Mark-XPR

A Computation Engine to Assess

Multi Phases Reward Markov Models

Mark-XPR User Manual

ARBoost Technologies
24, Allée Chabrier
Marseille, F-13008
FRANCE
contact@arboost.com

Author Antoine Rauzy
Version 1.3
Date 21/11/2006

Table of Contents

1. Introduction

Mark-XPR is a computation engine to assess multi phases reward Markov models. This tool has been realized by the author as a part of a partnership between ARBoost Technologies and Total S.A. Although an entirely new tool, Mark-XPR is inspired by the seminal work done by J.P. Signoret, on Mark-EXP/Mark-SMP [Sig93].

Mark-XPR implements various algorithms to assess Markov graphs, i.e. to compute their steady state and transient solutions. A set of commands makes it possible to create multi phase scenarios by applying these algorithms for various mission times, by transferring the probabilities from one graph to another, by repeating a given series of assessments and so on.

Beyond probabilities of presence, Mark-XPR computes automatically sojourn times in each state. It allows also the definitions of properties, or rewards, i.e. of functions from states to real numbers. In other words, properties are random variables defined over states. Expectations of these quantities can computed for a given time or for a given period (using respectively probabilities of presence and sojourn times).

Mark-XPR is developed in ANSI C++. It runs under Windows and Linux as a command with two parameters: the name of a file that describes the model and the name of a file that describes the commands (i.e. the computations) to be applied on the model. Both files are at the XML format. A call to Mark-XPR is as follows.

```
mark-xpr model.xml commands.xml
```

Results are printed on the standard output (and may be redirected using the directives “>” and “>>”).

The remainder of this document is organized as follows. Section 2 briefly recalls some basic definitions and properties of Markov models. Section 3 presents the syntax of model files. Section 4 describes syntax (and semantics) of command files. Section 5 is devoted to the various algorithms Mark-XPR implements. Appendix gives a complete example.

We assume the reader is familiar with Markov graphs. The book by W. J. Stewart [Ste94] has been a constant reference for our work.
We assume also that the reader is familiar with XML. Good introductions to that formalism can be found on the Web and many books are available.

2. Markov Graphs

This section gives some basic definitions and properties of Markov graphs. Some mathematical details are mandatory.

2.1. Continuous-Time Homogeneous Markov Chain

Mark-XPR deals with Continuous-Time Homogeneous Markov Chain. A Continuous-Time Markov Chain is a stochastic process, i.e. a random variable $\{X(t),\ 0 \leq t \leq +\infty\}$. The values assumed by $X(t)$ are called states and belong to a finite set. Moreover, $X(t)$ must satisfies the so-called Markov property, i.e. for all integers n and for any sequence $t_0 < t_1 < ... < t_n < t$, we have

$$Pr\{ X(t) = x \mid X(t_0) = x_0,\ X(t_1) = x_1,\ ...,\ X(t_n) = x_n \} = Pr\{ X(t) = x \mid X(t_n) = x_n \} \qquad (2.1)$$

where $Pr\{ E \}$ denotes the probability of the event E. This property is called the memory less property for the fact that the system was in state x_0 at t_0, x_1 at t_1 and so on is completely irrelevant.
When the probabilities of transitions out of the state $X(t)$ do not depend on the time t, the Continuous-Time Markov Chain is said homogeneous.
In the sequel, we shall simply say Markov process for Continuous-Time Homogeneous Markov Chain.

Graphically, Markov processes are represented as graphs. Transitions are associated with rates. The rate λ_{ij} of a transition from state e_i to state e_j equals the conditional probability $Pr\{ X(t+dt) = e_j \mid X(t) = e_i \}$ for a sufficiently small time period dt.

As an illustration consider a system made of two pumps P1 and P2 as presented Fig. 2.1. Pump P2 is a spare unit: it is started only when Pump P1 is failed. Moreover, we assume that when both pumps are failed, pump P2 is repaired first.

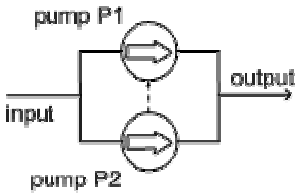

Figure 2.1. A system of two pumps

The system is failed when both pumps are failed. A Markov graph that represents this system is pictured Fig 2.2. A bar above the name of a pump indicates that this pump is failed. For instance, in state 2, the pump P1 is failed and the pump P2 is working.

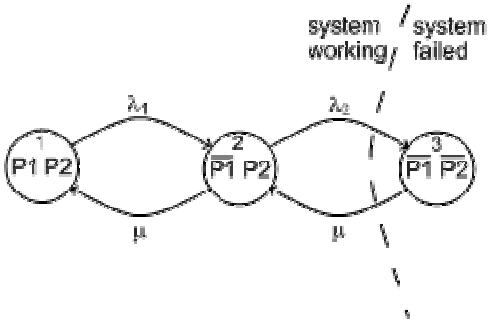

Figure 2.2. A Markov graph that represents the system of Fig. 2.1.

The Markov graph pictured Fig. 2.2 assumes a failure λ_1 for pump P1, a failure rate λ_2 for pump P2 and a repair rate μ for both.

2.2. Transient probabilities

Let us first simplify the notation by writing

$$p_{ij}(t,t+s) = Pr\{ X(t+s)=j \mid X(t)=i \}$$

After our hypotheses, that $p_{ij}(t,t+s)$ does not depend on t, so we can write it as $p_{ij}(s)$. It follows that

$$\sum_{all\ j} p_{ij}(s) = 1 \text{ for all values of } s \qquad (2.2)$$

Formally, the rate q_{ij} of the transition of the transition from state i to state j (at time any t) is defined as follows.

$$q_{ij} = \lim_{\Delta t \to 0} \left\{ \frac{p_{ij}(\Delta t)}{\Delta t} \right\}, \text{ for } i \neq j \qquad (2.3)$$

It is easy to verify that

$$q_{ii} = -\sum_{j \neq i} q_{ij} \qquad (2.4)$$

The matrix Q whose ij^{th} element is q_{ij} is called the infinitesimal generator matrix or transition rate matrix for the Markov process. Similarly, the matrix $P(\Delta t)$ whose ij^{th} element is $p_{ij}(\Delta t)$ is called the transition probability matrix. In terms of matrices, we have

$$Q = \lim_{\Delta t \to 0} \left\{ \frac{P(\Delta t) - I}{\Delta t} \right\} \tag{2.5}$$

where I denotes the identity matrix.

For instance, the infinitesimal generator matrix for graph of Fig. 2.2. is as follows.

$$Q = \begin{pmatrix} -\lambda_1 & \lambda_1 & 0 \\ \mu & -\lambda_2 - \mu & \lambda_2 \\ 0 & \mu & -\mu \end{pmatrix}$$

It can be shown, by applying the Chapman-Kolmogorov equations, that we have

$$\frac{dP(dt)}{dt} = P(dt).Q \tag{2.6}$$

Let $\pi(t)$ be the vector of length n of probabilities to be in state i at time t. $\pi(t)$ is called the vector of transient probabilities (a time t). $\pi(0)$ thus denotes the vector of initial probabilities. It can be shown, from equality 2.6, that we have

$$\pi(t) = e^{t.Q}.\pi(0) \tag{2.7}$$

where $e^{t.Q}$ is defined as follows.

$$e^{t.Q} = \lim_{n \to \infty} \sum_{k=0}^{n} \frac{(t.Q)^k}{k!} \tag{2.8}$$

We shall see section 5, that equation 2.8 is the basis of all of the algorithms to compute transient probabilities. Mark-XPR implements 6 algorithms to compute transient probabilities: the forward Euler method (FEM), the matrix exponentiation method (EXP), two Runge-Kutta methods (RK2 and RK4) and two Adams-Bathforth methods (AB2 and AB4), see section 5 for more details.

2.3. Beyond transient probabilities

Beyond transient probabilities at time t, it is possible to extract more information from a Markov model.

2.3.1. Steady state probabilities

We shall say that a graph is cyclic if, for any two states i and j, there is a path from i to j and a path from j to i. The graph is acyclic (or tree decomposable) otherwise.

It can be shown that if the underlying graph is cyclic, there exists a time T from which transient probabilities don't vary, i.e. $\pi(T+t) = \pi(T)$ for all $t>0$. $\pi_s = \pi(T)$ is then called the vector of steady state probabilities.

The evolution of probability to be in the failed state of our example is picture Fig. 2.3 for $\lambda_1 = 0.001$, $\lambda_2 = 0.002$ and $\mu = 0.1$. This figure shows clearly the steady state.

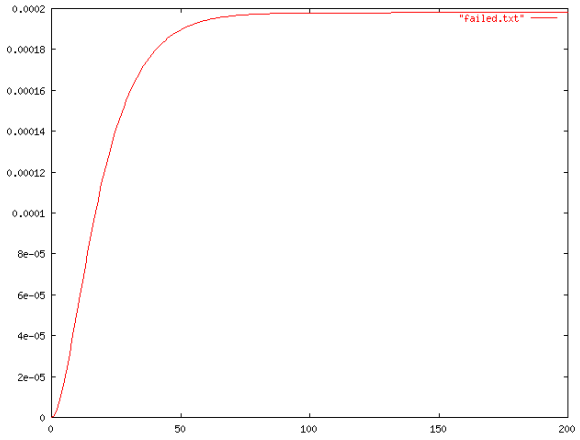

Figure 2.3. Probability to be in failure state for $\lambda_1 = 0.001$, $\lambda_2 = 0.002$ and $\mu = 0.1$.

When they exist, it is possible to compute steady state probabilities without computing $\pi(t)$ for various t. Mark-XPR implements two algorithms to compute steady state probabilities (if any): the bi-conjugate gradient (BCG) and conjugate gradient square (CGS).

2.3.2. Sojourn Times

Transient probabilities are the probabilities of presence in each state at time *t*. It is often of a great interest to assess not only these probabilities, but also the sojourn times in each state from the origin to time *t*. Formally, the sojourn time in state *i* at time *t*, denoted by $s_i(t)$, are defined as follows.

$$s_i(t) = \int_0^t \pi_i(t).dt \qquad (2.9)$$

Mark-XPR computes automatically sojourn times while computing transient probabilities. This computation can be performed at almost no cost (except the storage of the vector of sojourn times).

2.3.3. Reward Models

Each state of the Markov graph represents a state of the system under study. Various quantities may depend on the latter. For instance, in our example, the amount of fluid pumped by the system may depend on the state. It could by 100% (of the nominal amount) when pump P1 is working, 80% when pump P2 is working and 0% when none of the pumps is working.

Mark-XPR makes it possible to define so called properties. A property *r* is a real valued function of states (a Random variable defined over the states). The point expectation of *r* at time *t* is defined as follows.

$$E_p[r(t)] = \sum_{state\,i} r_i.\pi_i(t) \qquad (2.10)$$

Similarly, the cumulated expectation of *r* at time *t*, which represents the average value of *r* from the origin to time *t* is defined as follows.

$$E_c[r(t)] = \frac{1}{t} \sum_{state\,i} r_i.s_i(t) \qquad (2.11)$$

Mark-XPR computes, for each defined property, both quantities from transient probabilities and sojourn states.

Consider for instance our example at time t=100. Transient probabilities and sojourn times are as follows.

P1	P2	probability	sojourn time
working	working	0.989904	99.0942
failed	working	0.00989816	0.889981
failed	failed	0.000197856	0.0158211

Point and cumulated expectations of the amount of pumped fluid are as follows.

$$E_p[production] = 0.989904 \times 100 + 0.00989816 \times 80 + 0.000197856 \times 0 = 99.7823$$

$$E_s[production] = 1/100 \times (99.0942 \times 100 + 0.889981 \times 80 + 0.0158211 \times 0) = 99.8062$$

2.4. Equivalent failure rates and related notions

Mark-XPR provides instructions to compute generalizations to reward models of equivalent failure rates. To introduce these instructions, we need some preliminary definitions.

2.4.1. Preliminaries

Consider first a continuous, homogenous Markov model for a system S with a partition of states into two groups: working states and failure states. Let T denote the date of the first failure of S. T is a random variable. It is called the lifetime of S. We assume that components of S were as good as new at time *0* and that they are as good as new after a repair.

Reliability $R_S(t)$ and unreliability $F_S(t)$: the reliability of S at t is the probability that S experiences no failure during time interval [0,t], given that all its components were working at 0. Formally,

$$R_S(t) \overset{def}{=} Pr\{t < T\}$$

The unreliability, or cumulative distribution function $F_S(t)$, is just the opposite.

$$F_S(t) \overset{def}{=} Pr\{t \geq T\} = 1 - R_S(t)$$

The curve $R_S(t)$ is a survival distribution. This distribution is monotonically decreasing. Moreover, the following asymptotic properties hold.

$$lim_{t\to 0}\, R_S(t) = 1$$
$$lim_{t\to \infty}\, R_S(t) = 0$$

Failure density $f_S(t)$: The failure density refers to the probability density function of the law of *T*. It is the derivative of $F_S(f)$.

$$f_S(t) \overset{def}{=} \frac{d\,F_S(t)}{dt}$$

For sufficiently small dt's, $f_S(t).dt$ *expresses the* probability that the system fails between t and t+dt, given its was working at time *0*.

Failure rate $r_S(t)$: the failure rate or hazard rate is the probability the system fails for the first time per unit of time at age t. Formally,

$$r_S(t) \overset{def}{=} lim_{dt\to 0} \frac{Pr\{\text{the system fails between } t \text{ and } t+dt \,/\, C\}}{dt}$$

where C denotes the event "the system experienced no failure during the time interval [0,t]". As before, for sufficiently small *dt*'s, one can deduce from definition of $r_S(t)$, the following expression.

$$r_S(t).dt = \frac{Pr[(t<T\le t+dt)\cap(t<T)]}{Pr(t<T)} = \frac{Pr(t<T\le t+dt)}{Pr(t<T)} = \frac{f_S(t)}{R_S(t)} = \frac{\left(\frac{d\,R_S(t)}{dt}\right)}{R_S(t)} \qquad (2.12)$$

By integrating each member of equality (2.12), one obtains immediately the following property.

$$R_S(t) = exp\left[-\int_0^t r_S(u)\,du\right] \qquad (2.13)$$

Availability $A_S(t)$ and unavailability $Q_S(t)$: the availability of S at t is the probability that S is working at t, given that all its components were working at 0.

$$A_S(t) \overset{def}{=} Pr\{S \text{ is working at } t\}$$

The unavailability is just the opposite.

$$Q_S(t) \overset{def}{=} 1 - A_S(t)$$

The following properties hold.

$A_S(t) \geq R_S(t)$, for general systems. (2.14)
$A_S(t) = R_S(t)$, for systems with only non-repairable components. (2.15)

Conditional failure intensity $\lambda_S(t)$: the conditional failure intensity refers to the probability that the system fails per unit time at time t, given that it was working at time 0 and is working at time t. Formally,

$$\lambda_S(t) \overset{def}{=} lim_{dt \to 0} \frac{Pr\{\text{the system fails between } t \text{ and } t+dt / D\}}{dt}$$

where D denotes the event "the system S was working at time 0 and is working at time t". The conditional failure intensity is sometimes called Vesely rate. $\lambda_S(t)$ is an indicator of how the system is likely to fail.

Unconditional failure intensity $w_S(t)$: the unconditional failure intensity refers to the probability that the system fails per unit of time at time t, given it was working at time 0. Formally,

$$w_S(t) \overset{def}{=} lim_{dt \to 0} \frac{Pr\{\text{the system fails between } t \text{ and } t+dt / E\}}{dt}$$

where E denotes the event "the system was working at time 0". This parameter is called "failure frequency" by some authors.

In the case of systems with non-repairable components, the following property holds.

$w_S(t) = f_S(t)$, for systems with only non-repairable components.

Definitions of $\lambda_S(t)$ and $w_S(t)$ induce the following properties.

$$\begin{aligned} \lambda_S(t).dt &= Pr\{\text{the system fails between } t \text{ and } t+dt / D\} \\ &= Pr\{A / D\} = Pr\{A / E \cap F\} \\ &= Pr\{(A / E) / F\} \end{aligned} \quad (2.16)$$

Where *F* denotes the event "the system is working at time *t*".

$$\begin{aligned} w_S(t).dt &= Pr\{\text{the system fails between } t \text{ and } t+dt / E\} \\ &= Pr\{A / E\} \end{aligned} \quad (2.17)$$

Let *B* be the conditional event *(A/E)*. *B* can be expressed as follows.

$$B = (B \cap F) \cup (B \cap \overline{F})$$

Now, if we consider that at most one failure can occur during the small time interval *dt*, the event $(B \cap \overline{F})$ is not realizable (it corresponds to a repair followed with a failure). From the above equality, we can conclude that the compound event $(B \cap F)$ reduces to event *B*. Property (2.16) can be rewritten as follows.

$$\lambda_S(t).dt = \Pr\{B/F\} = \frac{\Pr\{B \cap F\}}{\Pr\{F\}} = \frac{\Pr\{B\}}{\Pr\{F\}} = \frac{\Pr\{A/E\}}{\Pr\{F\}} = \frac{w_S(t).dt}{A_S(t)}$$

Therefore, the following property holds.

$$\lambda_S(t) = \frac{w_S(t)}{A_S(t)} \qquad (2.18)$$

Pagès and Gondrand showed that, under certain reasonable conditions, $\lambda_S(t)$ can be used as an approximation of $r_S(t)$ [PG80].

2.4.2. Calculation

Consider again a continuous, homogenous Markov model for a system S with a partition of states into two groups: working states and failure states. Assume moreover that there is no transition from failure states to working states. In that case, $R_S(t) = A_S(t)$ and $r_s(t) = \lambda_S(t)$. There are two ways to assess $\lambda_S(t)$.

First, $w_S(t)$ can be assessed by numerical differentiation. For sufficiently small dt, the following property holds.

$$w_S(t) \approx \frac{Q_S(t+dt) - Q_S(t)}{dt} = \frac{A_S(t) - A_S(t+dt)}{dt} \qquad (2.19)$$

Using then equation (2.18), we get an approximation of $\lambda_S(t)$. Since all of the methods work by calculating probabilities of presence in states for successive dt (from 0 to the mission time), this first method is almost costless. However, $A_S(t)$ and $A_S(t+dt)$ may be very close and their difference is very small. Errors due to floating point numbers rounding are therefore likely to occur.

The other way to assess $w_S(t)$ uses critical transitions. A transition is critical if its source is a working state and its target is a failure state. Let Crit(S) be the set of critical transitions of the system S. Then, $w_S(t)$ can be approximated as follows.

$$w_S(t) \approx \sum_{\tau \in Crit(S)} Pr(src(\tau)) \times rate(\tau) \quad (2.20)$$

This second way of assessing $w_S(t)$ requires a loop over (critical) transitions. It is therefore more costly than the former. However, it does involve any numerical difference and is therefore less subject to numerical errors.

In the case there are transitions from some of the failure states to some of the working states, the two calculations give different results. The latter measures the fraction of probability that goes from working states to failure states. The former measures the balance of the exchange.

2.4.3. Generalization to Reward Models

In reward models, properties associated with states are not necessarily Boolean. Rather, a property P is a function from states to real numbers. The expectation of a property P at time t is defined as follows.

$$E_P(t) = \sum_{i \in states_s} E_i(t) = \sum_{i \in states_s} p_i(t) \times P(i)$$

The characterization of states into two disjoint groups (working and failure) can be interpreted as a special reward model ($A(s)=1$ if s is a working state and 0 otherwise). Under this interpretation, transitions fall into exclusive three categories: those that increase the expectation of the property, those that decrease it and those that neither increase nor decrease it. Critical transitions are the transitions that decrease the expectation. Let us denote by G_P and L_P the set of transitions that respectively increase and decrease the expectation of P (G stands for gain and L for loss). Formally, G_P and L_P is defined as follows.

$$G_P = \{\tau \,;\, P_{src(\tau)} < P_{tgt(\tau)}\}$$
$$L_P = \{\tau \,;\, P_{src(\tau)} > P_{tgt(\tau)}\}$$

Now, we can define the instantaneous gain $g_P(t)$ and the instantaneous loss $l_P(t)$ of a property P as follows.

$$g_P(t) \stackrel{def}{=} lim_{dt \to 0} \frac{\sum_{\tau \in G_P} \left[E_{tgt(\tau)}(t+dt) - E_{src(\tau)}(t)\right]}{dt}$$
$$l_P(t) \stackrel{def}{=} lim_{dt \to 0} \frac{\sum_{\tau \in L_P} \left[E_{src(\tau)}(t) - E_{tgt(\tau)}(t+dt)\right]}{dt}$$

By definition, the following property holds.

$$lim_{dt \to 0} \frac{dE_P}{dt} = g_P(t) - l_P(t)$$

Mark-XPR provides means to assess these three quantities, or more exactly these quantities normalized with $E_P(t)$. Namely, the following quantities can be computed for a property (a reward).

$$E_P(t)$$

$$\Delta_P(t) \stackrel{def}{=} \frac{lim_{dt \to 0} \frac{dE_P(t)}{dt}}{E_P(t)}$$

$$\Gamma_P(t) \stackrel{def}{=} \frac{g_P(t)}{E_P(t)}$$

$$\Lambda_P(t) \stackrel{def}{=} \frac{l_P(t)}{E_P(t)}$$

$\Delta_P(t)$, $\Gamma_P(t)$ and $\Lambda_P(t)$ are called respectively the variation, the gain and the loss of P at t.

2.5. Multi phase Markov models

Mark-XPR models can be extended in two ways (that are related one another). First, immediate states can be introduced. Second, multi phase models can be handled.

2.5.1. Immediate states

Consider again the example of Fig. 2.1. When the pump P1 fails, the pump P2 is started. It could be the case that it fails to start with a given probability γ. In order to take this phenomenon into account, one may add a transition from the nominal state (1) to the fail state (3) and modify rates accordingly. Another solution consists in adding a fake state with two out transitions: one going to the failure state (and labeled with γ), the other going to the degraded state (and labeled with $1-\gamma$), as illustrated on Fig 2.4. When the system enters into this fake state, it immediately leaves it. Hence, such a state is said immediate. A non immediate state is said stochastic.

Fig. 2.4 illustrates our graphical convention: transitions out of immediate states are represented with dashed arrows.

It is worth to notice that the sum of probabilities labeling out transitions of an immediate state must equal 1.

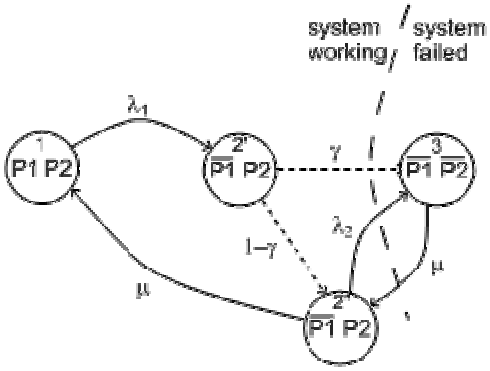

Figure 2.4. A model for the system of Fig. 2.1 with an immediate state.

Mark-XPR algorithms to compute transient probabilities (and sojourn times) are able to deal with immediate states. It is worth to notice however that this does not change essentially the expressiveness of the formalism: it is always possible to translate a model with immediate states into a model without immediate states. To do so, it suffices to remark that a transition from state *i* to state *j* with a rate λ followed by an immediate transition from state *j* to state *k* with a probability γ is equivalent to a transition from state *i* to state *k* with a rate $\lambda \cdot \gamma$.

The model pictured Fig. 2.4 is therefore equivalent to the model pictured Fig. 2.5.

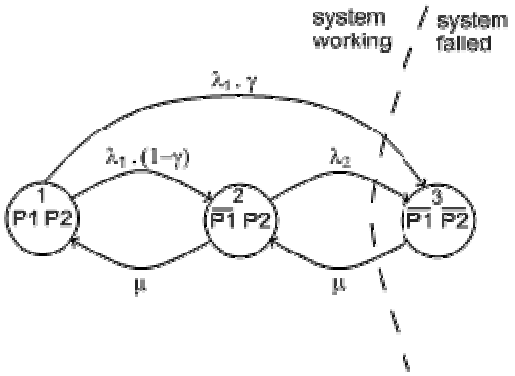

Figure 2.5. A model equivalent to the model of Fig. 2.4

2.5.2. Multi phase models

In Markov models, rates are kept constant through the time. It is sometimes convenient however to distinguish several phases of operations, with different rates and even different underlying graphs. This is exemplified, for instance, by systems with two alternating phases: normal operation and maintenance.

Mark-XPR makes it possible to deal with multi phase models. Each phase corresponds to a Markov graph. The same graph may be used for different phases. Phases are chained to create scenarii. Each phase has its own duration. Chaining phases *i* and *i+1* means to transfer probabilities of presence in states of the graph *i* at the end of the phase *i* into probabilities of presence in states of the graph *i+1* at the beginning of the phase *i+1*. In Mark-XPR, this transfer is described via so-called transition matrices. If the graph at phase *i* has *m* stochastic states and the graph at phase *i+1* has *n* (stochastic or immediate) states, the corresponding transition matrix is a *m×n* stochastic matrix such that the sum of probabilities on each row equals 1.

As an illustration consider a system with two phases, normal operation and maintenance. The system is shut down during maintenance. It has a failure rate λ and repair rate μ. It is tested at the beginning of the maintenance (this test is assumed to be perfect). The system is restarted at the end of the maintenance, with a probability of failure ω. Fig. 2.6 and 2.7 picture respectively the transitions from normal operation to maintenance and vice versa. States W, R and F stand respectively for Working, Repair and Failed.

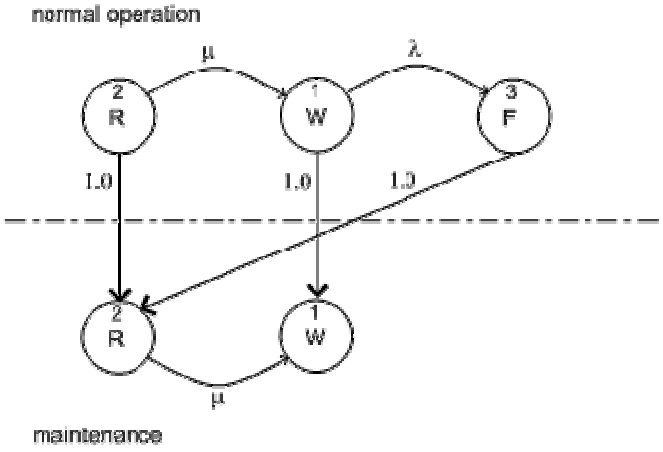

Figure 2.6. Transition from normal operation to maintenance

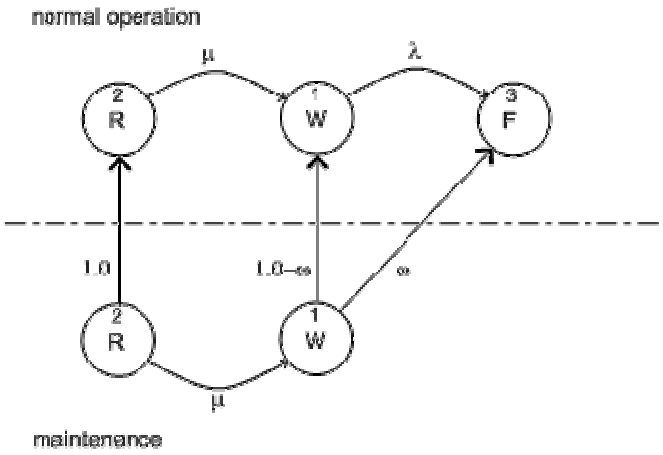

Figure 2.7. Transition from maintenance to normal

In terms of matrices, the transition pictured Fig. 2.6. and 2.7 are as follows.

From normal operation to maintenance	From maintenance to normal operation
$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1.0-\omega & 0.0 & \omega \\ 0.0 & 1.0 & 0.0 \end{pmatrix}$

3. Models

Mark-XPR model files and command files are both at the XML format. In model files, tags are used to declare items (number of graphs, states, transitions…) in advance. In that way, data structures can be allocated before they are used.

3.1. Introductive example and structure of descriptions

To start with consider the first model discussed in the previous section (and pictured Fig. 2.2). For the sake of the convenience, this model is redrawn Fig. 3.1.

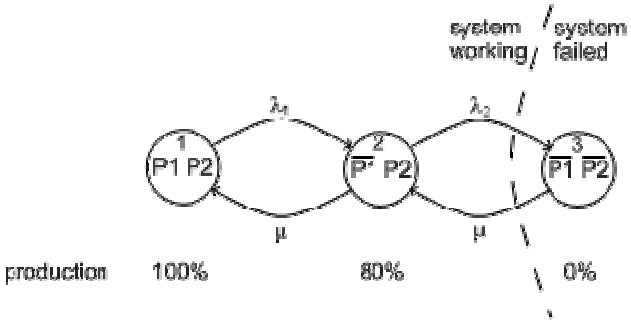

Figure 3.1. A Markov model.

A Mark-XPR description for the model pictured Fig. 3.1 is given Fig. 3.2.

Model files describe three types of objects: parameters, graphs and transition matrices. Therefore, the structure of a description is as follows.

```
<?xml version="1.0">
<!DOCTYPE xprmodel >
<xprmodel parameters="3" graphs="1" matrices="0">

  <!--
     description of parameters, graphs and transition matrices
  -->

</xprmodel >
```

The two first lines are just the header of the description. Numbers of parameters, graphs and transition matrices are given as tags in the outer most bloc.

Objects of Mark-XPR descriptions (parameters, graphs, transition matrices, states, transitions, properties...) are numbered from 1 to the number of objects of the same type in the corresponding bloc. Objects are always referred by their number (and there must be no unused numbers).

```
<?xml version="1.0"?>
<!DOCTYPE xprmodel>
<xprmodel parameters="3" graphs="1" matrices="0">

  <parameter number="1" name="lambda1" value="0.001" />
  <parameter number="2" name="lambda2" value="0.002" />
  <parameter number="3" name="mu"      value="0.1" />

  <graph number="1" name="pumps"
         states="3" transitions="4" properties="1">
    <states>
      <state number="1" name="P1P2" initial="1.0" />
      <state number="2" name="p1P2" />
      <state number="3" name="p1p2" />
    </states>
    <transitions>
      <transition number="1" source="1" target="2" parameter="1" />
      <transition number="2" source="2" target="1" parameter="3" />
      <transition number="3" source="2" target="3" parameter="2" />
      <transition number="4" source="3" target="2" parameter="3" />
    </transitions>
    <properties>
      <property number="1" name="production">
        <state number="1" value="100"/>
        <state number="2" value="80"/>
      </property>
    </properties>
  </graph>

</xprmodel>
```

Figure 3.2. A Mark-XPR description of the model pictured Fig. 3.1.

3.2. Graphs

3.2.1. Structure of graph descriptions

Markov graphs as handled by Mark-XPR are made, as expected, of states and transitions. Moreover, properties may be defined, as explained section 2.3.3. The numbers of states, transitions and properties must be given in advance, in order to allocate data structures. Hence the bloc that defines a graph opens with tags "`number`", "`states`", "`transitions`" and "`properties`". E.g.

```
<graph number="1" states="3" transitions="4" properties="1">
  ...
</graph>
```

The number of a graph ranges from 1 to the number of graphs in the description. The tag "name" is optional. The above piece of description declares a graph with three states, four transitions and one property.

3.2.2. States

States are numbered from 1 to the number of states in the graph (tag "`number`"). They may be named (tag "`name`"). They may also receive an initial probability (tag "`initial`"). The bloc that declares states comes first in the graph description. E.g.

```
<states>
  <state number="1" name="P1P2" initial="1.0" />
  <state number="2" name="p1P2" />
  <state number="3" name="p1p2" />
</states>
```

The declaration of immediate states is described section 3.2.5.

3.2.3. Transitions

The bloc that declares transitions comes in second in the graph description. E.g.

```
<transitions>
  <transition number="1" source="1" target="2" parameter="1" />
  <transition number="2" source="2" target="1" parameter="3" />
  <transition number="3" source="2" target="3" parameter="2" />
  <transition number="4" source="3" target="2" parameter="3" />
</transitions>
```

Transitions are numbered from 1 to the number of transitions in the graph (tag "`number`"). Each transition has a source state (tag "`source`") and a target state (tag "`target`"). Both must be legal state numbers. In a Markov graph, each transition must be given a rate. Mark-XPR provides the user with two different ways to associate a rate to a transition: either the rate (a floating point number) can be declared with the transition, or the transition can be labeled with a parameter. Different transitions may be labeled with the same parameter. In the above bloc, all the transitions are labeled with a parameter. A transition labeled with a rate would be as follows.

```
<transition number="1" source="1" target="2" value="1.0e-4" />
```

Floating point numbers can be given using the scientific notation, as illustrated above.

3.2.4. Properties

Mark-XPR computes, beyond the probabilities of presence and sojourn times in states, the expectations, the instantaneous variation, the instantaneous gain and the instantaneous loss of properties (see section 2.4). A property is a floating point number associated with each state of the graph (see section 2.3.3). The bloc that defines properties comes third (and last) in the graph description. E.g.

```
<properties>
  <property number="1" name="production">
    <state number="1" value="100"/>
    <state number="2" value="80"/>
  </property>
</properties>
```

Properties are numbered from 1 to the number of properties in the graph (tag "`number`"). A property may have a name (tag "`name`") and defines a value for each state. By default, the value of a property is 0.0. So, if a property is not defined in a given state, it takes the value 0 for that state. In our example, the following statement is therefore useless.

```
<state number="3" value="0.0" />
```

The bloc "`properties`" may be omitted if no property is defined (note that this is also the case for blocs "`states`" and "`transitions`" although a graph with no state or no transition is somehow meaningless).

By default, only the expectation of the property is computed. A Boolean attribute must be set to get the instantaneous variation, the instantaneous gain and the instantaneous loss. E.g.

```
<property number="1" name="failed"
  variation="yes" gain="yes" loss="yes">
  ...
</property>
```

If the attribute "expectation" is set to false, the expectation is not computed.

3.2.5. Immediate states

As explained section 2.4.1, Mark-XPR extends the pure Markov graph formalism by handling so called immediate states. Fig. 3.3 shows a model with such an immediate state (state 3).

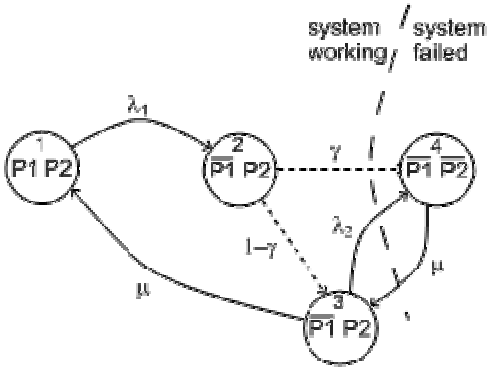

Figure 3.3. A model with an immediate state.

```
<?xml version="1.0"?>
<!DOCTYPE xprmodel>
<xprmodel graphs="1" matrices="0" parameters="5">

  <parameter number="1" name="lambda1" value="0.001" />
  <parameter number="2" name="lambda2" value="0.002" />
  <parameter number="3" name="mu"      value="0.1" />
  <parameter number="4" name="gamma"   value="0.01" />
  <parameter number="5">
     <sub> <constant value="1.0"/> <parameter number="4"/> </sub>
  </parameter>

  <graph number="1" name="pumps"
         states="4" transitions="6" properties="1">
    <states>
      <state number="1" name="P1P2" initial="1.0" />
      <state number="2" name="p1P2" immediate="yes" />
      <state number="3" name="p1P2" />
      <state number="4" name="p1P2" />
    </states>
    <transitions>
      <transition number="1" source="1" target="2" parameter="1" />
      <transition number="2" source="2" target="3" parameter="5" />
      <transition number="3" source="2" target="4" parameter="4" />
      <transition number="4" source="3" target="1" parameter="3" />
      <transition number="5" source="3" target="4" parameter="2" />
      <transition number="6" source="4" target="3" parameter="3" />
    </transitions>
    <properties>
      <property number="1" name="production">
        <state number="1" value="100"/>
        <state number="3" value="80"/>
      </property>
    </properties>
  </graph>
</xprmodel>
```

Figure 3.4. A Mark-XPR description of the model pictured Fig. 3.3.

A Mark-XPR description for the graph pictured Fig. 3.3 is given Fig 3.4.

In order to declare that a state is immediate, it suffices to add the tag "`immediate`" with the value "`yes`" in its declaration:

```
<state number="2" name="p1P2" immediate="yes" />
```

Transitions going out an immediate state are assumed to be labeled with probabilities and not rates. Moreover, the sum of their probabilities must equal 1.0. Otherwise, these transitions are declared the same way as stochastic transitions:

```
<transition number="2" source="2" target="3" parameter="5" />
<transition number="3" source="2" target="4" parameter="4" />
```

3.3. Transition matrices

Mark-XPR makes it possible to deal with multi phase models, see section 2.4.2. Transition matrices are used to transfer the probabilities from one graph to another when the phase of the computation changes. The probability of each stochastic state of the source graph is transferred into one or more states of the target graph. At the end of a phase, the probability to be in an immediate state always equals 0. Therefore immediate states of the source graph are not involved in transition matrices.

A transition matrix involves a source graph, a target graph and some transitions. Each transition has a number, a source state (in the source graph) and a target state (in the target graph) and a probability.

As an illustration consider again the example of section 2.4.2. It consists of a system with two alternating phases: normal operation and maintenance. Graphs and transitions matrices for this system are given Fig. 3.5 and 3.6.

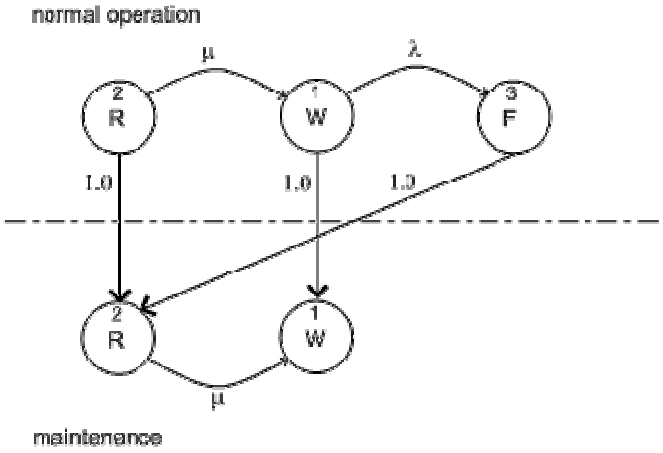

Figure 3.5. Transition from normal operation to maintenance

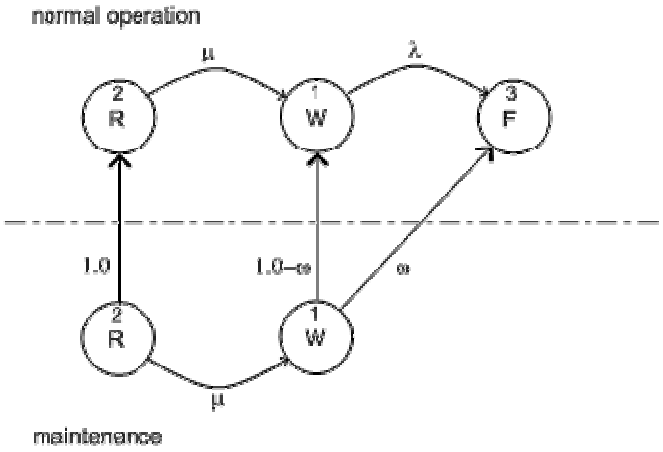

Figure 3.6. Transition from maintenance to normal

A Mark-XPR description for that model is given Fig. 3.7.

Each matrix has a number which is given by the tag "`number`". The number of a matrix ranges from 1 to the number of matrices in the description. The matrix description is opened with the numbers of its source and target graphs (tags "`source`" and "`target`") as well as and the number of transitions it is made of (tag "`transitions`"). A matrix may be named (tag "`name`"). The transitions follow. E.g.

```
<matrix number="1" source="1" target="2" transitions="3">
  <transitions>
    ...
  </transitions>
</matrix>
```

The syntax of transitions of matrices is essentially the same as the syntax of transitions of graphs. Each transition is identified with a unique number that ranges from 1 to the number of transitions in the matrix. Either values or parameters can be used to describe probabilities of transitions. E.g. the two last transitions of the second matrix.

```
<transition number="2" source="1" target="3" parameter="3" />
<transition number="3" source="2" target="2" value="1.0"   />
```

Note that parameters are global and can be used in different graphs and/or matrices.

```
<?xml version="1.0"?>
<!DOCTYPE xprmodel>
<xprmodel parameters="4" graphs="2" matrices="2">

  <parameter number="1" name="lambda" value="1.0e-4" />
  <parameter number="2" name="mu"     value="0.1"    />
  <parameter number="3" name="omega"  value="0.01"   />
  <parameter number="4">
    <sub> <constant value="1.0"/> <parameter number="3"/> </sub>
  </parameter>

  <graph number="1" name="operation"
    states="3" transitions="2" properties="0">
    <states>
      <state number="1" name="W" initial="1.0" />
      <state number="2" name="R" />
      <state number="3" name="F" />
    </states>
    <transitions>
      <transition number="1" source="1" target="3" parameter="1" />
      <transition number="2" source="2" target="1" parameter="2" />
    </transitions>
  </graph>

  <graph number="2" name="maintenance"
    states="2" transitions="1" properties="0">
    <states>
      <state number="1" name="W" />
      <state number="2" name="R" />
    </states>
    <transitions>
      <transition number="1" source="2"  target="1"  parameter="2" />
    </transitions>
  </graph>

  <matrix number="1" source="1" target="2" transitions="3">
    <transitions>
      <transition number="1" source="1" target="1" value="1.0" />
      <transition number="2" source="2" target="2" value="1.0" />
      <transition number="3" source="3" target="2" value="1.0" />
    </transitions>
  </matrix>

  <matrix number="2" source="2" target="1" transitions="3">
    <transitions>
      <transition number="1" source="1" target="1" parameter="4" />
      <transition number="2" source="1" target="3" parameter="3" />
      <transition number="3" source="2" target="2" value="1.0"   />
    </transitions>
  </matrix>

</xprmodel>
```

Figure 3.7. A Mark-XPR description for the model pictured Fig. 3.5 and 3.6.

3.4. Parameters

Parameters are declared at the beginning of the description. They can be used to set rates and probabilities in graphs and transition matrices. They can also be used to define other parameters. The description of Fig. 3.7 contains some parameter definitions:

```
<parameter number="1" name="lambda" value="1.0e-4" />
<parameter number="2" name="mu"     value="0.1"    />
<parameter number="3" name="omega"  value="0.01"   />
<parameter number="4">
  <sub> <constant value="1.0"/> <parameter number="3"/> </sub>
</parameter>
```

A parameter is uniquely referred by a number (tag "number") that ranges from 1 to the number of parameters in the description. It may be named (tag "name"). Parameter definitions are in one the two forms given above. Either a value (a floating point number) is given to the parameter with the tag "value", e.g. parameters lambda, mu and omega. Or the parameter is defined by an arithmetic expression. Arithmetic expressions are built using the following constructs.

- <constant value="v">, where v is any floating point number.
- <parameter number="n">, where n is any valid parameter number. Note that this should introduce no loop in the definitions.
- <add> exp_1 ... exp_n </add>, to get the sum of any number of expressions.
- _{exp_1 exp_2}, to get the difference of two expressions.
- <mul> exp_1 ... exp_n </mul>, to get the product of any number of expressions.
- exp_1 exp_2 , to get the quotient of two expressions.

Consider, for instance, the rate "$\lambda_1.(1-\gamma)$" of the transition from state 1 to state 2 of the graph pictured Fig. 2.5. The definition of a parameter that encodes this rate could be as follows.

```
<parameter number="1" name="lambda1" value="1.0e-4" />
<parameter number="2" name="gamma"   value="0.01"   />
<parameter number="3" name="lambda1*(1-gamma)" >
  <mul>
    <parameter number="1"/>
    <sub> <constant value="1.0"/> <parameter number="2"/> </sub>
  </mul>
</parameter>
```

3.5. Short cuts

Most of keywords have a short version that can be used instead of the full keyword. Table 3.8 gives the short version of keywords. As an illustration, Fig. 3.9 gives the

description of the second matrix of Fig. 3.7. using short cuts rather than full keywords.

Keywords	Shortcuts
constant	cst
expectation	xpc
gain	gan
graph	grp
immediate	imm
initial	ini
loss	lss
matrix	mat
name	nam
number	num
parameter	prm
property	prp
source	src
state	stt
target	tgt
transition	trs
value	val
variation	vrt

Table 3.8. Shortcuts for keywords

```
<matrix number="2" source="2" target="1" transitions="3">
  <transitions>
    <trs num="1" src="1" tgt="1" prm="4"   />
    <trs num="2" src="1" tgt="3" prm="3"   />
    <trs num="3" src="2" tgt="2" val="1.0" />
  </transitions>
</matrix>
```

Figure 3.9. Description of the second transition matrix of Fig. 3.7 using short cuts.

4. Commands

This chapter describes the syntax of command files.

4.1. Structure of command files

Mark-XPR command files define one or more independent series of computations called histories. The structure of a command file is illustrated 4.1.

```
<?xml version="1.0">
<!DOCTYPE xprstudy>
<xprstudy>

  <history>
     <!-- definition of 1st the history -->
  </history>

  <history>
     <!-- definition of 2nd the history -->
  </history>

  <!-- and so on ... -->

</xprstudy>
```

Figure 4.1. Structure of the XML description of a Mark-XPR command file

Inside each bloc "`history`", commands are called to create scenarios. The current version of Mark-XPR provides four commands:

- The command "`compute`" that performs a computation on a graph. E.g.

```
<compute method="EXP" dtratio="0.1" graph="1" />
```

- The command "`let`" that is used to set parameters. This command acts like parentheses. E.g.

```
<let method="EXP" dtratio="0.1" graph="1">
  <!-- commands in which method is set by default to EXP,
       dtratio to 0.1 and graph to 1 -->
</let>
```

- The command "`transfer`" that applies a transition matrix to a source and a target graph. E.g.

```
<transfer matrix="1" />
```

- The command "`repeat`" that is used to iterate a series of computations. E.g.

```
<repeat iterations="12">
  <!-- what is to be repeated -->
</repeat>
```

- The command "`print`" is used to print results. E.g.

```
<print properties="yes" states="no" />
```

- The command "`aggregate`" is used to aggregate results obtained on different graphs. E.g.

```
<aggregate property="2" />
```

In the following sections, we shall show how to use these commands to create complex scenarios.

4.2. Basic scenarios: commands "`compute`" and "`print`"

Consider the model of Fig. 3.1 and 3.2. Assume that we want to assess it on a one year (=8760 hours) period. To do so we shall create a history with two commands: a command "`compute`" to perform the computation and a command "`print`" to print the results. A command file that defines such a history is given Fig. 4.2.

```
<?xml version="1.0">
<!DOCTYPE xprstudy>
<xprstudy>
  <history>
    <compute graph="1" method="EXP" dtratio="0.1" duration="8760" />
    <print graph="1" states="yes" properties="yes" />
  </history>
</xprstudy>
```

Figure 4.2. A basic command file

The command file of Fig. 4.2 first applies the matrix exponentiation method (tag "method") to compute transient probabilities of graph number 1 (tag "graph"). The transient probabilities are computed at t=8760 (tag "duration"). The tag "dtratio" is used to set the time step. A complete description of available algorithms is given in the next chapter. Their various options and parameters are summarized section 5.5. The command "print" is then used to display the results. Since, tags "states" and "properties" are set to "yes", it prints for transient probabilities and sojourn times as well as point and cumulated expectations of properties (see sections 2.2 and 2.3 for a precise definition of these notions). The output of Mark-XPR for the description of Fig. 3.2 and the command file of Fig 4.2 is given Fig. 4.3.

```
File generated by Mark-XPR version 1.1
Date      Fri Oct 21 14:28:24 2033
Model file      pumps0.xml
Command file    mission0.xml


Period: 0 --> 8760 [8760] probabilities and sojourn times
-------------------------------------------------
graph     state probability sojourn times
                            period      cumulated
-------------------------------------------------
pumps     P1P2  0.989903    8671.65     8671.65
pumps     p1P2  0.00989903  86.6155     86.6155
pumps     p1p2  0.000197981 1.73033     1.73033
-------------------------------------------------
pumps     all   8760  1     8760  8760
-------------------------------------------------


Period: 0 --> 8760 [8760] expectations
-------------------------------------------------
graph     property    expectations
                      point period      cumulated
-------------------------------------------------
pumps     production  99.7822     99.7825     99.7825
-------------------------------------------------

Global running time  0
bye
```

Figure 4.3. Output of Mark-XPR for the model of Fig. 3.2 and the commands of Fig. 4.2.

Note that the graph on which commands apply must be given with each command (tag "graph"). Note also that, in that case, the history consists of single period. Therefore, sojourn times and expectations for the last period and cumulated from the origin are the same.

It is possible to get some information about the computation by adding the tag "verbose="yes"" to the command "compute". In that case, a banner is displayed once the computation is done, which looks like the following.

```
Transient states computed with the Matrix Exponentiation method (EXP).
dt ratio  0.1
epsilon   1.0e-9
Current date    8760
Convergence date      226.45
#vector operations    4621
#matrix operations    982
```

Note that, in that case, the computation converged at t=226.45.

4.3. Getting intermediate results: commands "let" and "repeat"

Assume now that we want 3 intermediate results at t=2190, t=4380 and t=6570. The command file to obtain these results could be as shown Fig. 4.4.

```
<?xml version="1.0">
<!DOCTYPE xprstudy>
<xprstudy>
  <history>
    <compute graph="1" method="EXP" dtratio="0.1" duration="2190" />
    <print graph="1" states="no" properties="yes" />
    <compute graph="1" method="EXP" dtratio="0.1" duration="2190" />
    <print graph="1" states="no" properties="yes" />
    <compute graph="1" method="EXP" dtratio="0.1" duration="2190" />
    <print graph="1" states="no" properties="yes" />
    <compute graph="1" method="EXP" dtratio="0.1" duration="2190" />
    <print graph="1" states="no" properties="yes" />
  </history>
</xprstudy>
```

Figure 4.4. A (rough) command file to get intermediate results.

However, the command file of Fig. 4.4 can be simplified in two ways.
First, the command "let" can be used to set the various parameters. The command "let" gives default values to the tags it contains. These default values are set in all the inner blocs of the command. In that way, tags need not to be repeated.
Second, since the four successive periods are identical, the command "repeat" can be used to repeat the same period four times. The tag "iterations" is used to set the number of iterations of the inner commands.
The simplified version of the command file is given Fig. 4.5.

```
<?xml version="1.0"?>
<!DOCTYPE xprstudy>
<xprstudy>
  <history>
    <let graph="1"
         method="EXP" dtratio="0.1" duration="2190"
         states="no" properties="yes" >
      <repeat iterations="4">
        <compute />
        <print />
      </repeat>
    </let>
  </history>
</xprstudy>
```

Figure 4.4. A simplified command file to get intermediate results.

The output of Mark-XPR for that command file is given Fig. 4.5.

```
File generated by Mark-XPR version 1.1
Date      Fri Oct 21 14:28:24 2033
Model file      pumps0.xml
Command file    mission1.xml


Period: 0 --> 2190 [2190] expectations
-------------------------------------------------
graph     property    expectations
                      point period      cumulated
-------------------------------------------------
pumps     production  99.7822     99.7833     99.7833
-------------------------------------------------



Period: 2190 --> 4380 [2190] expectations
-------------------------------------------------
graph     property    expectations
                      point period      cumulated
-------------------------------------------------
pumps     production  99.7822     99.7822     99.7828
-------------------------------------------------



Period: 4380 --> 6570 [2190] expectations
-------------------------------------------------
graph     property    expectations
                      point period      cumulated
-------------------------------------------------
pumps     production  99.7822     99.7822     99.7826
-------------------------------------------------



Period: 6570 --> 8760 [2190] expectations
-------------------------------------------------
graph     property    expectations
                      point period      cumulated
-------------------------------------------------
pumps     production  99.7822     99.7822     99.7825
-------------------------------------------------

Global running time  0
bye
```

Figure 4.5. Output of Mark-XPR for the model of Fig. 3.2 and the commands of Fig. 4.4.

4.4. Multi phase models: commands "`transfer`" and "`aggregate`"

Consider now the system presented Fig. 3.5, 3.6 and 3.7. It has two alternating phases (normal operation and maintenance) described by two different graphs (and two transition matrices). We may assume that the maintenance phase takes 4 hours a day (think for instance to a subway). A command file to assess that model over a one year period could be as given Fig. 4.6.

```
<?xml version="1.0"?>
<!DOCTYPE xprstudy>
<xprstudy>
  <history>
    <let method="EXP" dtratio="0.01" verbose="no">
      <repeat iterations="365">
        <repeat iterations="20">
          <compute graph="1" duration="1" />
          <print graph="1" properties="no" states="yes" />
        </repeat>
        <transfer matrix="1" />
        <repeat iterations="4">
          <compute graph="2" duration="1" />
          <print graph="2" properties="no" states="yes" />
        </repeat>
        <transfer matrix="2" />
      </repeat>
    </let>
  </history>
</xprstudy>
```

Figure 4.6. Mark-XPR command file for the system with two alternating phases.

The command file of Fig. 4.6 defines a history. The "`let`" bloc defines some parameters. The outer most "`repeat`" bloc is used to iterate a one day period through one year. Inside that bloc, there is a first "`repeat`" bloc to assess the first graph (that represents the normal operation phase) each hour through twenty hours. Then a call to a transition matrix (command "`transfer`") transfers the probabilities from the first graph to the second one. Then a second "`repeat`" bloc assesses the second graph (that represents the maintenance phase) each hour for four hours. Finally, a call to the second transition matrix transfers back the probabilities to the first graph. This command file therefore invokes the "`compute`" command 365 × (20 + 4) times.

Assume that we add to the description of Fig. 3.7 the definition of a property "`unavailable`" that takes the value 1 on every state but the state "`W`" of the first graph (where it takes the value 0). The description is modified sketched Fig 4.7.

```
...
<graph number="1" name="operation"
  states="3" transitions="2" properties="1">
  ...
  <properties>
    <property number=”1” name=”unavailable”>
      <state number=”2” value=”1”>
      <state number=”3” value=”1”>
    </property>
  </properties>
</graph>
...
<graph number="2" name="operation"
  states="3" transitions="2" properties="1">
  ...
  <properties>
    <property number=”1” name=”unavailable”>
      <state number=”2” value=”1”>
      <state number=”3” value=”1”>
    </property>
  </properties>
</graph>
...
```

Figure 4.7. Modifications of description of Fig. 3.7 to introduce a property

Now, it is of interest to get the global unavailability of the system and, for instance, the global time it is in repair. These measures can be obtained with the command “`aggregate`”. This command takes one parameter which is either a state number or a property number. It aggregates the sojourn times (or expectations) of this state or property over the different graphs.

For instance, the two following lines could be added at the end of the history defined by command file of Fig. 4.6.

```
<aggregate state=”2”/>
<aggregate property=”1”/>
```

The Mark-XPR output for these two commands is as shown Fig. 4.8. When applied to a state, the command “`aggregate`” gives the cumulated sojourn times in the graphs in which the state shows up and in the state itself. When applied to a property, it gives the cumulated sojourn times in the graphs in which the property is defined and the cumulated expectation of the property.

```
Aggregated state 2 [8760]
-------------------------------------------------
          sojourn times
graph     in graph    in state 2
-------------------------------------------------
operation 7300  27.5017
maintenance     1460  15.6862
-------------------------------------------------
total     8760  43.1878
-------------------------------------------------


Aggregated property 1 [8760]
-------------------------------------------------
          sojourn times     expectation
graph     in graph    of property 1
-------------------------------------------------
operation 7300  0.0146373
maintenance     1460  0.0107439
-------------------------------------------------
total     8760  0.0139884
-------------------------------------------------
```

Figure 4.8. Mark-XPR output for the command "`aggregate`"

4.5. Drawing curves: the command "`plot`"

It is sometimes of interest to draw curves of values of properties. The command "`plot`" saves all the points computed for a property into a file.

```
<plot property="failed" file="failed.txt" >
  commands to compute values
</plot>
```

In the above example, all the points computed for the property "`failed`" into the file "`failed.txt`". These points are saved in two columns, one for the date, the other for values.

By default, the plotted value is the expectation of the property. By setting the attribute "observation" to "variation", "gain" or "loss", it is possible to plot the corresponding values (see section 2.4). Indeed, these values can be plotted only if they are observed, which it turns requires that they have been declared with the property (see section 3.2.4).

4.6. Summary of Mark-XPR commands

The table 4.9 summarizes Mark-XPR commands and their options. The table 4.10 summarizes options and their potential values.

Commands	Options (tags)
history	none
compute	graph, method, duration, dtratio, espilon, verbose
print	graph, states, properties
let	all tags
repeat	iterations
transfer	matrix
aggregate	state, property
plot	property, file, observation

Table 4.9. Summary of Mark-XPR commands and their options.

Options (tags)	Commands	Values
duration	compute, let	floating point number
dtratio	compute, let	floating point number
epsilon	compute, let	floating point number
graph	compute, print, let	valid graph number
iterations	repeat, let	integer
matrix	transfer	valid matrix number
method	compute, let	EXP, FEM, RK2, RK4, AB2, AB4, CGS, BCG
observation	plot	expectation, variation, gain, loss
property	aggregate	valid property number
properties	print, let	yes, no
state	aggregate	valid state number
states	print, let	yes, no
verbose	compute, let	yes, no

Table 4.10. Summary of Mark-XPR options and their values.

5. Algorithms

Throughout this chapter, we shall use the same notations as in chapter 2.

5.1. Introduction

Consider a Markov process with n states. Let Q be the $n \times n$ matrix whose elements $q_{i,j}$ denote the rate of transition of the chain from state i to state j. Let $\pi(t)$ be the vector of length n of probabilities to be in state i at time t. $\pi(0)$ thus denotes the vector of initial probabilities. It can be shown (see section 2, equation 2.7) that,

$$\pi(t) = e^{t.Q}.\pi(0) \tag{5.1}$$

where $e^{t.Q}$ is defined as follows.

$$e^{t.Q} = \lim_{n\to\infty} \sum_{k=0}^{n} \frac{(t.Q)^k}{k!} \tag{5.2}$$

In the case of Markov chains, this series converges, at least theoretically. The problem is twofold. First, in order to assess $\pi(t)$ efficiently one should perform only products of matrices by vectors. Second, because of rounding errors, coefficients of matrices should be kept small (preferably between 0 and 1). To tackle this problem, two ideas can be applied [Ste94].

First, the following equality holds.

$$\frac{(t.Q)^k}{k!} \times \pi = \frac{t.Q}{k} \times \left(\frac{(t.Q)^{k-1}}{(k-1)!} \times \pi \right) \tag{5.3}$$

Therefore, the k-th term of the series (2) can be obtained from the $(k-1)$-th term.

Second, the following equality holds.

$$e^{t.Q} \times \pi = e^{dt.Q} \times \left(e^{(t-dt).Q} \times \pi \right) \tag{5.4}$$

Therefore, the computation of $e^{t.Q}.\pi(0)$ can be replaced successive computations of $\pi(t_i)=e^{dti.Q}.\pi(t_{i-1})$ such that $t_0=0$, $t_i=t_{i-1}+dt_i$, $dt_1+dt_2+\ldots=t$ and the dt_i's are small.

Combining these two ideas lead to several algorithms. These algorithms differ on the way they assess $e^{dt.Q}.\pi$. The choice of dt is also an important issue.

5.2. The basic algorithmic scheme

Equalities (5.1) to (5.4) can be used to define the following algorithmic scheme

$$
\begin{array}{l}
select\ \pi(0) \\
\pi \leftarrow \pi(0) \\
t \leftarrow 0 \\
while\ t < T\ do \\
\quad select\ dt \\
\quad t \leftarrow t + dt \\
\quad \pi \leftarrow e^{dt.Q}.\pi
\end{array}
\qquad (5.5)
$$

Where T denotes the mission time and Q stands for the infinitesimal generator of the problem.

To make the scheme (5.5) a concrete algorithm, it remains to answer the following questions.

- How to select $\pi(0)$? In Mark-XPR, $\pi(0)$ is up to the user. So, we won't discuss this point here.
- How to select dt ? We shall discuss this major issue below.
- How to assess $e^{dt.Q}.\pi$? The different ways to assess this quantity define the different algorithms implemented in Mark-XPR.

If T is large enough, the stationary probabilities may be reached before the loop is completed. It is therefore a good idea to introduce a convergence test. This test aims to exit the loop as soon as two consecutive values of π can be considered as sufficiently close one another. In Mark-XPR, this is achieved as follows. We select a threshold ε. The loop is exited if the following inequality holds (assuming $\pi_i(t+dt) \neq 0$).

$$
\max_i \left(\frac{\left| \pi_i(t) - \pi_i(t+dt) \right|}{\pi_i(t+dt)} \right) \leq \varepsilon \qquad (5.6)
$$

where π_i denotes the i-th value in the vector π. Other norms can be defined to test the convergence [QSS00]. This one works fine. ε must be chosen quite low (say 10^{-9}) in order not to stop too early. The tag "`epsilon`" is used to set this parameter.

5.3. Algorithms to compute transient solutions

Full matrix exponentiation: The first algorithm based on scheme (5.5) we can consider consists in computing successively the terms of the series (5.2) until the

convergence criterion (5.6) is satisfied. This algorithm, so-called (full) matrix exponentiation, is denoted by "EXP". It is often considered as too costly [Ste94]. This is the reason why approximations of $e^{dt.Q}.\pi$ have been suggested.

Forward Euler Method: The first approximation consists in remarking that, since *dt* has anyway to be small, the whole series can be approximated by its first two terms.

$$e^{dt.Q}.\pi \approx \pi + dt.Q.\pi \tag{5.7}$$

This method, so called forward Euler method in the literature, has a bad reputation. Many textbooks claim that it is dubious because *dt* must be chosen very small to achieve a good accuracy. None of the experiments we performed confirms this claim. This method is denoted by "FEM".

Explicit Runge-Kutta methods: Runge-Kutta methods are probably the most popular methods to solve ordinary differential equations [PTVF95,SAP97]. Mark-XPR implements the Runge-Kutta methods of order 2 and 4, denoted by "RK2" and "RK4". RK2 is as follows (the reader may refer to references [Ste94], [PTVF95] and [SAP97] for a mathematical justification of RK2 and RK4).

$$e^{dt.Q}.\pi \approx \pi + \frac{1}{2}(\kappa_1 + \kappa_2) \tag{5.8}$$

with

$$\begin{aligned} \kappa_1 &= dt.Q.\pi \\ \kappa_2 &= dt.Q.(\pi + \kappa_1) \end{aligned}$$

RK4 is as follows.

$$e^{dt.Q}.\pi \approx \pi + \frac{1}{6}(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4) \tag{5.9}$$

with

$$\begin{aligned} \kappa_1 &= dt.Q.\pi \\ \kappa_2 &= dt.Q.(\pi + \frac{1}{2}\kappa_1) \\ \kappa_3 &= dt.Q.(\pi + \frac{1}{2}\kappa_2) \\ \kappa_4 &= dt.Q.(\pi + \kappa_3) \end{aligned}$$

Explicit Adams-Bashforth multi-steps methods: The underlying idea of multi-steps methods is orthogonal to the one of Runge-Kutta methods. The value of $\pi(t+dt)$ is computed from the values of $\pi(t)$, $\pi(t-dt)$, $\pi(t-2.dt)$ …$\pi(t-k.dt)$.
The Adams-Bashforth method of order 2 ("AB2") is as follows.

$$\pi(t+dt) = \pi(t) + \frac{1}{2}\left(3.\rho(t) - \rho(t-dt)\right) \quad (5.10)$$

where $\rho(s)$ denotes $dt.Q.\pi(s)$.
The Adams-Bashforth method of order 4 ("AB4") is as follows.

$$\pi(t+dt) = \pi(t) + \frac{1}{24}\left(55.\rho(t) - 59.\rho(t-dt) + 37.\rho(t-2.dt) - 9.\rho(t-3.dt)\right) \quad (5.11)$$

See references [Ste94], [QSS00] and [SAP97] for mathematical justifications.

Coccoza-Eymard-Mercier method: This method is very close to the Forward Euler Method. The idea is to substitute the matrix dt.Q by the matrix Q' defined as follows.

$$q'_{i,j} = \begin{cases} 1 - e^{-dt.q_{i,i}} & \text{if } i = j \\ (1 - e^{-dt.q_{i,i}}).\dfrac{q_{i,j}}{q_{i,i}} & \text{if } i \neq j \end{cases} \quad (5.12)$$

where $q_{i,i}$ is the sum of the $q_{i,j}$ ($i \neq j$).

This method is denoted by "CEM".

5.4. Discussion

The methods presented above are said explicit for the value of $\pi(t+dt)$ is computed from the value of $\pi(t)$. In implicit methods, the value of $\pi(t+dt)$ is defined by a relation between $e^{dt.Q}.\pi(t)$ and $\pi(t)$. A system of equations has to be solved to get it. An iteration of an implicit method is therefore much more costly than one of an explicit method. Experiments we performed showed that it is very dubious that, in practice, implicit methods can be more efficient than explicit ones.
The order of an explicit method characterizes the error it makes. A method is of order k if the error is in $O(dt^k)$. It can be shown [Ste94,SAP97] that the forward Euler method is of order 1, while Runge-Kutta methods presented above are respectively of order 2 and 4 (hence their names). It is often claimed that the estimation of the error is a major issue of the computation of transient solutions. However, according to reference [Ste94], methods proposed in the literature can estimate only the local error, i.e. the error made by one step of the algorithm. Just summing up the local

errors is certainly a very rough approximation of the global error. No satisfactory method has been proposed to estimate the global error.

The algorithmic scheme (5.5) assumes that a new value is selected for *dt* at each iteration. This value depends indeed on *Q* and possibly on π. If *dt* depends only on *Q*, then it remains constant through the whole process. *dt.Q* can be computed once for all, which saves a lot of multiplications. If *dt* depends also on π, then its value must be actually computed at each iteration. To be interesting, such a computation must save more time than it costs. We didn't see any heuristics that could justify this overhead. So all the algorithms implemented in Mark-XPR work with constant *dt's*.

The choice of an appropriate *dt* is a major issue of the implementation of iterative methods. On the one hand, if *dt* is chosen too large, either the approximation of $e^{dt.Q}.\pi$ is too rough (for `FEM`, `RK2`, `RK4`, `AB2` and `AB4` methods), or the coefficients of the matrix are too large and rounding errors make the computation diverge. On the other hand, if *dt* is chosen too small, the computation is very expensive. Moreover, as noticed in reference [SAP97], the impact of rounding errors may increase as the number of operations increases.
If all the coefficients of the matrix *dt.Q* range between 0 and 1, the expectation that rounding errors cause problems is minored. Hence, the following rule of thumb can be applied: choose *dt* such the product $\rho = dt.\max_i(q_{i,i})$ is in the range [0,1]. The product ρ provides a mean to choose *dt* in a uniform way:

$$dt = \frac{\rho}{\max_i q_{i,i}} \qquad (5.13)$$

Note that equality (5.13) depends only on the matrix *Q* (therefore *dt* can be kept constant through the computation). It is worth noticing that, strictly speaking, to make the system solvable, the matrix *I+qt.Q* must be stochastic, which in turn means that ρ must be smaller than 1. In practice however, values greater than *1* can sometimes be used, even with much care. The value of ρ is set through the tag `dtratio`.

Once the value of *dt* (or equivalently of ρ) is selected, the value of $\pi(t)$ can be assessed using any of the six algorithms presented above. However, it is not possible to estimate the error with a single run. References [Ste94] and [PTVF95] suggest to perform a second calculation with a smaller value for *dt*, say *dt/2*. If the two results are not close enough, the process is reiterated. It is worth noticing that one of the arguments in favour of the Runge-Kutta methods is that they can come with some means to estimate the error [Ste94, QSS00, SAP97]. We found this argument of a little help in practice for rounding errors make this estimation too rough to be actually useful.

5.5. Steady state probabilities

The steady state probabilities are reached when the derivative of the differential equation that rules the Markov process equals 0. In other words, steady state probabilities are the solution of the following equation.

$$Q.\pi(\infty) \quad = \quad 0 \tag{5.14}$$

From a theoretical viewpoint, in order to solve this equation, it suffices to invert the matrix Q (in which a line of 1 has been substituted for any line). In practice, such an operation is intractable for it introduces too many non-zero cells.

Fortunately, there exist algorithms that work by iterations and make it possible to converge to the solution by making only products matrix × column vector.
Mark-XPR implements two such algorithms. They are described below. For a mathematical justification of these algorithms see the Stewart's book or the "Numerical Recipes in C".

Bi-Conjugate Gradient Method ("BCG"):

$$\begin{aligned}
&\textit{select an initial solution } \pi \textit{ and a threshold } \varepsilon \\
&\tau \leftarrow -Q.\pi \\
&V \leftarrow \tau,\ \tilde{\upsilon} \leftarrow \tau, \tilde{\tau} \leftarrow \tau \\
&\textit{while } \tau^T.\tau > \varepsilon \\
&\quad \tau_p \leftarrow \tau, \quad \tilde{\tau}_p \leftarrow \tilde{\tau} \\
&\quad a \leftarrow \frac{\tilde{\tau}^T.\tau}{\tilde{\upsilon}^T.Q.\upsilon} \\
&\quad \tau \leftarrow \tau - a.Q.\upsilon \\
&\quad \tilde{\tau} \leftarrow \tilde{\tau} - a.Q^T.\tilde{\upsilon} \\
&\quad \pi \leftarrow \pi + a.\upsilon \\
&\quad b \leftarrow \frac{\tilde{\tau}^T.\tau}{\tilde{\tau}_p{}^T.\tau_p} \\
&\quad \upsilon \leftarrow \tau + b.\upsilon \\
&\quad \tilde{\upsilon} \leftarrow \tilde{\tau} + b.\tilde{\upsilon}
\end{aligned} \tag{5.15}$$

Conjugate Gradient Squared Method ("CGS") :

$$
\begin{aligned}
&\textit{select an initial value } \pi \textit{ and a threshold } \varepsilon \\
&\tau \leftarrow -Q.\pi \\
&\tilde{\tau} \leftarrow \tau, \quad \upsilon \leftarrow 0 \\
&\\
&\textit{while } \tau^T.\tau > \varepsilon \textit{ do} \\
&\quad r_p \leftarrow r, \quad r \leftarrow \tilde{\tau}^T.\tau, \quad b \leftarrow \frac{r}{r_p} \\
&\quad \nu \leftarrow \tau + b.Q \\
&\quad \upsilon \leftarrow \nu + b.(Q + b.\upsilon) \\
&\quad \omega \leftarrow Q.\upsilon \\
&\quad s \leftarrow \tilde{\tau}^T.\omega, \quad a \leftarrow \frac{r}{s} \\
&\quad \upsilon \leftarrow \nu - a.\omega \\
&\quad \tau \leftarrow \tau - a.Q.(\nu + \upsilon) \\
&\quad \pi \leftarrow \pi + a.(\nu + \upsilon)
\end{aligned}
\tag{5.16}
$$

It must noticed that these algorithms do not work if the graph is acyclic (i.e. if there exists a state *i* and a state *j* such that *j* is not reachable from *i*). Moreover, they cannot be applied to graphs with immediate states.

5.6. Summary

The six available algorithms to compute transient probabilities are recalled table 5.1. These algorithms are invoked by using the command "`compute`". E.g.

```
<compute method="EXP" duration="8760" dtratio="0.1" graph="1" />
```

Options and parameters of these algorithms are recalled table 5.2.

Keywords	Methods
FEM	Forward Euler method
EXP	Matrix Exponentiation
RK2	Explicit Runge-Kutta method of order 2
RK4	Explicit Runge-Kutta method of order 4
AB2	Explicit Adams-Bashforth method of order 2
AB4	Explicit Adams-Bashforth method of order 4
CEM	Coccoza-Eymard-Mercier method

Table 5.1. Available algorithms to compute transient probabilities

Tags	Meanings
`method`	Algorithm to be invoked
`graph`	Number of the graph on which the algorithm is to be invoked
`duration`	Mission time
`dtratio`	Parameter ρ described by equation 5.12. This parameter is used to set the time step.
`epsilon`	Parameter ε of inequality 5.6. This parameter is used to determine when a computation has reached its fixpoint
`verbose`	To print a banner with information (such as the time step, the convergence date ...) at the end of the computation (the value of this tag should be "`yes`" or "`no`").

Table 5.2. Options of the algorithms to compute transient probabilities

Algorithms to compute steady state probabilities are recalled table 5.3. These algorithms are invoked by using the command "`compute`". E.g.

```
<compute method="BCG" graph="1" />
```

Options and parameters of these algorithms are the same as those of algorithms to compute transient probabilities. The mission time (tag "`duration`") is used to compute sojourn times.

Keywords	Methods
`BCG`	Bi-Conjugate Gradient Method
`CGS`	Conjugate Gradient Squared Method

Table 5.3. Available algorithms to compute steady state probabilities

6. References

[PG80] A. Pagès and M. Gondrand. *Fiabilité des systèmes*, volume 39 of *Collection de la Direction des études et Recherches d'électricité de France*. Eyrolles, 1980. ISSN 0399-4198.

[PTVF95] W. H. Press, S. A. Teukolski, W. T. Vettering and B. P. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1995. ISBN 0-521-43108-5.

[QSS00] A. Quateroni, R. Sacco and F. Saleri. *Méthodes numériques pour le calcul scientifique*. Springer Verlag, 2000. ISBN 2-287-59701.

[SAP97] L.F. Shampine, R.C. Allen and Jr. S. Pruess. Fundamentals of Numerical Computing. John Wiley & Sons. 1997. ISBN 0-471-16363-5.

[Sig93] J.-P. Signoret. *Manuel d'utilisation des logiciels de calcul de processus de Markov MARK-EXD et MARK-SMP*. Rapport Technique Elf-Aquitaine Production. Référence EP/P/PRO/SES-JPS93032. 1993.

[Ste94] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994. ISBN 0-691-03699-3.

7. Appendix: a periodically tested component

7.1. The model

To illustrate the features of Mark-XPR, we treat here the case of a periodically tested component. Such a component may be in three states: either working, failed or in repair. We consider three phases:

1. The phase from time 0 to the date of the first test, i.e. θ.
2. The test phase whose duration is π.
3. The phase between tests. The delay between two consecutive tests is called τ.

Phases 2 and 3 are repeated over and over. Figure 7.1 illustrates this process.

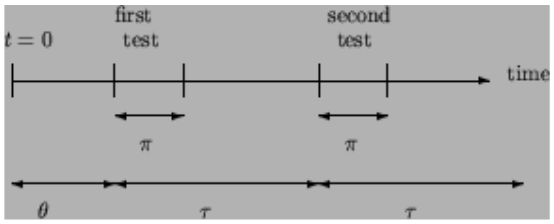

Figure 7.1. Phases of a periodically tested component.

The second phase spreads from times $\theta+n.\tau$ to $\theta+n.\tau+\pi$, with n any positive integer. The third phase spreads from times $\theta+n.\tau+\pi$ from $\theta+(n+1).\tau$.

The Markov graphs for these three phases and the transition matrices are pictured Fig. 7.2. The meanings of the symbols showing up on that figures are given table 7.3. Ai's , Fi's, Ri's states correspond respectively to states where the component is available, failed and in repair. Dashed lines correspond to immediate transitions.

The Mark-XPR description for the model of Fig. 7.2 is given 7.4.

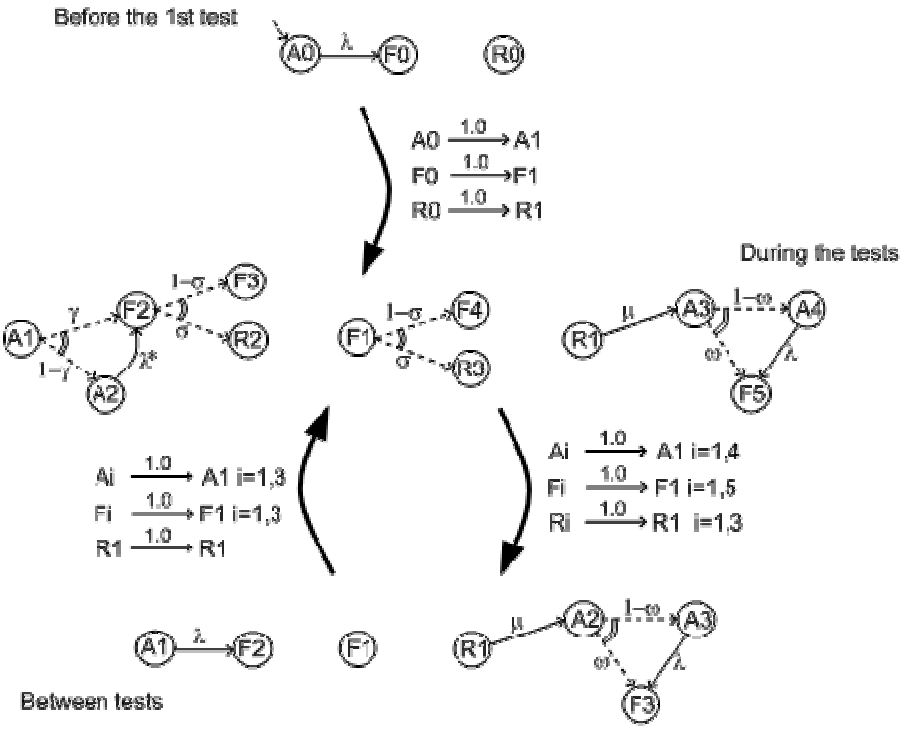

Figure 7.2. Markov graphs for test case

Parameters	Meanings	Values
λ	failure rate when the component is working.	1.0e-4
λ*	failure rate when the component is tested.	1.0e-5
μ	repair rate (once the test showed that the component is failed).	0.1
γ	probability of failure due to the (beginning of the) test.	0.01
σ	test covering: probability that the test detects the failure, if any.	0.01
ω	probability that the component is badly restarted after a test or a repair.	0.01

Table 7.3. Meanings and values of parameters

```
<?xml version="1.0"?>
<!DOCTYPE xprmodel>
<xprmodel parameters="9" graphs="3" matrices="3">

  <parameter number="1" name="lambda"      value="1.0e-4" />
  <parameter number="2" name="lambda-star" value="1.0e-5" />
  <parameter number="3" name="mu"          value="0.1"    />
  <parameter number="4" name="gamma"       value="0.01"   />
  <parameter number="5">
    <sub> <constant value="1.0"/> <parameter number="4"/> </sub>
  </parameter>
  <parameter number="6" name="sigma" value="0.99" />
  <parameter number="7">
    <sub> <constant value="1.0"/> <parameter number="6"/> </sub>
  </parameter>
  <parameter number="8" name="omega" value="0.01" />
  <parameter number="9">
    <sub> <constant value="1.0"/> <parameter number="8"/> </sub>
  </parameter>

  <graph number="1" name="init" states="3" transitions="1" properties="1">
    <states>
      <state number="1" name="A0" initial="1.0" />
      <state number="2" name="F0" />
      <state number="3" name="R0" />
    </states>
    <transitions>
      <transition number="1" source="1" target="2" parameter="1" />
    </transitions>
    <properties>
      <property name="unavailable" number="1">
        <state number="2" value="1" />
        <state number="3" value="1" />
      </property>
    </properties>
  </graph>
```

Figure 7.4 (a). The Mark-XPR description for a periodically tested component

```
<graph number="2" name="during-test"
  states="12" transitions="11" properties="1">
 <states>
   <state number="1" name="A1" immediate="yes" />
   <state number="2" name="A2" />
   <state number="3" name="A3" immediate="yes" />
   <state number="4" name="A4" />
   <state number="5" name="F1" immediate="yes" />
   <state number="6" name="F2" immediate="yes" />
   <state number="7" name="F3" />
   <state number="8" name="F4" />
   <state number="9" name="F5" />
   <state number="10" name="R1" />
   <state number="11" name="R2" />
   <state number="12" name="R3" />
 </states>

<transitions>
   <transition number="1" source="1"  target="6"  parameter="4" />
   <transition number="2" source="1"  target="2"  parameter="5" />
   <transition number="3" source="2"  target="6"  parameter="2" />
   <transition number="4" source="6"  target="7"  parameter="7" />
   <transition number="5" source="6"  target="11" parameter="6" />
   <transition number="6" source="5"  target="8"  parameter="7" />
   <transition number="7" source="5"  target="12" parameter="6" />
   <transition number="8" source="10" target="3"  parameter="3" />
   <transition number="9" source="3"  target="4"  parameter="8" />
   <transition number="10" source="3" target="9"  parameter="9" />
   <transition number="11" source="4" target="9"  parameter="1" />
 </transitions>

 <properties>
   <property name="unavailable" number="1">
     <state number="5"  value="1" />
     <state number="6"  value="1" />
     <state number="7"  value="1" />
     <state number="8"  value="1" />
     <state number="9"  value="1" />
     <state number="10" value="1" />
     <state number="11" value="1" />
     <state number="12" value="1" />
   </property>
 </properties>
</graph>
```

Figure 7.4 (b). The Mark-XPR description for a periodically tested component

```
<graph number="3" name="between-test"
  states="7" transitions="5" properties="1">

  <states>
    <state number="1" name="A1" />
    <state number="2" name="A2" immediate="yes" />
    <state number="3" name="A3" />
    <state number="4" name="F1" />
    <state number="5" name="F2" />
    <state number="6" name="F3" />
    <state number="7" name="R1" />
  </states>

  <transitions>
    <transition number="1" source="1" target="4" parameter="1" />
    <transition number="2" source="7" target="2" parameter="3" />
    <transition number="3" source="2" target="6" parameter="8" />
    <transition number="4" source="2" target="3" parameter="9" />
    <transition number="5" source="3" target="6" parameter="1" />
  </transitions

  <properties>
    <property name="unavailable" number="1">
      <state number="4" value="1" />
      <state number="5" value="1" />
      <state number="6" value="1" />
      <state number="7" value="1" />
    </property>
  </properties>

</graph>
```

Figure 7.4 (c). The Mark-XPR description for a periodically tested component

```
  <matrix number="1" source="1" target="2" transitions="3">
    <transitions>
      <transition number="1" source="1" target="1"  value="1.0" />
      <transition number="2" source="2" target="5"  value="1.0" />
      <transition number="3" source="3" target="10" value="1.0" />
    </transitions>
  </matrix>

  <matrix number="2" source="2" target="3" transitions="8">
    <transitions>
      <transition number="1" source="2"  target="1" value="1.0" />
      <transition number="2" source="4"  target="1" value="1.0" />
      <transition number="3" source="7"  target="4" value="1.0" />
      <transition number="4" source="8"  target="4" value="1.0" />
      <transition number="5" source="9"  target="4" value="1.0" />
      <transition number="6" source="10" target="7" value="1.0" />
      <transition number="7" source="11" target="7" value="1.0" />
      <transition number="8" source="12" target="7" value="1.0" />
    </transitions>
  </matrix>

  <matrix number="3" source="3" target="2" transitions="6">
    <transitions>
      <transition number="1" source="1" target="1"  value="1.0" />
      <transition number="2" source="3" target="1"  value="1.0" />
      <transition number="3" source="4" target="5"  value="1.0" />
      <transition number="4" source="5" target="5"  value="1.0" />
      <transition number="5" source="6" target="5"  value="1.0" />
      <transition number="6" source="7" target="10" value="1.0" />
    </transitions>
  </matrix>

</xprmodel>
```

Figure 7.4 (d). The Mark-XPR description for a periodically tested component

7.2. Study

Assume that the system is to be studied over a 5 year period (43800 hours), with a delay before the first test θ of 4380 hours, a delay between tests τ of 8760 hours and a 24 hours test phase.

Tests can performed for instance with the EXP method with the parameter ρ (`dtratio`) set to 0.01. The command file for such a test is given Fig. 7.5.

The command file of Fig. 7.5 first completes the first phase (before the first test) drawing a point each 6 hours. Then, a one year period is repeated 5 times. Each one year period consists of a test phase of 24 hours, then a phase between tests of 8736 hours. In both cases, a point is drawn each 6 hours.

Results obtained with Mark-XPR have been checked against those obtained with the fault tree analysis tool Aralia that implements a basic event distribution for periodically tested components. Fig. 7.6 shows the curves obtained with Mark-XPR and Aralia. Aralia results have been shifted to the right in order not to have the same curve for both tools.

```
<?xml version="1.0"?>
<!DOCTYPE xprstudy>
<xprstudy>
  <history>
    <let method="EXP" dtratio="0.01" verbose="no">
      <repeat iterations="730">
        <compute graph="1" duration="6" />
        <print graph="1" properties="yes" states="no" />
      </repeat>
      <transfer matrix="1" />
      <repeat iterations="5">
        <repeat iterations="4">
          <compute graph="2" duration="6" />
          <print graph="2" properties="yes" states="no" />
        </repeat>
        <transfer matrix="2" />
        <repeat iterations="1456">
          <compute graph="3" duration="6" />
          <print graph="3" properties="yes" states="no" />
        </repeat>
        <transfer matrix="3" />
      </repeat>
    </let>
    <aggregate property=”1”>
  </history>
</xprstudy>
```

Figure 7.5. A command file for the study of a periodically tested component

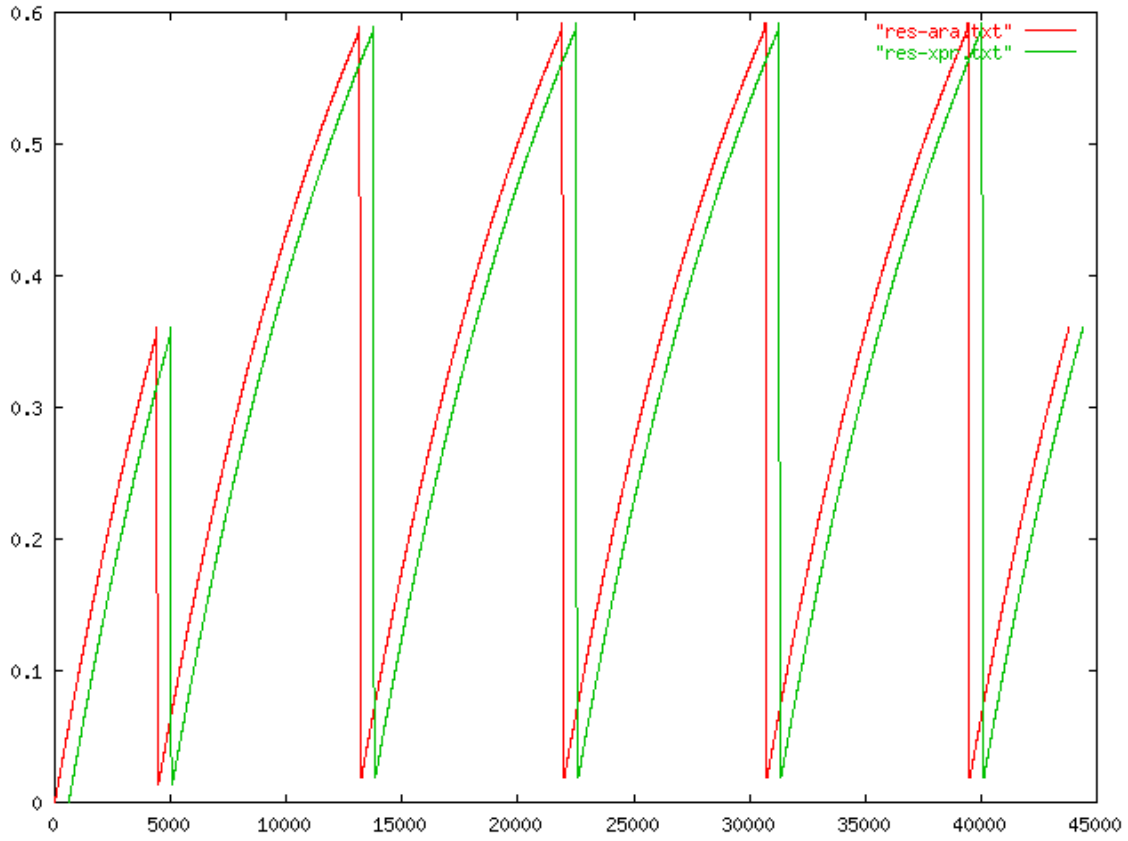

Figure 7.6. Curves drawn from Aralia and Mark-XPR results (the curve of Aralia results has been shifted to the right in order not to have the same curve for both tools).

The “aggregate” command that ends the history makes it possible to compute the cumulated expectation of the unavailability over the different phases of the five year period. The results are presented on Fig. 7.7.

```
Aggregated property 1 [48180]
-------------------------------------------------
          sojourn times     expectation
graph     in graph    of property 1
-------------------------------------------------
init      4380  0.190241
during-test     120   0.544999
between-test    43680 0.340644
-------------------------------------------------
total     48180 0.32748
-------------------------------------------------
```

Figure 7.7. Cumulated expectation of the unavailability.