# The Backtracking Process Algorithm: A Dynamic Probabilistic Risk Assessment Method for Autonomous Vehicle Control Systems

**Mohammad Hejase*[a], Arda Kurt[a], Tunc Aldemir[a], and Umit Ozguner[a]**
[a] The Ohio State University, Columbus, Ohio, U.S.A.

**Abstract:** There is an increasing demand for quantitative risk assessment tools capable of providing safety assurances for autonomous vehicle control systems. This demand is due to the recent rise of autonomous functions that are being incorporated into aerospace and automotive domains. A deductive implementation of Markov Cell-to-Cell Mapping Technique is proposed for the identification of risk significant scenarios that may lead to violations of safety goals. Challenges of the methodology are identified, and a breadth-first implementation of the Backtracking Process Algorithm (BPA) is proposed to meet these challenges. Three autonomous vehicle case studies on which BPA was successfully implemented are summarized.

**Keywords:** Dynamic PRA, PRA, Autonomous Vehicles, Markov

## 1. INTRODUCTION

Emerging vehicles in today's market are becoming increasingly intelligent due to recent advances in the development of autonomous functions. Such vehicles have multiple interconnected Electronic Control Units (ECUs) that have to realize possibly thousands of features. As the level of autonomous functions keeps increasing, so does the need for Validation and Verification (V&V) methods that act as alternatives to physical testing for ensuring safe operations of these functions.

In control system theory of autonomous vehicles, reachability analysis techniques are typically used to provide assurance of control system performance. In [1], the authors utilize reachability analysis for the computation of safe sets for platoons of heavy duty vehicles. Authors of [2] apply reachability analysis for formal verification of automated ground vehicle safety via online prediction of vehicle occupancies on highway scenarios. Reachability analysis was used in [3] for the determination of states of inevitable collisions using car-like kinematic models in arbitrary environments. Two of the authors of the present paper have proposed [4] a reachability based controller design method for the computation of exact reachable sets from unsafe sets. Here we take a different approach.

Autonomous vehicles are equipped with many components that are prone to different types of failures with different probability distributions associated with such failures. A component failure, in most cases, changes system behavior either through a change of parameters, or a complete change of dynamics. This makes the task of implementing analytical reachability analysis techniques a very challenging one, as system switching needs to be incorporated into analysis which requires utilizing system-specific methods with high levels of complexity. Additionally, such an analysis only serves to indicate whether a system violates a bound or not, rather than quantifying the likelihood of the system to do so.

In an effort to overcome the problems associated with system dynamics, researchers have been extending the traditional Probabilistic Safety Analysis (PSA) techniques, such as the event-tree/ fault-tree to include some dynamic elements. Authors of [5] utilize Fault Tree Analysis (FTA) for the generation of binary decision diagrams that are used to manipulate and quantify expressions used in the calculation of failure probabilities in decision making strategies for multi-phased missions. Authors of [6,7] use FTA in conjunction with dynamic simulations to assess Unmanned Aerial System (UAS) collision avoidance performance as part of a safety case framework constructed for sense and avoid functions. In [8], authors employ model-based simulations for the generation of system-level Failure Mode Effects and Analysis (FMEA) for systems within aircraft. In [9], FMEA is performed for small

UAS for the identification of critical fault models. System dynamic analysis was then used to study the change of system capabilities and vulnerabilities throughout operational envelopes.

In an effort to develop more advanced techniques that are naturally capable of utilizing Model Based Designs (MBDs) of systems, Dynamic Probabilistic Risk Assessment (DPRA) methods have been developed. DPRA approaches offer an increase in the realism in modeling of stochastic system evolution in quantifying risk. These methods are capable of providing frameworks that allow considering epistemic and aleatory uncertainties in physical processes and system safety responses, including software behavior, on a common platform.

DPRA methods are generally classified into three main categories [10]: (i) continuous-time methods, (ii) discrete-time methods, and (iii) methods utilizing graphical interfaces. Even though Category (iii), in essence, is a subset of Category (ii), it can be regarded as a separate category due to its user-friendly features when dealing with input preparation. A comprehensive Category (i) approach is the Continuous Event Tree (CET) methodology [11], which uses integral equations to capture dependencies among failure events within process/hardware/software/firmware interactions. Markov Cell-to-Cell Mapping Technique (Markov/CCMT) approach is a discrete state-space version of CET. It is worthwhile to mention that even though Markov/CCMT can be derived from CET, it was independently developed [12]. Markov/CCMT has been proposed as both a Category (i) [13] and a Category (ii) method in the literature (e.g. [14-17]).

Monte Carlo Simulation methods are among the most popular Category (ii) methods [18, 19]. These methods mainly emulate the actual processes involved in a given system. Another popular Category (ii) method is the Dynamic Event Tree (DET) [20, 21] approach. DET methods are similar to the traditional ET method in determination of consequences of system responses. The main difference lies in the modeling of the sequencing of events, which is determined subjectively (possibly with the help of a dynamic system model) by the analyst in ET methods, and determined mechanistically through a scenario generator (e.g. [21-24]) coupled to the dynamic system model in DETs.

Within Category (iii) methods, system evolution is modeled using graphs that represent information transmission among nodes. The Dynamic Flowgraph Methodology (DFM) [25-28] is a di-graph based technique, where continuous process variables are discretized into a finite number of states for the representation of the system. Cause-effect relationships among such states are used along with system hardware/firmware states to constitute a deterministic mapping of the system state space on itself for the deductive and inductive tracing of fault propagation. Dynamic Fault Tree (DFT) methods [29-31] use timed housing events, or functional dependency gates for the representation of dynamic dependencies among events. A recent overview of DPRA methods is described in [10].

An instance of the Markov/CCMT implementation falling within the Category (ii) methods will be presented in this paper. Markov/CCMT is capable of accurately capturing mobile system dynamics, the associated control system, and system configuration interactions in a computationally feasible fashion (e.g. see [32] for a case of UAS dynamics.) The method utilizes a discrete state-space representation of the system and models transitions among different states via simulation and fault injection. It can be used both in inductive [14, 33] and in deductive manner [15].

The Backtracking Process Algorithm (BPA) [15] was developed in order to overcome challenges associated with the deductive implementation of Markov/CCMT. The algorithm is deductive in a sense that event sequences leading to specified undesirable consequences (Top Events) are identified. BPA allows quantification of probabilistic system evolution in time, as well as tracing of fault propagation through the system. The BPA can be thought of as a search tree that uses a probabilistic map of the system state-space onto itself. This search tree structure is achieved by recursive enumeration of sub-trees from a Top Event and the traversal of possible paths through a branching process. In order to avoid a computational explosion, only risk significant scenarios with probabilities above a user-specified cut-off value are identified. The algorithm is capable of accounting for nonlinear system dynamics, configuration, and stochastic uncertainties, making it an ideal tool to use in the quantitative identification

of risk significant scenarios in the autonomous vehicles domain. System evolution in time is represented through a series of discrete-time transitions among computational cells that partition the system state-space in a manner similar to finite element or finite difference methods. Each cell can be regarded as accounting for the uncertainty in the system location in the state space at a given point in time. A transition probability from one system cell to another is determined via system dynamics, controller behavior, or system constituent malfunction. Such transitions produce a probabilistic mapping of the system state-space onto itself, including system hardware normal or faulted states, over a user defined time-step.

Section 2 of this paper presents an overview of Markov/CCMT. In Section 3, a breadth-first instance of BPA is proposed and presented. Section 4 discusses the case studies on which the method has been implemented, and presents a sample of the results obtained from one of the case studies. Section 5 contains the conclusion.

## 2. THE MARKOV CELL-TO-CELL MAPPING TECHNIQUE

In Section 2.1, an overview of the Markov/CCMT history is presented. Section 2.2 contains the assumptions required for Markov/CCMT. Section 2.3 outlines the generalized procedure for the global analysis of the system. Challenges with Markov/CCMT are presented in Section 2.4.

### 2.1. Markov/CCMT History

The CCMT was proposed by [34] in the early 1980s for the global analysis of nonlinear dynamical systems. The technique mainly relies on definition of cells that partition the system state space, followed by a mapping which captures the entire system evolution process in terms of transitions among cells. Markov/CCMT was then developed by Aldemir [33] in the late 1980s for the reliability modelling of process control systems by interpreting CCMT as a Markovian process model and extending it to include possible component failures.. This work was one of the earliest to propose and deal with the dynamic PSA (DPSA) of control systems. The methodology was then further extended by [16] to utilize databases rather than differential equations to represent system physics. A continuous-time version of Markov/CCMT was developed in 1996 by authors of [13]. In 2006, Markov/CCMT was utilized for the dynamic reliability modelling and PRA of digital instrumentation and control systems for nuclear reactors [17]. In an effort to tackle the computational and memory challenges faced by Markov/CCMT for deductive analysis, BPA was proposed in 2016 [15], and demonstrated for a level control system.

The surge of autonomous vehicle capability and applications in civil applications calls for techniques capable of providing quantitative assurance on control system performance for such vehicles. In 2017, the Markov/CCMT methodology was introduced to the autonomous vehicle domain in [32]. In this work, BPA was used for the identification of risk significant scenarios leading to hazardous Top Events of interest for UAS operating in the National Airspace as part of the National Aeronautics and Space Administration (NASA) Systems-wide Safety and Assurance Technologies (SSAT) program. Most recently, the method was extended to cope with risk significant scenario identification for UAS with adaptive elements across multiple phases of operation [35]. The method has also been implemented for an automotive application involving the identification of scenarios of risk-significance under hardware failures [36].

### 2.2. Assumptions

As indicated in Section 2.1, Markov/CCMT requires system discretization into a set of cells that partition the system state-space. Such a state-space possibly includes continuous states representing the physics of the system, along with possible system configurations. System discretization is conducted in a manner similar to finite element, or finite difference methods. Cells within the discretized system allow for accounting uncertainty in the system location within the state-space at a given point in time.

A system probabilistic cell-to-cell map is constructed based on system dynamics, configuration, and controller behavior. Transitions in such a map reflect the mapping of the system onto itself over a user-defined time step $\Delta t$ and include system normal or faulted states.
Two main assumptions need to be placed on the system of interest in order to employ Markov/CCMT:

1) The system configurations are fixed over $[t, t+\Delta t)$, but can change at $t+\Delta t$.
2) Transitions among cells or hardware states do not depend on system history.

The first assumption means that the system components can only fail or change their mode of operation once during the interval $\Delta t$. Through proper selection of the time-step $\Delta t$, the system configuration changes and the probabilities of those changed can be realistically modeled and captured. The second assumption implies that the system has Markov property. However, the second assumption can be relaxed via the use of sufficient number of auxiliary state variables.

### 2.3. Generalized Procedure for Global System Analysis

A generalized form of an autonomous control system is presented in Fig. 1. The vehicle dynamics are represented in a continuous $L$ dimensional state space represented by $\mathcal{X} \triangleq \mathbb{R}^L$. This form includes the continuous states $x \in \mathcal{X}$ representing system states such as velocity, angular rate, Euler positions, etc. System components, or configuration, are represented by an $M$ dimensional discrete space $\mathcal{N} \triangleq \mathbb{Z}^M$. This space includes the discrete states $n \in \mathcal{N}$ that are representative of system component conditions, modes of operation, etc. The combination of both $\mathcal{X}$ and $\mathcal{N}$ represents the overall system state-space. A control law $u(x,n)$ is implemented as a function of both $x$ and $n$ to steer the system to the desired set-points and scenario objectives based on the implemented controllers and feedback obtained from the measurements and estimation module.

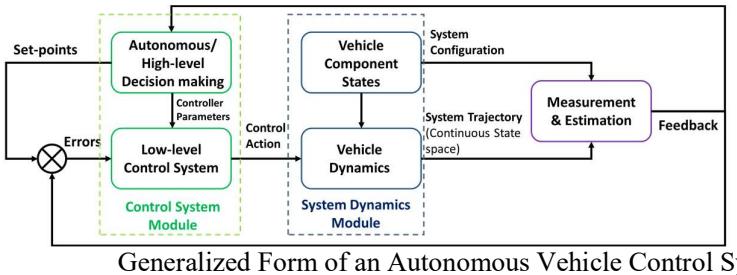


Fig. 1 Generalized Form of an Autonomous Vehicle Control System.

The space $\mathcal{X} \triangleq \mathbb{R}^L$is discretized by partitioning each continuous variable $x_l$ $(l = 1, \dots, L)$ into intervals of $J_l$ partitions and considering combinations of those partitions to form the cells. Knowledge of the state-space upper bounds $\overline{x}$, and lower bounds $\underline{x}$ of interest for the state-space is required for the partitioning. The cells can be regarded as means to accommodate epistemic uncertainties (such as model uncertainties) or aleatory uncertainties (such as process noise and minor environmental disturbances). Possible states of each hardware component $M$ of interest are then defined (e.g. operational, degraded, failed), with each component $m$, having $N_m$ possible states, each denoted by $n_m$ $(m = 1, \dots, M)$.

The unique combinations of the partitioned $\mathcal{X} \triangleq \mathbb{R}^L$ along with the discrete system component configurations forms the complete state-space of the system, denoted by $\mathcal{V}$. Each cell in the cell space is represented by an $(L+M)$ dimensional vector $[\mathbf{j}\ \mathbf{n}] \equiv [j_1, \dots, j_l, \cdots, j_L, n_1, \dots, n_m, \cdots, n_M]$, where $(j_l = 1,2, \dots, J_l; l = 1, \dots, L)$ enumerate the partitioning of the interval $\underline{x_l} \le x_l < \overline{x_l}$, and $n_m$ represents the state of component $m$ $(n_m = 1, \dots, N_m;\ m = 1, \dots, M)$. The cell space $\mathcal{V}$ is composed of $J \times N$ unique cells with $J = J_1 \times \cdots \times J_L$ and $N = N_1 \times \dots \times N_M$ with$\mathcal{V}_\mathcal{X} \triangleq \mathbb{Z}^L$ denoting a subspace of $\mathcal{V}$ containing the vectors $\mathbf{j}$. Let $\mathcal{V}_\mathcal{N} \triangleq \mathbb{Z}^M$ be a subspace of $\mathcal{V}$ containing the vectors $\mathbf{n}$. Note that $\mathcal{V}_\mathcal{X} \cup \mathcal{V}_\mathcal{N} = \mathcal{V}$. The discretized system, along with the relevant notations, is illustrated in Fig 2.

Using the Chapman-Kolmogorov equation under the assumptions stated in Section 2.2, and as derived in [12], the cell-to-cell probabilities over a single time-step transition $\Delta t$ can be calculated from

$$q(\mathbf{j}, \mathbf{n} | \mathbf{j}', \mathbf{n}', \Delta t) = h(\mathbf{n}|\mathbf{n}', \mathbf{j}' \to \mathbf{j}, \Delta t) \times g(\mathbf{j}|\mathbf{j}', \mathbf{n}', \Delta t) \quad (1)$$

where $g(\mathbf{j}', \mathbf{n}', \Delta t)$ represents the transition probability from cell $\mathbf{j}'$ to $\mathbf{j}$ over $\Delta t$ under configuration $\mathbf{n}'$, and $h(\mathbf{n}|\mathbf{n}', \mathbf{j}' \to \mathbf{j}, \Delta t)$ quantifies the system configuration transition probabilities over $\Delta t$. For each component of interest $m$, a component state transition probability matrix $H_{n_m}$ is constructed. Contents

of this matrix represent the probability of component state transitions over $\Delta t$. These probabilities can be based on hardware component data, such as failure rates, or expert opinion in the absence of reliable data. An example of such a matrix can be seen in Table 1 where $\lambda_{n'_m,n_m}$ denotes the transition rate from $n'_m$ *to* $n_m$.

The cell-to-cell state transition probabilities $g(\mathbf{j}|\mathbf{j}',\mathbf{n}',\Delta t)$ $\Delta t$ can be found from [12, 14, 15, 33],

$$g(\mathbf{j}|\mathbf{j}',\mathbf{n}',\Delta t) = \frac{1}{v_{j'}}\int_{v_{j'}} u_{\mathbf{j}}[\mathbf{x}(\mathbf{x}',\mathbf{n}',\Delta t)]dx' \quad (2)$$

$$u_{\mathbf{j}}[\mathbf{x}(\mathbf{x}',\mathbf{n}',\Delta t)] = \begin{cases} 1 & if\ \mathrm{x} \in v_{\mathbf{j}} \\ 0 & otherwise \end{cases} \quad (3)$$

where $v_{\mathbf{j}}$ is the volume of the cell $\mathbf{j}$,

$$\mathbf{x}(\mathbf{x}',\mathbf{n}',\Delta t) = \int_t^{t+\Delta t} f\big(\mathbf{x}(t'),\ \mathbf{n}'\big)\boldsymbol{dt}' + \mathbf{x}' \quad (4)$$

and $f\big(\mathbf{x}(t'),\ \mathbf{n}'\big)$ represents the system dynamics. For most practical applications the integral in Eq.(2) needs to be quantified using numerical techniques.

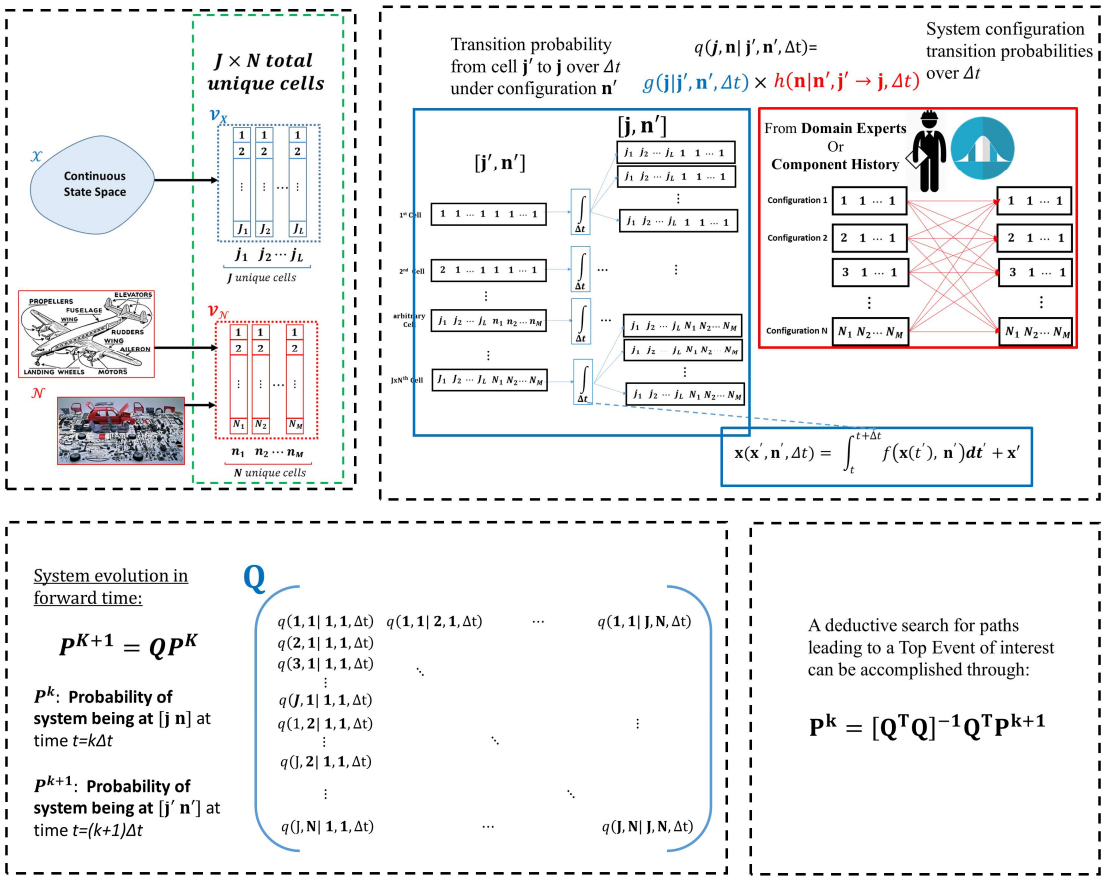


Fig. 2 Deductive Markov/CCMT Implementation Stages

Table 1. Sample System Configuration Transition Matrix ($H_{n_m}$)

| | | Final System Configuration State | | | |
|---|---|---|---|---|---|
| | | Normal State (N) | Fail State 1 ($\boldsymbol{F_1}$) | | Fail State N (FN) |
| Initial System Configuration State | Normal State (N) | $\lambda_{N,N}\Delta t$ | $\lambda_{N,F_1}\Delta t$ | .. | $\lambda_{N,F_N}\Delta t$ |
| | Fail State 1 ($\boldsymbol{F_1}$) | $\lambda_{F_1,N}\Delta t$ | $\lambda_{F_1,F_1}\Delta t$ | .. | $\lambda_{F_1,F_N}\Delta t$ |
| | : | : | : | — | : |
| | Fail State N (FN) | $\lambda_{F_N,N}\Delta t$ | $\lambda_{F_N,F_1}\Delta t$ | .. | $\lambda_{F_N,F_N}\Delta t$ |

A possible approach is to use an equal weight quadrature scheme to sample multiple points from each cell, and run these samples in forward simulation over a single time-step to represent cell-to-cell mappings [14]:

$$g(\mathbf{j}|\mathbf{j}', \mathbf{n}', \Delta t) = \frac{\#\ of\ sampled\ points\ in\ \text{cell}\ \mathbf{j}'\ arriving\ in\ cell\ \mathbf{J}\ over\ \Delta t}{\#\ of\ points\ sampled\ from\ \text{cell}\ \mathbf{j}'} \quad . \tag{5}$$

For an inductive implementation of Markov/CCMT, system trajectories can be obtained through the construction of the system probabilistic cell-to-cell transition matrix $\mathbf{Q} = \{q(\boldsymbol{j}, \mathbf{n}|\, \mathbf{j}', \mathbf{n}', \Delta t), \forall\, \mathbf{j}, \mathbf{j}', \mathbf{n}, \mathbf{n}' \in \mathcal{V}\}$. This matrix is depicted in Fig. 2. Trajectories are recursively obtained via Eq. (6) below

$$\mathbf{P}^{k+1} = \mathbf{Q}\mathbf{P}^{k} \tag{6}$$

where $\mathbf{P}^{k} = \{p^k(1,1),\ p^k(2,1), \cdots, p^k(J,1), \ldots p^k(\mathbf{j},\mathbf{n}), \cdots, p^k(J,N)\}$ is a column vector with elements $p^k(\mathbf{j},\mathbf{n})$ representing the probability of the system continuous variables being at $\mathbf{j}$ with a system configuration $\mathbf{n}$ at a time $k\Delta t$ ($k = 1,2, \ldots$).

For a deductive implementation of BPA, system backwards trajectories can be, in principle, recursively inferred from Eq. (7) as (also see Fig. 2)

$$\mathbf{P}^{k} = [\mathbf{Q}^{T}\mathbf{Q}]^{-1}\mathbf{Q}^{T}\mathbf{P}^{k+1} \quad . \tag{7}$$

### 2.4. Challenges with the Deductive Implementation of Markov/CCMT

There are two main challenges associated with the deductive implementation of Markov/CCMT as presented in Section 2.3:

- The first challenge lies in the construction of $\mathbf{Q}$. This matrix contains as many as$(J \times N)^2$ entries, meaning that a substantial amount of memory space could be required for its storage and substantial processing time to determine its elements.
- The second challenge lies in utilizing $\mathbf{Q}$ to perform backtracking in Eq. (6). The matrix $\mathbf{Q}^{T}\mathbf{Q}$ may not be invertible, meaning that backtracking is not always possible.

## 3. BPA

In Section 3.1, solutions to Markov/CCMT challenges are proposed. A breadth-implementation of BPA is presented in Section 3.2. BPA challenges and proposed solutions are discussed in Section 3.3.

### 3.1. Proposed Solutions to Challenges

BPA was designed to overcome the challenges described in Section 2.4. The BPA algorithm includes the addition of two main components to Markov/CCMT (see Fig. 3):

- In BPA, there is no need for the storage of the $\mathbf{Q}$ matrix, it is sufficient to only store the sampled points sent to the simulator, and the results after one time-step of simulation. This reduces the required storage space for cell to cell transitions from $(J \times N)^2$ to $\leq [2S(J+N)]$, where S is the number of points sampled from each cell in the quadrature scheme.
- Rather than relying on matrix operations to identify event sequences leading to a Top Event of interest, a breadth-first search scheme is developed to construct paths of risk significance leading to the Top Event.

### 3.2. A Breadth-First Implementation of BPA

In this section, a deductive breadth-first search algorithm is presented for the identification of scenarios of risk significance leading to a user defined Top Event of interest. In order to adequately describe the algorithm, several variables need to be introduced. These variables are used for the construction of the BPA search tree and can be seen in Fig. 4. The top node of the search tree is the Top Event of interest, each subsequent search depth represents backwards evolution of system dynamics and configurations over a time-step $\Delta t$. Each depth has $M_i$ cells indexed by $m_i \in [1, M_i]$. For each cell $m_i$, the variable $L^k_{m_i}$ indicates the number of cells that cell $m_i$ branches into at a search depth $i$. A path probability $P_{m_0|m_1|\cdots|m_{K-k+1}}$ to the Top Event is defined at each cell node.

The stepwise implementation of BPA is explained in detail in the pseudocode provided in Table 2. A flowchart of the BPA breadth-first algorithm can be seen in Fig. 5.

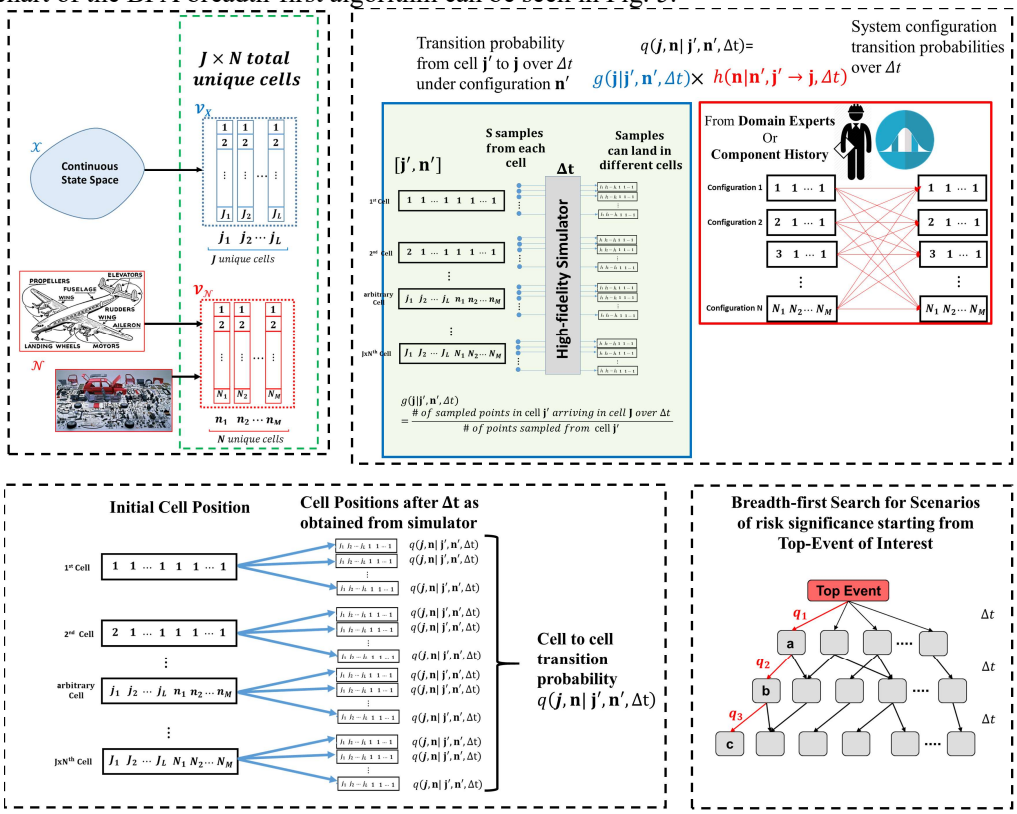


Fig. 3 Deductive BPA Implementation Stages

Table 2: Breadth-first Pseudocode of BPA

| Step | Description |
|---|---|
| **1** | Enter the desired search depth $K$.<br>Initialize the path probability. Initialize the current search depth to 0 (at Top Event). $k = 0$ |
| **2** | Identify all cells leading to the current search depth.<br>There are $M_{K-k+1}$ such cells, where $m_{K-k+1} \in [1,\ M_{K-k+1}]$ is used as an index to these cells. |
| **3** | Initialize $m_i = 1$ for all search depths |
| **4** | Calculate the transition probability of the current path (Top Event to current cell). |
| **5** | Does the calculated probability fall below the user-specified truncation value?<br>If YES, go to Step 6. If NO, store the path and go to Step 6 |
| **6** | Were path probabilities calculated for all of the cells corresponding to the current parent cell?<br>If YES, go to Step 7. If NO, go to Step 9 |
| **7** | Were path probabilities calculated for all cells within the current search depth?<br>If YES, go to Step 8. If NO, go to Step 10 |
| **8** | Is the current search depth at the desired depth?<br>If YES, then END. If NO, go to Step 11 |
| **9** | Go to the next cell along the same level. Go to Step 4 |
| **10** | Go to the next cell along the same level, and the next cell on the level above. Go to Step 4. |
| **11** | Go one level deeper. Go to Step 2. |

### 3.3. Challenges with BPA and Proposed Solutions

While BPA can alleviate challenges associated with Markov/CCMT, use of a BPA breadth-first search scheme has two main limitations.

- Large scale control systems that involve high levels of autonomy and numerous hardware are generally hard to accurately set up and initialize using single BPA implementations.

- For autonomous systems with large state spaces such as platoons or formations of vehicles, combinatorial and computational issues are prone to appear.

There are several extensions to BPA that can help minimize the aforementioned challenges. Two solutions that have been explored by the authors to address the challenges are the following:

- Use of phase-specific BPA implementations, and the integration of analysis results that are obtained from runs over multiple phases.
- Reduction of the system cell-to-cell map by use of a coarser partitioning scheme, and compensating for the coarser scheme by increasing the number of samples taken from each cell in the quadrature scheme. This in turn would reduce the number of computations needed to construct the search tree. Additionally, an upper limit can be imposed on the number of scenarios to be investigated through careful selection of the truncation criterion. Such a bound can be relaxed in new BPA runs once the initially identified scenarios of high risk significance are mitigated.

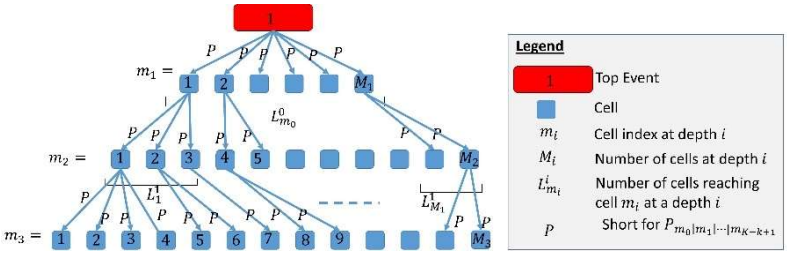


Fig. 4 BPA Search Tree Relevant Notations

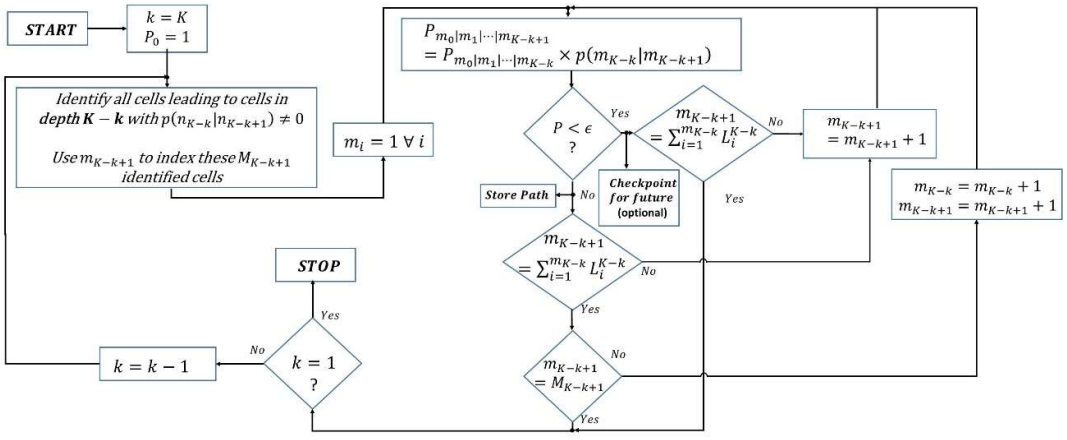


Fig. 5 Breadth-first BPA Flowchart

## 4. BPA Autonomous Vehicle Case Studies

In this section, three autonomous vehicle cases where BPA was implemented for the identification of risk significant scenarios will be presented, along with sample results from one of the implementations. These case studies were taken from [32, 35, 36].

### 4.1. Case Study 1: UAS Lost Link Scenario

A hybrid-state control system was developed for a UAS performing a mission involving take-off, navigation through a series of waypoints, and then landing [32]. The high-level decision making module of the system was augmented with emergency and contingency actions that serve to guide the UAS in the case of a lost link scenario. The system continuous states were defined to be the UAS longitudinal states, where the system component prone to failure was the UAS link state. A Top Event was defined as the UAS falling out of the flight plan. BPA was implemented to identify scenarios of risk significance under the implemented contingency actions that lead to the Top Events of interest. An illustration of the case-study can be seen in Fig. 6a below.

### 4.2. Case Study 2: Unmanned Ground Vehicle (UGV) Collision Avoidance Scenario

A hybrid-state control system was developed for an UGV autonomously operating in highway and urban scenarios [36]. Contingency actions were augmented to the decision making module for collision avoidance with obstacles. The model-based framework introduced in [33, 37] was implemented. BPA was used to identify scenarios of risk significance leading to crashes with obstacles. The information of these risk-significant event sequences was then used to address weaknesses in the implemented contingency actions. BPA was run again to verify that the corrected contingency actions did not lead to a risk-significant violation of the Top Event. An illustration of the scenario in this case-study can be seen in Fig. 6b.

### 4.3. Case Study 3: UAS Sub-nominal Landing Scenario under Icing Conditions

Multiple phase-specific and integrated BPA implementations were used to provide a safety assurance case for a UAS adaptive flight control system capable of handling variations in flight dynamics, hardware components, and the surrounding environment [35]. The case-study presented was for a UAS cruising at altitude that makes it prone to icing, and subsequent nominal/sub-nominal landing. Multiple mission phases were considered: cruise, initial descent, final descent, and flare. Icing accretion was also modeled in the system, along with possible engine failures during landing as a result of the accreted ice. Four instances of BPA were implemented, where each implementation was tailored for each phase of the mission and set up by possibly different domain experts. The Top Event of interest was defined to be a failed landing scenario during flare. The first instance of BPA was implemented in the Flare phase, which contains the Top Event of interest, and subsequent implementations were used to identify event sequences or risk significance that originate in the cruise phase and propagate through intermediate phases, eventually leading to a failure in the flare stage. An illustration of the scenario for this case-study can be seen in Fig. 6c.

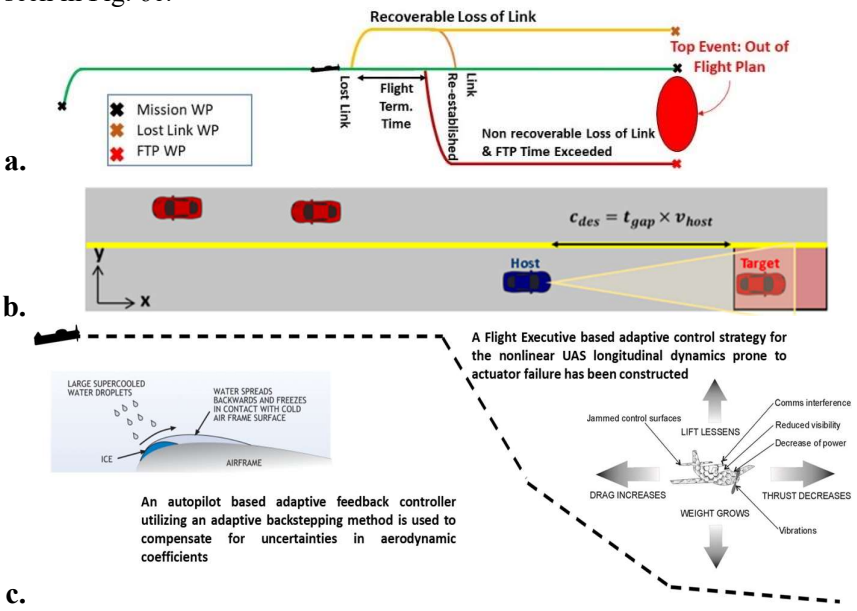


Fig. 6 (a) UAS Lost Link Scenario [32], (b) Autonomous Ground Vehicle Urban Scenario [36], (c) UAS Adaptive Landing Scenario [35]

### 4.4. Sample BPA Results

Figure 7 below provides a table which details the discretization scheme used in the case study presented in Section 4.1. In Fig. 9, the user opts to define the velocity to be in the range of 40 to 60 m/s, the flight path angle $-\pi/6$ to $\pi/6$ rad, the pitch $-\pi/6$ to $\pi/6$, and the pitch rate -0.5 to 0.5 rad/s. The user partitions each of these variables into 1, 1, 1, 1, 8, and 4 partitions respectively. Lost link elapsed time is augmented to the system and defined to be in the range of 0 to 300s based on the implemented contingency actions. Three states are defined for the system component 'Link State': Normal, Recoverable Failure, and Non-recoverable failure. Transitions among system states is determined using a MBD of the system over a

time-step $\Delta t$ =100s, along with domain expert provided data on the system component failure rate probabilities. The Top Event of interest is defined spatially as the UAS being at a location that is not among the defined waypoints. A breadth first scheme of BPA is run over three time-steps for the identification of scenarios that lead to the violation of the Top Event with risk significant probabilities. Results of this implementation can be seen in Fig. 8. Note that each cell is represented by an 8-tuple, where each integer corresponds the partition number of the discretized variables in Fig. 7.

| Cell # | | $x_1$ Vel. (m/s) | $x_2$ FPA (rad) | $x_3$ Pitch Rate (rad/s) | $x_4$ Pitch (rad) | $x_5$ Alt. (m) | $x_6$ X-Pos. (m) | $x_7$ Lost Link Elapsed T. (s) | $s_1$ Link State |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | *All Range* $40 \rightarrow 60$ | *All Range* $-\pi/6 \rightarrow \pi/6$ | *All Range* $-0.5 \rightarrow 0.5$ | *All Range* $-\pi/6 \rightarrow \pi/6$ | $5500 \rightarrow 5875^{-}$ | $0 \rightarrow 4500^{-}$ | $0 \rightarrow 100^{-}$ | *Normal* |
| | 2 | | | | | $5875 \rightarrow 6250^{-}$ | $4500 \rightarrow 9000^{-}$ | $100 \rightarrow 200^{-}$ | *Rec. failure* |
| | 3 | | | | | $6250 \rightarrow 6625^{-}$ | $9000 \rightarrow 13500^{-}$ | $200 \rightarrow 300^{-}$ | *Unrec. Failure* |
| | 4 | | | | | $6625 \rightarrow 7000^{-}$ | $13500 \rightarrow 18000$ | | |
| | 5 | | | | | $7000 \rightarrow 7375^{-}$ | | | |
| | 6 | | | | | $7375 \rightarrow 7750^{-}$ | | | |
| | 7 | | | | | $7750 \rightarrow 8125^{-}$ | | | |
| | 8 | *Variables with a single cell defined over the entire range means that the user opts to "not care" about the values they take. | | | | $8125 \rightarrow 8500^{-}$ | | | |

Fig. 7 Continuous Space Representation of Discretized Cell Space [32]

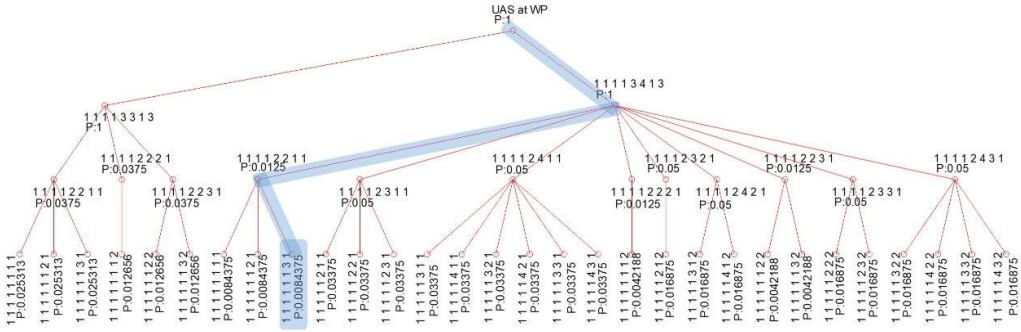


Fig. 8 Breadth-first BPA Flowchart [32]

Taking the shaded branch in Fig. 10 as an example to interpret the results from the search tree, we observe the following:

**[1 1 1 1 1 1 1 2]**– The UAS initially has a velocity of 40-60 m/s, a flight path angle of –π/6 to π/6 rad, a pitch rate of -0.5 to 0.5 rad/s, a pitch angle of –π/6 to π/6 rad, an altitude of 5500 to 5875m, an x-Position of 0 to 4500m, and a recoverable loss of link after a 0-100s lost link elapsed time.

**[1 1 1 1 2 2 2 1]** – One time step later, the UAS has a velocity of 40 to 60 m/s, a flight path angle of –π/6 to π/6 rad, a pitch rate of -0.5 to 0.5 rad/s, a pitch angle of –π/6 to π/6 rad, an altitude of 5875 to 6250m, an x-Position of 4500 to 9000m, and normal link state after a 100 to 200s lost link elapsed time.

**[1 1 1 1 3 4 1 3]**– One time step later, the UAS has a velocity of 40 to 60 m/s, a flight path angle of –π/6 to π/6 rad, a pitch rate of -0.5 to 0.5 rad/s, a pitch angle of –π/6 to π/6 rad, an altitude 6250 to 6625m, x-Position of 13,500 to 15,000m, and non-recoverable link failure after 0 to 100s lost link elapsed time.

**Out of Flight Plan** – One time step later the UAS reaches the final destination at a point that is not within the Flight Plan. This scenario occurs with a conditional probability of 0.0084375.

## 5. CONCLUSION

The development of a set of procedures and methods for autonomous vehicle control system assurance is vital for the safe deployment of such systems in civilian applications. In this paper, an overview was presented for a deductive implementation of Markov/CCMT for such a purpose. The BPA was proposed as an algorithm that can be used to overcome some of the challenges typically faced by Markov/CCMT implementations. The capability of BPA to consider the epistemic and uncertainties on a phenomenologically and stochastically consistent platform is especially advantageous when considering the insufficient operating experience of autonomous vehicles. A summary of the different autonomous systems case studies on which BPA was successfully implemented was presented in this paper. In all

case studies, BPA was capable of identifying risk significant event sequences leading to undesirable Top Events. Future work will involve further development of BPA to efficiently handle large-scale systems, and systems with large state spaces.

**Acknowledgements**
The work is partially funded by the National Science Foundation (NSF) Cyber-Physical Systems (CPS) project under contract 60046665. An application BPA to Unmanned Aircraft Systems (UAS) was developed with ASCA Inc. as part of a project funded by the NASA Ames Research Center (ARC). Discussions in that project with Drs. Sergio Guarro, and Michael Yau from ASCA Inc., and Dr. Matt Knudson from NASA ARC are gratefully acknowledged.

**References**

[1] Alam, A., Gattami, A., Johansson, K. H., & Tomlin, C. J. *"Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations."* Control Engineering Practice, 24, 33-41, (2014).

[2] Althoff, M., & Dolan, J. M. *"Online verification of automated road vehicles using reachability analysis."* IEEE Transactions on Robotics, 30(4), 903-918, (2014).

[3] Lawitzky, A., Nicklas, A., Wollherr, D., & Buss, M. *"Determining states of inevitable collision using reachability analysis."* In International Conference on Intelligent Robots and Systems, IROS 2014, pp 4142-4147, (2014).

[4] Park, J., Kurt, A., & Özgüner, Ü. *"Hybrid Systems Modeling and Reachability-Based Controller Design Methods for Vehicular Automation."* Unmanned Systems, 2(02), pp. 101-119, (2014).

[5] Andrews, J. D., Prescott, D. R., & Remenyte-Prescott, R. *"A systems reliability approach to decision making in autonomous multi-platform systems operating a phased mission."* In Annual Reliability and Maintainability Symposium 2008. pp. 8-14, (2008).

[6] Zeitlin, A., Lacher, A., Kuchar, J., & Drumm, A. "*Collision avoidance for unmanned aircraft: Proving the safety case"* (No. MP-060219). MITRE CORP MCLEAN VA, (2006).

[7] Kuchar, J. K. *"Safety analysis methodology for unmanned aerial vehicle (UAV) collision avoidance systems."* In USA/Europe Air Traffic Management R&D Seminars Vol. 12, (2005).

[8] Snooke, N. A. *"Automated failure effect analysis for PHM of UAV"*. In Handbook of Unmanned Aerial Vehicles. Springer Netherlands. pp. 1027-1051, (2015).

[9] Freeman, P., & Balas, G. J. *"Actuation failure modes and effects analysis for a small UAV."* In American Control Conference (ACC), 2014 pp. 1292-1297, (2014).

[10] Aldemir, T.*"A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants"*. Annals of Nuclear Energy, *52*, pp. 113-124. (2013).

[11] Devooght, J., & Smidts, C. *"Probabilistic reactor dynamics—I: the theory of continuous event trees"*. Nuclear science and engineering, 111(3), pp. 229-240, (1992).

[12] Aldemir, T. *"Computer-assisted Markov failure modeling of process control systems"*. IEEE Transactions on reliability, 36(1), pp. 133-144, (1987).

[13] Tombuyses, B., & Aldemir, T. *"Dynamic PSA of process control systems via continuous cell-to-cell mapping."* In Probabilistic Safety Assessment and Management, New York: Springer-Verlag, pp. 1541-1546 (1996).

[14] Aldemir, T. *"Utilization of the cell-to-cell mapping technique to construct Markov failure models for process control systems"*. PSAM Proceedings, Elsevier Publishing Company Co. Inc., N*Y*, pp. 1431-1436, (1991).

[15] Yang, J., & Aldemir, T. *"An algorithm for the computationally efficient deductive implementation of the Markov/Cell-to-Cell-Mapping Technique for risk significant scenario identification"*. Reliability Engineering & System Safety, 145, pp. 1-8. (2016).

[16] Hassan, M., & Aldemir, T. *"A data base oriented dynamic methodology for the failure analysis of closed loop control systems in process plant"*. Reliability Engineering & System Safety, 27(3), 275-322. (1990).

[17] Aldemir, T., Stovsky, M., Kirschenbaum, J., Mandelli, D., Bucci, P., Mangan, L. & Yau, M. *"NUREG/CR-6942: Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments"*. Office of Nuclear Regulatory Research, US Nuclear Regulatory Commission, Washington, DC, (2007).

[18] Krystul, J., & Blom, H. *"A. Sequential Monte Carlo simulation of rare event probability in stochastic hybrid systems."* IFAC Proceedings Volumes, 38(1), pp. 176-181, (2005).
[19] Stroeve, S. H., Blom, H. A., & Bakker, G. B. *"Systemic accident risk assessment in air traffic by Monte Carlo simulation"*. Safety science, 47(2), pp. 238-249, (2009).
[20] Cojazzi, G. "The DYLAM approach for the dynamic reliability analysis of systems". Reliability Engineering & System Safety, 52(3), pp. 279-296, (1996).
[21] Catalyurek, U., Rutt, B., Metzroth, K., Hakobyan, A., Aldemir, T., Denning, R., & Kunsman, D. *"Development of a code-agnostic computational infrastructure for the dynamic generation of accident progression event trees"*. Reliability Engineering & System Safety, 95(3), pp. 278-294, (2010).
[22] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, M. Woltereck, *"An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties"*, Reliability Engineering and System Safety, 77: pp. 229-238, (2002).
[23] C. Picoco, T. Aldemir, V. Rychkov, A. Alfonsi, D. Mandelli, C. Rabiti, *"Coupling of RAVEN and MAAP5 for the Dynamic Event Tree analysis of Nuclear Power Plants"*, Proceedings of the European Safety and Reliability Conference, Portoroz, p.407, (2017).
[24] Y.H. Chang, A. Mosleh. *"Dynamic PRA using ADS with RELAP5 code as its thermal hydraulic module"*. Mosleh, A., Bari, R. (Eds.). PSAM 4, Springer-Verlag, New York: pp. 2468-2473, (1998).
[25] Yau, M., S. Guarro, and G. Apostolakis. *"Demonstration of the dynamic flowgraph methodology using the Titan II space launch vehicle digital flight control system."* Reliability Engineering & System Safety 49, no. 3, pp. 335-353, (1995).
[26] Guarro, S., M. Yau, and S. Dixon. *"Applications of the dynamic flowgraph methodology to dynamic modeling and analysis."* In Proceedings of the 11th International Conference on Probabilistic Safety Assessment and Management, (2012).
[27] Yau, Michael, George Apostolakis, and Sergio Guarro. "*The use of prime implicants in dependability analysis of software controlled systems."* Reliability Engineering & System Safety 62, no. 1, pp. 23-32, (1998).
[28] Aldemir, Tunc, Sergio Guarro, Diego Mandelli, Jason Kirschenbaum, L. Anthony Mangan, Paolo Bucci, Michael Yau et al. *"Probabilistic risk assessment modeling of digital instrumentation and control systems using two dynamic methodologies."* Reliability Engineering & System Safety 95, no. 10, pp. 1011-1039, (2010).
[29] Geng, Qichuan, Haibin Duan, and Shuangtian Li. *"Dynamic fault tree analysis approach to safety analysis of civil aircraft."* In Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference on, pp. 1443-1448, (2011).
[30] Yiping, Yao, Yang Xiaojun, and Li Peiqiong. *"Dynamic fault tree analysis for digital fly-by-wire flight control system."*, 15th AIAA/IEEE Digital Avionics Systems Conference. pp.479-484, (1996).
[31] Yevkin, Olexandr. *"An improved Monte Carlo method in fault tree analysis"*. In Reliability and Maintainability Symposium (RAMS), 2010 Proceedings-Annual, IEEE, pp. 1-5, (2010).
[32] Hejase, M., Kurt, A., Aldemir, T., Ozguner, U., Guarro, S. B., Yau, M. K., & Knudson, M. D. *"Quantitative and Risk-Based Framework for Unmanned Aircraft Control System Assurance"*. Journal of Aerospace Information Systems, pp. 1-15, (2017).
[33] Guarro, S., Yau, M. K., Ozguner, U., Aldemir, T., Kurt, A., Hejase, M., & Knudson, M. *"Formal Framework and Models for Validation and Verification of Software-Intensive Aerospace Systems"*. In AIAA Information Systems-AIAA Infotech@ Aerospace, p. 0418, (2017).
[34] Hsu, C. S. *"A theory of cell-to-cell mapping dynamical systems"*. Journal of Applied Mechanics, 47(4), pp. 931-939, (1980).
[35] Hejase, M., Kurt, A., Aldemir, T., Ozguner, U., Guarro, S., Yau, M. K., & Knudson, M. *"Dynamic Probabilistic Risk Assessment of Unmanned Aircraft Adaptive Flight Control Systems"*. In 2018 AIAA Information Systems-AIAA Infotech@ Aerospace, p. 1982, (2018).
[36] Hejase, M., Kurt, A., Ozguner, U., Aldemir, T. *"Identification of Risk Significant Automotive Scenarios Under Hardware Failures"*. To appear in proceedings of 2nd International Workshop on Safe Control of Autonomous Vehicles (2018).
[37] Guarro, S., Yau, M. K., Ozguner, U., Aldemir, T., Kurt, A., Hejase, M., & Knudson, M. *"Risk Informed Safety Case Framework for Unmanned Aircraft System Flight Software Certification"*. In AIAA Information Systems-AIAA Infotech@ Aerospace, p. 0910, (2017).