Reliability Engineering and System Safety 139 (2015) 179–187

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Automated generation of partial Markov chain from high level descriptions

P.-A. Brameret [a,*], A. Rauzy [b], J.-M. Roussel [a]

[a] Lurpa, Ens Cachan, Univ Paris-Sud, F-94235 Cachan, France
[b] Chaire Blériot-Fabre, LGI École Centrale de Paris, Grande voie des vignes, 92295 Châtenay-Malabry cedex, France

ARTICLE INFO

Article history:
Received 19 August 2014
Received in revised form
12 February 2015
Accepted 22 February 2015
Available online 3 March 2015

Keywords:
Model Based Safety Assessment
Markov chains
State space build
AltaRica

ABSTRACT

We propose an algorithm to generate partial Markov chains from high level implicit descriptions, namely AltaRica models. This algorithm relies on two components. First, a variation on Dijkstra's algorithm to compute shortest paths in a graph. Second, the definition of a notion of distance to select which states must be kept and which can be safely discarded.

The proposed method solves two problems at once. First, it avoids a manual construction of Markov chains, which is both tedious and error prone. Second, up the price of acceptable approximations, it makes it possible to push back dramatically the exponential blow-up of the size of the resulting chains.

We report experimental results that show the efficiency of the proposed approach.

1. Introduction

Markov chains are pervasive in Probabilistic Safety Analyses. They make it possible to assess performance indicators for systems with complex control structures such as cold spare units, or systems with limited number of resources. However, they suffer from the exponential blow-up of the number of states and transitions. This drawback has two aspects. First, the manual construction of Markov chains is both tedious and error prone. Second, assessment of large Markov chains is very resource consuming.

A way to solve the first problem consists in generating Markov chains from higher level descriptions, typically Generalized Stochastic Petri Nets [1] or AltaRica models [2]. These descriptions represent the state space in an implicit way. To obtain the Markov chain, a space exploration algorithm is used: starting from the initial state, states and transitions are progressively added to the resulting chain, until no more state or transition can be added.

However, only some of the many states of a very large Markov chain are relevant to the calculation of reliability indicators. The odds of reaching them is very low. Therefore, they have almost no influence on the calculated quantities and can be safely ignored. The same idea is behind algorithms that consider failure sequences in turn, while keeping only probable enough sequences; see e.g. [3–5]. What we propose here is rather to generate a relevant fraction of the whole Markov chain. Technically, the idea is to explore the underlying state graph at a bounded depth, i.e. to keep states (and transitions between these states) that are at the shortest distance from the initial state. Our algorithm relies on two components:

- An efficient way to explore the underlying graph in order to avoid revisiting states. To do so, we apply a variation of Dijkstra's algorithm to determine on-the-fly shortest paths in a graph [6].
- A suitable notion of distance which is basically the probability of the path and that is used as an indicator of relevance for states.

The combination of these two components proves extremely efficient. We present here examples for which a partial chain, whose size is a tiny fraction of the complete chain, makes it possible to approximate system unreliability with a relative error less than 0.25%.

It is not possible to guarantee a priori the quality of the approximation (to get a "probably approximately correct" result according to Valiant's scheme for approximation algorithms [7]). However, we show that it is possible to calculate a posteriori an upper bound of the probability of discarded states. This bound provides the analyst with a means to assess the accuracy of the approximation.

The method we propose in this paper is a contribution to the so-called Model-Based Safety Analyses: it makes Markov chains an effective tool to assess large high level models. This tool is of paramount interest for systems that show dependencies amongst

* Corresponding author.
E-mail addresses: pierre-antoine.brameret@lurpa.ens-cachan.fr (P.-A. Brameret), Antoine.Rauzy@ecp.fr (A. Rauzy), jean-marc.roussel@lurpa.ens-cachan.fr (J.-M. Roussel).

http://dx.doi.org/10.1016/j.ress.2015.02.009
0951-8320/

failures, i.e. systems for which combinatorial representations (such as Fault Trees) are not suitable.

The remainder of this paper is organized as follows. Section 2 introduces the context of the present work, and discusses related works. Section 3 presents the algorithm. Section 4 discusses issues regarding the practical implementation of the algorithm and the accuracy of the approximation. Finally, Section 5 presents experimental results.

2. Problem statement

2.1. Context

Classical formalisms used in safety analyses, such as Fault Trees and Markov chains, are well mastered by analysts. Moreover, they provide a good tradeoff between the expressiveness of the modeling formalism and the efficiency of assessment algorithms. They stand however at a low level. As a consequence, there is a significant distance between the specifications of the system under study and the safety models of this system. This distance is both error prone and a source of inefficiency in the modeling process. Not only are models difficult to share amongst stakeholders but any change in the specifications may require a tedious review of safety models.

Hence the idea is to describe systems with high level modeling formalisms and to compile these high level descriptions into lower level ones, typically Fault Trees and Markov chains, for which efficient assessment algorithms exist. AltaRica 3.0 is such a high level formalism (see e.g. [8]).

The semantics of AltaRica 3.0 is defined in terms of Guarded Transition Systems [9]. Prior to most of any assessment, including compilation into Markov chains, AltaRica 3.0 models are flattened into Guarded Transition Systems as illustrated Fig. 1 which gives an overview of the AltaRica 3.0 project.

As defined in [8], a Guarded Transition System (GTS for short) is a quintuple $\langle V, E, T, A, \iota \rangle$, where

- $V = S \uplus F$ is a set of variables, divided into two disjoint subsets: the subset S of state variables and the subset F of flow variables.
- E is a set of events.
- T is a set of transitions. A transition is a triple $\langle e, G, P \rangle$, denoted as $e : G \to P$, where $e \in E$ is an event, G is a guard, i.e. a Boolean formula built over V, and P is an instruction built over V, called the action of the transition. The action modifies only state variables.
- A is an assertion, i.e. an instruction built over V. The assertion modifies only flow variables.
- ι is the initial assignment of variables of V.

In a GTS, states of the system are represented by variable assignments. A transition $e : G \to P$ is said to be fireable in a given state σ if its guard G is satisfied in this state, i.e. if $G(\sigma) = \texttt{true}$. The firing of that transition transforms the state σ into the state $\sigma' = A(P(\sigma))$, i.e. σ' is obtained from σ by applying successively the action of the transition and the assertion.

Guarded Transition Systems are implicit representations of labeled Kripke structures, i.e. of graphs whose nodes are labeled by variable assignments and whose edges are labeled by events. The so-called reachability graph $\Gamma = \langle \Sigma, \Theta \rangle$ of a GTS $\langle V, E, T, A, \iota \rangle$ is the smallest Kripke structure such that

- $\iota \in \Sigma$.
- If $\sigma \in \Sigma$, $e : G \to P$ is a transition of T and $G(\sigma) = \texttt{true}$ (the transition is fireable in σ), then $\sigma' = A(P(\sigma)) \in \Sigma$ and $e : \sigma \to \sigma' \in \Theta$.

If exponential distributions are associated with events of E, the Kripke structure $\Gamma = \langle \Sigma, \Theta \rangle$ can be interpreted as a Continuous Time Homogeneous Markov Chain (for sake of brevity we shall just write Markov Chain in the remainder of the paper). The reliability indicators (such as system unavailability) can be defined by associating a reward (a real number) with each state of the chain.

The reachability graph may be very large, even for small GTS. Assume for instance that we model a system made of n independent, repairable components. Then, the number of variables of V is n, the number of transitions of T is $2 \times n$, but the number of states of Σ is 2^n and the number of transitions of Θ is $n \times 2^n$. Even when the components of the system are not fully independent, safety models tend to show the same picture, i.e. a number of states which is exponential in the number of components (or the variables in the GTS) and a number of transitions which is a small multiple of the number of states.

The idea is thus to generate (still starting from the initial state and applying the above principle) only a fraction of the Kripke structure,

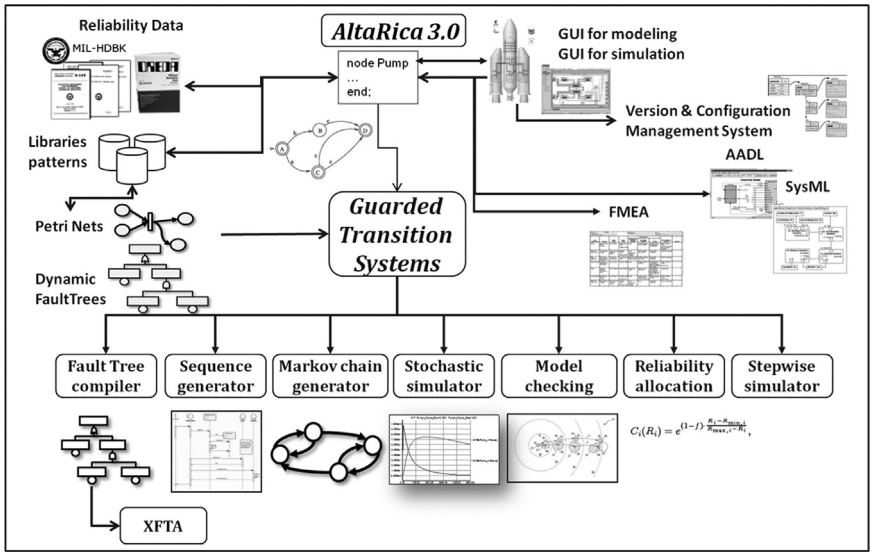

Fig. 1. Overview of the AltaRica 3.0 project.

keeping only states that are relevant with respect to the calculation of reliability indicators, and transitions between these states.

2.2. Related work

Several strategies have been proposed to overcome, or at least to contain, the combinatorial explosion of the size of Markov chains. Establishing a full review of the literature on this topic is quite difficult, because Markov chains are pervasive in many areas of science. We review here a few ideas that are related to the proposed work, but our aim is not to provide a comprehensive review of related research.

As already pointed out in the Introduction, methods have been developed to assess implicitly described Markov chains sequence by sequence (see e.g. [3–5]). The calculated quantity is obtained by summing-up individual values obtained for sequences. Sequences with a too low probability are discarded. This method is used by Bouissou and Bon to assess the so-called Boolean-Driven Markov Processes [10]. Of course, Generating a partial Markov chain induces a higher memory consumption than considering sequences in turn (if sequences are not saved), but subsequent calculations are much easier. The partial chain is actually generated independently of any specific target indicator or mission time. It can then be used to calculate any indicator defined in terms of rewards and transient probabilities, steady state probabilities or sojourn times.

Another approach has been proposed by Plateau et al. for the assessment of Stochastic Automata Network – SAN are another high level formalism to describe finite state automata –, see e.g. [11,12]. The idea is to express the Markov chain in terms of a generalized tensor product, using the modularity of SAN models. This approach makes it possible to reduce the size of the chain, but it does not reduce its number of states.

Pribadi et al. introduced a method to reduce the size of Markov chains while guaranteeing the exact assessment of the rewards [13]. This method seems efficient to reduce the chain but it relies on a strong ergodicity hypothesis. This hypothesis is not always verified, in particular in the case of non-repairable systems.

Fourneau et al. proposed a theoretical framework to study state-space truncation, i.e. to estimate bounds on the error made by applying such censoring (see e.g. [14,15]). These theoretical developments are of interest although it seems difficult to apply them to our concrete problem because of the complexity of underlying algorithms.

Mercier developed in [16] approximate bounds to quickly calculate transient and steady-state probabilities. Unfortunately, the method uses the inversion of a matrix which is similar to the transition matrix, so the method is not scalable.

Carrasco proposed in [17] approximate bounds with error control to calculate transient rewards of a Markov chain. This method is interesting because it uses a smaller Markov chain to approximate the bigger one, and is based on sequences. It is an optimized algorithm to calculate rewarded Markov chains. However, the algorithm to build the smaller chain uses matrix product involving the transition matrix of the bigger chain, so the method is not scalable.

Muntz et al. developed in [18] a method to bound steady-state performance indicators of a system using approximate aggregation in Markov chains. They build the Markov chains from (simple) high level description, and avoid the construction of the whole chain. The method we develop in this paper is close to that one: we generate also a partial Markov chain from a higher level modeling language. However, in contrast to Muntz et al., we do not make any assumption about the input model. The method proposed by Muntz et al. works only for models in which the number of possible failures in each state of the system is known prior to the calculation of the chain and only one component can be repaired at a time.

To the best of our knowledge, the approach we propose in this paper is original. It can evaluate any performance indicator of any system, while the full Markov chain is never generated.

3. Partial construction of the reachability graph

The algorithm to build a partial reachability graph relies on two components: first, a variation of Dijkstra's shortest path algorithm, second the calculation of suitable notion of distance, i.e. a relevance factor for states. In this section, we shall present them in turn.

3.1. Variation on Dijkstra's shortest path algorithm

Let $\langle V, E, T, A, \iota \rangle$ be a Guarded Transition System and $\Gamma = \langle \Sigma, \Theta \rangle$ be its reachability graph. Assume that a length $l(e : \sigma \to \tau)$, i.e. a positive real number, is associated with each transition of Θ (no matter what it means for now).

The distance from a state σ to any state τ of Σ is defined as usual as the minimum, over the paths from σ to τ, of the lengths of the paths. The length of a path is the sum, over the transitions of the path, of the length of the transitions.

The Dijkstra's algorithm [6] calculates distances of all states from a source state. The basic idea is as follows. At any time, there is a set C of candidates, i.e. of states that have been reached but not treated yet. Initially C contains only the source state. The calculation is completed when there are no more candidates. If C is not empty, the algorithm picks up the candidate σ that is at the shortest distance from the source, removes it from C and then considers in turn all of the successors of σ. Let τ be such a successor. If τ is already treated, there

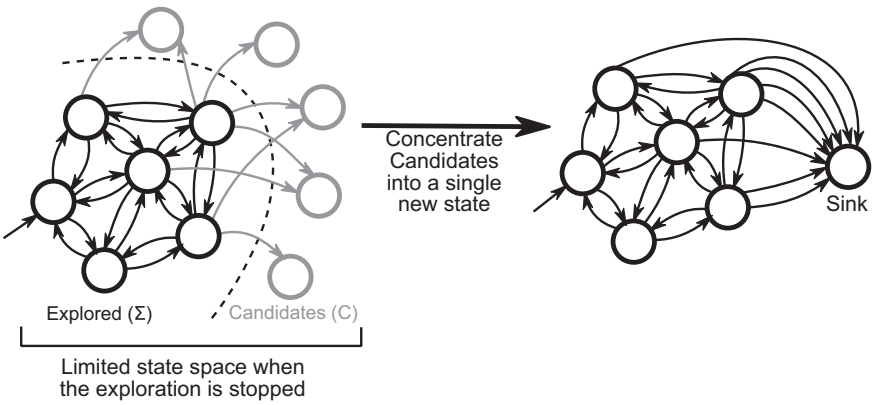

Fig. 2. Building the Markov chain with a sink state to gather discarded candidate states.

is nothing to do. If τ is not in C, then it is added to C and its distance to the source $d(\tau)$ is set to $d(\sigma)+l(\sigma,\tau)$, where $l(\sigma,\tau)$ denotes the length of the edge (σ,τ). Finally, if τ already belongs to C, then its distance to the source is updated, i.e. is replaced by $d(\sigma)+l(\sigma,\tau)$ if this value is smaller than the previous value of $d(\tau)$.

This strategy ensures that each state is treated only once and therefore the algorithm is linear in the number of transitions of the graph. To understand why it works, it suffices to remark that there cannot be a shortest path from the source to the selected candidate σ going through non-treated states because all of these states are already at the longer distance from the source than σ.

We designed a variation of Dijkstra's algorithm to build on-the-fly a partial reachability graph up to a given size S. Algorithm 1 sketches the way it works. The limiting size S of the graph can be either the number of states, or the number of transitions, or the number of bytes of the objects in memory or any other convenient measure. In our experiments, we used the number of transitions.

Algorithm 1. Algorithm for the construction of a partial reachability graph of size at most S.

Input: A GTS $\langle V,E,T,A,\iota\rangle$
Input: A function $l(e:\sigma\to\tau)$ that calculates the length of a transition
Input: A threshold S on the size of the reachability graph
Output: $\Gamma_S=(\Sigma,\Theta)$ the partial reachability graph
Local: C the set of candidate states
Local: $d(\sigma)$ the distance of each state σ to the initial state ι

```
1  begin
       //  Initialization
2      C ← {ι}, Σ ← ∅, Θ ← ∅, d(ι) ← 0.0;
       //  Construction of the state space
3      while C ≠ ∅ and |Γ| ≤ S do
           //  Selection of the best candidate
4          let σ be the candidate with the minimum value d(σ);
5          C ← C\{σ};
6          Σ ← Σ ∪ {σ};
           //  Calculation of its successors
7          for each fireable transition e : G→P of T do
8              let τ = A(P(σ));
9              let d = d(σ)+l(e : σ→τ);
10             if τ ∈ C and d(τ) > d then
11             ⌊ d(τ) ← d;
12             else if τ ∉ C and τ ∉ Σ then
13                 d(τ) ← d;
14                 C ← C ∪ {τ};
15             Θ ← Θ ∪ {e : σ→τ};
           //  Removal of discarded candidates
16     create a sink state ω and add it to Σ;
17     for each transition e : σ→τ in Θ s.t. τ ∉ Σ do
18         remove e : σ→τ from Θ;
19         add e : σ→ω to Θ;
```

The initial state ι is the initial assignment of variables of the given GTS $\langle V,E,T,A,\iota\rangle$. The algorithm builds the reachability graph by adding in order states that are at the shortest distance from ι. It is thus possible to stop the exploration at any time while keeping only the "best" states.

The last part of the algorithm redirects to a sink state transitions whose target state has been discarded (see Fig. 2). An alternative consists in just discarding these transitions. We shall explain in the next section the interest of the sink state ω.

3.2. Suitable notion of distance

In Algorithm 1, the notion of distance is rather abstract. From a mathematical viewpoint, we just need a set of values D together with a comparison operation $<$ and a binary aggregation (or sum) operation $+$ so that for any two elements a and b of D, the following properties hold:

- $<$ is a total order over D, i.e. either $[a\le b$ or $b\le a]$.
- The sum of two distances is a distance, i.e. $a+b\in D$.
- By adding a distance to a distance we get a longer distance, i.e. $a\le a+b$.

Assume that each event e is associated with a transition rate $\lambda(e)$. Let σ be a state of Σ with out transitions $e_1:\sigma\to\tau_1,\ldots,e_p:\sigma\to\tau_p$. Then, the probability $p(e_i:\sigma\to\tau_i)$ to take the transition $e_i:\sigma\to\tau_i$ $(1\le i\le p)$ is as follows:

$$p(e_i:\sigma\to\tau_i)=\frac{\lambda(e_i)}{\sum_{1\le j\le p}\lambda(e_j)} \tag{1}$$

Moreover, the mean time $\theta(\sigma)$ to get out of the state σ is as follows:

$$\theta(\sigma)=\frac{1}{\sum_{1\le j\le p}\lambda(e_j)} \tag{2}$$

Our original idea was to define a notion of distance that takes into account both $p(e_i:\sigma\to\tau_i)$ and $\theta(\sigma)$. Multiple experiments with various combinations of these two quantities showed however that taking into account $\theta(\sigma)$ is of no help for our objective. Eventually, we define the distance just as $p(e_i:\sigma\to\tau_i)$ with the multiplication (of conditional probabilities) as the aggregation operation, and $>$ as order relation. It is easy to verify that these operations verify the properties stated above.

This relevance indicator for states is essentially heuristic. It is fully compatible with Dijkstra's algorithm, therefore making the generation of the chain very efficient.

4. Discussion

In this section, we discuss two important issues regarding our algorithm: first, its practical implementation; second, the accuracy of approximations.

4.1. Practical implementation

To implement efficiently the algorithm proposed in the previous section, we have to choose carefully data structures to encode states of Σ, the set Σ itself, transitions of Θ, the set Θ itself, and the set C of candidates. We shall examine them in turn.

Each state σ must encode a value for each variable of V. In Guarded Transition Systems (and more generally in AltaRica 3.0), the value of flow variables can be calculated (by means of the assertion A) from the value of state variables. Therefore it is possible to store values of state variables only, up to the cost of the recalculation of the assertion (which is linear in the number of flow variables). We simply encoded states as arrays of values. More elaborated data structures such as Binary Decision Diagrams [19] that makes it possible to share information amongst the different states and therefore to get more compact encoding could be tested in future implementation. States embed also a real number to encode their distance to the origin.

We encoded the set Σ by means of an AVL Tree (see e.g. [20]), so that testing whether a given state belongs to the set, insertions and removals can be performed in $O(|V|\times\log(|\Sigma|))$ (the factor $|V|$ is the cost of a comparison between two states). The size of the encoding of Σ is in $O(|V|\times|\Sigma|)$.

Transitions are encoded as triples of pointers (to source and target states and to the event). The set Θ can be just implemented as a list so that insertions can be done in constant time. Removals of the last part of the algorithm can be done efficiently as well for the whole list has to be traversed anyway. The encoding of Θ is therefore linear in the size of this set.

Finally, the set C must be implemented in such a way that insertions and removals as well as the selection of the state at the shortest distance of the initial state are efficient. We chose a binary heap to do so (see e.g. [21]) which makes the former insertions and removals in $\mathcal{O}(|V| \times \log(|C|))$ and selection of the candidate in $\mathcal{O}(1)$. C is also backed up with an AVL Tree, because testing whether a given state is a candidate would not be efficient with the binary heap.

As already pointed out, the algorithm visits each transition at most once. With the chosen data structures, the cost of a visit of a transition is in $\mathcal{O}(|V| \times \log(|\Sigma|))$ (given that C is in any case smaller than the final size of Σ). So eventually, the whole algorithm runs in $\mathcal{O}(|V| \times \log(|\Sigma|) \times |\Theta|)$.

The number of transitions is in general in $\mathcal{O}(|V| \times |\Sigma|)$ (for the reasons given Section 2.1). So, the size of the encodings of Σ and Θ are comparable and related. Again, the really limiting factor is the number of states of the reachability graph.

4.2. Accuracy of the approximation

The algorithm we propose here is a heuristic to build a partial Markov chain. It would be desirable to guarantee the accuracy of this approximation, i.e. to be able, given a model (a GTS) M and a maximum percentage of error ϵ, to determine a priori (prior to the construction of the chain) a threshold S on the size, such that the error made by calculating reliability indicators from Γ_S rather than from Γ is at most ϵ. Of course S should be small enough, i.e. typically be polynomially related to the size of M. Unfortunately, guaranteeing the accuracy is not possible. To prove this negative result, we use a counterexample. We can observe first that a Fault Tree can be easily encoded as a GTS:

- Each basic event is encoded by means of Boolean state variable and a failure transition that turns the state variable from false to true.
- Each gate is encoded by means of a flow variable and an instruction of the assertion that updates the value of the variable according to the value of its fan-ins.

Now, if we were able to build a polynomial size Markov chain for that GTS that provides a predictable degree of approximation, we would be able to this degree of approximation in polynomial time for the probability of the top event of the Fault Tree. However, the calculation of the probability of the top event of a Fault Tree is P-hard [22] and even approximations of P-hard problems are hard, as predicted by the computational complexity theory (see e.g. [23]). It follows that obtaining predictable approximations is not possible.

Note also that in practice, the question must be set in different terms. In fact, we have a computer (or network of computers) with a given memory. So the question is actually what can we do within this memory? In this respect, the value of S does not depend on the model, but on the calculation resources at hand.

Does it mean that the algorithm we propose here is just a heuristic method without any possibility to monitor the error? Not exactly.

Here comes into the play the sink state ω we introduced in the last part of our algorithm. The reward for that state can be set to 0 (or to any convenient value) so that it does not influence the calculation of the reliability indicator of interest. The probability to be in any other state of the partial chain state is necessarily smaller than the probability to be in the same state in the complete chain, because the sink state absorbs a fraction of this probability without restituting anything (the same reasoning applies to sojourn times). As a consequence, the error made by considering the partial chain is bounded by the probability to be in the sink state (or the sojourn time in this state) times the suitable value of the reward.

Note that we can consider the absolute value of this indicator to obtain an absolute bound on the error, or to compare it with the approximated value of the reliability indicator, so to get a relative bound on the error. Of course, the latter error is in general much higher than the former one.

This is only an a posteriori result about the accuracy of the approximation (as it comes after the construction of the partial chain), but it is better than nothing. As we shall see in the next section, this indicator is of real practical interest. It could be used for instance to seek for a suitable value of S by means of a dichotomic search, in case one tries to get the smallest partial chain that approximates the complete chain with a given degree of approximation.

5. Experiments

In this section, we report experimental results we obtained with our algorithm in two test cases of the literature. First, a non-repairable system for which the complete state space can be explored (so that the accuracy of approximations can be assessed directly). Second, a repairable system for which the complete state space is too big to be fully explored. On this example, the sink state technique makes it possible to show that the accuracy of the approximation is good enough, even for small maximum sizes.

5.1. A computing system

This example comes from Malhorta et al. [24]. It was used in [25] to compare three safety tools which assess the unreliability of a system, based on different approaches: DBNet [25], DRPFTproc [26] and Galileo [27]. The safety model, technical data and mission times we use here were described in [25].

The objective is to assess the (transient) unreliability of the system at different mission times.

5.1.1. Description

The system pictured Fig. 3 is a non-repairable multiprocessor computing system made of two computing modules CM1 and CM2. Table 1 gives reliability data of components.

- CM1 consists of a processor P1, a memory M1, a primary hard disk D11 and a backup disk D12.
- CM2 consists of a processor P2, a memory M2, a primary hard disk D21 and a backup disk D22.
- M3 is a spare memory. It can replace M1 or M2 in case of failure, but not both.
- A unique bus connects CM1, CM2 and M3.
- The power supply PS is used by both processors.

The disks and the memory are warm spares, which deteriorate, even when unused.

5.1.2. Experimental results

Table 2 shows unreliabilities for several mission times calculated with different tools. Values given for DBNet, DRPFTproc, and Galileo come from [25]. The complete Markov chain is made of

3328 states and 17,152 transitions. As the reader can see, results obtained with the four methods are almost identical.

As the size of the Markov chain is roughly determined by its number of transitions, we shall study partial Markov chains for successive values (fractions) of the maximum number of transitions (1/2, 1/5, 1/10…).

To start with, we shall consider the Markov chain obtained by merging all discarded candidate transitions into a sink node. As explained in Section 4.2, this chain is used to obtain a lower bound and an upper bound for the unreliability of the system. The lower bound is obtained by summing up the probability of failed states. The upper bound is obtained by adding the probability to be in the sink state to the lower bound.

Results are reported in Table 3. The first column gives the maximum number of transitions before discarding candidate states. The second one expresses this number as a fraction of the number of transitions of the full chain. The third column gives the number of states of the partial chain. The fourth column gives its number of transitions. This number is slightly bigger than the threshold for it incorporates also the transitions to the sink state. The fifth and sixth columns give respectively the lower and the upper bound obtained for the unreliability calculated at $t=10{,}000$ h. Finally, the last two columns give the running time in second (the first number is the running time for the generation of the partial chain, the second one is the running time for the assessment of the chain).

Two lessons can be learned from these results:

- Both lower and upper bounds are very close to the actual unreliability, even for partial chains which represent tiny fractions (1/20, 1/50) of the full chain.
- The actual unreliability is systematically much closer to the lower bound.

These two lessons apply to all (reasonable) thresholds and mission times we tested.

Next, we consider the partial chains obtained by ignoring transitions going to discarded candidate states. These chains are approximations of the original one in the full sense (the calculated unreliability is neither a lower nor an upper bound of the unreliability). Table 4 gives the calculated approximations for the same thresholds as previously.

Although the partial chains without sink state cannot provide bounded approximations, these approximations are in practice even better than the bounds obtained previously, again with partial chains which represent only tiny fractions of the full chain.

To deepen these results, we measured the error percentage $\delta(t)$ of the unreliability $U'(t)$ calculated with a partial Markov chain made of 74 states and 470 transitions. This error percentage $\delta(t)$ is defined as follows:

$$\delta(t) = \left|\frac{U'(t) - U(t)}{U(t)}\right| \times 100$$

Fig. 4 presents the evolution of $\delta(t)$ w.r.t. the mission time. The curve is drawn from 1000 measures (one every 50 h).

It is noticeable that the error percentage never exceeds 0.25%. Its evolution is influenced by absorbing states in the Markov chain (the system is made of non-repairable components). The overall evolution of the error percentage significantly depends on the system.

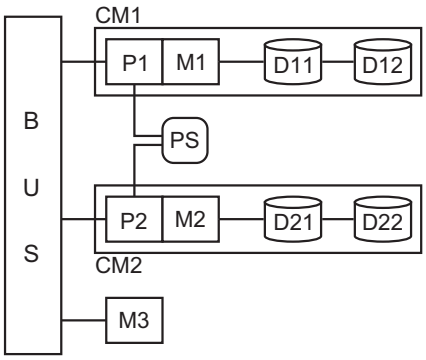

Fig. 3. A multiprocessor computing system (taken from [25]).

Table 1
Failure rates and dormancy factors for the computing system.

Component	Failure rate (h^{-1})	Dormancy factor
BUS	2.0×10^{-9}	–
P1, P2	5.0×10^{-7}	–
PS	6.0×10^{-6}	–
D11, D12, D21, D22	8.0×10^{-5}	0.5
M1, M2, M3	3.0×10^{-8}	0.5

Table 2
Unreliability of the computing system, assessed with different tools.

Time (h)	DBNet	DRPFTproc	Galileo	AltaRica
1000	0.0060086	0.0060088	0.0060088	0.0060088
2000	0.0122452	0.0122455	0.0122455	0.0122456
3000	0.0191820	0.0191832	0.0191832	0.0191833
4000	0.0273523	0.0273548	0.0273548	0.0273548
5000	0.0372379	0.0372413	0.0372413	0.0372413

5.2. An electric power supply system

The second example is taken from [28,10,29]. It is an electric power system with repairable components, failures on demand, cold redundancies, and common cause failures. We shall use here an augmented version of the model given in [29].

Table 3
Bounds on the unreliability obtained for the computing system with different partial Markov chains with a sink state, $t=5000$ h.

Threshold	Fraction	states	transitions	Lower bound	Upper bound	Generation (s)	Assessment (s)
17,152	1/1	3,328	17,152	0.0372413	0.0372413	0.9	1.1
8576	1/2	1433	8637	0.0372413	0.0372413	0.6	0.6
3430	1/5	556	3,473	0.0372412	0.0372413	0.3	0.2
1715	1/10	274	1,719	0.0372339	0.0372430	0.2	0.1
858	1/20	145	876	0.0371127	0.0373886	0.1	0.1
343	1/50	55	344	0.0361117	0.0405602	0.1	0.0

Table 4
Approximations of the unreliability obtained for the computing system with different partial Markov chains without sink state.

Threshold	Fraction	states	transitions	Unreliability	Relative error
17,152	1/1	3,328	17,152	0.0372413	(reference)
8576	1/2	1,432	5,859	0.0372413	–
3430	1/5	555	1,888	0.0372413	–
1715	1/10	273	797	0.0372384	7.8×10^{-5}
858	1/20	144	345	0.0371881	1.4×10^{-3}
343	1/50	54	105	0.0366998	1.7×10^{-2}

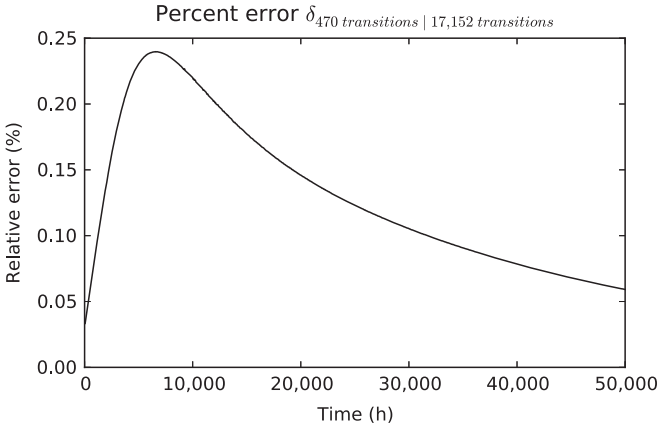

Fig. 4. Evolution of the error percentage δ with the time.

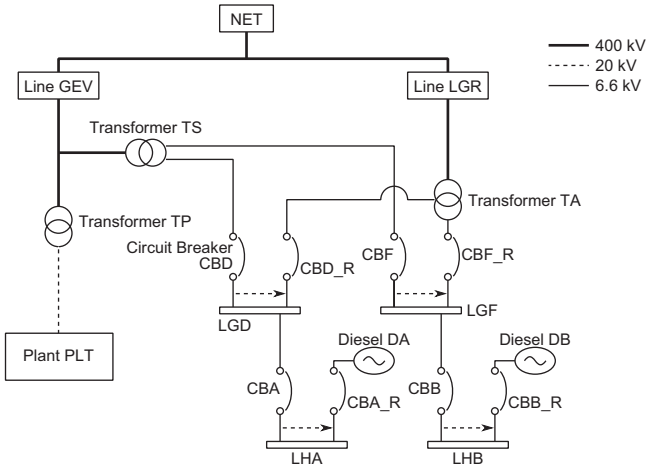

Fig. 5. An electric power supply system.

5.2.1. Description

The system is pictured Fig. 5. The role of the system is to supply electricity out of the boards LHA or LHB. The regular power supply of boards LHA and LHB comes from the transformer TS. TS is supplied by the NET and by the plant PLT. When the NET is available, PLT works in regular mode. Otherwise the PLT works in standalone mode, which is rather unstable. LHA and LHB can also be powered by the NET alone through transformer TA. Diesels generators DA and DB supply respectively LHA and LHB when these boards are not powered by LGD and LGF. Boards LGD, LGF, LHA and LHB may fail.

In this paper, we extended the original system with circuit breakers whose role is to protect the boards. They were chosen to have an influence on the overall availability while not being the prevailing components. Careful attention was given to avoid symmetries, useless components, useless components, and other means to simplify the model.

Table 5 gives the reliability data taken from [29]. Components are repairable and have a failure rate λ, a repair rate μ. The plant PLT may fail on demand to switch to standalone mode with probability γ. Diesels may fail on demand and have a common cause failure with rate λ_{cc} and repair rate μ_{cc}. Diesels which failed on demand are repaired with rate μ_d.

The problem at hand is to assess the unavailability of the system, typically for a mission time of 10,000 h, i.e. about one year.

The system is made of the following elements: 18 “binary” repairable components, the plant which has 2 failure modes (failure and failure on demand), and the 2 diesel generators which have 3 failure modes. The estimated number of states is then about 2^{26} (67 millions) and the estimated number of transitions is 26×2^{26} (1.7 billions).

5.2.2. Bounds on unavailability

As in the previous section, bounds on the unavailability of the system are first calculated by means of partial chains with a sink state. It is important to note that the sink state absorbs an increasing proportion of probability as the mission time grows. Therefore, the lower and upper bounds calculated with this technique tend respectively to 0 and 1 as the mission time tends to infinity. Fig. 6 shows a typical evolution of the bounds (calculated with a threshold $\xi = 400,000$ on the number of transitions). For a realistic mission time however (such as 10,000 h), bounds are very significant.

Table 6 gives the bounds calculated at $t = 10{,}000$ h with different thresholds on the number of transitions. Columns give the same information as previously. These results show that it is possible to obtain accurate bounds within reasonable amounts of computing resources (time and memory). Note that generating and assessing a Markov chain with more than 1 million states and 20 millions of transitions is at the current limit of the technology:

Table 5
Reliability data for the electric power supply system.

	λ (h^{-1})	μ (h^{-1})	γ	μ_d (h^{-1})	λ_{cc} (h^{-1})	μ_{cc} (h^{-1})
NET	1.0×10^{-6}	8.0×10^{-3}				
GEV, LGR	5.0×10^{-6}	1.0×10^{-2}				
TP, TS, TA	2.0×10^{-6}	1.0×10^{-3}				
LGD, LGF, LHA, LHB	2.0×10^{-7}	1.0×10^{-1}				
PLT (regular mode)	1.0×10^{-4}	1.0×10^{-1}	0.5			
PLT (standalone mode)	1.0×10^{-1}	1.0×10^{-3}				
DA, DB	1.0×10^{-4}	2.0×10^{-2}	0.001	1.0×10^{-1}	1.0×10^{-4}	1.0×10^{-2}
CBD, CBD_R, CBF, CBF_R	1.0×10^{-4}	1.0×10^{-1}				
CBA, CBA_R, CBB, CBB_R						

it took us about 6 h to perform the whole computation on a rather fast PC. Two-third of this calculation time was taken by the assessment of the Markov chain.

Table 7 reports approximations obtained with chains without sink state. These chains involve significantly less transitions (and therefore are faster to assess) than the previous ones. The approximations are very close to the lower bounds obtained previously.

Fig. 7 plots the "relative" error (as explained Section 4.2) with different thresholds on the number of transitions. It is calculated as follows: at mission time, the probability to be in the sink state ω (which is the absolute error) is divided by the upper bound. It is not the relative error, as the latter involves the unavailability, which cannot be computed for this system. However, this value shows how close the bounds are, relatively to the calculated unavailability of the system. For instance, with 2,000,000 transitions, the error is 1.3×10^{-4}, so the unavailability is calculated accurately with 3 significant figures. The overall evolution of the error is interesting. The first part shows that there is a minimum number of transitions from which a limited Markov chain is able to represent the behavior of the system. The second part shows that the error decreases regularly when the number of transitions of the Markov chain is increased.

5.3. Discussing stiff Markov chains

The core idea of the method is to keep only the most probable states of the system, assuming that those states are close enough to the initial state. Therefore, its efficiency depends strongly on whether this assumption is verified or not.

If components of the system are very unreliable (e.g. failure rates are close to repair rates), then all the states are about equally probable and the partial generation cannot give accurate results.

If, on the contrary, the components of the system are highly reliable (failure rates are low, repair rates are high), then the Markov chain is stiff and the method gives good results because the probability is concentrated in a few states.

To illustrate this point, let us consider again the electric power supply in which all the failure rates are divided by ten and all the repair rates are multiplied by ten (all components are therefore 100 times more reliable than in the original system).

Results for that system (at $t=10{,}000$ hours) are presented in Table 8. The overall reliability of the system is indeed greatly improved and the bounds are much closer one to another.

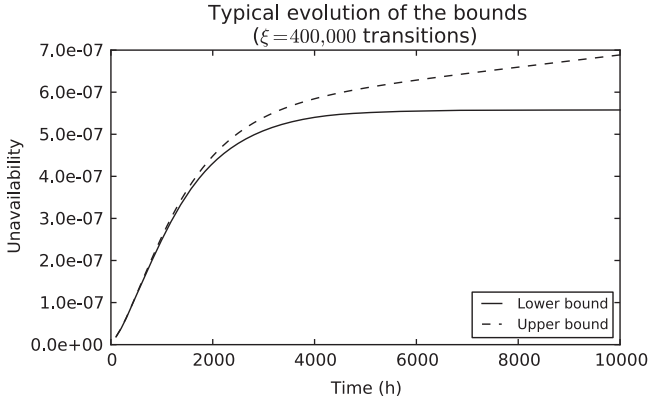

Fig. 6. Evolution of bounds on unavailability as a function of the mission time.

6. Conclusion

In this paper, we proposed an algorithm to build partial Markov chains from high level descriptions, namely AltaRica 3.0 models. This algorithm actually applies not only to AltaRica or Guarded

Table 7
Approximations of the unreliability obtained for the power unit with different partial Markov chains without sink state.

Threshold	states	transitions	Approximation	Time (s)
20,000,000	1,259,490	13,279,823	5.5798×10^{-7}	7200 + 14,000
2,000,000	123,721	989,637	5.5798×10^{-7}	990 + 1,200
800,000	49,081	353,408	5.5798×10^{-7}	380 + 330
400,000	23,752	160,920	5.5806×10^{-7}	180 + 120
200,000	12,007	73,248	5.6407×10^{-7}	86 + 43

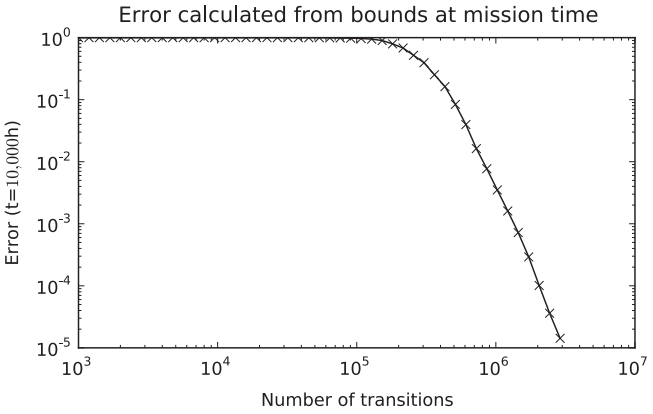

Fig. 7. Error of the unavailability obtained for the power unit at mission time, with different partial Markov chains with a sink state.

Table 8
Bounds on the unavailability obtained for the modified power unit with more reliable components.

Threshold	Lower bound	Upper bound
800,000	4.9333×10^{-14}	4.9341×10^{-14}
400,000	4.9333×10^{-14}	5.0674×10^{-14}
200,000	4.9332×10^{-14}	2.3141×10^{-13}

Table 6
Bounds on the unavailability obtained for the power unit with different partial Markov chains with a sink state.

Threshold	states	transitions	Lower bound	Upper bound	Time (s)
20,000,000	1,259,491	20,000,115	5.5798×10^{-7}	5.5798×10^{-7}	7200 + 19,000
2,000,000	123,722	2,001,075	5.5798×10^{-7}	5.5805×10^{-7}	990 + 2,200
800,000	49,082	800,292	5.5796×10^{-7}	5.6401×10^{-7}	380 + 680
400,000	23,753	400,004	5.5771×10^{-7}	6.8857×10^{-7}	180 + 270
200,000	12,008	200,439	5.5496×10^{-7}	2.1205×10^{-6}	86 + 130

Transition Systems but to any implicit representation of a Markov chain. It relies on two principles: first, a variation on Dijkstra's algorithm to compute shortest path in a graph; second, a suitable definition of distance based on the probability to leave a state by a given transition. Eventually, the algorithm is a heuristic method to select which states to keep and which to discard. We showed that, although it is not possible to bound a priori the accuracy of the approximation, it is possible to assess it a posteriori, i.e. once the partial chain has been generated.

We show, by means of test cases taken from the literature, that it is possible to obtain very accurate approximations, even with partial Markov chains that represent only a tiny fraction of the complete chain.

In a word, the compilation into partial chains plays the same role for Markov models as calculation of Minimal Cutsets with cutoffs plays for combinatorial models (Fault Trees).

The results presented here are very important in the perspective of the deployment of Model-Based Safety Assessment. Systems for which combinatorial models are not suitable (typically because they involve dependencies amongst events) can be assessed by means of Markov chains, thanks to the proposed method.

Future research should explore several potential improvements and extensions of the proposed algorithm. For instance

- Binary Decision Diagrams-like data structures could be used to encode states of the reachability graph to obtain more compact representations.
- Other notions of distances could be designed and compared with the one we proposed here.
- Deterministic transitions could be taken into account, typically periodic maintenances, with the idea of compiling the AltaRica model into a multiphase Markov chain.

The present work opens many interesting perspectives that we shall explore in the near future.

References

[1] Ajmone-Marsan M, Balbo G, Conte G, Donatelli S, Franceschinis G. Modelling with generalized stochastic Petri nets, Wiley series in parallel computing. West Sussex, England: John Wiley and Sons; 1995.

[2] Boiteau M, Dutuit Y, Rauzy A, Signoret J-P. The altarica data-flow language in use: assessment of production availability of a multistates system. Reliab Eng Syst Saf 2006;91(7):747–55.

[3] Bon J-L, Bouissou M. Fiabilité des grands systèmes séquentiels: résultats thèoriques et applications dans le cadre du logiciel gsi. Rev Stat Appl 1992;39(2):45–54.

[4] Collet J, Renault I. Path probability evaluation with repeated rates. In: Proceedings of annual reliability and maintainability symposium, RAMS'97. Philadelphia, PA, USA: IEEE; 1997. p. 184–7.

[5] Bouissou M, Lefebvre Y. A path-based algorithm to evaluate asymptotic unavailability for large Markov models. In: Proceedings of annual reliability and maintainability symposium, RAMS'2002. Seattle, USA: IEEE; 2002. p. 32–9.

[6] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Mathematik 1959;1(1):269–71.

[7] Valiant LG. Probably approximately correct: nature's algorithms for learning and prospering in a complex world. New York, NY, USA: Basic Books; 2013.

[8] Prosvirnova T, Batteux M, Brameret P-A, Cherfi A, Friedlhuber T, Roussel J-M, et al. The altarica 3.0 project for model-based safety assessment. In: Proceedings of 4th IFAC workshop on dependable control of discrete systems, DCDS'2013. Great Britain: International Federation of Automatic Control, York; 2013. p. 127–32, iSBN: 978-3-902823-49-6, ISSN: 1474-6670.

[9] Rauzy A. Guarded transition systems: a new states/events formalism for reliability studies. J Risk Reliab 2008;222(4):495–505.

[10] Bouissou M, Bon J-L. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic-driven Markov processes. Reliab Eng Syst Saf 2003;82(2):149–63.

[11] Fourneau J-M, Plateau B. A methodology for solving Markov models of parallel systems. J Parallel Distrib Comput 1991;12:370–87.

[12] Fernandes P, Plateau B, Stewart WJ. Efficient descriptor-vector multiplications in stochastic automata networks. J Assoc Comput Mach 1998;45(3):381–414.

[13] Pribadi Y, Voeten JPM, Theelen BD. Reducing markov chains for performance evaluation. In: Proceedings of PROGRESS'01, STW Technology Foundation; 2001. p. 173–9.

[14] Fourneau J-M, Pekergin N, Younès S. Censoring Markov chains and stochastic bounds. In: Formal methods and stochastic models for performance evaluation. Lecture notes in computer science, vol. 4748; 2007. p. 213–27.

[15] Busic A, Djafri H, Fourneau J-M. Bounded state space truncation and censored markov chains. In: Proceedings of IEEE 51st annual conference on decision and control (CDC). Maui, HI, USA: IEEE; 2012. p. 5828–33.

[16] Mercier S. Bounds and approximations for continuous-time Markovian transition probabilities and large systems. Eur J Oper Res 2008;185(1):216–34.

[17] Carrasco JA. Computation of bounds for transient measures of large rewarded Markov models using regenerative randomization. Comput Oper Res 2003;30(7):1005–35.

[18] Muntz RR, de Souza e Silva E, Goyal A. Bounding availability of repairable computer systems. SIGMETRICS Perform. Eval. Rev. 1989;17(1):29–38. http://dx.doi.org/10.1145/75372.75376.

[19] Brace KS, Rudell RL, Bryant RS. Efficient Implementation of a BDD Package. In: Proceedings of the 27th ACM/IEEE design automation conference. Orlando, FL, USA: IEEE; 1990. p. 40–5.

[20] Cormen TH, Leiserson CE, Rivest RL. Introduction to algorithms. Cambridge, MA, USA: The MIT Press; 1990.

[21] Atkinson MD, Sack J-RW, Santoro N, Strothotte TE. Min-max heaps and generalized priority queues. Program Tech Data Struct 1986;29(10):996–1000.

[22] Valiant LG. The complexity of enumeration and reliability problems. SIAM J Comput 1979;8(3):410–21.

[23] Papadimitriou CH. Computational complexity. Boston, MA 02116, USA: Addison Wesley; 1994.

[24] Malhotra M, Trivedi KS. Dependability modeling using petri-nets. IEEE Trans Reliab 1995;44(3):428–40.

[25] Montani S, Luigi Portinale AB, Varesio M, Codetta-Raiteri D. A tool for automatically translating dynamic fault trees into dynamic bayesian networks. In: Proceedings of annual reliability and maintainability symposium, RAMS'06. Newport Beach, CA, USA: IEEE; 2006. p. 434–41.

[26] Bobbio A, Raiteri DC. Parametric fault trees with dynamic gates and repair boxes. In: Proceedings of the annual reliability and maintainability symposium (RAMS'2004). Los Angeles, CA, USA: IEEE; 2004. p. 459–65.

[27] Dugan JB, Sullivan KJ, Coppit D. Developing a low-cost high-quality software tool for dynamic fault-tree analysis. IEEE Trans Reliab 2000;49(1):49–59.

[28] Gondran M, Pagès A. Fiabilité des systèmes, Collection de la Direction des Études et Recherches d'Électricité de France, Eyrolles, vol. 39; 1980.

[29] Rauzy A. An experimental study on six algorithms to compute transient solutions of large Markov systems. Reliab Eng Syst Saf 2004;86(1):105–15.