Submitted, IEEE Real-Time Systems Symposium (RTSS)
http://www.cds.caltech.edu/~murray/preprints/ccom16-rtss_s.pdf

Synthesis of Control Protocols for Switched Electrical Power Systems for Commercial Applications with Safety Specifications

Benson Christalin[1], Michele Colledanchise[2], Petter Ögren[2] Richard Murray[1]

***Abstract*— This paper presents a method for synthesizing fault tolerant control protocols for a deterministic discrete event system subject to safety specifications. The system discussed in the paper is modeled as a finite state machine (FSM) and Behavior Tree (BT). The synthesis procedure involves formulating the policy problem as a shortest path dynamic programming problem, and performing a backward search from the desire sates or behavior to the initial configuration. The search is performed over all possible states when applied to the FSM, or over all possible actions when applied to the BT. The resulting strategy minimizes the number of actions performed to meet operational objectives without violating safety conditions. The effectiveness of the procedure on FSMs and BTs is demonstrated through three examples of switched electrical power systems for commercial applications.**

I. Introduction and Motivation

In an effort to facilitate the increasing complexity and automation of smart systems, there has been a growing demand for reliable and efficient electrical power systems (EPS). Computational complexity theory provides tools to handle modeling, analyzing, and ensuring reliability of the intricate circuitry and operation of these safety critical electrical power systems. These tools enable the synthesis of control protocols that allow an electrical power system to quickly actuate to meet system objectives, while maintaining safe operating conditions. The synthesis of reactive control protocols involves modeling the system, safety, and mission specifications in a tractable form, and designing an algorithm that uses the specifications and outputs an optimal policy. This paper details the construction of two independent models of the electrical power system, provides an algorithm to synthesize reactive controllers, evaluates complexity of the algorithm in relation to the models, and demonstrates the realizability of the control protocols.

Synthesizing the controls protocols requires examining the event space, and constructing sequential decisions to achieve goals under uncertainty. There has been work conducted that implements correct-by-construction control synthesis to power allocation and distribution in aircraft electric power systems [1] [2]. These papers use linear temporal logic (LTL) specifications to synthesize a controller that is guaranteed, by construction, to satisfy formalized properties. The LTL specifications accounted only for safety and requires GR(1) design. Using the collection of mathematical tools afforded by dynamic programming this paper demonstrates the synthesis of a reactive controller for an electrical power system, modeled by a finite state machine (FSM) diagram and a behavior tree (BT), and outlines the extension needed to account for liveness specifications. Furthermore, multiple electrical power system topologies are discussed in this paper. The first allow readers to gain an intuition for the synthesis algorithm, the second example is motivated by an experimental test fixture used to validate automatically synthesized reactive control protocols [3], and the last is an industrial application from [4], seen in figure 11, demonstrating feasibility and scalability of the synthesis procedure.

The electrical power system exemplifies a discrete event system. Such a modeling approach is fitting since this paper only considers actuation of an electrical power system occurring at finite time intervals, assumes the state of the electric power system is static unless triggered, and the information is represented in discrete form. A FSM and BT was used to model the electrical power system. The FSM for the electrical power system can be represented by a directed graph known as a FSM diagram. The nodes of the graph are the states of the system and the edges are transitions. With such an abstraction one can describe the necessary conditions to reach certain states using modal logic [5]. Furthermore, state transition diagrams are intrinsically sequential in nature, which aligns with the assumptions for actuation of the electrical power system. However, state transition diagrams can be infeasible. The conventional FSM diagram formalism requires explicit representation of all states, and as the model grows linearly, the number of states grows exponentially. There have been techniques developed to address the exponential complexity such as binary decision diagrams [6]. An alternative to state transition diagrams are behavior trees (BT). First introduced in the computer gaming industry [7] to meet their needs of compactness, modularity, and reusability in the artificial intelligence for non player characters [8], [9], BTs are a recent alternative to Controlled Hybrid Systems (CHSs) for fault tolerant execution of tasks. Different studies highlight the advantages of BTs over more classical CHSs in various applications [10]–[14], in particular robotic applications [15]–[17]. BTs are often used to describe fully reactive systems in a convenient and compact way [17]. A comparative analysis of the use of behavior trees and transition diagrams as models of the electrical power system is provided, demonstrating the advantages of the compactness of the BTs to reduce the computational complexity synthesizing the controls.

[1]The authors are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, USA bchrista@caltech.edu

[2]The authors are with the Computer Vision and Active Perception Lab., Centre for Autonomous Systems, School of Computer Science and Communication, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden { miccol|petter}@kth.se

The remainder of the paper provides: a description of the electrical power system and problem description, a simple illustration of the proposed solution, an exposition of the mathematical setting, followed by the formal problem statement, a demonstration of the synthesis procedure including two examples used to compare the transition diagram and behavior tree, and concluding remarks.

II. Framework

A. Electrical Power System

An electrical power system is a modular collection of components and circuits necessary for the generation, transmission and distribution of power. The typical components of an electrical power system are generators/alternators, rectifier units, transformers, contactors, batteries, and loads. Generators provide the source of power that the system will convert into electrical energy. Rectifier units convert alternating current (AC) to direct current (DC). Transformers use electromagnetic induction through a "step-up" or "step-down" process that involves the increase or decrease of voltages to transfer electrical energy. Batteries store electrical energy. Contactors connect and disconnect circuit components to maintain system requirements while allowing the electrical power system to provide energy to the loads. And loads consume the electrical energy to perform a function. These components can be arranged to form various topologies in order regulate and control the conversion and transmission of electrical energy for consumption. Maintaining power flow is the key objective of the electrical power system for safety critical applications. For this paper it is assumed the electrical power system operates under a Gray code specification and therefore only one contactor is actuated at each finite time interval.

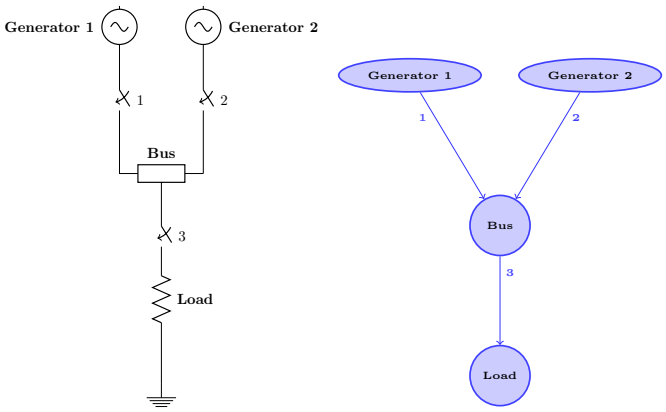
B. Problem Description

Given an electrical power system topology, an initial state, and objective, determine the minimal number of actions required that do not violate safety specification and yield the final optimal contactor configuration. For a configuration to be optimal means that the EPS satisfies system, safety, and mission specifications.

III. Proposed Solution

This section provides a colloquial description of the proposed solution. Section VI of the paper provides a formal presentation of the solution. The first step of the proposed approach is to construct a graphical model of the EPS where each vertex represents a component and each edge represents a contactor. Next, from the graphical model identify all contactor configurations, highlighting those that satisfy the mission and violate safety requirements. Contactor configuration is denoted by a vector where if the i^{th} entry is positive the i^{th}-contactor is on and if negative the i^{th}-contactor is off. Actuation of the system corresponds to a transition and the observed contactor configuration corresponds to a state of the system. Use the states and the transitions to create a FSM and FSM diagram. Now, derive the control policy. For the FSM-based policy, a search is performed on the FSM diagram to identify a path from the initial configuration to the a goal configuration. Transitions are assigned cost values to ensure specifications are satisfied. In the BT-based policy a recursive search is performed to find the sequence of actions needed to satisfy the mission and safety requirement and tree ordering is used to address liveness specifications. Regardless of the EPS model, a backward search approach from the goal configuration identifies the actions that yield the desired results. Before performing an action, the BT ensures that it does not violate any safety requirement (e.g. it turns off some other contactors to avoid safety violation). Among all the possible BTs or paths on the FSM diagram, the one that represents the least number of actions from the initial configuration to the terminal configuration is chosen.

Example 1: Consider the EPS in figure 1(a). The graph representation of the EPS is depicted in figure 1(b). The mission is to power the load without short-circuits. The initial condition of the system is all contactors opened, $x_0 = [-1, -2, -3]$. The indexed set of terminal configurations that satisfy the mission is denoted by, $\mathcal{M}$; thus, for this example, $\mathcal{M} = \{[1, -2, 3] \cup [-1, 2, 3]\}$. The sates that correspond to a safety violation are denoted by $\mathcal{S}$ and therefore, for this example, $\mathcal{S} = \{[1, 2, 3] \cup [1, 2, -3]\}$.



(a) EPS. (b) Graph modeling the EPS.

Fig. 1. EPS and Graph for Example 1. The EPS has four components: two generators, one bus, and one load, and three contactors: contactor 1, 2, and 3.

Notice that all the possible states and transitions of the electrical power system FSM are depicted in the FSM diagram shown in figure 2. In the figure, the dotted lines denote the transitions that yield a safety violation, the boxed vertices are viable terminal configurations, and the arrow indicates the initial state. A weighted value of infinity is assigned to edges related to transitions to unsafe states. Weighted values are also assigned to edges indicating preferential states that relate to the liveness specifications. These values are stored in a cost matrix and used to when computing value and policy iteration, which provide the optimal set of actions from every state to every state. The distant from one state to another is the sum of the weighted edges that make up the path connecting the two nodes. Examining example 1, assume that all non-dotted edges in figure 2 have a value of one;

one can immediately identify the paths through safe states, and the related actions, that result in a satisfactory terminal state: close contactor 2 and then close contactor 3 or close contactor 1 and then close contact 3. Since both paths are equidistant from the initial position and no preference was indicated, a probabilistic approach or which ever sequences was queued first. The size of the FSM diagram is $2^{\#\text{components}}$-nodes and $\#\text{components} \times 2^{\#\text{components}}$-edges.

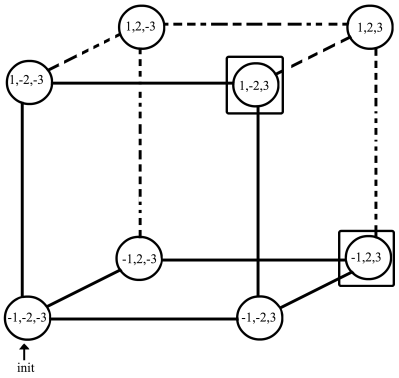


Fig. 2. FSM Diagram of Example 1

For the BT, figure 3 shows the construction of the tree $\mathcal{T}_{M_1}$ step by step. Algorithm 4 creates the initial $\mathcal{T}_{M_1}$ depicted in figure 3(a). $\mathcal{T}_{M_1}$ returns failure since the condition *is* 1 *CLOSE* is not satisfied. Then, Algorithm 4 expands the tree as seen in figure 3(b). Now, $\mathcal{T}_{M_1}$ returns failure since the condition *is* 3 *ON* is not satisfied. Again, Algorithm 4 expands the tree as shown in figure 3(c). $\mathcal{T}_{M_1}$ finally returns success. $\mathcal{T}_{M_2}$ is constructed in a similar way and depicted in figure 3(d). Given the initial state, both $\mathcal{T}_{M_1}$ and $\mathcal{T}_{M_2}$ results in two operations performed hence their order in $\mathcal{T}$ is irrelevant. If there is a preference indicated by a liveness specification, simply order the tree accordingly. $\mathcal{T}$ is depicted in figure 4.

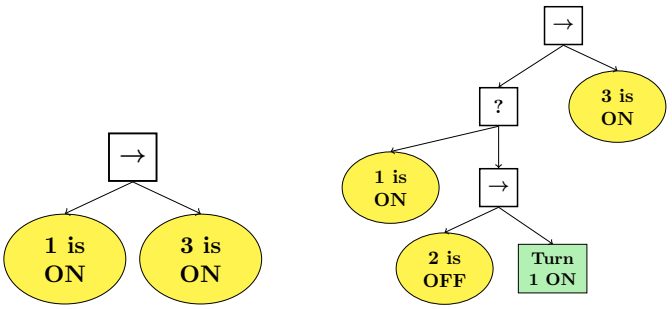
In order to demonstrate a rigorous form of the aforementioned solution to the control synthesis problem there are mathematical preliminaries necessary, which are presented in the next section.

IV. Mathematical Background

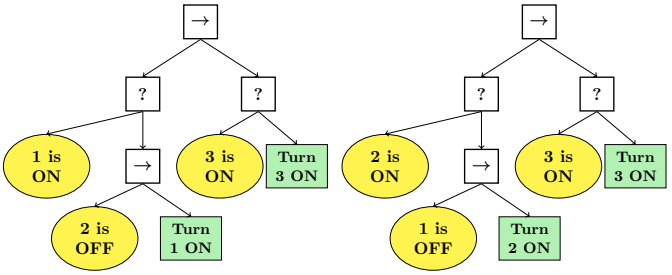
A. EPS → Graph → Finite State Machine

This section summarizes the mathematical background needed for the formulation of the control synthesis problem and solution. First, using the usual definition, let $\mathcal{G} = (V, E)$ be a undirected *graph*. V refers to the *nodes* or *vertices* of the graph, denoted by $v \in V$, and E refers to the *edges* of the graph, which do not have direction, and therefore are denoted using set notation, $\{v, w\} \in E$ or $\{w, v\} \in E$. A *path* is a finite sequence of nodes $\langle v_0, v_1, \ldots, v_\kappa \rangle$. The set of paths of $\mathcal{G}$ is $\mathrm{Paths}(\mathcal{G})$. The electrical power system is a network of components and an inherent topological representation of a network is a graph. Table I shows the circuit symbols and their corresponding graphical representation.

A single path on the graph denotes connection between circuit components and the flow of power. $\mathrm{Paths}(\mathcal{G})$ represents



(a) First $\mathcal{T}_{M_1}$ of Example 1. (b) Updated $\mathcal{T}_{M_1}$ of Example 1



(c) Final $\mathcal{T}_{M_1}$ of Example 1 (d) Final $\mathcal{T}_{M_2}$ of Example 1

Fig. 3. Construction of $\mathcal{T}_{M_1}$ and $\mathcal{T}_{M_2}$ for Example 1.

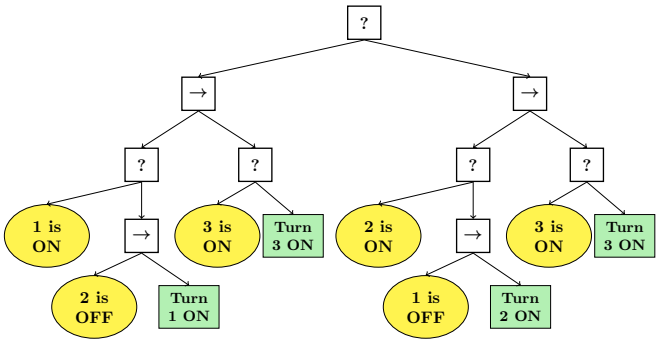


Fig. 4. $\mathcal{T}$ of Example 1

all configurations/states of the electrical power system. For the electrical power system discussed in this paper only one switch/contactor can be actuated at a time time step thus for each state and input there is only one transition. The states and transition of the electrical power system can be represented with a deterministic finite state machine.

Definition 1: A deterministic finite state machine is a quadruple (X, Σ, f, Y) where:

- a finite set of states, X
- a finite set called the input alphabet, Σ
- a transition function mapping pairs of a state and an input symbol to the corresponding next state, $f : X \times \Sigma \to X$
- a finite set of final states, $Y \subset X$

The FSM too has a graphical representation and moving forward, vertices shall refer to components on the graphical abstraction of the EPS and nodes shall refer to the states on the graphical abstraction on the FSM or as defined on a behavior tree, detailed in the next subsection.

TABLE I. Symbols used and their graphical representations

Symbol	Description	Graphical Symbol
$\text{ACGen} \subseteq V$	AC Generators	
$\mathcal{C} \subset E$	Contactors	
		AC
$\text{DCLoad}, \text{ACLoad} \subseteq V$	DC, AC Loads	DC
$\text{RU} \subseteq V$	Rectifier Units	RU

B. Behavior Trees

This subsection presents a digest of BTs, describing the execution of the nodes used throughout the paper. Refer to [9] for a more detailed description.

A BT is a directed tree where the internal nodes are classified as *control flow nodes* and the leaf nodes as *execution nodes*, using the usual definition of parent and children for each connected nodes. Graphically, the children of a control flow node are sorted from its bottom left to its bottom right, as depicted in Figures 5-6. The execution of a BT starts from the root node (i.e. the control flow nodes with no parents), which sends *ticks* to its children. When a node in a BT receives a tick, its execution starts and it returns to its parent a status of *running* if its execution is under completion; *success* if its execution is accomplished; or *failure* if the execution cannot be accomplished. Here we describe the execution of the two control flow nodes (selector, sequence) and the execution nodes (action and condition).

Fallback node (also known as Selector or Priority): When a fallback node receives a tick, then it ticks its children in succession from left to right, until a child returns the status of success or running. Then this status is returned to the parent of the fallback node. A fallback node returns failure only when all the children return a status failure. The purpose of the fallback node is to carry out a task that can be performed using different approaches (e.g. powering a bus can be either done by switching the generator on or by plugging the external battery). A fallback node is graphically represented as a box with a "?", as in Fig. 5.

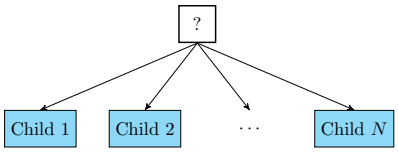


Fig. 5. Graphical representation of a fallback node with N children.

Sequence node: When a sequence node receives a tick, then it ticks its children in succession from left to right, until a child returns the status of failure or running. Then this status is returned to the parent of the sequence node. A sequence node returns success only when all the children return a status success. The purpose of the sequence node is to carry out a task that is defined as a strict sequence of sub-tasks (e.g. powering a load need to have the load connected and rhe generator on) A fallback node is graphically represented as a box with a "$\rightarrow$", as in Fig. 6..

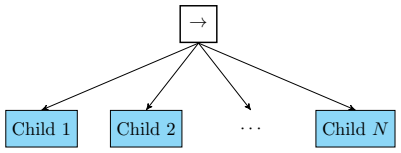


Fig. 6. Graphical representation of a sequence node with N children.

(a) Action node. (b) Condition node.

Fig. 7. Graphical representation of action and condition nodes.

Action node: When an action node receives a tick, it returns the status of success if the action is completed or failure if the action cannot be completed. Whit it is performing the action, it returns the status of running. An action node is graphically represented as in Fig. 7(a)

Condition: When a condition node receives a tick, it returns the status of success if the condition is satisfied or failure otherwise. A condition node never returns running. A condition node is graphically represented as in Fig. 7(b).

C. Dynamic Programming

To find the appropriate set of actions to satisfy the mission objective a search is performed on both the FSM and BT model. However, for the FSM the search is performed over states. Consider the FSM definition previously provided. X is a non-empty state space. From each state, $x_k \in X$, there is a input $\sigma_k \in \Sigma(x)$ associated with a transition from state x_k to $x_{k+1} = f(x_k, \sigma_k)$. Also, there is a cost $g(x_k, \sigma_k)$ additive cost associate with each state with each action. For the purpose of this paper the cost to-go is captured by an adapted adjacency matrix from the graph of the FSM diagram with values corresponding to the weighted edges of the FSM diagram. There exist several policies or control laws made of a sequence of actions $\mu = \{\sigma_0, \ldots, \sigma_{N-1}\}$. Therefore our problem can be formulated for the FSM model as given an initial state x_0, find the optimal policy π^* that actuates the system from state x_0 to x_{final} by minimizing the cost

$$J_{mu^*}(x_0) = \min_{\mu \in U} J_\mu(x_0).$$

The optimal cost function is computed by backward value iteration, provided in the next section, which is then use in policy iteration to find the optimal policy, μ^*, that minimizes the cost for a given initial conditional to a final state. For more detail on value and policy iteration refer to [18].

V. Problem Statement

The objective is to minimize the expected number of switches to get to safe-on final state while always remaining in $X_{safe-on} \cup X_{safe-off}$ where $\mathcal{M} \subseteq X_{safe-on}$. In this case, the transition cost is 1 if the transition is to safe space of the events space, 0 if the transition is $\mathcal{M}$ or ∞ otherwise.

VI. Design and Implementation

Using successive iteration, we converge on a value function and policy that account for the entire event space and is

Algorithm 1: value iteration

Input: cost function, states, & actions
Output: value function

initialization Set $k = 0$ **repeat**
  foreach k **do**
    foreach *state* x **do**
      compute vector

$$J_{k+1}(x) = \arg\min_{\sigma \in \Sigma} [g(x, \sigma) + \sum_{j-1}^{n} g_{x,j}(\sigma) J_k(j) \text{ [18]}$$

      $J^*(x) \leftarrow J(x)$
until $J_{k+1} \approx J_k$;
return J_k^*

Algorithm 2: policy iteration

Input: initial value function vector
Output: policy

initialization Set $k = 0$. Find an admissible policy μ^0;
repeat
  foreach k **do**
    foreach *state* x **do**
      compute vector

$$\mu^{k+1} = \arg\min_{\sigma \in \Sigma} [g(x, \sigma) + \sum_{j-1}^{n} c_{i,j}(\sigma) J_{\mu^k}(j) \text{ [18]}$$

      $\mu^*(x) \leftarrow \mu(x)$
until $\mu_{k+1} \approx \mu_k$;
return μ_k^*

therefore the convergent rate is dependent on the event space. Under some regularity assumptions outlines in [19], there is a unique value function corresponding stationary policy π that can arrive at the optimal value function by value iteration.

A. BT part

We aim to create a BT that returns success if and only if the current contactor configuration satisfies the mission requirement without violating the safety specifications.

For each contactor configuration $M \in \mathcal{M}$ we create a BT $\mathcal{T}_M$ that returns success if and only if the such configuration is met. Initially this $\mathcal{T}_M$ is composed by conditions only (Algorithm 4 Line 9). We execute $\mathcal{T}_M$ on the graph starting with the initial configuration M_0. If the the contactor configuration M is not met, $\mathcal{T}_M$ returns failure. We identify which single element $m_i \in M$ is not met (Algorithm 4 Line 12). For each m_i we identify a BT that will met such single contactor configuration without violating any safety requirements (Algorithm 4 Line 15). We repeat the procedure until $\mathcal{T}_M$ return success. Finally we order $\mathcal{T}$ to achieve optimality. In the ordered BT, each fallback node has its children ordered by the the number of action in an ascending fashion.

Algorithm 3: get_safe_subtree

input : Single contactor configuration: m
Mission Configuration: M
Violating Configurations: $\mathcal{S}$
output: Subtree to safely satisfy m

for $S \in \mathcal{S}$ **do**
  $\mathcal{T}_S \leftarrow$ `fallback_node()`
  if $m \in S$ **then**
    $\tilde{S} \leftarrow S \backslash M$
    for $\tilde{s} \in \tilde{S}$ **do**
      if $\tilde{s} > 0$ **then**
        $\tilde{c} \leftarrow$ `condition_node( Contactor` $\tilde{s}$ `Is OFF)`
      else
        $\tilde{c} \leftarrow$ `condition_node( Contactor` $\tilde{s}$ `Is ON)`
      $\mathcal{T}_S$`.add_child(`$\tilde{c}$`)`
  if $m > 0$ **then**
    $a \leftarrow$ `action_node(Turn Contactor` m `ON)`
    $c \leftarrow$ `condition_node( Contactor` m `Is ON)`
  else
    $a \leftarrow$ `action_node(Turn Contactor` m `OFF)`
    $c \leftarrow$ `condition_node( Contactor` m `Is OFF)`
  $\mathcal{T}_a \leftarrow$`sequence_node()`
  $\mathcal{T}_a$`.add_child(`$\mathcal{T}_S$`)`
  $\mathcal{T}_a$`.add_child(`a`)`
  $\mathcal{T} \leftarrow$`fallback_node()`
  $\mathcal{T}$`.add_child(`c`)`
  $\mathcal{T}$`.add_child(`$\mathcal{T}_a$`)`
return $\mathcal{T}$

Algorithm 4:

input : Initial configuration: M_0
Mission Configurations: $\mathcal{M}$
Violating Configurations: $\mathcal{S}$
output: BT $\mathcal{T}$

```
T ← fallback_node()
for M ∈ M do
    T_M ← sequence_node()
    for m ∈ M do
        if m > 0 then
            c ← condition_node(Is Contactor m ON)
        else
            c ← condition_node(Is Contactor m OFF)
        T_M.add_child(c)
    status ← T_M.execute_on_graph(M_0)
    while status ≠ success do
        C ← T_M.failed_conditions()
        for c ∈ C do
            c̃ ← get_value_of(c)
            T_c ← get_safe_subtree(c̃, M, S)
            T_M.replace_child(c, T_c)
        status ← T_M.execute_on_graph(M_0)
    T.add_child(T_M)
T.reorder()
return T
```

VII. Algorithm Analysis

A. Computation of FSM Control Policy

The abstraction used to perform calculations on the FSM was extended from the EPS graph. The complexity of the construction of the EPS graph is $O(V+E)$. The nodes of the transition diagram are 2^V and the edges are $V \times 2^V$ therefore the complexity to construct the graph is $O(2^V \times (1+V))$. The computational complexity of the value iteration procedure is $O(|V \times 2^V||2|^V)$ and a space complexity of $O(|2^V|)$. For policy Iteration the computation complexity is $O(|2^V| + |V \times 2^V||(2^{2\times V}|)$ and the space is $O(|2^V|)$.

B. Computation of BT

Given the mission set $\mathcal{M}$ and the safety violations set $\mathcal{S}$ Algorithm 4 computes a number $|\mathcal{M}|$ of BTs $\mathcal{T}_M$ for each contactor configuration $M \in \mathcal{M}$. In each $\mathcal{T}_M$, Algorithm 4 finds a subtree $\mathcal{T}_m$ to safely satisfy each single contactor configuration $m \in M$. Each $\mathcal{T}_m$ is computed in $O(|\mathcal{S}||E|)$ time where $|E|$ is the cardinality of the edges of the graph (number of contactors). For each $\mathcal{T}_M$ we compute at most a number $|E|$ of $\mathcal{T}_m$ (worst case where we have to perform an action on each contactor). Finally Algorithm 4 sorts the children of $\mathcal{T}$. The sorting is done in $O(|\mathcal{M}||E|log(|\mathcal{M}| + |E|)))$ time, as for each $\mathcal{T}_M$ we sort each subtree $\mathcal{T}_m$. times. Hence the proposed approach computes a BT in $O(|\mathcal{M}||\mathcal{S}||E|^2 + |\mathcal{M}||\mathcal{S}||E|log(|\mathcal{M}| + |E|))$ time.

VIII. Examples

A. Experimental Test Fixture

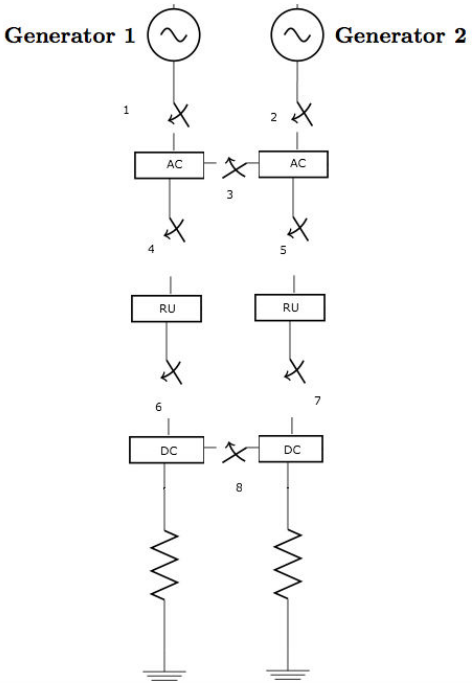


Fig. 8. EPS of Example VIII-A

Consider the EPS in Fig. 8 where the mission requirement is to power the four loads and the safety specification is to not parallel the two generator. The mission set is $\mathcal{M} = \{M_1, M_2, M_3, M_4, M_5\}$ where:

- $M_1 = [1, 2, 4, 5, 6, 7]$
- $M_2 = [1, 3, 5, 6, 7, 8]$
- $M_3 = [2, 3, 4, 6, 8]$
- $M_4 = [1, 2, 4, 6, 8]$
- $M_5 = [1, 2, 5, 7, 8]$

The safety violation set is $\mathcal{S} = S_1, S_2$ where:

- $S_1 = [1, 2, 3]$
- $S_2 = [1, 2, 4, 5, 6, 7, 8]$
- $S_2 = [1, 2, 3, 4, 5, 6, 7, 8]$

The initial contactor configuration is $M_0 = [1, 2, -3, 4, 5, 6, 7, -8]$ that is all the contactors except 3 and 8 are turned off. We choose such initial configuration to show how the framework satisfy the safety conditions

The resulting BT $\mathcal{T}$ is shown in Fig. 9.Each child in $\mathcal{T}$ achieves a contactor configuration in the mission set $\mathcal{M}$. The children are ordered by the number of actions they execute. Some children have to turn off contactors to satisfy the safety requirement. In the nominal case the actions executed are: *Turn* 1 *ON*, *Turn* 5 *ON*, *Turn* 7 *ON*. None of this action violate a safety constraint. Note that if, due to several faults, the $\mathcal{T}$ executes its third child (i.e. it performs the third best plan), the contactor 3 has to be turned off before tuning on the contactor 2 to avoid the

two generators being paralleled. For FSM diagram yield the same results, however it require additional computing power provide by Amazon Web Services and the diagram is too large to display. It contain 256 nodes and 2048 edges. Nodes pertaining to safety violation shown below as contact configuration were removed in order to reduce computational complexity.

11110010	11110110	11110111	11110101	11110100
11111100	11111101	11111111	11111110	11111010
11111011	11111001	11111000	11101000	11101001
11101011	11101010	11101110	11101111	11101101
11101100	11100100	11100101	11100111	11100110
11100010	11100011	11100001	11100000	11110000

11110001 11011111

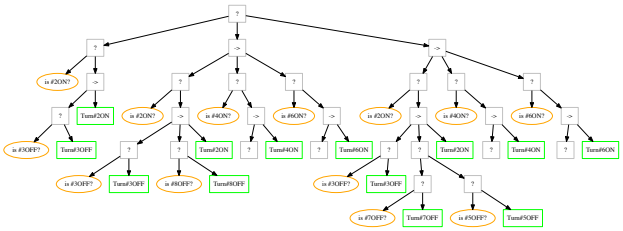


Fig. 10. BT of Example VIII-B. Second Mission

B. Large-Scale Electrical Power System

In this example we consider an electrical power system for commercial applications. The electrical power system is depicted in figure 11. For this example the objective was to power the component *LVDC Bus 3*. The safety specification is to not parallel any two generators. The initial contactor configuration is such that only the contactors 1, 6 and 26 are on. We chose to not enumerate the sets $\mathcal{M}$ and $\mathcal{S}$ due to magnitude of the cardinality of the event space to search, $|X| = 10^{13}$.

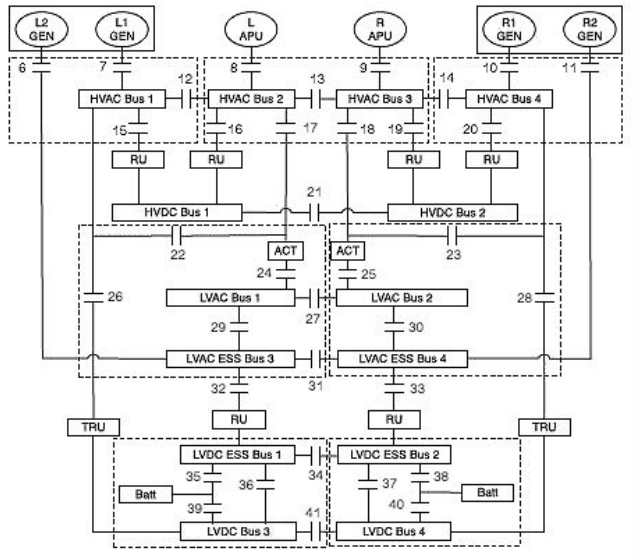


Fig. 11. Electrical Power System Schematic for Example VIII-B

The resulting $\mathcal{T}$ has children equal to $|\mathcal{M}|$ children. We chose to depict only the most left one (nominal path) in Fig. 12. Intuitively to power *LVDC Bus 3*, the contactors 1, 27, and 31 must be turned on. Since at the initial condition the contactors 11, and 31 are on, turning 6 on will yield to a unsafe contactor configuration (Generator L2 and Generator R2 in parallel). To avoid such configuration, either the contactor 6 or 26 have to be turned off before turning on the contactor 6. The FSM formulation was found intractable for this example.

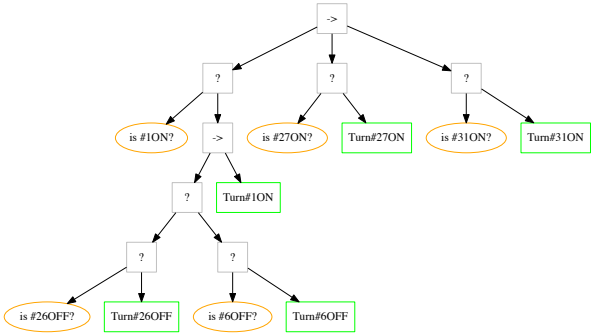


Fig. 12. First child of BT of Example VIII-B.

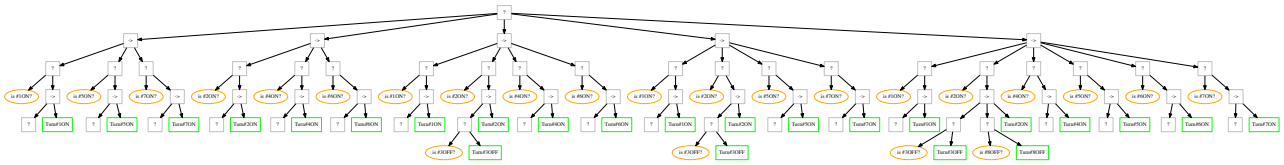


Fig. 9. $\mathcal{T}$ of Example VIII-A. The precondition that are satisfied on the initial state are not shown for space limitation (fallback nodes without children).

IX. Conclusion and Future Direction

In this paper we presented a methodology to synthesize fault tolerant control protocols apropos aircraft EPS modeled as discrete event systems. The electrical power systems were modeled as FSMs and BTs. The algorithm used dynamic programming techniques on graphical abstractions of the systems to synthesize a switching protocol. The results demonstrated, for small scale systems, that the algorithm was effective with a FSM diagram model. However, unless additional modeling techniques were implemented to compress the representation of the system's event space, the problem became intractable. BTs provide a tractable formulation for the large-scale electrical power systems. The proposed approach computed a minimal switching protocol for an electrical power system whose switching is defined by Gray code convention. Future work will seek to improve the algorithm and formulation presented in this paper and use the results to address other problems related to heuristic decision-making for electrical powers systems such as sensor placement, fault detect and isolation, and synthesis of built-in-test.

Appendix

Proposition 1: Algorithm 3 finds a BT in finite time.

Proof: The sets $\mathcal{S}$ and $\tilde{S}$ are finite. Hence the loops inside Algorithm 3 executes a finite number of operations ■

Proposition 2: Algorithm 4 finds an optimal BT in finite time. *Proof:* The set $\mathcal{M}$ is finite set. Hence the number of $\mathcal{T}_M$ computed is finite. The $\mathcal{T}_M$ is updated until it return success. To prove that it will return success in finite time we need to prove that an action required to satisfy the safety specification do not conflict with an action required to perform the mission. Algorithm 4 computes the actions required to satisfy the safety specification using Algorithm 3. Algorithm 3 consider only those contactors that are not listed in the mission specification (Line 4) Hence no confliction action are performed. The optimality is ensured by the sorting algorithm. ■

Acknowledgments

The authors would like to thank Scott Livingston, Ivan Papusha, and the anonymous reviewers for helpful comments. This work was supported in part by IBM and UTC via the iCyPhy consortium

References

[1] N. Ozay, U. Topcu, R. M. Murray, and T. Wongpiromsarn, "Distributed synthesis of control protocols for smart camera networks," in *Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. IEEE Computer Society, 2011, pp. 45–54.

[2] H. Xu, U. Topcu, and R. M. Murray, "A case study on reactive protocols for aircraft electric power distribution," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 1124–1129.

[3] R. Rogersten, H. Xu, N. Ozay, U. Topcu, and R. M. Murray, "An aircraft electric power testbed for validating automatically synthesized reactive control protocols," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 2013, pp. 89–94.

[4] R. G. Michalko, "Electrical starting, generation, conversion and distribution system architecture for a more electric vehicle," Oct. 21 2008, uS Patent 7,439,634.

[5] J. Hopcroft, *Introduction to automata theory, languages, and computation*. Boston: Pearson/Addison Wesley, 2007.

[6] R. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 293–318, 1992.

[7] D. Isla, "Handling Complexity in the Halo 2 AI," in *Game Developers Conference*, 2005.

[8] I. Millington and J. Funge, *Artificial Intelligence for Games*. Taylor & Francis, 2009. [Online]. Available: https://books.google.com/books?id=1OJ8EhvuPXAC

[9] S. Rabin, *Game AI Pro: Collected Wisdom of Game AI Professionals*. Natick, MA, USA: A. K. Peters, Ltd., 2013.

[10] R. d. P. Pereira and P. M. Engel, "A framework for constrained and adaptive behavior-based agents," *arXiv preprint arXiv:1506.02312*, 2015.

[11] K. R. Guerin, C. Lea, C. Paxton, and G. D. Hager, "A framework for end-user instruction of a robot assistant for manufacturing," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 6167–6174.

[12] M. Colledanchise, R. Parasuraman, and P. Ögren, "Learning of behavior trees for autonomous agents," *arXiv preprint arXiv:1504.05811*, 2015.

[13] A. Klöckner, "Behavior trees for uav mission management," in *INFORMATIK 2013: Informatik angepasst an Mensch, Organisation und Umwelt*. Köllen Druck + Verlag GmbH, Bonn, 2013, pp. 57–68.

[14] D. Hu, Y. Gong, B. Hannaford, and E. J. Seibel, "Semi-autonomous simulated brain tumor ablation with ravenii surgical robot using behavior tree," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 3868–3875.

[15] A. Klökner, "Interfacing Behavior Trees with the World Using Description Logic," in *AIAA conference on Guidance, Navigation and Control, Boston*, 2013.

[16] M. Colledanchise and P. Ogren, "How Behavior Trees Modularize Robustness and Safety in Hybrid Systems," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 1482–1488.

[17] P. Ögren, "Increasing Modularity of UAV Control Systems using Computer Game Behavior Trees," in *AIAA Guidance, Navigation and Control Conference, Minneapolis, MN*, 2012.

[18] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[19] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1. IEEE, 1995, pp. 560–564.