BOOLEAN LOGIC DRIVEN MARKOV PROCESSES:
A POWERFUL NEW FORMALISM FOR SPECIFYING AND
SOLVING VERY LARGE MARKOV MODELS

Marc Bouissou
EDF R&D, 1 avenue du Général de Gaulle,
92141 Clamart Cedex, France
or
Laboratoire d’Analyse et de Mathématiques Appliquées
CNRS UMR 8050 - Université de Marne-la-Vallée

ABSTRACT
This paper introduces a modeling formalism that enables the analyst to combine concepts inherited from fault trees and
Markov models in a new way. We call this formalism Boolean logic Driven Markov Processes (BDMP). It has two
advantages over conventional models used in dependability assessment: it allows the definition of complex dynamic models
while remaining nearly as readable and easy to build as fault-trees, and it offers interesting mathematical properties, which
enable an efficient processing for BDMP that are equivalent to huge Markov chains. We give a mathematical definition of
BDMP, their main properties, and several examples to illustrate how powerful and easy to use they are.

KEYWORDS
Modeling, fault-tree, dynamic fault-tree, Markov chain, Petri net, reliability, availability, approximation

1. INTRODUCTION
Fault-trees are undoubtedly the easiest and most often used technique in complex systems dependability assessment. Many
people have refined this technique which has been applied to various industries, including aerospace, medical, and nuclear.
However, conventional fault-trees (we will call them «static» fault-trees) are not at all suited to modeling systems in which
there are strong dependencies between components.

In order to be able to model component dependencies, one has to recur to dynamic models. The most popular are Markov
chains, because of their numerous nice mathematical properties. In practice, the direct use of Markov chains has practically
been abandoned, to be replaced by some higher level formalisms that enable the automatic generation of a (potentially
huge) Markov chain.

The problem with these higher level representations, like stochastic Petri nets, is that they are much too general. They do
not allow to infer any interesting property of the Markov graph, that could be used to simplify its processing, from the
model input by the user.

Therefore it appears that some kind of trade-off must be chosen between static fault-trees, which have a low modeling
power, but are extremely easy to process, and general dynamic models such as Petri nets, which enable the construction of
much more accurate, but unfortunately untractable models.

The purpose of BDMP is precisely to provide the reliability analyst with such a trade-off.

Although BDMP may seem similar to dynamic fault-trees [5] [6], they are in fact quite different. Instead of adding new
kinds of gates, they assign a new semantics to the traditional graphical representation of fault-trees, augmented only by a
new kind of links (represented by dotted arrows). They enable the analyst to combine conventional fault trees and Markov
models in a brand new way. In fact, they offer much more modeling power, than a simple juxtaposition of these
formalisms, as can be seen in the examples we provide in this article.
Moreover, BDMP have very interesting mathematical properties, which allow a dramatic reduction of combinatorial problems in operational applications, especially when they are processed using a method based on sequences exploration. This method not only is able to process BDMP equivalent to huge Markov chains, but also gives very interesting qualitative results: the most probable sequences that lead to the undesirable event.

From a mathematical point of view, a BDMP is nothing more than a certain way to define a global Markov process, as the result of several elementary processes which can interact in a given manner. An extreme case is when the processes are independent. Then we simply have a fault-tree, the leaves of which are associated to independent Markov chains.

2. BDMP DEFINITION AND BASIC PROPERTIES

2.1 Definition of the elements of a BDMP

A BDMP \((\mathcal{F}, r, T, (P_i))\) is made of:

- a multi-top coherent fault-tree \(\mathcal{F}\)
- a main top event \(r\) of \(\mathcal{F}\)
- a set of triggers \(T\)
- a set of "triggered Markov processes" \(P_i\) associated to the basic events (i.e. the leaves) of \(\mathcal{F}\)
- the definition of two categories of states for the processes \(P_i\)

The multi-top coherent fault-tree \(\mathcal{F} = (E, L, g)\) can be decomposed in:

- a set \(E = G \cup B\), where \(G\) is a set of gates, and \(B\) a set of "basic events" such that \(G \cap B = \emptyset\).
- \(L \subset G \times E\), a set of oriented edges, such that: \((E, L)\) is a directed acyclic graph, \(\forall i \in G, \text{ sons}(i) \neq \emptyset\), and \(\forall j \in B, \text{ sons}(j) = \emptyset\).
- the definition of the gate types. Since any coherent fault-tree can be defined using only \(k/n\) gates (\(n/n\) and \(1/n\) are respectively the "and" and "or" gates), this definition is a function \(g\) \(G \rightarrow \mathbb{N}\), with \(g(i) = \text{ the parameter } k \text{ of the gate (this suffices, because } n \text{ is the number of sons of the gate).}

The main top event of the BDMP, \(r\), can be any top event of \(\mathcal{F}\) (a top event is a gate without any entering edge). This top event will be used to define the set of failure states for the global Markov process.

The set of triggers \(T\) of the BDMP is simply a subset of \((E - \{r\}) \times (E - \{r\})\), such that:

\(\forall (i, j) \in T, \ i \neq j \text{ and } \forall (i, j) \in T, \forall (k, l) \in T, i \neq k \Rightarrow j \neq l\). The first element of a trigger will be called its origin, and the second element will be called its target. The constraints defined above simply mean that the origin and the target of a trigger must be different, and that two triggers cannot have the same target.

Figure 1 is an example of graphical representation of all the notions we have introduced so far, in two versions (the second one refers to the traditional representation of fault-trees). In this example, we have a fault-tree with two tops: \(r\) (the main one) and \(G1\). \(r\) is an AND gate, while \(G1\) and \(G2\) are OR gates. The basic events are \(f1\), \(f2\), \(f3\), and \(f4\). There is only one trigger, from \(G1\) to \(G2\).

![Figure 1: a simple BDMP - theoretical representation, and a more familiar one](image)

The important thing we must still define is the notion of "triggered Markov process": we have such a process \(P_i\) associated to each basic event \(i\) of the fault-tree. \(P_i\) is the following set of elements: \(\{Z^s(t), Z^f(t), f^s_{0..1}, f^f_{1..0}\}\)

\(^1\) To simplify notations, we need to define the function: \(E \xrightarrow{\text{sons}} P(E)\) \(\text{ sons}(i) = \{j \in E, (i, j) \in L\}\)
\( \{Z'_0(t), Z'_i(t)\} \) are two Markov processes. For \( k \in \{0,1\} \), the state space of \( Z'_i(t) \) is \( A'_i \). Note that we do not assume any specific property of this state space (in particular, it may be infinite).

For each \( A'_k \ (i \in B, k \in \{0,1\} ) \) we will need to refer to a part \( F'_i \) of the state space \( A'_i \). In general, \( F'_i \) will correspond to failure states of the component or subsystem modeled by the process \( P_i \).

\[ f_{i_{0 \rightarrow 1}} \text{ and } f'_{i_{0 \rightarrow 0}} \text{ are two probability transfer functions:} \]

- for any \( x \in A'_0 \), \( f'_{i_{0 \rightarrow 1}}(x) \) is a probability distribution on \( A'_i \), such that if \( x \in F'_0 \), then \( \Pr(f'_{i_{0 \rightarrow 1}}(x) \in F'_i) = 1 \),
- for any \( x \in A'_1 \), \( f'_{i_{1 \rightarrow 0}}(x) \) is a probability distribution on \( A'_i \), such that if \( x \in F'_1 \), then \( \Pr(f'_{i_{1 \rightarrow 0}}(x) \in F'_0) = 1 \).

Such a process is said to be "triggered" because it switches instantaneously from one of its modes to the other, via the relevant transfer function, according to the state of some externally defined Boolean variable, called "process selector" (see variables \( X_i \) below). Moreover, in each mode, the process can be "trimmed", which means that its transitions entering \( F'_i \) can be inhibited. The fact that the process is/is not trimmed at time \( t \) depends on the value of another externally defined variable called "relevant event indicator" (see variables \( Y_i \) below).

Section 3 gives some useful illustrations of the concept of "triggered Markov process".

Before going any further into mathematical details, we can now tell what is the meaning of the BDMP of Figure 1: it could represent a system made of three physical components, \( f_1 \), \( f_3 \) and \( f_4 \), which is said to be failed if: \( f_1 \) is failed and \((f_3 \text{ or } f_4 \text{ is failed}) \). Whenever gate \( G_1 \) enters/leaves a failure state, it instantaneously changes the mode of \( G_2 \) and thus of the triggered Markov processes \( f_3 \) and \( f_4 \). The mode they are in at time \( t \) is defined by the value of the Boolean assumption: \((f_1 \text{ or } f_2 \text{ is failed}) \). Whenever gate \( G_1 \) enters/leaves a failure state, it instantaneously changes the mode of \( G_2 \) and thus of the triggered Markov processes \( f_3 \) and \( f_4 \). The mode they are in at time \( t \) is defined by the value of the Boolean assumption: \((f_1 \text{ or } f_2 \text{ is failed}) \).

### 2.2 The global stochastic process specified by a BDMP

Let us now explain the dynamics of a BDMP, in terms of the global stochastic process that it defines. In order to do so, we need to introduce a few extra notations to define its semantics.

\( J \) is a structure that enables us to define recursively three families of Boolean functions of the time, each of these families having a member associated to each element (basic event or gate) of \( E \). To simplify notations, the time, which should appear everywhere, will be omitted.

#### 2.2.1 The structure functions

The first family is, as is usually the case in fault-trees, the family of structure functions \( (S_j)_{j \in E} : \)

\[ \forall i \in G, \quad S_i = \sum_{j_{1 \rightarrow 0}(i)} S_j \geq g(i) \quad (1) \]

\[ \forall j \in B, \quad S_j = Z_{X_j} \in F'_{X_j} \text{ (}X_j(0 \text{ or } 1) \text{ is the mode the process } P_j \text{ is in at time } t \text{; see below}) \quad (2) \]

Note that most of the time, the value 1 for \( S_j \) will represent a failure, as is usual in fault-tree modeling. In particular, the set of failure states for the whole system will be defined by \( S_1 = 1 \).

#### 2.2.2 The process selectors

The second family is the family of process selectors \( (X_j)_{\in E} \). They are used to answer the question: which mode is chosen for each basic process?

If \( i \) is a root of \( J \), then \( X_i = 1 \) else

\[ X_i = \neg \left( \left( \forall x \in E_i, (x,i) \in L \Rightarrow X_s = 0 \right) \lor \left( \exists x \in E \setminus (x,i) \in T \land S_x = 0 \right) \right) \quad (3) \]

This means that \( X_i = 1 \) except in the two following cases:

- the origin of a trigger pointing at \( i \) has its structure function equal to 0,
- \( i \) has at least one father and all fathers of \( i \) have their process selector equal to 0.
2.2.3 The trimming of the process

The third and final family is the family of relevant event indicators \( (Y_i)_{i \in E} \). They are used to answer the question: is each basic process to be "trimmed"? Trimming the processes usually results in a dramatic reduction of the combinatorial explosion. This simplification, depending on the situations, may yield approximate or exact results (see § 2.4 and 4.1.1).

If \( i \) is a root of \( F_i \), then \( Y_i = 1 \), else

\[
Y_i \equiv C_i \lor (\exists x \in E / (x, i) \in L \land Y_i \land S_i = 0) \lor (\exists y \in E / (i, y) \in T \land S_y = 0)
\]  

(4)

\( C_i \) is an arbitrary Boolean constant. This means that \( Y_i = 1 \) (i.e. failures of process \( i \) are "relevant", and hence, must not be trimmed) if and only if at least one of the following conditions is true:
- \( C_i = 1 \) (this is a modeling choice made by the analyst; see § 4.1.1 for examples),
- \( i \) is a root of \( F_i \),
- \( i \) has at least one "relevant father" whose structure function is equal to 0,
- \( i \) is the origin of at least one trigger pointing at a gate or basic event whose structure function is equal to 0.

2.2.4 BDMP dynamics

It is now possible to explain the semantics of a BDMP.

At time \( t = 0 \), the global process is initialized, by assigning a probability distribution to a set of initial configurations. Each configuration is an (arbitrary) set of values of the modes \( X_i \) and states \( Z_{x,i} \) of all \( P_i \).

Conditionally to an initial configuration, using equations (1) to (4), all functions \( X_i, S_i, Y_i \) are evaluated (this is always possible: see [9] for the proof).

If the modes of some processes \( P_i \) change, the corresponding transfer functions are used to determine the probability distribution of a new set of configurations. This may again change the modes of some processes, which requires a new application of transfer functions...

A configuration \((S, X, Y)\) is said to be stable if no instantaneous change is possible from this configuration in conformity with previous definitions. It is reasonable to assume that the system admits at least one stable configuration, and thus, in most cases, a probability of 1 is assigned to a stable configuration, to start the process.

However, in the general case, the exploration of all possibilities leads to the construction of a tree representing all the instantaneous cascading configuration changes that can happen.

Any branch of this tree is necessarily finite. Here is the proof: the "monotonicity property" we required from transfer functions and from \( F \) ensures that along this branch, all \( S_i \) are increasing functions; when they have reached the value 1, they become stable. Therefore, the functions \( X_i, Y_i \) become stable too, since they depend only on the \( S_i \).

As for the number of branches of the tree, from a purely theoretical point of view, it might be infinite, but in fact this does not happen in practical applications.

The whole (and complex!) initialization process finally produces a set of stable configurations with a probability distribution on these configurations at \( t = 0+ \).

Any of these configurations remains stable until one of the \( P_i \) goes through a border between its two categories of states.

Formally, the first stopping time of the global process is

\[
T_i = \text{Inf} \left\{ t / \exists i \in B, Z_{x_i} (t) \notin D_{x_i} (0+) \right\}.
\]

where \( D_{x_i}^j (0+) \) denotes \( F_{x_i}^j (0+) \) or its complementary in \( A_{x_i}^j (0+) \); it is the category of states the process \( P_i \) was in at time \( 0+ \). Let \( i_0 \) be the index of the corresponding process. The definition of \( S_{i_0} \) implies that it changes, and this change may propagate in equations (1) to (4), causing other changes in \((S, X, Y)\). If the modes of some \( P_i \) change, the corresponding probability transfer functions give a set of new configurations. This may again change the modes and so on. This means that at time \( T_i \), the same processing as the one that was used to determine the distribution on initial configurations must be applied.

The subsequent stopping times of the global process are determined according to the same principle as the one defining \( T_i \).

At any time, the processes \( P_i \) for which the relevant event indicator equals zero are trimmed: their transitions to failure states are inhibited until their indicator takes again the value 1. All their other transitions (including, of course, the repair transitions) are still allowed. The trimming takes a particular meaning in the case of the application of a probability transfer function: it implies that probabilities of reaching new states become conditional to the fact that transitions leading to failure states are inhibited. § 3.2 gives an example of how a transfer function is modified when it applies to a trimmed process.

2.3 Illustration of the definitions

In the example of Fig. 1, the three function families (taking the value 0 for all constants \( C_i \)) are defined according to table 1.
Assuming that the basic events are all of the type described in § 3.1, except f3 which is of the type described in § 3.2, Fig. 2 is the Markov chain corresponding to the global process specified by this BDMP. To simplify the graph, the component f2 is assumed to be perfect (λf2 = 0), and all standby failure rates are taken equal to 0. The dotted arrows correspond to transitions of the graph obtained will all Ci equal to 1, which are suppressed by choosing the value 0 for all Ci.

In a more complex BDMP each set of values for the constants Ci might correspond to a different Markov chain.

<table>
<thead>
<tr>
<th>Structure functions</th>
<th>Process selectors</th>
<th>Relevant event indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sf = Sf1 \land Sg2</td>
<td>Xf = 1</td>
<td>Yf = 1</td>
</tr>
<tr>
<td>Sg2 = Sf3 \lor Sf4</td>
<td>Xg2 = Sg1</td>
<td>Yg2 = \neg Sf</td>
</tr>
<tr>
<td>Sg1 = Sf1 \lor Sf2</td>
<td>Xg1 = 1</td>
<td>Yg1 = \neg Sf</td>
</tr>
<tr>
<td>Sf1 = 1 \iff Pf1 in a failure state</td>
<td>Xf1 = Xg1 \lor Xf = 1</td>
<td>Yf1 = \neg Sg1 \land Yg1</td>
</tr>
<tr>
<td>Sf2 = 1 \iff Pf2 in a failure state</td>
<td>Xf2 = Xg1 = 1</td>
<td>Yf2 = \neg Sg1 \land Yg1</td>
</tr>
<tr>
<td>Sf3 = 1 \iff Pf3 in a failure state</td>
<td>Xf3 = Xg2 = Sg1</td>
<td>Yf3 = Yg2 \land \neg Sg2</td>
</tr>
<tr>
<td>Sf4 = 1 \iff Pf4 in a failure state</td>
<td>Xf4 = Xg2 = Sg1</td>
<td>Yf4 = Yg2 \land \neg Sg2</td>
</tr>
</tbody>
</table>

Table 1: the functions Sf, Xf, Yf of the BDMP of Figure 1

Figure 2: the Markov chain defined by the BDMP of Figure 1 (assuming f2 has no failure)

2.4 Properties of a BDMP

The proofs of all the properties we are going to state are in [9].

Property 1: Any BDMP is a valid description of a global Markov process.

Property 2: For a given BDMP modeling a non repairable system, the Markov chain obtained by assigning an arbitrary set of values to constants Ci is strictly equivalent in terms of reliability (i.e. distribution of the instant at which Sf changes from 0 to 1) to the “full” Markov chain obtained with all Ci equal to 1. In practice, they will all be set to 0, which will produce a much smaller Markov chain, in which each failure transition is “relevant”. The reduced Markov chain can of course be processed by any solving method valid for Markov chains in general. However a method based on the exploration of sequences leading to the failure states is the most interesting because it yields meaningful qualitative results: most of the obtained sequences are “minimal”.

Property 3: A BDMP can yield minimal cutsets of the modeled system. Of course, since any BDMP includes a coherent fault-tree, it is possible to compute the minimal cutsets of this fault-tree. For example, the BDMP of Figure 1 has two minimal cutsets: \{failure of f1, failure of f3\} and \{failure of f1, failure of f4\}. Although the cutsets of a BDMP can be interesting from a qualitative point of view, they are generally of little use for probability calculations. This is obvious in our example, where the component f2 is totally absent in the cutsets.

3. EXAMPLES OF TRIGGERED MARKOV PROCESSES

The next paragraphs depict the few standard triggered Markov processes which are sufficient to build many BDMP, including models of quite complex systems. Each of these processes can be associated to a leaf of the BDMP fault-tree.
3.1 The warm standby repairable leaf

This process is used to model a component that can fail both when it is in standby and when it works (this mode corresponds to a process selector equal to 1), but with different failure rates. This component can be repaired whatever its mode. When $\mathcal{A} = 0$, the model represents in fact a cold standby repairable component.

$$\begin{align*}
S \xleftarrow{\lambda} & F, \\
W \xleftarrow{\lambda} & F
\end{align*}$$

Process 0 Process 1

The transfer functions simply state that when the value of the process selector changes, the component goes from state Standby to Working (or vice-versa) or remains in Failure state with probability 1.

$$
\begin{align*}
f_{0\rightarrow1}(S) &= \{Pr(W) = 1, Pr(F) = 0\}, & f_{0\rightarrow1}(F) &= \{Pr(F) = 1, Pr(W) = 0\} \\
f_{1\rightarrow0}(W) &= \{Pr(W) = 1, Pr(F) = 0\}, & f_{1\rightarrow0}(F) &= \{Pr(F) = 1, Pr(S) = 0\}
\end{align*}
$$

3.2 The on-demand repairable failure leaf

This model is used to represent an on-demand failure, that can happen (with probability $\gamma$) when the process selector changes from state 0 to state 1.

$$\begin{align*}
W \xrightarrow{\mu} & F, \\
W \xrightarrow{\mu} & F
\end{align*}$$

Process 0 Process 1

$$
\begin{align*}
f_{0\rightarrow1}(W) &= \{Pr(W) = 1 - \gamma, Pr(F) = \gamma\}, & f_{0\rightarrow1}(F) &= \{Pr(F) = 1, Pr(W) = 0\} \\
f_{1\rightarrow0}(W) &= \{Pr(W) = 1, Pr(F) = 0\}, & f_{1\rightarrow0}(F) &= \{Pr(F) = 1, Pr(W) = 0\}
\end{align*}
$$

If the transfer function $f_{0\rightarrow1}$ must be applied to a trimmed process, it becomes conditional to the fact that the process cannot enter a failure state. Then, $f_{0\rightarrow1}(W) = W, \ f_{0\rightarrow1}(F) = F$.

A "symmetrical" version of this model (used to model failures that can happen when the process goes "back" from mode 1 to mode 0) is obtained by exchanging the two transfer functions.

3.3 The repairable leaf with an increasing failure rate

Any distribution of a time to failure can be approximated by the "phase method" [7]. This approximation uses a Markov chain. The simplest structure for the chain is a series of states $i$, with a transition from $i$ to $i+1$, but better approximations can be obtained with a little more complicated structures.

Here is a very simple triggered Markov process that could be used to model an aging repairable component. With this model, degradation of the component is assumed to be possible only in mode 1, when the component is in operation. The only failure state is the state number 0. After a repair, the component is "as good as new".

$$\begin{align*}
\begin{array}{cccc}
\mu & n-1 & \ldots & 1 \\
\lambda & 0
\end{array}
\end{align*}$$

Process 0 Process 1

Both transfer functions state that the subsystem remains in the state it was in just before the change of the process selector.

4. APPLICATION EXAMPLES

In this section, we explain the BDMP solution for two generic modeling problems that occur very often in dependability studies, and how to model an electrical system showing a very dynamic behavior.

4.1.1 Failure modes of a component: mutually exclusive, or independent?

The component A has 3 failure modes a, b, c. Each of them results in the unavailability of A. This is obviously modeled by any of the two structures of Fig. 3. Various degrees of dependency may exist between the three failure modes. All the possibilities can be modeled, using a proper combination for the constants $C_i$ defined in § 2.2.3.

Using the first structure, if we want independent failure modes, we must set $C_a = C_b = C_c = 1$. If we want mutually exclusive failure modes, we must set $C_a = C_b = C_c = 0$.

Now if we want a and b to be mutually exclusive, but independent from c, we must use the second structure, with the following values $C_a = C_b = 0$, and $C_c = C_{ab} = 1$. 

Figure 3: two representations of a component A with 3 failure modes

Because of property 2 given in § 2.4 those subtleties are relevant only for repairable systems. Fortunately, it happens that the model with mutually exclusive failures (which is much less combinatorial) is often the most realistic. It is the case in the following example: when an electrical sub-system is lost, it is no longer energized, and therefore it cannot suffer from additional failures.

4.1.2 Common cause failures

Modeling common cause failures is usually difficult in repairable systems models. The difficulty lies in the fact that a common cause failure can affect at the same time any subset of a set of components, and that the corresponding repairs must be performed individually. Figure 4 gives the BDMP solution for modeling a set of three components subject to common cause failures. Each of them can fail either for independent reasons or because of a "shock" that breaks the component. In this excerpt of a BDMP involving components A, B and C (the fathers of the "or" gates are not represented), the process "shocks", which is both a root and a leaf of the BDMP is of the type described in § 3.1, as well as A2, B2, and C2 which model independent failures of the components. Its function is only to generate at random times a mode change (thanks to the 3 triggers) for processes A1, B1, and C1 which are of the type described in § 3.2 (failure on demand). Each time a shock occurs, any combination of simultaneous failures of A, B and C may happen. The corresponding repairs are independent. The parameter \( \mu \) of the "shocks" process is given a very high value; thus, the occurrence of shocks is very close to a Poisson process.

A realistic example

4.1.3 A realistic example

Figure 5: a simple system with cascade redundancies, and the corresponding BDMP

Here is an example, showing how a BDMP can be used to model a simple electrical system with two levels of standby redundancies. Figure 5 gives the description of the system (on the left), and the corresponding BDMP as it can be input in the KB3 tool (shortly described in the next section).
This example shows how powerful and easy to use BDMP are. The fault-tree of the BDMP is exactly the same as a static fault-tree that could be used to get the minimal cutsets of the system. The magic is that by just adding two triggers, we obtain a fully dynamic model, taking into account all reconfigurations of the system, including those consecutive to repairs.

5. AN OPERATIONAL TOOL TO INPUT AND PROCESS BDMP

EDF has been developing since 1989 a set of tools for dependability analyses, based on a modeling language called the FIGARO language [3]. In particular, the KB3 tool enables the user to input a model graphically, which, thanks to the behavior of components cast in a knowledge base written in FIGARO language, is equivalent to a Markov chain.

It has been quite easy to implement all the theoretical concepts we have given in the previous paragraphs (plus a few little extras...), thanks to the high modeling power of the FIGARO language, by writing a suitable "knowledge base". The "components" of this knowledge base are the graphical elements that can be seen on Figure 5, plus the elements of Petri nets: places and transitions. Petri nets can be used to describe non-standard processes associated to the leaves of the BDMP.

How does it work?
KB3 transforms a graphical model input by the user into a FIGARO language description. The processing of this textual model relies on the FIGSEQ tool, which automatically finds sequences of events leading to an undesirable state, according to the methods described in [1], [2], [4]. Thanks to probability based pruning of sequences, it is possible to process BDMP that are equivalent to huge Markov chains, like the BDMP with more than 70 leaves that we briefly describe in [4]. FIGSEQ is able to compute a precise approximation of the reliability of the system depicted in Figure 5 in a few seconds on a PC.

6. CONCLUSION

We have defined a new modeling formalism: Boolean logic Driven Markov Processes (BDMP). This formalism is nearly as simple to use as fault-trees. However, it offers powerful shortcuts for modeling the behavior of complex dynamic systems.

We have given a formal definition of BDMP, which enabled us to demonstrate (in [9]) some very important mathematical properties of this formalism. In particular, thanks to those properties, it is possible to dramatically reduce combinatorial explosion problems inherent to Markov models.

Using the KB3 and FIGSEQ tools, we were able to implement the concepts defined in this paper in an operational modeling and assessment environment. This tool, which includes all the generic functions of KB3 (like the interactive simulation with graphical display of state changes, model configuration management, automatic model documentation, etc.) could be developed in only a few days, on the basis of existing KB3 knowledge bases; although this is not the main subject of this article, it is worth mentioning the rapidity of this development because it shows the power of KB3 and FIGSEQ.

7. REFERENCES