $c_0 \longrightarrow c_1 \longrightarrow c_2 \longrightarrow \cdots$ where $c_0 = c$. For a natural number k, we write $c \stackrel{k}{\longrightarrow} c'$ if there is a sequence $c_0 \longrightarrow c_1 \longrightarrow \cdots \longrightarrow c_\ell$ with $\ell \leq k$, $c_0 = c$ and $c_\ell = c'$, i.e., we can reach c' from c in k or fewer steps. Notice that $c \stackrel{*}{\longrightarrow} c'$ iff $c \stackrel{k}{\longrightarrow} c'$ for some k. We lift the above notation to sets of configurations. For sets $C_1, C_2 \subseteq C$ of configurations,

* This tutorial is based on common work with Noomene Ben Henda, Richard Mayr, and Sven Sandberg.

J.-P. Katoen and B. König (Eds.): CONCUR 2011, LNCS 6901, pp. 1–16, 2011.

we write $C_1 \longrightarrow C_2$ if $c \longrightarrow c'$ for some $c \in C_1$ and $c' \in C_2$. We use $C_1 \xrightarrow{k} C_2$ and $C_1 \xrightarrow{*} C_2$ in a similar way. We also mix the notations, so we we write for instance $c \xrightarrow{*} C_2$ instead of $\{c\} \xrightarrow{*} C_2$ We say that C_2 is *reachable* from C_1 if $C_1 \xrightarrow{*} C_2$. A transition system $\mathcal{T}$ is said to be *k-spanning* wrt. a give set F of configurations if for any configuration c, we have that $c \xrightarrow{*} F$ implies $c \xrightarrow{k} F$. In other words, for any configuration c, either c cannot reach F or it can reach F within k steps. We say that $\mathcal{T}$ is *finitely spanning* wrt. F if there is a k such that $\mathcal{T}$ is k-spanning wrt. F. In other words, if $\mathcal{T}$ is *finitely spanning* wrt. F then $\exists k.\ \forall c \in C.\ c \xrightarrow{*} F \supset c \xrightarrow{k} F$. We define $\widetilde{F} := \left\{c|\ c \not\xrightarrow{*} F\right\}$, i.e., $\widetilde{F}$ is the set of configurations from which F is not reachable. For a set $U \subseteq C$, we define $Pre(U) := \{c|\ \exists c' \in U.\ c \longrightarrow c'\}$, i.e., $Pre(U)$ is the set of configurations that can reach U through the execution of a single transition. We assume familiarity with the temporal logic CTL^*. Given a CTL^* path-formula ϕ, we use $(c \models \phi)$ to denote the set of c-runs that satisfy ϕ.

3 Petri Nets

We illustrate some ideas of our methodology, using the model of *Petri Nets*. After recalling the standard definitions of Petri nets, we describe the transition system induced by a Petri net. We describe how checking safety properties can be translated to the reachability of sets of configurations which are upward closed wrt. a natural ordering on the set of configurations[1]. We give a sketch of an algorithm to solve the reachability problem, and show that Petri nets are finitely spanning with respect to upward closed sets of configurations. Finally, we briefly mention a model closely related to Petri nets, namely that of Vector Addition Systems with States (VASS).

3.1 Model

A Petri net $\mathcal{N}$ is a tuple (P, T, F), where P is a finite set of *places*, T is a finite set of *transitions*, and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*. If $(p, t) \in F$ then p is said to be an *input* place of t; and if $(t, p) \in F$ then p is said to be an *output* place of t. We use $In(t) := \{p|\ (p, t) \in F\}$ and $Out(t) := \{p|\ (t, p) \in F\}$ to denote the sets of input places and output places of t respectively.

Figure 1 shows an example of a Petri net with three places (drawn as circles), namely `L`, `W`, and `C`; and two transitions (drawn as rectangles), namely t_1 and t_2. The flow relation is represented by edges from places to transitions, and from transitions to places. For instance, the flow relation in the example includes the pairs $(\texttt{L}, t_1)$ and $(t_2, \texttt{W})$, i.e., `L` is an input place of t_1, and `W` is an output place of t_2.

The transition system induced by a Petri net is defined by the set *configurations* together with the *transition relation* defined on them. A *configuration* c

[1] Reachability of upward closed sets of configurations is referred to as the *coverability problem* in the Petri net literature.

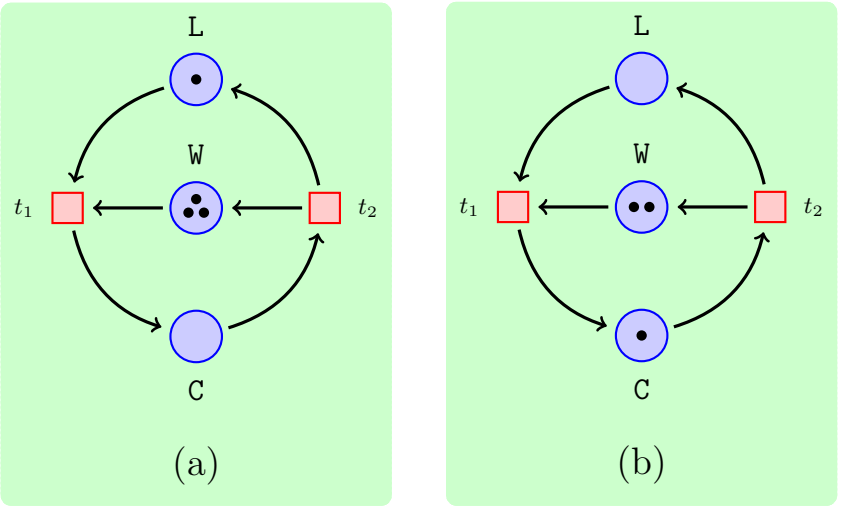

Fig. 1. (a) A simple Petri net (b) The result of firing t_1

of a Petri net[2] is a multiset over P. The configuration c defines the number of *tokens* in each place. Figure 1 (a) shows a configuration where there is one token in place `L`, three tokens in place `W`, and no token in place `C`. The configuration corresponds to the multiset $\left[\mathtt{L}, \mathtt{W}^3\right]$.

The operational semantics of a Petri net is defined through the notion of *firing* transitions. This gives a transition relation on the set of configurations. More precisely, when a transition t is fired, then a token is removed from each input place, and a token is added to each output place of t. The transition is fired only if each input place has at least one token. Formally, we write $c_1 \longrightarrow c_2$ to denote that there is a transition $t \in T$ such that $c_1 \geq In\,(t)$ and $c_2 = c_1 - In\,(t) + Out\,(t)$.

A set $U \subseteq C$ of configurations is said to be *upward closed* if $c \in U$ and $c \leq c'$ implies that $c' \in U$. For a configuration $c \in C$, define the *upward closure* of c by $\widehat{c} := \{c' |\ c \leq c'\}$. We extend the definition to a set $C_1 \subseteq C$ of configurations by $\widehat{C_1} := \cup_{c \in C_1} \widehat{c}$.

The Petri net of Figure 1 can be seen as a model of a simple mutual exclusion protocol, where access to the critical section is controlled by a global lock. A process is either *waiting* or is in its *critical section*. Initially, all the processes are in their waiting states. When a process wants to access the critical section, it must first acquire the lock. This can be done only if no other process has already acquired the lock. From the critical section, the process eventually releases the lock and moves back to the waiting state. The numbers of tokens in places `W` and `C` represent the number of processes in their waiting states and critical sections respectively. Absence of tokens in `L` means that the lock is currently taken by some process.

[2] A configuration in a Petri net is often called a *marking* in the literature.

The set C_{init} of *initial configurations* are those of the form $[\mathtt{L}, \mathtt{W}^n]$ where $n \geq 0$. In other words, all the processes are initially in their waiting states, and the lock is free. The transition t_1 models a process moving to its critical section, while the transition t_2 models a process going back to its waiting state. For instance, if we start from the configuration $\left[\mathtt{L}, \mathtt{W}^4\right]$, we can fire the transition t_1 to obtain the configuration $\left[\mathtt{C}, \mathtt{W}^3\right]$ from which we can fire the transition t_2 to obtain the configuration $\left[\mathtt{L}, \mathtt{W}^4\right]$, and so on.

3.2 Safety Properties

We are interested in checking a safety property for the Petri net in Figure 1. In a safety property, we want to show that "nothing bad happens" during the execution of the system. Typically, we define a set *Bad* of configurations, i.e., configurations which we do not want to occur during the execution of the system. In this particular example, we are interested in proving mutual exclusion. The set *Bad* contains those configurations that violate mutual exclusion, i.e., configurations in which at least two processes are in their critical sections. These configurations are of the form $\left[\mathtt{L}^k, \mathtt{W}^m, C^n\right]$ where $n \geq 2$. Checking the safety property can be carried out by checking whether we can fire a sequence of transitions taking us from an initial configuration to a bad configuration, i.e., we check whether the set *Bad* is reachable.

To analyze safety properties, we study some aspects of the behavior of Petri nets. First, we observe that the behavior of a Petri net is *monotonic*: if $c_1 \longrightarrow c_2$ and $c_1 \leq c_3$ then there is a c_4 such that $c_3 \longrightarrow c_4$ and $c_4 \geq c_2$.

We will work with sets of configurations that are upward closed with respect to $\leq$. Such sets are interesting in our setting for two reasons. First, all sets of bad configurations that occur in our examples are upward closed. For instance, in our example, whenever a configuration contains two processes in their critical sections then any larger configuration will also contain (at least) two processes in their critical sections, so the set *Bad* is upward closed. In this manner, checking the safety property amounts to deciding reachability of an upward closed set. Second, each upward closed set U can be uniquely represented by its set of minimal elements. This set, which we refer to as the set of *generator* of U, is finite due to Dickson's lemma [8]. In fact, since the ordering $\leq$ is anti-symmetric, it follows that each upward closed set has a unique generator. Finally, we observe that, due to monotonicity, if U is upward closed then $Pre(U)$ is upward closed. In other words, upward closedness is preserved by going backwards in the transition relation.

Below, we give a sketch of backward reachability algorithm for checking safety properties.

3.3 Algorithm

As mentioned above, we are interested in checking whether it is the case that *Bad* is reachable. The safety property is violated iff the question has a positive

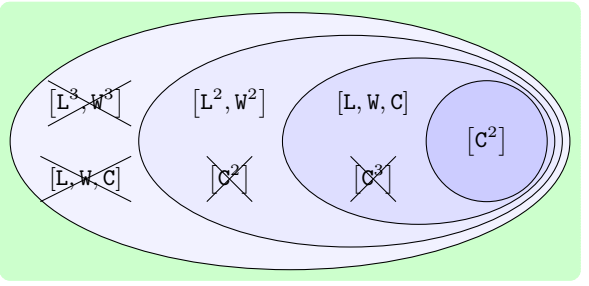

Fig. 2. Running the backward reachability algorithm on the example Petri net. Each ellipse contains the configurations generated during one iteration. The subsumed configurations are crossed over.

answer. The algorithm, illustrated in Figure 2, starts from the set of bad configurations, and tries to find a path backwards through the transition relation to the set of initial configurations. The algorithm operates on upward closed sets of configurations. An upward closed set is symbolically represented by a finite set of configurations, namely the members of its generator. In the above example, the set $gen\,(Bad)$ is the singleton $\left\{\left[\mathtt{C}^2\right]\right\}$. Therefore, the algorithm starts from the configuration $c_0 = \left[\mathtt{C}^2\right]$. From the configuration c_0, we go backwards and derive the generator of the set of configurations from which we can fire a transition and reach a configuration in $Bad = \widehat{c_0}$. Transition t_1 gives the configuration $c_1 = [\mathtt{L}, \mathtt{W}, \mathtt{C}]$, since $\widehat{c_1}$ contains exactly those configurations from which we can fire t_1 and reach a configuration in $\widehat{c_0}$. Analogously, transition t_2 gives the configuration $c_2 = \left[\mathtt{C}^3\right]$, since $\widehat{c_2}$ contains exactly those configurations from which we can fire t_2 and reach a configuration in $\widehat{c_0}$. Since $c_0 \leq c_2$, it follows that $\widehat{c_2} \subseteq \widehat{c_0}$. In such a case, we say that c_2 is *subsumed* by c_0. Since $\widehat{c_2} \subseteq \widehat{c_0}$, we can discard c_2 safely from the analysis without the loss of any information. Now, we repeat the procedure on c_1, and obtain the configurations $c_3 = \left[\mathtt{L}^2, \mathtt{W}^2\right]$ (via t_1), and $c_4 = \left[\mathtt{C}^2\right]$ (via t_2), where c_4 is subsumed by c_0. Finally, from c_3 we obtain the configurations $c_5 = \left[\mathtt{L}^3, \mathtt{W}^3\right]$ (via t_1), and $c_6 = [\mathtt{L}, \mathtt{W}, \mathtt{C}]$ (via t_2). The configurations c_5 and c_6 are subsumed by c_3 and c_1 respectively. The iteration terminates at this point since all the newly generated configurations were subsumed by existing ones, and hence there are no more new configurations to consider. In fact, the set $B = \left\{\left[\mathtt{C}^2\right], [\mathtt{L}, \mathtt{W}, \mathtt{C}], \left[\mathtt{L}^2, \mathtt{W}^2\right]\right\}$ is the generator of the set of configurations from which we can reach a bad configurations. The three members in B are those configurations which are not discarded in the analysis (they were not subsumed by other configurations). To check whether Bad is reachable, we check the intersection $\widehat{B} \cap C_{init}$. Since the intersection is empty, we conclude that Bad is not reachable, and hence the safety property is satisfied by the system.

3.4 Finite Span

An interesting consequence of the above algorithm is that Petri nets are finitely spanning with respect to an upward closed set of configurations. First, the reachability algorithm is guaranteed to terminate by Dickson's lemma [8]. Suppose that the algorithm starts by an upward closed set U (represented by its generator) and suppose that it terminates in k steps. Then, we claim that the Petri net is k-spanning wrt. U. To see this, consider a configuration c such that $c \xrightarrow{*} U$. Then, the algorithm will generate some c' such that $c' \leq c$. Suppose that c' is generated in step $\ell \leq k$. Then $c' \xrightarrow{\ell} U$. By monotonicity, we have that $c \xrightarrow{\ell} U$. For instance, the span of the Petri net of Figure 1 wrt. $\left[\mathtt{C}^2\right]$ is equal to 2.

3.5 VASS

A VASS is simply a Petri net equipped with a finite set of control states. Each transition has exactly one input control state and one output control state. Thus a transitions changes the control state of the VASS (in addition to changing the numbers of the tokens in the places). We can also think of a VASS as a counter machine where the counters are allowed to be tested for equality with zero.

4 Markov Chains

A *Markov chain* $\mathcal{M}$ is a tuple (C, P) where C is a (potentially infinite) set of *configuration*, and $P : C \times C \to [0, 1]$, such that $\sum_{c' \in C} P(c, c') = 1$, for each $c \in C$. A Markov chain induces a transition system, where the transition relation consists of pairs of configurations related by positive probabilities. In this manner, concepts defined for transition systems can be lifted to Markov chains. For instance, for a Markov chain $\mathcal{M}$, a run of $\mathcal{M}$ is a run in the underlying transition system, and $\mathcal{M}$ is *finitely spanning* w.r.t. given set F if the underlying transition system is finitely spanning w.r.t. F, etc.

We use $Prob_c(\phi)$ to denote the measure of the set of c-runs ($c \models \phi$) (which is measurable by [12]). Sometimes, we refer to $Prob_c(\phi)$ as the probability by which ϕ holds at c. For instance, given a set $F \subseteq C$, $Prob_c(\Diamond F)$ is the measure of c-runs which eventually reach F. We say that *almost all* runs of a Markov chain satisfy a given property ϕ if $Prob_c(\phi) = 1$. In this case one says that ($c \models \phi$) holds *almost certainly*. For formulas ϕ_1, ϕ_2, we use $Prob_c(\phi_1 \mid \phi_2)$ to denote the probability that ϕ_2 holds under the assumption that ϕ_1 holds.

5 Decisive Markov Chains

In this section, we introduce *decisive Markov chains*, present two sufficient conditions for decisiveness, and show examples of models that induce decisive Markov chains.

Consider a Markov chain $\mathcal{M} = (C, P)$ and a set F of configurations. We say that $\mathcal{M}$ is *decisive* wrt. F if each run of the system almost certainly will

eventually either reach F or reach $\widetilde{F}$. Formally, for each configuration c, it is the case that $Prob_c\left(\Diamond F \vee \Diamond \widetilde{F}\right) = 1$. Put differently, if F is always reachable along a run ρ then ρ will almost certainly eventually reach F, i.e., $Prob_c(\Diamond F \mid \Box\exists\Diamond F) = 1$. Figure 3 shows an illustration of a run in a decisive Markov chain.

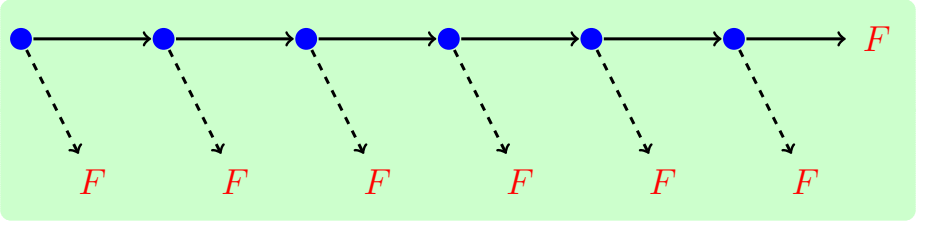

Fig. 3. Illustration of a run in a decisive Markov chain

Notice that all finite Markov chains are decisive (wrt. any given set of configurations). On the other hand, Figures 4–5 show examples of infinite Markov chains that are not decisive. Let us consider the Markov chain of Figure 4. The configuration F is reachable from each configuration in the Markov chain. Therefore $\widetilde{F} = \emptyset$, and hence $Prob_{Init}\left(\Diamond F \vee \Diamond \widetilde{F}\right) = Prob_{Init}(\Diamond F) = \frac{2}{3} < 1$.

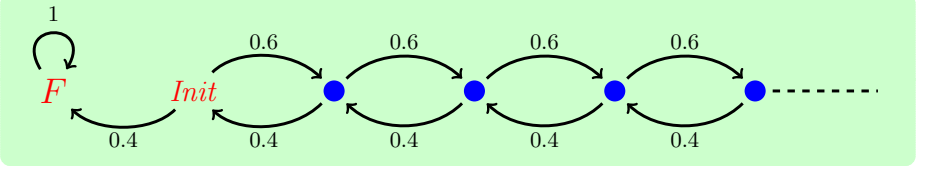

Fig. 4. A Markov chain that is not decisive

Next, let us consider the Markov chain of Figure 5. Again, the configuration F is reachable from each configuration in the Markov chain. Therefore, $Prob_{Init}\left(\Diamond F \vee \Diamond \widetilde{F}\right) = Prob_{Init}(\Diamond F) < 0.2$.

5.1 Sufficient Condition I

A configuration c is said to be of *coarseness* β if for each $c' \in C$, $P(c, c') > 0$ implies $P(c, c') \geq \beta$. A Markov chain $\mathcal{M} = (C, P)$ is said to be of *coarseness* β if each $c \in C$ is of coarseness β. We say that $\mathcal{M}$ is *coarse* if $\mathcal{M}$ is of coarseness β, for some $\beta > 0$. Notice that if $\mathcal{M}$ is coarse then the underlying transition system is finitely branching; however, the converse is not necessarily true. For instance, the Markov chain of Figure 4 is coarse (it is of coarseness 0.4), while the Markov chain of Figure 5 is not coarse.

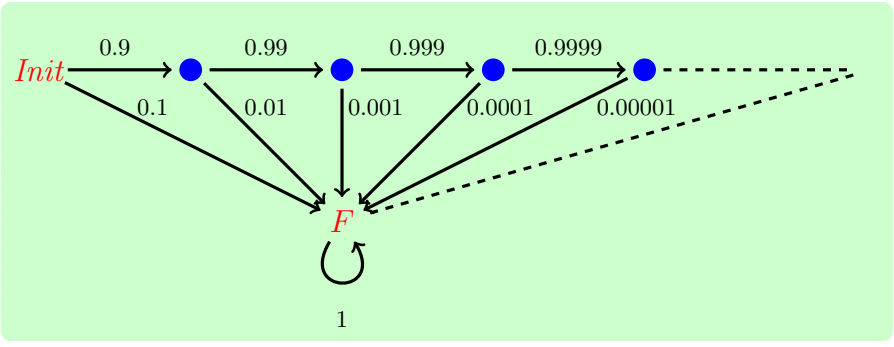

Fig. 5. Another Markov chain that is not decisive

A sufficient condition for decisiveness is the combination of coarseness and finite spanning. If a Markov chain $\mathcal{M}$ is both coarse and finitely spanning wrt. to set F of configurations then $\mathcal{M}$ is decisive wrt. F. The situation is illustrated in Figure 6 that shows a run in a Markov chain with coarseness 0.1 and span 3 (wrt. some F). The idea is that if we have a run ρ from which F is always

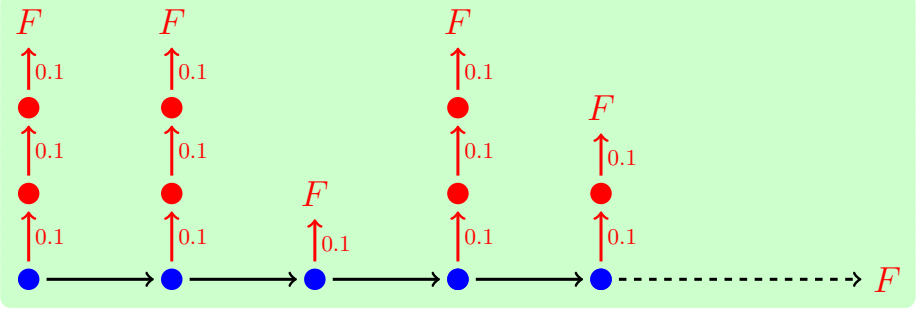

Fig. 6. A run in a Markov chain that is both coarse and finitely spanning

reachable, then from each configuration along the run, the probability of hitting F within the next 3 steps is at least 0.001. Therefore, the probability that ρ will avoid F forever is equal to 0. In other words, ρ will almost certainly eventually reach F.

5.2 Sufficient Condition II

An *attractor* $A \subseteq C$ is a set of configurations, such that each run of $\mathcal{M}$ will almost certainly eventually reach A. Figure 7 illustrates an attractor. Formally, for each $c \in C$, we have $Prob_c(\Diamond A) = 1$, i.e., the set A is reached from c with probability one.

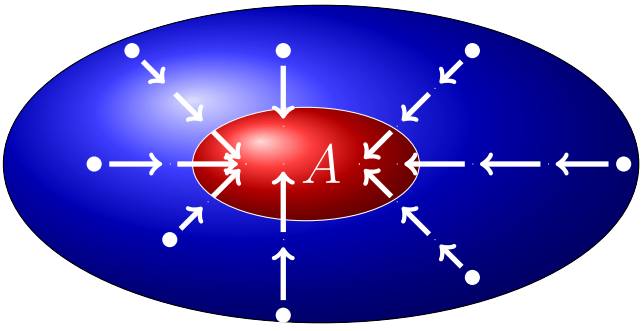

Fig. 7. An attractor

In fact, any run of the system will almost certainly visit A infinitely often. The reason (illustrated in Figure 8) is the following. Consider a run ρ. By definition of an attractor, ρ will almost certainly eventually reach a configuration $c_1 \in A$. We apply the definition of an attractor to the continuation of ρ from c_1. This continuation will almost certainly eventually reach a configuration $c_2 \in A$. The reasoning can be repeated infinitely thus obtaining an infinite sequence $c_1, c_2, \ldots$ of configurations inside A that will be visited. This means that A will be visited infinitely often with probability 1.

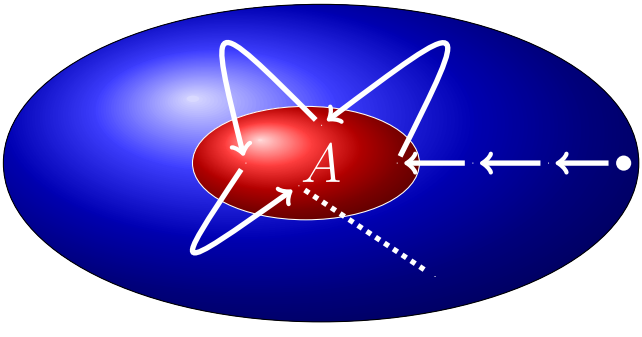

Fig. 8. Repeated reachability of an attractor

The existence of a finite attractor is a sufficient condition for decisiveness wrt. any set F of configurations. Assume a finite attractor A (see Figure 9). We partition A into two sets: $A_0 := A \cap \widetilde{F}$ and $A_1 := A \cap \neg\widetilde{F}$. In other words, the configurations in A_0 cannot reach F (in the underlying transition system), while from each configuration in A_1 there is a path to F. Consider a run ρ. We show that ρ will almost certainly eventually either reach $\widetilde{F}$ or reach F. We know that ρ will almost certainly visit A infinitely often. If ρ reaches F at some point then we are done. Otherwise, ρ will visit A_1 infinitely often with probability 1. Since

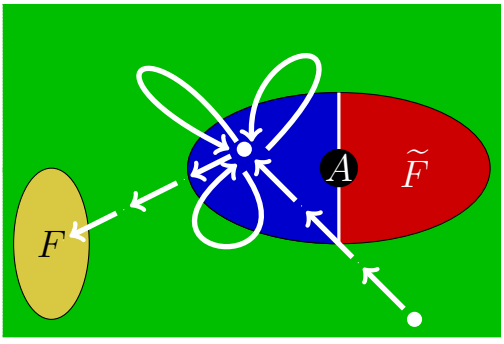

Fig. 9. Decisiveness due to a finite attractor

A_1 is a finite set, with probability 1, there is a configuration $c \in A_1$ that will be visited by ρ. By definition, we know that F is reachable from c, i.e., there is path (say of length k) from c to F. Let β be the probability that this path is taken during the next k steps of the run. This means that each time ρ visits c, it will reach F during the next k steps with probability at least β, which implies that ρ cannot avoid F forever. Thus ρ will almost certainly eventually reach F.

5.3 Probabilistic Petri Nets

To induce Markov chains from Petri nets, we associate weights (natural numbers) with the transitions of the net. If several transitions are enabled from a given configuration then a particular transition will be fired with a probability that reflects its weight relative to the weights of the rest of the enabled transitions.

For instance, Figure 10 shows a weighted version of the Petri net of Figure 1, where the transitions t_1 and t_2 have weights that are 3 and 2 respectively. Consider the configuration $c_1 = \left[\mathtt{L}, \mathtt{W}^3\right]$. From c_1 only transition t_1 is enabled. Therefore the probability of moving from c_1 to the configuration $c_2 = \left[\mathtt{C}, \mathtt{W}^2\right]$ is given by $P(c_1, c_2) = 1$, while $P(c_1, c_1') = 0$ for all other configurations c_1'. On the other hand, both t_1 and t_2 are enabled from the configuration $c_3 = \left[\mathtt{L}, \mathtt{C}, \mathtt{W}^3\right]$. Therefore, the probability of moving from c_3 to the configuration $c_4 = \left[\mathtt{C}^2, \mathtt{W}^2\right]$ is given by $P(c_3, c_4) = \frac{3}{2+3} = \frac{3}{5}$; while, for $c_5 = \left[\mathtt{L}, \mathtt{C}, \mathtt{W}^3\right]$, we have $P(c_3, c_5) = \frac{2}{2+3} = \frac{2}{5}$. In such a manner, we obtain an infinite-state Markov chain, where the configurations are those of the Petri net, and the probability distribution is defined by the weights of the transitions as described above. On the one hand, this Markov chain is finitely spanning wrt. an upward closed set F of configurations, since the underlying transition system is finitely spanning wrt. to F (as explained in Section 3). On the other hand, the Markov chains is coarse. In fact, the Markov chain is at least of coarseness $\frac{1}{w}$ where w is the sum of weights of all the transitions in the Petri net. It follows that the Markov chain induced by a Petri net is decisive wrt. any upward closed set F.

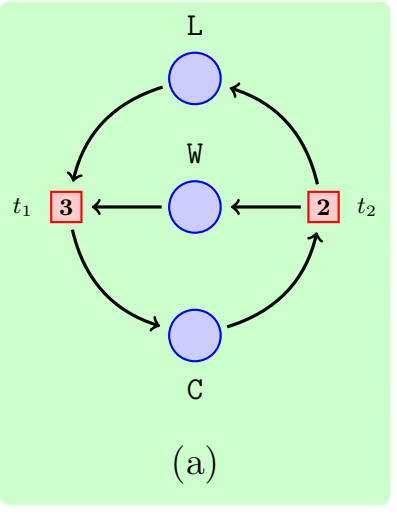

Fig. 10. A weighted Petri net

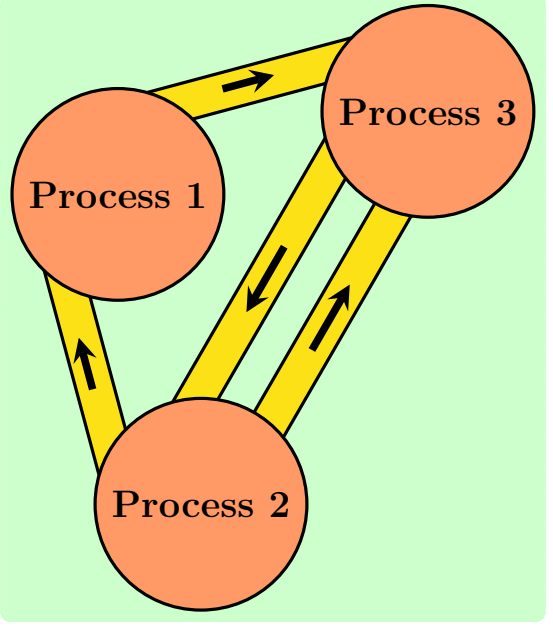

Fig. 11. Communicating finite-state processes

5.4 Communicating Processes

We consider systems that consist of finite sets of finite-state processes, communicating through unbounded channels that behave as FIFO queues (see Figure 11).

During each step in a run of the system, a process may either send a message to a channel (in which case the message is appended to the tail of the channel), or receive a message from a channel (in which case the message is fetched from the head of the channel). Choices between different enabled transitions are resolved probabilistically by associating weights to the transition in a similar manner to the case of Petri nets. Furthermore, after each step in a run of the system, a given message may be *lost* (removed from the buffer) by a predefined probability λ. The probability of loss is identical (equal to λ) for all messages that reside in the channels. The induced Markov chain is infinite-state, since the sizes of the buffers are not bounded. However, such a Markov chains always contains a finite attractor. In fact, the finite attractor is given by the set of configurations in which all the channels are empty. This is due to the fact that the more messages inside a buffer, the higher the probability that "many of" these message will be lost in the next step. Thus, each run of the system will almost certainly reach a configuration where all the channels are empty. Consequently, the induced Markov chain is decisive wrt. any set F of configurations.

6 Qualitative Analysis

In this section, we consider *qualitative analysis* of (infinite-state) Markov chains. We are given an *initial configuration Init* and a set of final (target) configurations F. In *qualitative reachability analysis*, we want to check whether $Prob_{Init}(\Diamond F) = 1$, i.e., whether the probability of reaching F from *Init* is equal to 1. In *qualitative repeated reachability analysis*, we want to check whether $Prob_{Init}(\Box\Diamond F) = 1$, i.e., whether the probability of repeatedly reaching F from *Init* is equal to 1. In the case of decisive Markov chains, we reduce the problems to corresponding problems defined on the underlying transition systems.

6.1 Reachability

For sets of configurations C_1, C_2 and a run $\rho = c_0 \longrightarrow c_1 \longrightarrow c_1 \longrightarrow \cdots$, we say that ρ satisfies C_1 $\underline{Before}$ C_2 if there is an $i \geq 0$ such that $c_i \in C_1$ and for all $j : 0 \leq j \leq i$ it is the case that $c_j \notin C_2$. Then, for a configuration c, we have $c \models \exists.\ C_1$ $\underline{Before}$ C_2 iff there is a c-run that reaches C_1 before reaching C_2. We will show that $Prob_{Init}(\Diamond F) = 1$ iff $Init \not\models \exists.\ \widetilde{F}$ $\underline{Before}$ F. One direction of the proof is illustrated in Figure 12. In fact, this direction holds for any Markov chain and is not dependent on the Markov chain being decisive.

Suppose that $Init \models \exists.\ \widetilde{F}$ $\underline{Before}$ F, i.e., there is an *Init*-run that reaches $\widetilde{F}$ before reaching F. This means that there is a prefix ρ' of ρ that will reach $\widetilde{F}$ before reaching F. The prefix ρ' has a certain probability β. Notice that the measure of all runs that are continuations of ρ' (that have ρ' as a prefix) is equal to β. Furthermore, all these continuations will not reach F (since ρ' visits $\widetilde{F}$ from which F is not reachable). This means that the measure of computations that will never reach F is larger than β, and hence the measure of computations that will reach F is smaller than $1 - \beta < 1$.

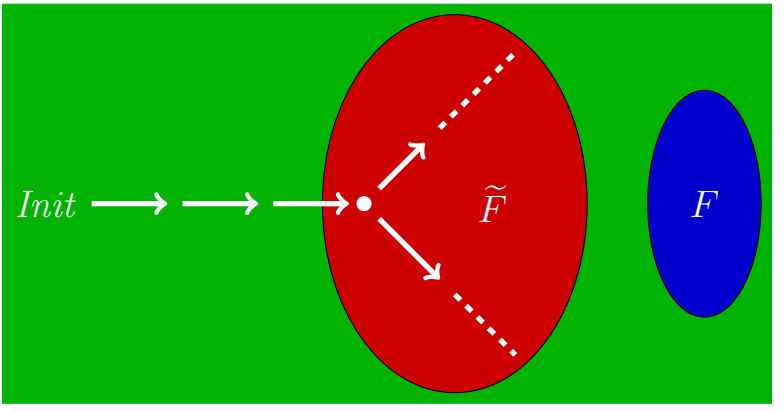

Fig. 12. Reaching $\widetilde{F}$ before F

The other direction of the proof, namely that $\mathit{Init} \not\models \exists.\ \widetilde{F}\ \underline{\mathit{Before}}\ F$ implies $\mathit{Prob}_{\mathit{Init}}(\Diamond F) = 1$ does not hold in general. As a counter-example, consider the Markov chain of Figure 4. In this example $\widetilde{F} = \emptyset$ since F is reachable from every configuration in the Markov chain. Therefore, $\mathit{Init} \not\models \exists.\ \widetilde{F}\ \underline{\mathit{Before}}\ F$ holds trivially. However, as mentioned in Section 5, we have $\mathit{Prob}_{\mathit{Init}}(\Diamond F) = \frac{2}{3} < 1$.

This direction of the proof holds for Markov chain that is decisive wrt. F (Figure 13). Consider any *Init*-run ρ. By decisiveness, ρ will almost certainly

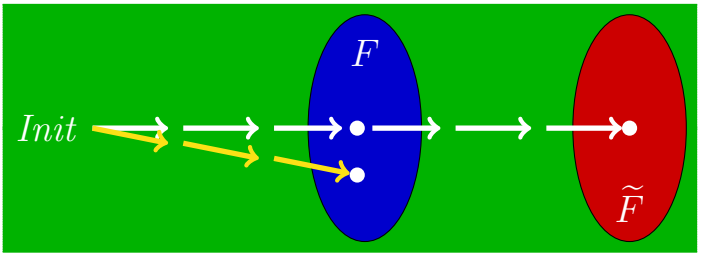

Fig. 13. Reaching F before $\widetilde{F}$

reach either F or $\widetilde{F}$. In the first case, the claim holds trivially. In the second case, since ρ visits $\widetilde{F}$ and since all runs must visit F before visiting $\widetilde{F}$, we know that ρ must have visited F (before visiting $\widetilde{F}$).

Thus, we have reduced the problem of checking whether $\mathit{Prob}_{\mathit{Init}}(\Diamond F) = 1$ in a Markov chain that is decisive wrt. F to that of checking whether $\mathit{Init} \models \exists.\ \widetilde{F}\ \underline{\mathit{Before}}\ F$ in the underlying transition system. In fact, as clear from the structure of the proof, the two problems are equivalent.

The problem of checking $Init \models \exists.\ \widetilde{F}\ \underline{Before}\ F$ is decidable for Petri nets in case F is given by a set of control states (we are asking about the reachability of a set of control states in a VASS). However, the problem is undecidable in case F is given by an arbitrary upward closed set of configurations. This is quite surprising since the control state reachability problem and upward closed set reachability problem are equivalent for all other models (whether probabilistic or not). The problem is also decidable for lossy channel systems which is the underlying transition system model for communicating processes (as described in Section 5).

6.2 Repeated Reachability

We will show that, for a configuration *Init* and a set F of configurations in a decisive Markov chain, we have that $Prob_{Init}(\Box\Diamond F) = 1$ iff $Init \models \forall\Box\ \exists\Diamond\ F$. The formula states that the set F remains reachable along all *Init*-runs. Notice that this is equivalent to $Init \not\models \exists\Diamond\ \widetilde{F}$. (it is not the case that there is an *Init*-run that leads to $\widetilde{F}$). Also, in this case, one direction of the proof (illustrated in Figure 12) holds for any Markov chain (it is not dependent on the Markov chain being decisive). Suppose that $Init \not\models \forall\Box\ \exists\Diamond\ F$ (i.e. $Init \models \exists\Diamond\ \widetilde{F}$). This means that there is a *Init*-run ρ that reaches $\widetilde{F}$. The run ρ is similar to the one shown in Figure 12 (with the difference that ρ is now allowed to visit F before visiting $\widetilde{F}$). In a similar manner to the case of reachability, there is a prefix ρ' that will reach $\widetilde{F}$ and that has a certain probability β. Furthermore, none of the continuations of ρ' will reach F. This means that the measure of computations that will not repeatedly visit F is smaller than $1 - \beta < 1$.

Also here, the other direction of the proof, namely that $Init \models \forall\Box\ \exists\Diamond\ F$. implies $Prob_{Init}(\Box\Diamond F) = 1$ does not hold in general. For instance in the Markov chain of Figure 4, we have $\widetilde{F} = \emptyset$. Therefore, $Init \models \forall\Box\ \exists\Diamond\ F$ holds. However, we have $Prob_{Init}(\Box\Diamond F) \leq Prob_{Init}(\Diamond F) = \frac{2}{3} < 1$. This direction of the proof holds for Markov chain that is decisive wrt. F (Figure 14). Consider any *Init*-run ρ. Since $\widetilde{F}$ is not reachable from *Init*, it follows by decisiveness that ρ will almost certainly reach some configuration $c_1 \in F$. We apply the definition of an attractor to the continuation of ρ from c_1. This continuation will almost certainly eventually reach a configuration $c_2 \in F$. The reasoning can be repeated infinitely thus obtaining an infinite sequence $c_1, c_2, \ldots$ of configurations inside the F that will be visited. This means that F will be visited infinitely often with probability 1.

We have then reduced the problem of checking whether $Prob_{Init}(\Box\Diamond F) = 1$ in a Markov chain that is decisive wrt. F to that of checking whether $Init \models \forall\Box\ \exists\Diamond\ F$ in the underlying transition system.

The problem of checking $Init \models \forall\Box\ \exists\Diamond\ F$ is decidable for Petri nets in case F is an arbitrary upward closed set of configurations. This is again surprising since it means that repeated reachability is a simpler problem than simple reachability (as we have seen, the former is decidable for upward closed sets while the latter is undecidable). The problem is also decidable for lossy channel systems.

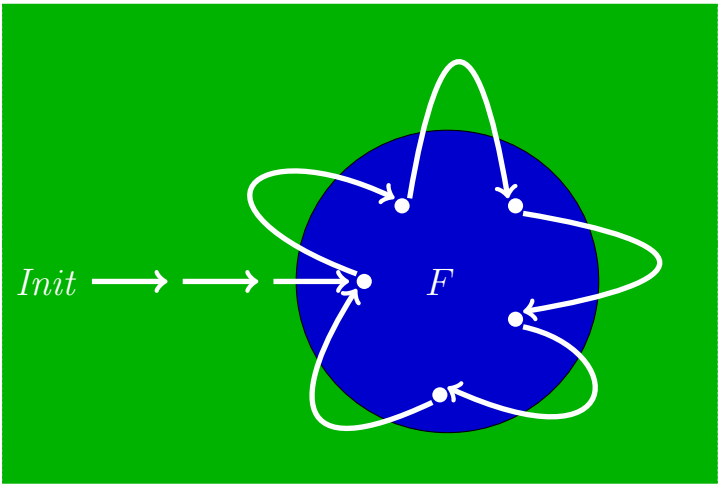

Fig. 14. Repeatedly reaching F

7 Quantitative Analysis

We give a sketch of a algorithm that computes the probability $Prob_{Init}(\Diamond F)$ up to an arbitrary given precision ϵ. The algorithm builds the reachability tree starting from *Init* as the root of the tree. It maintains two variables, namely the variable *yes* that under-approximates $Prob_{Init}(\Diamond F)$, and *no* that under-approximates $Prob_{Init}(\neg\Diamond F)$. Each time the algorithm picks a leaf from the tree (corresponding to a configuration c), it computes the successors of the node. For each successor c' such that $c' \notin F \cup \widetilde{F}$, it creates a child labeled with c' and labels the edge between the nodes by $P(c, c')$. If $c' \in F$ then it will close the node and increases the value *yes* by the weight the path from the root to the current node (equal to the product of the probabilities on the edges along the path). If $c' \in \widetilde{F}$ it will close the node and increases the value *no* analogously. The algorithm terminates when $yes + no \geq \epsilon$. The algorithm is guaranteed to terminate in case the Markov chain is decisive wrt. F since, as more and more steps of the algorithm are executed, the sum $yes + no$ will get arbitrarily close to one.

References

1. Abdulla, P.A.: Well (and better) quasi-ordered transition systems. Bulletin of Symbolic Logic 16(4), 457–515 (2010)
2. Abdulla, P.A., Čerāns, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: Proc. LICS 1996, 11th IEEE Int. Symp. on Logic in Computer Science, pp. 313–321 (1996)
3. Abdulla, P.A., Henda, N.B., Mayr, R.: Decisive markov chains. Logical Methods in Computer Science (2007); A Preliminary Version appeared in Proc. LICS 2005

4. Abdulla, P.A., Ben Henda, N., Mayr, R., Sandberg, S.: Eager markov chains. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 24–38. Springer, Heidelberg (2006)
5. Abdulla, P.A., Henda, N.B., Mayr, R., Sandberg, S.: Limiting behavior of Markov chains with eager attractors. In: D'Argentio, P.R., Milner, A., Rubino, G. (eds.) 3rd International Conference on the Quantitative Evaluation of SysTems (QEST), pp. 253–262 (2006)
6. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P.: Model-checking algorithms for continuous-time markov chains. IEEE Trans. Software Eng. 29(6), 524–541 (2003)
7. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. Journal of the ACM 42(4), 857–907 (1995)
8. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. Amer. J. Math. 35, 413–422 (1913)
9. Esparza, J., Kučera, A., Mayr, R.: Model checking probabilistic pushdown automata. In: Proc. LICS 2004, 20th IEEE Int. Symp. on Logic in Computer Science, pp. 12–21 (2004)
10. Etessami, K., Yannakakis, M.: Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 340–352. Springer, Heidelberg (2005)
11. Huth, M., Kwiatkowska, M.: Quantitative analysis and model checking. In: Proc. LICS 1997, 12th IEEE Int. Symp. on Logic in Computer Science, pp. 111–122 (1997)
12. Vardi, M.: Automatic verification of probabilistic concurrent finite-state programs. In: Proc. FOCS 1985, 26th Annual Symp. Foundations of Computer Science, pp. 327–338 (1985)