Reduction and Refinement Strategies for Probabilistic Analysis

Pedro R. D'Argenio[1], Bertrand Jeannet[2], and
Henrik E. Jensen[3] and Kim G. Larsen[3]

[1] FaMAF, Universidad Nacional de Córdoba
Ciudad Universitaria. 5000 - Córdoba. Argentina
dargenio@mate.uncor.edu
[2] IRISA – INRIA
Campus de Beaulieu. F-35042 Rennes Cedex, France
Bertrand.Jeannet@irisa.fr
[3] BRICS - Aalborg University
Frederik Bajers vej 7-E. DK-9220 Aalborg, Denmark
ejersbo@least.dk, kgl@cs.auc.dk

Abstract. We report on new strategies for model checking quantitative reachability properties of Markov decision processes by successive refinements. In our approach, properties are analyzed on abstractions rather than directly on the given model. Such abstractions are expected to be significantly smaller than the original model, and may safely refute or accept the required property. Otherwise, the abstraction is refined and the process repeated. As the numerical analysis involved in settling the validity of the property is more costly than the refinement process, the method profits from applying such numerical analysis on smaller state spaces. The method is significantly enhanced by a number of novel strategies: a strategy for reducing the size of the numerical problems to be analyzed by identification of so-called *essential states*, and heuristic strategies for guiding the refinement process.

1 Introduction

Fully automatic verification of a given (traditionally) finite-state system with respect to a given temporal logic property is known as *model checking*. For finite state systems, model checking has reached a clear level of maturity as witnessed by a number of successful industrial cases (e.g. [18,11]). Model checking of finite state systems allows settlement of qualitative properties such as "the system will never reach an erroneous situation". However, it is often vital that additional quantitative properties are established in order for the system to be considered correct. Such properties include real-time requirements such as "a desired state will be reached within 105 seconds" and probabilistic properties of the type "a desired state will be reached with probability at least 99%". While real-time model checking tools have been subject to significant research efforts leading to mature tools such as UPPAAL [24] and Kronos [8], it was not until recently that attention was drawn to efficient tool implementations for probabilistic model checking despite theoretical studies carried out during the last decade [14,2,7,19,4, etc.]. In this paper we report on a significant technical improvement on a recent tool of ours [9] to model check probabilistic properties.

In our context systems are described in terms of Markov decision processes [29], also called probabilistic transition systems (PTS) or probabilistic automata. This model allows to combine probabilistic and non-deterministic steps providing a natural extension to traditional non-deterministic models. The choice of this model is partly due to the fact that it is closed under parallel composition (which facilitates modeling and compositional reasoning), but primarily because PTSs are amenable to abstractions. This is a key factor for the techniques introduced in this paper.

We focus on a restricted class of reachability properties. These properties allow to specify that the probability of reaching a particular final condition ϕ_f from any reachable state satisfying a given initial condition ϕ_i is smaller (or greater) than a given probability p *regardless* of how non-deterministic choices of the model are resolved. Though apparently restrictive, the use of test-automata allow the range of properties that can be specified to be broadened substantially.

Our method [9] is based on automatic abstraction and refinement. The basic idea is to use abstractions in order to reduce the high cost of the numerical analysis involved in computing the minimum and maximum reachability probabilities for PTSs. The abstractions considered are obtained via successive refinements, starting from an initial coarse partitioning of the state space derived from the property under study. For a given refinement the property is checked on the induced abstract model, hopefully settling the property. However, the verdict may be inconclusive, when threshold probability p happens to be between the calculated minimum and the maximum abstract probabilities. In this case, the abstraction is further refined and the property checked again. This process is successively repeated until either the property is settled, or no further refinement is possible. To efficiently store the state space, perform abstractions and process the refinement steps, we use BDDs and MTBDDs (or ADDs) [9,12,3].

The performance of our method depends intimately on the efficiency of the numerical analysis performed on abstract models as well as the choice of refinements and initial partitioning. As main contributions of this paper we provide strategies for reducing the size of the numerical problems to be analyzed (by identification of so-called *essential* states) as well as strategies for guiding the refinement process. Finally, a number of comparative experimental results demonstrate the effectiveness of our method and suggested strategies.

The paper is organized as follows: Sections 2, 3, and 4 recall the theoretical foundation of the method and the implemented tool, which has been presented in [9]. Sections 5 and 6 present our reduction and refinement strategies. Section 7 provides details on implementation and experimental results. Section 8 concludes.

Related Work. Other quantitative model checkers have been developed. The tool PROBVERUS [15] allows to check the validity of a PCTL formula [14] on a (discrete time) Markov chain. PRISM [22,28] is a quantitative model checker for PCTL formulas on (discrete time) Markov decision processes, i.e., nondeterminism is inherent to the model, and on continuous-time Markov chains. Like PRISM, we also do model-checking on Markov decision processes, but we restrict to a particular kind of PCTL formula. Another quantitative model checker is $\mathsf{E} \vdash \mathsf{MC}^2$ [17], which model checks probabilistic timed properties on continuous-time Markov chains.

2 Probabilistic Transition Systems

Probabilistic transition systems (PTS for short) generalize the well-known transition systems with probabilistic information. In a PTS, a transition does not lead to a single state but to a probability space whose sample space is a set of states. The model we define is widely used (see, e.g. [30,7,21]) and is also known as Markov decision processes [29].

Let $\mathsf{Distr}(\Omega)$ denote the set of all probability distributions over the sample space Ω.

Definition 1 (Probabilistic Transition Systems). *A* probabilistic transition system *(PTS for short) is a structure* $T = (S, \rightarrow)$ *where* S *is a set of states, and* $\rightarrow \subseteq S \times \mathsf{Distr}(S)$ *is the transition relation. We write* $s \rightarrow \pi$ *for* $(s, \pi) \in \rightarrow$*. A PTS is said to be a* fully probabilistic transition system *(FPTS for short) if* $(s \rightarrow \pi \wedge s \rightarrow \rho) \Rightarrow \pi = \rho$*. A* rooted PTS (resp. FPTS) (T, s_0) *is a PTS (resp. FPTS) equipped with an initial state* $s_0 \in S$*. A PTS may be equipped with a proposition assignment* $p : S \rightarrow \mathcal{P}(\mathrm{AP})$*, where* AP *is a finite set of atoms and* $\mathcal{P}(\mathrm{AP})$ *the set of propositional formula on* AP*. We define* $\models \subseteq S \times \mathcal{P}(\mathrm{AP})$ *by* $s \models g$ *iff* $p(s) \Rightarrow g$ *is a tautology.*

We write $s \rightarrow$ whenever there is a π such that $s \rightarrow \pi$; otherwise, we write $s \not\rightarrow$. We let $\mathrm{supp}(\pi) = \{s' \mid \pi(s') > 0\}$. We call s a *sink state* if $s \not\rightarrow$.

Figure 1 shows (a symbolic representation) of an example PTS $T = (S, \rightarrow)$ where $S = \{\mathsf{i}, \mathsf{f}\} \times \{0, 1, 2\}$. An example transition is $(\mathsf{i}, 0) \rightarrow \{(\mathsf{i}, 1) \mapsto 0.75, (\mathsf{f}, 0) \mapsto 0.25\}$. T is nondeterministic as exhibited by the additional transition $(\mathsf{i}, 0) \rightarrow \{(\mathsf{i}, 1) \mapsto 0.25, (\mathsf{f}, 0) \mapsto 0.75\}$ originating from state $(\mathsf{i}, 0)$. We can assume that T is equipped with a proposition assignment p such that $p(l, v) = (l \wedge x = v)$.

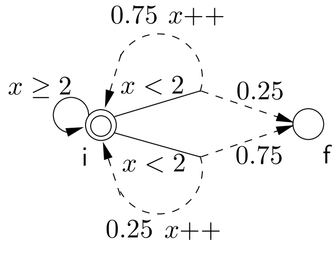

Fig. 1. A PTS

Let $T = (S, \rightarrow)$ be a PTS. A *simple path in* T is a finite sequence of states $\sigma = s_0 s_1 s_2 \ldots s_n$, where for each $0 \leq i < n$ there exists $\pi_i \in \mathsf{Distr}(S)$ such that $s_i \rightarrow \pi_i$ and $\pi_i(s_{i+1}) > 0$. Let $\sigma(i)$ denote the state in the i-th position. Let $|\sigma|$ be the length of σ and let $\mathit{first}(\sigma) = \sigma(1)$ and $\mathit{last}(\sigma) = \sigma(|\sigma|)$. A *simple path starting from* $s \in S$ is a simple path σ with $\sigma(1) = s$. A state t is *reachable* from another state s in T if there is a simple path in T with $s = \mathit{first}(\sigma)$ and $t = \mathit{last}(\sigma)$.

A *full path in* T is a sequence of states σ being either a simple path with $\mathit{last}(\sigma) \not\rightarrow$, or an infinite sequence. We denote by $\mathit{s\text{-}paths}(T)$ and $\mathit{f\text{-}paths}(T)$ the sets of simple paths and fullpaths in T, and by $\mathit{s\text{-}paths}(T, s)$ and $\mathit{f\text{-}paths}(T, s)$ the sets of simple paths and fullpaths in T starting from s. Let $\mathrm{reach}(T, s)$ denote the set of all states reachable from s in T.

We now define a probability measure on the full paths of a FPTS F. For any simple path $\sigma \in \mathit{s\text{-}paths}(F)$, define $\sigma^{\uparrow} = \{\pi \in \mathit{f\text{-}paths}(F) \mid \sigma \leq \pi\}$ where $\leq$ is the classical prefix order on sequences. Let $\mathcal{F}(F)$ be the smallest σ-field on $\mathit{f\text{-}paths}(F)$ which contains $\sigma^{\uparrow}$ for each $\sigma \in \mathit{s\text{-}paths}(F)$. Then for any state s of F, $\mathsf{P}_{F,s}$ is the uniquely defined probability measure on $\mathcal{F}(F)$ such that for any $\sigma = s_0 s_1 \ldots s_n \in \mathit{s\text{-}paths}(F)$ such that $s_i \rightarrow_F \pi_i$ for all i, $0 \leq i < n$:

$$\mathsf{P}_{F,s}(\sigma^{\uparrow}) \quad \triangleq \quad \textbf{if } (s = s_0) \textbf{ then } \pi_0(s_1) \cdot \pi_1(s_2) \cdot \ldots \cdot \pi_{n-1}(s_n) \textbf{ else } 0$$

We will write $\mathsf{P}_{F,s}(\sigma)$ to denote $\mathsf{P}_{F,s}(\sigma^{\uparrow})$. Intuitively, $\mathsf{P}_{F,s}(\sigma)$ is the probability of σ in F starting from s.

Any given PTS T defines a set of *probabilistic executions*, each one obtained by iteratively scheduling one of the possible post-state distributions from each pre-state, starting from a given state $s_0 \in S$. Notice that the same state s of T may occur more than once during a probabilistic execution and each time a different distribution from s may be scheduled. In order to distinguish such occurrences we include in all states s of a probabilistic execution the past history of s which is the unique path leading from the start state to s. Thus, a probabilistic execution essentially defines a finite or infinite *tree*.

Definition 2 (Probabilistic Execution). *A* probabilistic execution *of a PTS* $T = (S, \rightarrow_T)$ *is a FPTS* $F = (\mathit{s\text{-}paths}(T), \rightarrow_F)$ *such that* $(q \rightarrow_F \rho) \Leftrightarrow (\exists \pi : \mathit{last}(q) \rightarrow_T \pi \wedge \forall s \in S : \rho(qs) = \pi(s))$

We denote by $\mathit{execs}(T, s_0)$ the set of all probabilistic executions of T rooted in s_0.

3 Computing Extremum Probabilities

For a given rooted PTS (T, s_0) we are interested in the extremum probabilities of reaching some final condition from a given initial condition. For any given formula $\phi \in \mathcal{P}(\mathrm{AP})$ we define the set of all minimal simple paths of T that end in a state satisfying condition ϕ as:

$$\Sigma_{\phi}^{T} \triangleq \{\sigma \in \mathit{s\text{-}paths}(T) \mid \mathit{last}(\sigma) \models_T \phi \ \wedge\ \forall i, 0 < i < |\sigma| : \sigma(i) \models_T \neg\phi\}$$

By recording history information in states, the above set characterizes uniquely a set of simple paths of probabilistic executions of T. We also use Σ_{ϕ}^{T} to denote this alternative characterization. It should be clear from the context which alternative is used. We omit T in the notation whenever clear from context.

Definition 3 (Extremum Probabilities). *The* minimum *and* maximum probabilities *of reaching a final condition ϕ_f from an initial condition ϕ_i in a rooted PTS (T, s_0) equipped with a proposition assignment are defined respectively by*

$$\mathsf{P}^{\inf}_{T,s_0}(\phi_i, \phi_f) \triangleq \inf \big\{\mathsf{P}_{F,s}(\Sigma_{\phi_f}) \mid s \in \mathrm{reach}(T, s_0) \wedge s \models \phi_i \wedge (F, s) \in \mathit{execs}(T, s)\big\} \tag{1}$$

$$\mathsf{P}^{\sup}_{T,s_0}(\phi_i, \phi_f) \triangleq \sup \big\{\mathsf{P}_{F,s}(\Sigma_{\phi_f}) \mid s \in \mathrm{reach}(T, s_0) \wedge s \models \phi_i \wedge (F, s) \in \mathit{execs}(T, s)\big\} \tag{2}$$

We talk of an extremum probability *to refer to either the infimum or supremum probability.*

We use the shorthand $\mathsf{P}^{\inf}(s)$ and $\mathsf{P}^{\sup}(s)$ for $\mathsf{P}^{\inf}_{T,s_0}(s = s_0, \phi_f)$ and $\mathsf{P}^{\sup}_{T,s_0}(s = s_0, \phi_f)$, respectively. We denote by I and F the sets of states satisfying ϕ_i and ϕ_f, respectively. Our aim is to efficiently compute $\mathsf{P}^{\inf}(I) \triangleq \inf_{s\in I} \mathsf{P}^{\inf}(s)$ and $\mathsf{P}^{\sup}(F) \triangleq \sup_{s\in I} \mathsf{P}^{\sup}(s)$.

Consider again the PTS T of Figure 1. Take the initial condition $\phi_i = (\mathsf{i} \wedge x = 0)$ (which correspond to the initial state) and the final condition $\phi_f = \mathsf{f}$. It is easy to see that $\mathsf{P}^{\inf}(\phi_i, \phi_f)$ is obtained by always resolving the nondeterminism from state i in favor of the upper transition. Thus, $\mathsf{P}^{\inf}(\phi_i, \phi_f) = 0.25 + (0.75 \cdot 0.25) = 0.4375$. Analogously, $\mathsf{P}^{\sup}(\phi_i, \phi_f)$ is obtained by always resolving in favor of the lower transition and thus $\mathsf{P}^{\sup}(\phi_i, \phi_f) = 0.75 + (0.25 \cdot 0.75) = 0.9375$.

The equations 1 and 2 of definition 3 define extremum probabilities, but do not provide an effective way of computing them. However, it is well known [7,5] that $\mathsf{P}^{\inf}$ and $\mathsf{P}^{\sup}$ can be characterized as the least fixpoints of operators $F^{\inf}, F^{\sup} : (S \to [0,1]) \to (S \to [0,1])$ defined as follows. If $s \in F$ then $F^{\inf}(f)(s) = F^{\sup}(f)(s) = 1$. If $s \notin F$ then

$$F^{\inf}(f)(s) = \min_{s \to \pi} \sum_{s' \in S} \pi(s') \cdot f(s') \quad \text{and} \quad F^{\sup}(f)(s) = \max_{s \to \pi} \sum_{s' \in S} \pi(s') \cdot f(s') \tag{3}$$

Based on the above equations, two methods have been explored to compute $\mathsf{P}^{\inf}(s)$ and $\mathsf{P}^{\sup}(s)$. One can either compute the least fixpoints by iterative methods, or the equations can be transformed into a linear optimization problem that can be solved using classical techniques of linear programming. In this work we choose the linear programming method.

We use a standard precomputation of certain sets of system states in order to simplify the system before applying linear programming techniques. These sets are: the set of all reachable states *Reach*, and for each $p \in \{0,1\}$ the set of states having infimum (resp. supremum) probability p of reaching ϕ_f. These latter sets of states are denoted $\mathsf{P}^{\inf}_{=0}$, $\mathsf{P}^{\inf}_{=1}$, $\mathsf{P}^{\sup}_{=0}$, and $\mathsf{P}^{\sup}_{=1}$, respectively. All of the above sets can be computed using discrete fixpoint analysis [10] on a boolean abstraction of the system.

Based on the above precomputations our linear programming problems for computing $\mathsf{P}^{\mathrm{inf}}$ and $\mathsf{P}^{\mathrm{sup}}$ become as follows:

$$\text{maximize } \mathsf{P}^{\mathrm{inf}} \text{ under the constraints}\\ \begin{cases} \mathsf{P}^{\mathrm{inf}} \leq \mathsf{P}^{\mathrm{inf}}(s), & s \in I \\ \mathsf{P}^{\mathrm{inf}}(s) = 0, & s \in \mathsf{P}^{\mathrm{sup}}_{=0} \\ \mathsf{P}^{\mathrm{inf}}(s) = 1, & s \in (F \cup \mathsf{P}^{\mathrm{inf}}_{=1}) \\ \mathsf{P}^{\mathrm{inf}}(s) \leq \sum_{s' \in S} \pi(s') \cdot \mathsf{P}^{\mathrm{inf}}(s'), & s \to \pi, s \in S \setminus (\mathsf{P}^{\mathrm{sup}}_{=0} \cup \mathsf{P}^{\mathrm{inf}}_{=0} \cup F) \end{cases} \tag{4}$$

$$\text{minimize } \mathsf{P}^{\mathrm{sup}} \text{ under the constraints}\\ \begin{cases} \mathsf{P}^{\mathrm{sup}} \geq \mathsf{P}^{\mathrm{sup}}(s), & s \in I \\ \mathsf{P}^{\mathrm{sup}}(s) = 0, & s \in \mathsf{P}^{\mathrm{sup}}_{=0} \\ \mathsf{P}^{\mathrm{sup}}(s) = 1, & s \in F \cup \mathsf{P}^{\mathrm{sup}}_{=1} \\ \mathsf{P}^{\mathrm{sup}}(s) \geq \sum_{s' \in S} \pi(s') \cdot \mathsf{P}^{\mathrm{sup}}(s'), & s \to \pi, s \in S \setminus (\mathsf{P}^{\mathrm{sup}}_{=0} \cup \mathsf{P}^{\mathrm{sup}}_{=1} \cup F) \end{cases} \tag{5}$$

4 Simulations and Partitioning

Probabilistic simulation [20,30] is central to state the correctness of the abstraction technique proposed in this paper. For any $\delta \in \mathsf{Distr}(S \times S)$, $s \in S$ and $X \subseteq S$, $\delta(s, X)$ and $\delta(X, s)$ will denote resp. $\sum_{x\in X} \delta(s, x)$ and $\sum_{x\in X} \delta(x, s)$.

Definition 4 (Simulation). *Let $C \subseteq S \times S$ be a relation on states defining a* discrimination criterion. *A relation $R \subseteq S \times S$ is a C-*(probabilistic) simulation *if, whenever sRt,*

1. *$(s, t) \in C$, and*
2. *if $s \to \pi$, there exist ρ such that $t \to \rho$ and $\pi \sqsubseteq_R \rho$.*

*where $\pi \sqsubseteq_R \rho$ if there is $\delta \in \mathsf{Distr}(S \times S)$ such that for all $s, t \in S$, (i) $\pi(s) = \delta(s, S)$, (ii) $\rho(t) = \delta(S, t)$, and (iii) $\delta(s, t) > 0 \Rightarrow sRt$. s is C-*simulated *by t, notation $s \preceq_C t$, if there is a C-simulation R with sRt.*

Our interest is to check when a PTS reaches a goal ϕ_f starting from any state satisfying some initial condition ϕ_i ($\phi_i, \phi_f \in \mathcal{P}(\mathrm{AP})$). Let C_{ϕ_i,ϕ_f} be the discriminating criterion defined by

$$(s, t) \in C_{\phi_i,\phi_f} \iff (s \models \phi_f \Leftrightarrow t \models \phi_f) \wedge (s \models \phi_i \Leftrightarrow t \models \phi_i)$$

We write only C whenever ϕ_i and ϕ_f are clear from the context. Notice that C is equivalence relation. Simulation $\preceq_C$ provides a sufficient condition for preservation of extremum probabilities, as made precise by the following theorem.

Theorem 1. *Let (T_1, s^1_0) and (T_2, s^2_0) be two rooted PTSs such that none of them contains a sink state, and let $C = C_{\phi_i,\phi_f}$. Then $(T_1, s^1_0) \preceq_C (T_2, s^2_0)$ implies $\mathsf{P}^{\mathrm{sup}}_{T_1,s^1_0}(\phi_i, \phi_f) \leq \mathsf{P}^{\mathrm{sup}}_{T_2,s^2_0}(\phi_i, \phi_f)$ and $\mathsf{P}^{\mathrm{inf}}_{T_1,s^1_0}(\phi_i, \phi_f) \geq \mathsf{P}^{\mathrm{inf}}_{T_2,s^2_0}(\phi_i, \phi_f)$.*

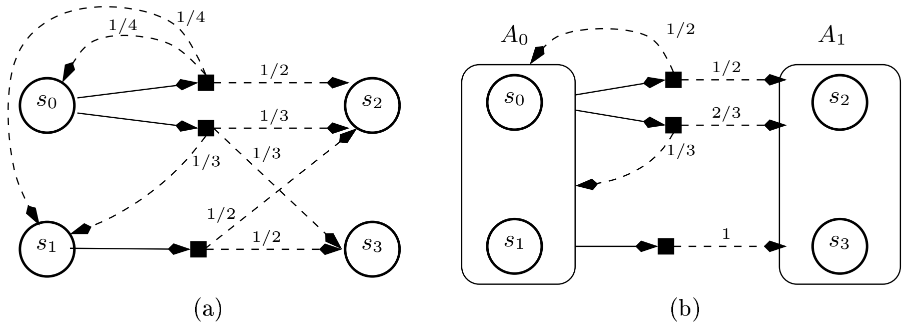

Fig. 2. A PTS and its quotient by the partition $\mathcal{A} = \{\{s_0, s_1\}, \{s_2, s_3\}\}$

The requirement that every state has a transition is not really harmful as each sink state can always be completed with a self-looping transition without affecting the properties of our interest on the original PTS.

We can abstract a PTS by partitioning its state space, and any such partitioning will induce an abstract PTS which will simulate the original (concrete) one. As a result extremum properties will be preserved by the abstract system.

Definition 5 (Quotient PTS). *Let $T = (S, \rightarrow_T)$ a PTS equipped with the proposition assignment p. Let $\mathcal{A} = (A_k)_{k \in K}$ be a partition of S. The* quotient PTS according to $\mathcal{A}$ *is the PTS $T/\mathcal{A} = (\mathcal{A}, \rightarrow_{\mathcal{A}}, p/\mathcal{A})$, where*

1. $A \rightarrow_{\mathcal{A}} \pi/\mathcal{A} \;\Leftrightarrow\; \exists s \in A : s \rightarrow \pi \,\wedge\, \forall A' \in \mathcal{A} : (\pi/\mathcal{A})(A') \triangleq \sum_{s' \in A} \pi(s')$, *and*
2. $p/\mathcal{A}(A) \triangleq \bigvee_{s \in A} p(s)$.

For a rooted PTS (T, s_0), its quotient is given by $(T, s_0)/\mathcal{A} \triangleq (T/\mathcal{A}, A)$ provided $s_0 \in A \in \mathcal{A}$.

Figure 2 gives an example of a quotient PTS. In the following we state the formal relationships between abstraction by partitioning and simulation. For any two partitions $\mathcal{A}$ and $\mathcal{B}$ of the same set, define $\mathcal{A} \leq \mathcal{B} \Leftrightarrow \forall A \in \mathcal{A}.\ \exists B \in \mathcal{B}.\ A \subseteq B$.

Theorem 2. *Let (T, s_0) be a rooted PTS with a set of states S and let C be an equivalence relation defining a partition $\mathcal{A}$ of S. Then for any partition $\mathcal{B}$ of S such that $\mathcal{B} \leq \mathcal{A}$, $(T, s_0)/\mathcal{B} \preceq_C (T, s_0)/\mathcal{A}$*

Notice the special case of the theorem where $\mathcal{B}$ partitions S into singleton sets. In this case $(T, s_0)/\mathcal{B}$ is isomorphic to (T, s_0). The following corollary states the relationship of abstraction by partitioning and preservation of extremum probabilities.

Corollary 1. *Let (T, s_0) be a rooted PTS equipped with a proposition assignment. Let ϕ_i and ϕ_f be the initial and final conditions. Let C be the equivalence relation $C = C_{\phi_i, \phi_f}$ defining a partition $\mathcal{C}$ of S. Then for any two partitions $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} \leq \mathcal{A} \leq \mathcal{C}$,*

$$\mathsf{P}^{\sup}_{(T,s_0)/\mathcal{B}}(\phi_i, \phi_f) \leq \mathsf{P}^{\sup}_{(T,s_0)/\mathcal{A}}(\phi_i, \phi_f) \quad \textit{and} \quad \mathsf{P}^{\inf}_{(T,s_0)/\mathcal{B}}(\phi_i, \phi_f) \geq \mathsf{P}^{\inf}_{(T,s_0)/\mathcal{A}}(\phi_i, \phi_f)$$

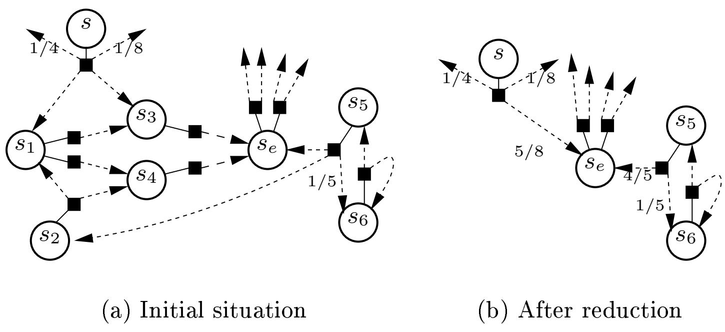

Fig. 3. Example of an essential state

5 Reduction Strategies

In this section we present new reduction strategies, that allow for simplifications of the linear programming problem characterizing the extremum probabilities of a PTS. First, we describe some optimizations which can be applied to the obtained linear programming problem. Then, we describe a new reduction technique which can be applied to a PTS before generating the corresponding linear programming problem. This reduction removes all but a particular type of essential states.

Our reduction techniques are orthogonal to our abstraction and refinement techniques and are therefore generally applicable as preprocessing steps for computing extremum probabilities of any PTS.

Optimizing Linear Programming Problems. Some optimizations are possible before solving an obtained linear programming problem. First, it is worth substituting the constant values 0 or 1 associated to special states. Notice that this a very simple case of Gaussian elimination and the substitution can make some inequations redundant; such inequations can be removed. Last and most important, when we obtain a single inequation of the form $\mathsf{P}^{\text{inf}}(s) \geq \ldots$ or $\mathsf{P}^{\text{sup}}(s) \leq \ldots$, we can replace the inequality by an equality, then perform Gaussian elimination, and finally remove new redundant inequations. Such transformations reduce the number of both variables and constraints.

Abstracting to Essential States. Suppose that a fragment of a PTS looks like the one depicted in Fig. 3(a). All probabilistic paths starting from s_1, s_2, s_3, s_4 are leading to the state s_e with probability 1 in a finite number of steps and therefore $\mathsf{P}^{\text{inf}}(s_i) = \mathsf{P}^{\text{inf}}(s_e)$ and $\mathsf{P}^{\text{sup}}(s_i) = \mathsf{P}^{\text{sup}}(s_e)$, for $i = 1, 2, 3, 4$. Thus, we could reduce the system by representing all of the above states via the single state s_e, and in addition merge (add) the probabilities for any distribution to enter the states represented by s_e. This analysis may at first seem quite identical to performing Gaussian elimination on the induced linear programming problem. However, this is not completely true. Besides reducing the size of the PTS, our analysis can also remove some non-determinism in the system, thus allowing to replace inequations by equations in the corresponding linear programming problem. In the figure on the right, we have $\mathsf{P}^{\inf}(s_1) \leq \mathsf{P}^{\inf}(s_3) = \mathsf{P}^{\inf}(s_e)$ and $\mathsf{P}^{\inf}(s_2) \leq \mathsf{P}^{\inf}(s_4) = \mathsf{P}^{\inf}(s_e)$. Gaussian elimination would remove $\mathsf{P}^{\inf}(s_3)$ and $\mathsf{P}^{\inf}(s_4)$, but would not replace the inequations on $\mathsf{P}^{\inf}(s_1)$ and $\mathsf{P}^{\inf}(s_4)$ by equations, thus limiting further eliminations.

Consider states s_5 and s_6 on Fig. 3(a). They also agree with state s_e on their infimum and supremum probabilities. However, due to the loop between s_5 and s_6, these properties appear only when taking into account infinite paths. The corresponding analysis is thus more expensive than the one in which only finite paths needs consideration. We consider only the analysis based on finite paths.

We now formalize the intuitive ideas of the previous paragraph. Let $T = (S, \rightarrow)$ be a PTS equipped with a set of final states F that are supposed to be sink. We define a *domination relation* $\preceq \subseteq S \times S$ with the intuition that $s_1 \preceq s_2$ whenever all probabilistic paths starting from s_1 will pass through s_2 with probability 1. This intuitive semantic definition can be characterized as a least fixpoint as follows.

Definition 6 (Domination Relation, Essential States, and Domination Equivalence). *Let $\preceq$ be the smallest relation satisfying the following. For all $s \in S$, $s \preceq s$, and for all $s, t \in S$ s.t. $s \neq t$,*

$$s \preceq t \;\Leftarrow\; (s \rightarrow) \wedge \forall \pi : (s \rightarrow \pi \Rightarrow (\forall s' \in \mathrm{supp}(\pi) : s' \preceq t)) \tag{6}$$

The relation $\preceq$ can be shown to be a partial order under the condition all states in $\mathsf{P}^{\sup}_{=0}$ have been removed. States that are maximal with respect to $\preceq$ are called essential states. *We let $\sim$ denote the relation induced from $\preceq$ as follows:*

$$s_1 \sim s_2 \;\Leftrightarrow\; \exists t : s_1 \preceq t \wedge s_2 \preceq t \tag{7}$$

In other words, states related by $\sim$ are dominated by the same state. Relation $\sim$ is an equivalence relation.

We will reduce a concrete PTS to an abstract one containing only essential states, and such that the abstract PTS preserves the exact extremum probabilities of the concrete PTS. We call the abstract PTS an *essential state abstraction*. To compute the essential state abstraction, we could try to use the quotient construction of Definition 5 with respect to the partition corresponding to the equivalence relation $\sim$. However, this will not guarantee an exact abstraction. Intuitively, the transitions linking states inside an equivalence class will generate loops on the abstract graph, and thus $\mathsf{P}^{\inf}$ may be lower than on the initial system. For states inside a class with a looping transition $\mathsf{P}^{\inf}$ will be 0.

However, by definition of $\sim$, we know that each equivalence class has a unique essential state. These essential states can be taken as abstract states representing all the equivalence classes. Also, any concrete distribution can be abstracted by merging (adding) all probabilities for states in the same equivalence class onto the essential state representing this class. The following definition formalizes this construction.

Definition 7 (Essential State Abstraction of a PTS). *Let $T = (S, \rightarrow)$ be a PTS equipped with a set of final states F supposed to be sink. Let $\mathcal{E}$ be the partition of S associated to $\sim$, and let $E \subseteq S$ be the set of essential states. For any $e \in E$ let $[e]$ denote the equivalence class of e in $\mathcal{E}$. The* essential PTS of T *is the PTS $T_\epsilon = (E, \rightarrow_\epsilon)$ with final condition $F \subseteq E$, and where $\rightarrow_\epsilon$ is defined by: $e \rightarrow_\epsilon \pi/\mathcal{E} \Leftrightarrow e \rightarrow \pi$ where $\pi/\mathcal{E} \in \mathsf{Distr}(E)$ is the distribution st. $\pi/\mathcal{E}(e) = \sum_{s \in [e]} \pi(s)$.*

Proposition 1 (Preservation of Extremum Probabilities). *Let $T = (S, \rightarrow)$ be a PTS equipped with a set of final states F that are sink, and let $T_\epsilon = (E, \rightarrow_\epsilon)$ be its essential abstraction. For $s \in S$ dominated by $e \in E$, we have:*

$$\mathsf{P}^{\inf}_{T}(s, F) = \mathsf{P}^{\inf}_{T_\epsilon}(e, F) \quad \textit{and} \quad \mathsf{P}^{\sup}_{T}(s, F) = \mathsf{P}^{\sup}_{T_\epsilon}(e, F)$$

Algorithm and Complexity. Given a PTS $T = (S, \rightarrow)$, we want to compute the essential elements of T, as well as their associated set of dominated elements. This is done by using the fixpoint definition of $\preceq$, and using the UNION-FIND data structure, with path compression and weighted union techniques [27]. The basic UNION-FIND data structure is an inverted tree (Fig. 4) where sons are pointing to their father and where the root represents both itself and the set of the nodes pointing to it. Figure 4 depicts the situation of the algorithm applied to the PTS of Fig. 3(a) after two steps. At that point we know that s_1, s_3, s_4 are dominated by s_e, and in the next step, s_2 will also be incorporated to the elements dominated by s_e, because all paths starting from s_2 lead to states that are dominated by s_e.

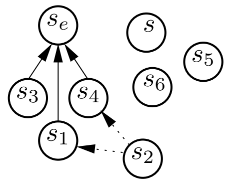

Fig. 4.

The only modification we have to bring to this algorithm is that we need to maintain, together with each tree, the essential state dominating all elements of the tree. Indeed, the root of a tree is not necessarily the essential state dominating all the elements of the tree, when weighted union is performed. According to [27], if m FIND operations and n UNION operations are performed, the global time to carry out these operations is $\mathcal{O}(n + m\alpha(m, n))$, where α is the inverse of the Ackerman function. Let us evaluate n and m in the algorithm sketched above. We have obviously $n \leq |S|$. For m, at each step, we do in the worst case $N \cdot D \cdot d$ FIND operations, where N is the number of equivalent classes, D is the maximum number of distributions outgoing from a state, and d the maximum size of their support. The maximum number of steps is $|S|$, and at each step we have $N \leq |S|$, so $m \leq |S|^2 \cdot D \cdot d$. As a result, an upper bound of the complexity of the algorithm is $\mathcal{O}(|S| + |S|^2 \cdot D \cdot d \cdot \alpha(|S|^2 \cdot D \cdot d, |S|))$. Notice that in practice the number of steps is likely to be much smaller than $|S|$.

6 Refinement Strategies

Our method of verification via abstraction follows the classical partition refinement scheme and it has been presented earlier in [9]. It starts with a coarse abstraction of the concrete system under investigation. If the analysis of the abstract system allows to conclude on the properties under investigation then the verification process is finished. Otherwise, a partition refinement step is performed in order to obtain more precise information. This process is iterated up to success or until all classes of the partition are stable.

This section reports on new results of ours to enhance the method of [9] with new strategies for constructing an initial partition, for refining individual classes, and for choosing intelligently the classes to refine.

Initial Partition. Before constructing the initial partition, we use the precomputations described in section 3 to simplify the system as follows: (1) We first restrict to the reachable (from the root) state space, since the definition of P^{inf} and P^{sup} involves only reachable states. (2) We then augment the set of final states F with $\mathsf{P}^{\text{inf}}_{=1}$, we compute $\mathsf{P}^{\text{sup}}_{=0}$ and we make all these states sink. (3) Finally, we restrict the state space to states reachable from the initial ones. Each of these transformations preserves extremum probabilities and can give large reductions in the state space.

Let $T = (S, \rightarrow)$ be a PTS on which the above preprocessing has been performed. Let I and F be the sets of states satisfying ϕ_i and ϕ_f. Since we want to compute safe approximations of $\mathsf{P}^{\text{inf}}_{T,s_0}(\phi_i, \phi_f)$ and $\mathsf{P}^{\text{sup}}_{T,s_0}(\phi_i, \phi_f)$, our initial partition should be at least a refinement of the partition $\{I,\, F,\, S \setminus (I \cup F)\}$. In addition, we distinguish the sets of states $\mathsf{P}^{\text{sup}}_{=0}$, $\mathsf{P}^{\text{inf}}_{=0}$ and $\mathsf{P}^{\text{sup}}_{=1}$ (the states $\mathsf{P}^{\text{inf}}_{=1}$ have already been merged with the final states). This is wise since we already have these sets of states from the precomputation, and since joining these states with other states will inevitably lead to a loss of precision in the extremum probabilities of the latter ones. In the case that the above sets are not disjoint, we make a proper partition.

We further refine the obtained partition according to the explicit control structure of our PTS, such that only the values of variables are abstracted. In our tool, this default behavior can be user modified by specifying the variables and the processes which should not be abstracted.

Refining Individual Classes. Our refinement method tries to stabilize classes in the standard way by splitting classes based on different futures for their contained sets of states. However, we allow for a more strategic splitting of a class than simply splitting it with respect to all its outgoing transitions.

Let T be the concrete PTS and let $T/\mathcal{A} = (\mathcal{A}, \rightarrow_{\mathcal{A}})$ be its quotient in the current refinement step. For any abstract transition $t : A \rightarrow_{\mathcal{A}} \Pi$ where $A \in \mathcal{A}$ and $\Pi \in \mathsf{Distr}(\mathcal{A})$, we define the *guard* of t as: $g(t) = \{s \in A \mid \exists s \rightarrow_T \pi.\ \forall i.\ \pi(A_i) = \Pi(A_i)\}$ where $\pi(A) = \sum_{s \in A} \pi(s)$. Now, if $g(t) \neq \emptyset$ and $g(t) \neq A$, then the class A can be split into two new classes: $g(t)$ and $A \backslash g(t)$.

For a given class A our method allows different strategies for choosing the transitions $A \rightarrow_{\mathcal{A}} \Pi$ to serve as basis for splitting A.

1. *Binary splitting:* This means that we do not split A with respect to all outgoing transitions, but we choose one particular transition. Candidates are transitions $A \rightarrow_{\mathcal{A}} \Pi$ where

(a) $\Pi(A) = 1$: Splitting with respect to these transitions is an attempt to raise the abstract infimum probabilities. If $\Pi(A) = 1$ then at least class A will have infimum probability 0. Transitions of this type are denoted L-transitions (L for looping).
(b) $\exists B \neq A.\ \Pi(B) = 1$: Splitting with respect to these transitions is an attempt to lower the abstract supremum probabilities. If $\Pi(B) = 1$ and the supremum probability of B is 1 then so is the supremum probability of A. Transitions of this type are denoted S-transitions (S for single).
(c) neither of the above hold. Transitions of this type are denoted O-transitions (O for other).

2. *N-ary splitting:* This means that we split A with respect to all its outgoing transitions, or to all the transitions of exactly one of the above types.

These types of splitting will be discussed in section 7.

Choosing Classes to Refine. We discuss here how we choose the classes to be refined. Our standard strategy tries to split once every class for which there exist a splitting guard, using the strategies from above to decide which transition to use as the basis for the split. However, we can do better if we take advantage of the dominance relation (section 5). Indeed, if $A \preceq A_e$ then any finite probabilistic path starting from A leads with probability 1 to A_e. As a consequence, refining class A without refining class A_e will not change the computed values of $\mathsf{P}^{\inf}(A)$ and $\mathsf{P}^{\sup}(A)$. For that reason, we allow the possibility to refine only essential classes in an abstract PTS.

Tradeoff between Refinement and Analysis. Stopping the refinement process as soon as possible, i.e. with the fewest number of classes enabling the proof of the property or its negation, requires to split only one class at a time, and then to perform an analysis to check the property. However, as probabilistic analysis is rather expensive, even on an abstract system, it should not be applied too often.

Our technique is to refine a partition $\mathcal{A}$ into a partition $\mathcal{A}'$ such that $\frac{|\mathcal{A}'|}{|\mathcal{A}|} \geq r$, with r specified by the user. A small value of r (close to 1) produces frequent analysis, whereas a large value favorizes refinement over analysis. If the goal is to produce the minimal stable partition of the concrete PTS, a infinite ratio allows to obtain it without any intermediate probabilistic analysis.

7 Implementation and Experiments

A tool called RAPTURE and based on the principles presented in this paper has been implemented. Its architecture (see Fig. 5) is the following: (1) the front-end parses the input language, that specifies both the system to be analyzed, the property and possibly the components (processes and variables) not abstracted in the initial abstraction. The output is a symbolic representation of the system (i.e., the probabilistic transition function and sets of root, initial and final states). (2) Boolean analysis is then performed. If it allows to prove or disprove the

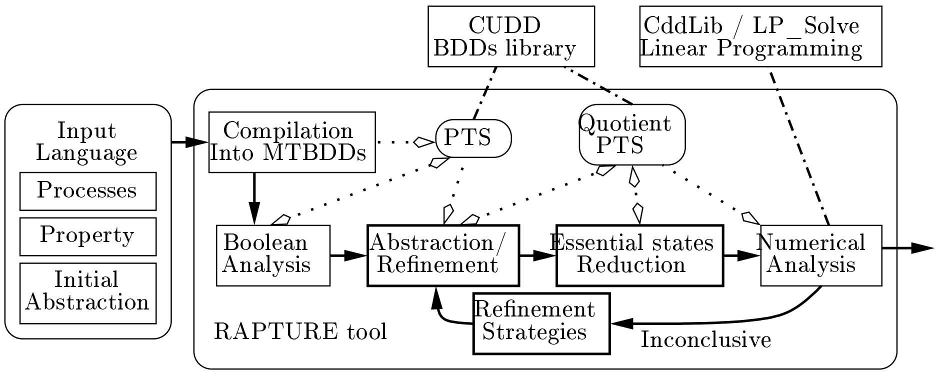

Fig. 5. Architecture of the RAPTURE tool

property, the verdict is emitted. (3) Otherwise, the initial abstraction is build, and the verification process alternating numerical analysis and refinement steps starts. The thick boxes on figure 5 indicate the new or updated modules since the prototype reported in [9].

As stated before, we use linear programming to compute extremum probabilities. Because of precision problems arising with some case studies, we offer the following possibilities in our tool :

– Use of the sparse matrix based solver LP_SOLVE [6], with coefficients being ordinary real numbers (1);
– Use of the dense matrix based solver CDDLIB [13], with coefficients being either exact rational numbers (2), multi-precision floating point numbers (3), or ordinary real numbers (4).

The possibilities that give the best results are (1) and (2). (1) is better whenever there is no precision problems, because it uses sparse matrices and our LP problems are sparse, whereas (2) is very useful when precision problems arise and/or when *exact results* are wanted.

Experiments. We have conducted several experiments in order to evaluate our reduction and refinement strategies as well as our implementation. The aim is to analyze the practical usefulness of our reduction strategies and the efficiency of our refinement strategies.

Our first case study is the Bounded Retransmission Protocol (BRP) [16]. The BRP is based on the well-known alternating bit protocol but allows for a bounded number of retransmissions of a chunk, i.e., part of a file, only. So, eventual delivery is not guaranteed and the protocol may abort the file transfer. We use the version presented in [9], where probabilities model the possible failures of the two channels used for sending chunks and acknowledgments, respectively. In Table 1 we check the maximum probabilities that the sender does not report a successful transmission. We consider a file composed of either 16 or 64 chunks, and N is the number of allowed retransmissions. We have to use here the dense matrix based solver with exact arithmetic, because probabilities of very different magnitude order appear in LP problems, which makes the usual floating point arithmetic unusable. The initial partitioning is here performed w.r.t. the explicit control structure of the specification: only variables are abstracted.

The meaning of row labels in the table is the following: `#reach` is the number of states reachable from root states, `#rel` is the number of relevant states after Boolean preprocessing, and `time` is the time needed for building and preprocessing. The three next sets of rows details the refinement process for different upper bounds for P^{sup}. `#refin` is the number of refinement steps, `#abst` the number of states of the most refined abstract PTS, `#ess` the number of its essential states, `psup` the computed probability, `verd` is the verdict (true or false), and `time(a+r)` gives the time spent in numerical analysis and in refinement process. When the verdict is false, the refinement has gone to the stable partitioning of the PTS and gives the actual P^{sup} of the concrete PTS.

The first observation is that Boolean preprocessing is here very efficient to reduce the state space: the reduction is one third in average. Second, the essential state reduction allows again a reduction of one third. The third observation is that it is nearly as easy to prove $\mathsf{P}^{\text{sup}} \leq 10^{-3}$ for big instances of BRP than for small ones: that means that the refinement strategy works well and will not perform too many useless splits. It can also be observed that checking smaller upper bounds can still be performed on very small abstract PTSs, compared to the concrete one, even reduced by preprocessing, and also compared to the stable partitioning (row $\mathsf{P}^{\text{sup}} \leq 10^{-90}$). Last, as we try to check bigger instances, time spent in refinement is much smaller than time spent in analysis, if we refine up to stabilization. Again, this shows the relevance of our method. We do not illustrate here the effect of the different options for refinement, because they give equivalent results on this example.

Our second example is the Probabilistic Dining Philosopher from [25], studied by [26] and analyzed by the the PRISM team [28]. In this example N philosophers are trying to eat and we want to prove a lower bound on the probability for some process to eat after a number of time units specified by value of deadline. As the philosophers perform asynchronous moves, we add the following *bounded fairness constraint*: a philosopher cannot stay idle for more than K steps[1]. Table 2 shows results for $N = 3$ and different values of K. The chosen deadline corresponds to the smallest one for which the property holds with a probability more than 0. We try to prove successively $\mathsf{P}^{\text{inf}} \geq \frac{1}{16}$ and $\mathsf{P}^{\text{inf}} \geq \frac{1}{8}$ (this last bound is the real one, according to [28]). Compared to the previous case study, the number of states is much bigger, as well as the BDDs and MTBDDs representing respectively sets of states and the transition relation. We give in the table not only the number of abstract and essential states, but also in each case the number of abstract distributions. We use here the sparse matrix based solver with ordinary

[1] Without the fairness constraint, the lower bound is zero and can be checked by the discrete fixpoint computations mentioned in section 3.

Table 1. Results in BRP

		file length 16				file length 64
	MAX	2	4	8	15	15
	#reach.	3908	6060	10364	17896	58024
	#relev.	1014	1790	3342	6058	26362
	time	0.94	1.10	1.59	1.75	5.20
$\leq 10^{-3}$	#refin.	4	5	6	6	6
	#abst.	52	89	161	161	161
	#ess.	24	42	85	85	85
	psup	3.09e-04	4.27e-06	7.89e-06	7.89e-06	7.89e-06
	verd.	T	T	T	T	T
	time(a+r)	0.07+0.88	0.72+0.83	2.96+1.68	2.95+1.72	2.97+2.62
$\leq 10^{-10}$	#refin.	9	10	7	7	7
	#abst.	375	675	242	247	247
	#ess.	152	272	108	115	115
	psup	2.65e-05	2.35e-08	7.01e-12	7.01e-12	7.01e-12
	verd.	F	F	T	T	T
	time(a+r)	0.58+2.56	3.30+5.42	3.15+2.07	3.54+2.33	3.51+3.55
$\leq 10^{-90}$	#refin.	9	10	11	12	16
	#abst.	375	675	1275	2325	9765
	#ess.	152	272	512	932	3908
	psup	2.65e-05	2.35e-08	1.85e-14	3.87e-25	3.87e-25
	verd.	F	F	F	F	F
	time(a+r)	0.58+2.56	3.30+5.42	15.87+11.00	186.58+22.06	1209.72+165.3

floating point arithmetic. We choose as the initial partition the one obtained by abstracting everything but the counter used for the deadline, as it is clear the value of the deadline is of fundamental importance for the studied property. Most of the encouraging observations made for the BRP are still true. The only exception is that essential state reduction is not very useful here. Execution times are much higher, because the abstract PTSs are much more complex, and the corresponding LP problems are very big. Still, refinement remains much cheaper than analysis, and state space reduction between the concrete PTS and the abstract one allowing to prove the property is impressive.

Table 3 shows verification of this case study for $K = 3$, with various refinement options and initial control structures. The first column corresponds to the options that work best and that were used in the previous table: the initial partition detail only the counter for the deadline, and we use n-ary division, giving priority to respectively O S and L types of probabilistic transitions, as described in section 6. Using binary divisions gives similar results (second column). Column 3 shows that inverting the priority of the different types of split in column 1 gives very bad results: a much more refined system is needed to prove the property. This is quite counter-intuitive (cf. our remark in section 6) but has been observed on nearly all case studies we performed. We conjecture that splitting a abstract state according to its looping transition most often leads to an unbal-

Table 2. Results in Dining Philosophers with $N = 3$

	K	4	5	6
	deadline	23	27	31
	#reach.	1.00e06	1.97e06	3.40e06
	#relev.	121041	271287	488859
	time	14.4	23.6	34
$\geq \frac{1}{16}$	#refin.	5	7	8
	#abst.	3064/11536	16903/52435	35780/111084
	#ess.	2778/11250	14442/49974	30361/105665
	pinf	0.0625	0.0625	0.0625
	verd.	T	T	T
	time(a+r)	49.6+79.5	2120+590	10353+1462
$\geq \frac{1}{8}$	#refin.	7	8	9
	#abst.	8512/22757	21011/59866	37542/114703
	#ess.	6668/20913	16996/55851	31656/108817
	pinf	0.125	0.125	0.125
	verd.	T	T	T
	time(a+r)	290+220	3683+712	20335+1575

anced division, that separates a few concrete states from a huge set of concrete states. Column 4 corresponds to the strategy where we fully stabilize all abstract states wrt. the current partition. Last, column 5 illustrates the importance of a good initial partition. Here, we generated it according to the explicit control structure of the philosopher, and it produces very bad result.

We also tried the IEEE FireWire root contention protocol [32,31], using the model developed by [28]. The property we want to prove is that a leader (root) is chosen before the time bound deadline is reached with some probability. Results are depicted in table 4, where we refine until we reach the probability given in row `pinf`. Here Boolean analysis is very expensive, because counters involved in the model make the BDDs huge ($\sim 4 \cdot 10^5$ nodes) and the number of iterations is also big ($> 200, 300, 400$). For deadline=200, Boolean analysis allows to show that $\mathsf{P}^{\mathrm{inf}} = 0$. The table shows also that the number of relevant states is here really smaller than the number of reachable states. That comes from the fact that at least all the states corresponding to deadline less than 200 satisfy the property with probability 0 and are removed, as shown by column 1.

8 Conclusion

In this paper we have introduced new efficient strategies for model checking quantitative reachability properties of Markov decision processes. The fundamental method of our approach is based on automatic abstraction and refinement [9], where properties are analyzed on abstractions rather than on the original system. The abstractions are safe with respect to the property under consideration and moreover they are expected to have significantly smaller state spaces than

Table 3. Results in Dining Philosophers with $N = K = 3$ and deadline $= 19$

	control option	deadline nary+osl	deadline bin+osl	deadline nary+lso	deadline nary+a	ctrl. struct. nary+osl
	#reach.	408397				
	#relev.	30018				
	time	6.14				
$\leq \frac{1}{4}$	#refin.	2	2	4	2	4
	#abst.	51/87	35/122	882/2880	140/680	5861/12972
	#ess.	51/87	35/122	827/2825	140/680	4109/11196
	pinf	0.25	0.25	0.25	0.25	0.25
	verd.	T	T	T	T	T
	time(a+r)	0.03+3.85	0.02+3.48	5.16+16.79	0.19+5.09	53.6+44.8

Table 4. Results in FireWire

deadline	200	300	400	400
#reach.	6.8e6	2.3e07	4.4e07	4.4e07
#relev.	-	21	129661	129661
time	512	3240	8018	8018
#refin.	-	2	2	11
#abst.	-	6/4	11/15	530/1295
#ess.	-	5/3	5/7	71/178
pinf	0.0	0.5	0.5	0.625
time(a+r)	-	0.01 + 0.05	0.01+39	0.26+104

their corresponding concrete system. If an abstraction cannot prove or disprove a considered property, the abstraction is refined and the analysis is repeated on the refined system.

The overall performance of our method depends crucially on the efficiency of the numerical analysis performed on abstract systems as well as on the choice of refinement method. A main contribution of this paper has been the development of strategies for reducing the size of the numerical problems to be analyzed as well as strategies for guiding the refinement process.

Our experiments have shown that our reduction and refinement strategies are relevant for several case studies of various type, and the method of abstraction and refinement allows considerable simplifications. This is especially true when the property of interest does not require the computation of exact probabilities, but also holds in general.

It is worth to mention that the techniques reported in this article considerably improved the performance of our tool with respect to the prototype reported in [9]. We highlight that the processing speed has increased around 3 orders of magnitudes: whereas checking the BRP (MAX=4, length=16) for unsuccessful transmission with probability $\leq 10^{-5}$ took about half an hour, now it only takes around 2 seconds. There are two fundamental reasons. The first one is due to the fact that the new implementation is smarter on doing the refinements and therefore less number of steps are required to reach the result. Second, the LP problem in [9] was constructed with one variable per abstract state and one inequality per transition on this abstract representation. Now, instead, considering a variable per essential state reduces the number of variable to the 50% (in the BRP, see Table 1). Of course this percentage is sensible to the problem under study. Compare the 90% in the Dining Philosophers to the 13% in the FireWire (Tables 2 and 4, respectively). Last but not least, we tuned more carefully the abstraction and refinement algorithms described in [9] (data-structures, custom operations and variable ordering in MTBDDs). These improvements were however too technical to be described here.

Our near future goal is to apply the results of this paper to model check probabilistic timed automata. Decidability of the model checking of properties on such automata has been proven, by resorting to their region graphs [23]. However, region graphs are known to be practically unusable, and our technique would allow to generate progressively a minimal *probabilistic* model, in the spirit of [1]. The different structure of the state space naturally requires the design of new abstraction algorithms, compared to the ones currently implemented in our tool.

References

1. R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. Minimization of timed transition systems. In *Proceedings of the Third Conference on Concurrency Theory CONCUR '92*, volume 630 of *LNCS*, pages 340–354. Springer-Verlag, 1992.
2. A. Aziz, V. Singhal, F. Balarin, R.K. Bryton, and A.L. Sangiovanni-Vincentelli. It usually works:the temporal logics of stochastic systems. In *Computer Aided Verification, CAV'95*, volume 939 of *LNCS*, July 1995.
3. R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3):171–206, April 1997.
4. C. Baier, J.-P. Katoen, and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In *CONCUR'99*, volume 1664 of *LNCS*, 1999.
5. Christel Baier. *On Algorithmic Verification Methods for Probabilistic Systems.* Habilitation thesis, Faculty for Mathematics and Informatics, University of Mannheim, 1998.
6. M. Berkelaar. LP_SOLVE: Mixed integer linear program solver. `ftp://ftp.ics.ele.tue.nl/pub/lp_solve`.
7. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science, FSTTCS'95*, volume 1026 of *LNCS*, 1995.
8. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: A model-checking tool for real-time systems. In *Computer Aided Verification, CAV'98*, volume 1427 of *LNCS*, July 1998.

9. P.R. D'Argenio, B. Jeannet, H.E. Jensen, and K.G. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Process Algebra and Probabilistic Methods - Performance Modelling and Verification, PAPM-PROBMIV 2001*, volume 2165 of *LNCS*, pages 39–56, 2001.
10. Luca de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.
11. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP (Cæsar / Aldébaran Development Package) : A protocol validation and verification toolbox. In *Computer Aided Verification, CAV'96*, volume 1102 of *LNCS*, July 1996.
12. M. Fujita, P.C. McGeer, and J.C.-Y. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169, April 1997.
13. K. Fukuda. CDDLIB. `ftp://ftp.ifor.math.ethz.ch/pub/fukuda/cdd`.
14. H.A. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
15. V. Hartonas-Garmhausen and S. Campos. ProbVerus: Probabilistic symbolic model checking. In *AMAST Workshop on Real-Time and Probabilistic Systems, AMAST'99*, volume 1601 of *LNCS*, 1999.
16. L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *Proc. International Workshop TYPES'93*, volume 806 of *LNCS*, 1994.
17. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2000*, volume 1785 of *LNCS*, 2000.
18. G. Holzmann. The model cheker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.
19. M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *Logic in Computer Science, LICS'97.* IEEE Computer Society Press, 1997.
20. B. Jonsson and K.G. Larsen. Specification and refinement of probabilistic processes. In *Procs. 6th Annual Symposium on Logic in Computer Science*, pages 266–277. IEEE Press, 1991.
21. B. Jonsson, K.G. Larsen, and W. Yi. Probabilistic extensions in process algebras. In J.A. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebras.* Elsevier, 2001.
22. M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In *TOOLS'2002*, volume 2324 of *LNCS*, April 2002.
23. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with probability distributions. In J.P. Katoen, editor, *Procs. of the 5th ARTS*, volume 1601 of *LNCS*, pages 75–95. Springer, 1999.
24. Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1(1/2), 1997.
25. D. Lehmann and M. Rabin. On the advantages of free choice: A symmetric fully distributed solution to the dining philosophers problem. In *Proc. 8th Symposium on Principles of Programming Languages*, 1981.
26. N. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proc. 13th ACM Symposium on Principles of Distributed Computing*, 1984.

27. K. Mehlhorn and A.K. Tsakalidis:. Data structures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A : Algorithms and Complexity, pages 301–342. Elsevier, 1990.
28. PRISM Web Page. `http://www.cs.bham.ac.uk/~dxp/prism/`.
29. M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley series in probability and mathematical statistics. John Wiley & Sons, 1994.
30. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Mathematics, Massachusetts Institute of Technology, 1995.
31. D. Simons. and M. Stoelinga. Mechanical verification of the IEEE1394a root contention protocol using Uppaal2k. To appear in *International Journal on Software Tools for Technlogy Transfer*, 2001.
32. M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. In *Proc. 5th AMAST Workshop on Real-Time and Probabilistic Systems (ARTS'99)*, volume 1601 of *LNCS*, 1999.