Practical course on Boolean logic Driven Markov Processes (BDMP)

Marc Bouissou
Electricité de France/R&D/MRI dept
CNRS

Detailed theoretical definition of BDMP available at the address:
http://www-math.univ-mlv.fr/users/bouissou/

Outline

- Reminder on standard fault-trees and Markov processes
- Introduction to BDMP
- How to use examples provided with KB3
- Explanation on examples
- Use of the Figseq tool to process the models
- Conclusion
A simple fault-tree

Logical operators of fault-trees

- A logical operator enables the analyst to express an event $S$ (top or intermediary event) as a combination of other events $E_i$ considered as its immediate causes.

- Most often used operators are:
  - Operator OR: $S$ is true if and only if at least one of the $E_i$ is true [other notations: $U$ or $+$]
  - Operator AND: $S$ is true if and only if all the $E_i$ are true [other notations: $\cap$ or $\cdot$]
  - Operator $k/n$: $S$ is true if and only if at least $k$ $E_j$ out of $n$ are true
    This operator is equivalent to a combination of one OR and several AND operators
Probability calculation on fault-trees

- A fault-tree is a particular case of structure function
- Structure function $S = \text{boolean function of the component's states, which has value}$
  - 1 if the system is failed
  - 0 if the system works
- Many methods enable the calculation of:
  - $\Pr(S=1)$ at time $t$ (i.e. system unavailability)

Consequences

- A fault-tree model makes it possible to compute the unavailability of a system and importance factors of its components, as a function of time
- It can be done whatever the evolution of the component's availability along the time axis. For example, non exponential distributions can be taken into account
- Components may be repairable or not; the only necessary hypothesis is their independence
- With this hypothesis, it is also possible to perform a reliability calculation for repairable systems
- But unfortunately, this independence is not true for many systems
Classical situations including dependences

A, B and C are 3 sub-systems ->
they may contain several components, with or without on-demand failure, with or without a possibility of failure in standby

- Several possibilities for the exploitation policy, e.g.:
  - A is active, B and C must start in case of a failure of A
  - A is active, B must start in case of a failure of A, C must start when both A and B are failed
- Common cause failures in a repairable system
- Startup sequences, with steps depending on the result of previous ones
- Preventive maintenance: forbidden in case of failure of a sub-system
- Failure rate depending on (a changing) operating environment
- Mutually exclusive failure modes
- etc

Markov graphs: basic notions

- Introduction
- Modelling small systems
- How about complex systems?
**Simple example**

System state = Combination of components states

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>ETATS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKING</td>
<td>WORKING</td>
<td>E1</td>
</tr>
<tr>
<td>WORKING</td>
<td>FAILED</td>
<td>E2</td>
</tr>
<tr>
<td>FAILED</td>
<td>WORKING</td>
<td>E3</td>
</tr>
<tr>
<td>FAILED</td>
<td>FAILED</td>
<td>E4</td>
</tr>
</tbody>
</table>

Working states

Failure states

Markov graph:
- node = state
- edge = event (failure, repair...)

*A Markov graph can support availability, reliability, performance calculations*

**Complete Markov graph**

P2 is repaired first (only one repairman)
The two kinds of solving methods for Markov graphs

- **Matrix calculations**
  - Advantages: give for each state the probability of the system being in this state as a function of time
  - Drawbacks: limited to graphs with about $10^6$ states, do not give any indication about the weak points of the system
- **Sequence based calculations (principle of Figseq)**
  - Advantages: see detailed slide later.
  - Drawbacks: the availability as a function of time cannot be computed. (Only the asymptotic value can be computed).

**Availability curves**

2 to 3 times the largest time to repair
Quantification of sequences

- **Two kinds of algorithms**
  - A: Exhaustive search (well suited to non repairable systems or for short mission times)
  - B: NRI algorithm (well suited to completely and quickly repairable systems -> used in most cases)

**A**:
\[
P(1,2,3;t) + P(1,2,1,2,3;t) < 1 - R(t) < P(1,2,3;t) + P(1,2,1,2,3;t) + P(1,2,1,2,1;t)
\]

**B**:
\[
R(t) = e^{-\frac{\lambda_{12} + \lambda_{23}}{\mu_{21}} t}
\]
Modeling on demand failures

Becomes (in order to allow matrix calculations)

The on demand failures can be kept explicitly in sequence based solving methods

Advantages of sequence based methods

- Ability to process huge graphs (even infinite)
- Qualitative validation of the models
- Most probable sequences = weak points of the system. The results give hints as how to improve the reliability or availability of the system
- Intuitive understanding of the results

The only limit: the failure probability must not be scattered into too many sequences, each of them having a very small contribution
Application of sequences: understanding an example of unacceptable approximation

Considered system
- 2 components N1, N2 in series
- Component S1: 1st spare (starts up if: N1 or N2 failed)
- Component S2: 2nd spare (starts up if: (N1 or N2 failed) and S1 failed)

Reliability data
- N1: \( \lambda_{N1} = 1 \times 10^{-3} \) MTTR \( N1 = 10 \) h
- N2: \( \lambda_{N2} = 1 \times 10^{-4} \) MTTR \( N2 = 200 \) h
- S1: \( \lambda_{S1} = \lambda_{S2} = 1 \times 10^{-3} \)
- S2: \( \mu_{S1} = \mu_{S2} = 1 \times 10^{-2} \)

Beware! Repair times for N1 and N2 are very different

The "true" system

Approximate model

Application of sequences: understanding an example of unacceptable approximation (2)

\[ \lambda_N = 1.10 \times 10^{-3} \]
\[ \mu_N \approx 3.67 \times 10^{-2} \]

« Equivalent » component \( N \)

\[ \lambda_N = \sum \lambda_{\lambda(x)} \]
\[ \mu_N \approx \frac{\sum \lambda_{\lambda(x)}}{2 \lambda_{\lambda(x)} x_0} \]
Application of sequences: understanding an example of unacceptable approximation (3)

- Initial system
  - Unreliability at t=1000: about 9.43e-4 (as. unavail: 4.89e-5)
  - Most probable sequences:
    - fail N2, fail S1, fail S2
    - variants with loops due to repair of N2 or to failure + repair of N1
  - fail N1, fail S1, fail S2 (contribution <3%)

- Approximate system
  - Unreliability at t=1000: about 6.09e-4 (as. unavail: 1.19e-5) (very optimistic!)
  - Most probable sequences:
    - fail N, fail S1, fail S2
    - variants with loops due to repair of N

- conclusion: 1) rare and serious failures have more impact than frequent and quickly repaired failures. 2) the aggregation of the two components N1 and N2 is a worse approximation than the fact of neglecting completely the failures of N1.

Boolean Logic Driven Markov Processes (BDMP)

A powerful new formalism for specifying and solving very large Markov models:
Taking the best from fault-trees and Markov graphs
Outline

- Advantages/drawbacks of FT, Markov
- BDMP theoretical definition
- Properties of BDMP
- Examples: your first use of KB3

Advantages of Fault-trees

- Simplicity. Implicit representation of a graph with $2^N$ states
- Hierarchical model
  - breakdown levels
  - supports local reasoning
- Enables the calculation of minimal cutsets
  But
- Static model: the quantification methods need the independence of basic events
Markov graphs advantages

- Ability to take into account complex dependencies
- The «perfect» mathematical framework

**But**

- Require a higher level (i.e. more concise) representation (e.g. Petri nets)
- The validation of a complex Petri net is very hard
- Combinatorial explosion

A new formalism: BDMP

- The total independence of leaves of a fault-tree is replaced by simple dependencies. **Each leaf has two modes:** required and not required. Transitions between those two modes define instantaneous states in which on demand failures can be triggered.
- Any "triggered Markov process" can be associated to a leaf

“Boolean logic Driven Markov Process”
(BDMP)
Graphical representation of a BDMP

- **Main top event**
- **Secondary top event**
- **Trigger**
- Triggered Markov processes $P_i$ + definition of failure states for each $P_i$

Examples of leaves behaviour
*(triggered Markov processes)*

<table>
<thead>
<tr>
<th>Not required</th>
<th>Transition</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
<td>$\mu$</td>
<td>$F$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Failure mode possible only if in required mode
- Failure mode with reduced rate if in non required mode
- On demand failure mode
Definition of required mode in a BDMP (1)

- Very powerful concept, because it is **hierarchical**
- Requirement signal transmitted by the triggers and the branches of the fault-tree

![Diagram](diagram1.png)

A gate or leaf is required except when it receives a signal of non req. from:
- all its fathers or
- directly via a trigger

Makes it easy to model cascade standby redundancies

Definition of required mode in a BDMP (2)

The only possible failures in the initial state of the BDMP
Mathematical properties

- A BDMP includes a fault tree (FT) => minimal cutsets (a complex leaf corresponds to a single event in the cutsets).
- This fault-tree gives a «structure» to the Markov graph equivalent to the BDMP: this makes it possible to reduce combinatorial explosion thanks to the concept of irrelevant event.
- An event is said to be irrelevant if the propagation of the effects of its fulfillment in the fault-tree only concerns gates which are already in the «true» state.

Exploitation of irrelevant events

- Trimming of irrelevant events:
  - Non repairable system -> dramatic reduction of the Markov graph size, with exact calculation of reliability.
  - Repairable system -> dramatic reduction of the Markov graph size, with approximate calculation of reliability and availability.
- Note that in some cases the model with trimming is more realistic than without (e.g.: electrical components, mutually exclusive failure modes).
A system with cascade standby redundancies

- GRID
- CB_up_1
- CB_up_2
- CB_dw_1
- CB_dw_2
- transfo1
- transfo2
- line_1
- line_2
- diesel generator
- CB_dies
- CB_up_1
- CB_dw_1
- Transfo1
- GRID
- CB_up_2
- Transfo2
- CB_dw_2
- dies_generator

Corresponding BDMP input in the KB3 tool

- OR
- LossOfLine_1
- CB_up_1
- Transfo1
- CB_dw_1
- LossOfLine_2
- OR
- LossOfAllBackups
- OR
- LossOfDieseline
- RO_CB_dw2
- RC_CB_dies
- RS_dies
Transitions                      Sequence   Contribution
Name                    Rate   Type  Duration

[failure OF GRID]                         1.00E-04 EXP
not occurrence OF RO_CB_dw_2,                   9.99E-01 INS
not occurrence OF RS_dies,                       9.99E-01 INS
occurrence OF RC_CB_dies]                    1.00E-03 INS 0.00E+00 2.46E-01

... (2 sequences)

[occurrence OF transfo1]                     1.00E-04 EXP
occurrence OF RC_CB_dw_2,                   1.00E-03 INS
not occurrence OF RO_CB_dw_2,                    9.99E-01 INS
not occurrence OF RC_CB_up_2]                    9.99E-01 INS
not occurrence OF RS_dies,                       9.99E-01 INS
occurrence OF RC_CB_dies]                   1.00E-03 INS 0.00E+00 2.46E-04

... (21 sequences)

[failure OF transfo2]                         1.00E-04 EXP
[not occurrence OF RO_CB_dw_2,                    9.99E-01 INS
not occurrence OF RC_CB_up_2]                    9.99E-01 INS
[failure OF transfo2]                         1.00E-04 EXP
[not occurrence OF RO_CB_dw_2,                    9.99E-01 INS
occurrence OF RS_dies,                      1.00E-03 INS
not occurrence OF RC_CB_dies]                    9.99E-01 INS 9.96E+00 2.45E-04

... etc

FIGSEQ output corresponding to this BDMP

<table>
<thead>
<tr>
<th>Transitions</th>
<th>Sequence</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Rate</td>
<td>Type</td>
</tr>
</tbody>
</table>

BDMP summary

- **BDMP**
  - Are a generalization of standard fault-trees
  - Are a generalization of Markov graphs
  - Enable lots of combinations, which would be:
    - **impossible** to represent with fault-trees
    - **very difficult** to represent with Markov graphs or Petri nets
  - They have very interesting mathematical properties which dramatically reduce the combinatorial problems
Getting started with KB3/BDMP

Screen dumps obtained with KB3 (v3) and a knowledge base that implements the BDMP concepts

Principles of KB3 and FIGSEQ

**KB3**
- Fault-tree generator
- FIGARO model

**EDF tools**
- FIGARO knowledge base
  - Generic component descriptions

**Non EDF tools**
- Fault-tree processing tool
  - minimal cutsets
  - probabilities
  - importance factors

**FIGSEQ**
- Sequences with their probabilities
Implementation of BDMP via KB3

- KB3 can be used with many knowledge bases
- One of them is designed in order to implement the BDMP concepts
- With this kb, the only possible exploitation of models is via Figseq

In the following examples, the symbols used for leaves refer to the « triggered Markov processes » defined in transparency n° 26

How to load and try the examples described hereafter

Launch KB3 (Start menu/KB3)
Click here then double-click on a study name
When the tab with the study name appears, click on it
Double-click on the yellow icon

The study is open!

NB: the toolbars can be hidden/shown via the View menu
System with standby redundancy

- In the initial state, F_2 and SF_1 are in non required mode => failure of F_2 impossible, failure of SF_1 has a reduced rate
- After failure of F_1, the failure of F_2 becomes possible, failure of SF_2 has an increased rate
- If the trigger is deleted, the BDMP models an active redundancy

Load study: Redund1

How to do an interactive simulation

Click on tab « processing »
Click here, then double-click on the icon Sim
If needed, choose a Profile, then launch the simulation

In the simulation window, you can choose a visualization option that will show the state changes with colours
Standby redundancy with a possibility of on demand failure

- The failure of B is explained by a sub-tree:
  - On demand failure, which can be triggered when the corresponding leaf changes from mode « non required » to mode « required »
  - Failure in function

Example of sequence dependency

- If switch fails after primary fails (and after spare is activated) then the system is still operational.
- If the switch fails before the primary fails, then the spare cannot be activated and the system fails, even though the spare is operational.
- Failure criteria depends on order in which failures occur. This system can be solved correctly via a Markov model.

(from J.B. Dugan, RAMS 2001)
Sequence dependency

This special gate changes from 0 to 1 when its right input changes from 0 to 1 while its left input is equal to 1.

Common cause failure

The system is lost when all three components A, B, C are lost, either by independent failure, or by common cause failure: each shock generated by the isolated leaf Shock induces one or more instantaneous failures. The «repair» rate of Shock is set to a very high value: this ensures that the first event to occur after a shock and its instantaneous consequences is this repair.

Study: CCF
Shared spare: problem statement

A, B are permanently in function

S can replace either A, or B

If it replaces A, it becomes unavailable for B and vice-versa

The loss of the function fulfilled by A occurs if A and S are failed, or if A fails after B (which will have utilized the spare capacity of S)

Study: Share1

Shared spare: solution with a BDMP

Several options can be chosen for the conditions in which the « then » gate goes back from 1 to 0. (first input to 0, both to 0, left input to 0, right input to 0)

The interest of this structure is that it could be used as the top part of a much more complex BDMP where A, B and S would be modelled by sub-BDMPs

Here, we choose the option: first input to 0
Model of a PC

The failure rate of the CPU increases when the cooling is lost, and a failure of the floppy drive can occur only when it is in use.

Advanced concepts of BDMP

- Fine tuning of trimming
- The « approx_OR » and « group » leaves
- How to model shared repair resources
Fine tuning of trimming: the example of the mutual exclusion of failure modes

According to the values given to constants $C_i$ of the various BDMP elements (gates or leaves), we can model:
- 3 mutually exclusive failure modes (all $C_i = 0$)
- 3 independent modes (all $C_i = 1$)
- $F_1$ and $F_2$ mutually exclusive, but this group independent from $F_3$

NB: the constants $C_i$ of the theoretical article are called « force_relevant_event » in KB3/BDMP

The « approximate OR » gate

This gate behaves:
- like a standard OR gate when its constant « agregation » is set to FALSE
- like a single f_leaf when its constant « agregation » is set to TRUE. The failure and repair rates of this « macro leaf » are calculated using the following formulas:
\[
\lambda_{eq} = \sum \lambda_{Ni}
\]
\[
\mu_{eq} \approx \frac{\sum \lambda_{Ni}}{\sum \lambda_{Ni} \tau_{Ni}}
\]

Go back to slide 17 to see the advantages (and risks !) of using this kind of leaf
The « group » leaf

- This leaf behaves as a group of identical components, sharing one or several repairmen, under a k/n gate
- Its characteristics are:
  - number_of_items
  - number_of_repairmen
  - min_acceptable (the min number of items that must be in working state to ensure that the struct_fn variable of the leaf has the value FALSE)

How to model shared repair resources and repair priorities

1. Create repair teams (like rep_1 and rep_2 here)
2. Specify the number of repairmen in each team
3. For each leaf, declare the repair team(s) necessary to repair it (for ex: electricians, mechanics) by editing the interface rep_team
4. For each leaf, declare the priority of repairs (1 is the most urgent, then 2, then 3, etc)

If no repair team is declared for a leaf, it is assumed to have its own repair resources

This is the only feature of the BDMP knowledge base which does not comply with the theoretical framework of BDMP. But some users require it!
FigSeq use

- From KB3/BDMP to FigSeq
- Definition of target states
- Choice of parameters
- Experiments on small examples

From KB3/BDMP to FigSeq

- In KB3, choose the « generate FIGARO 0 » processing
- Save the generated file (with a .fi extension) in a directory of your choice
- Launch the FigSeq GUI via the Start menu of Windows
- Click on the « browse » button and look for the FIGARO 0 file
- The GUI then proposes a parameter file (with extension .pa)
- Accept it or look for another one if you wish
- Now you are ready to check/modify the FigSeq parameters that will be used for the processing
Definition of target states

For a first run on a given model, you can accept all the default values proposed by the FigSeq GUI, except for the definition of the target state. This default definition is the constant expression TRUE, which means that any state is considered to be a target (i.e. failure) state. You must replace this expression by something like in the example below:

For a simple and safe definition, always choose a condition on the value of state variables named « struct_fn »

It is possible to use complex expressions, like:

\[ \text{struct}_\text{fn}(\text{object1}) \text{ AND NOT (struct}_\text{fn}(\text{object2}) \text{ OR struct}_\text{fn}(\text{object3})) \]

Use of NS algorithm

- Only for reliability calculation, for non repairable systems (i.e. when the repairable_system constant of the global object OPTIONS is set to FALSE)
Test case from A. Cabarbaye and L. Ngom

A required if C or D failed
B required if E or D failed

Example of non active event: failure of D after failure of C

nb of sequences with/without filtering of active events: 36 versus 60

Test case from A. Cabarbaye and L. Ngom: BDMP solution

One can check that exactly the same reliability is obtained with the two versions trimmed/not trimmed

Study: Cab2
Trimmed version: the global results

<table>
<thead>
<tr>
<th>Estimated values</th>
<th>Pessimistic values</th>
<th>Given results</th>
<th>Terminale</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.321084e+0000</td>
<td>9.321084e+0000</td>
<td>6.010061e-001</td>
<td>man of products paid. deposit duration</td>
</tr>
<tr>
<td>6.803001e+001</td>
<td>6.803001e+001</td>
<td>6.010061e-001</td>
<td>man of products returned</td>
</tr>
</tbody>
</table>

Mission time: 10000h

NON trimmed version

<table>
<thead>
<tr>
<th>Estimated values without trimmed sequences</th>
<th>Pessimistic values with trimmed sequences</th>
<th>Given results</th>
<th>Terminale</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.321084e+0000</td>
<td>9.321084e+0000</td>
<td>6.010061e-001</td>
<td>man of products paid. deposit duration</td>
</tr>
<tr>
<td>6.803001e+001</td>
<td>6.803001e+001</td>
<td>6.010061e-001</td>
<td>man of products returned</td>
</tr>
</tbody>
</table>

Average sequence time in initial state: 1 / Limit 3.921084e+0000

Display of sequences in Reliability not limited.
**NRI algorithm (will be used in most cases)**

- Principle of NRI
  - exponential approximation: calculation of a global failure rate for the whole system
  - method: exploration of sequences that lead from the initial state to the failure state, avoiding to traverse loops
- Results:
  - Approximations of reliability and MTTF
  - Has been extended in order to allow the asymptotic availability calculation
  - Very precise for systems with components of high availability
  - Those approximations are pessimistic

\[ R(t) \geq e^{-\lambda t} \]

**CONCLUSION**

- KB3 BDMP is an extremely powerful tool for:
  - Creating dynamic reliability models
  - Carrying out the « feared-events » calculation
  - Sorting results according to contribution criteria
  - Processing the results with standard PC tools