Chapter 5

Probabilistic Safety Analysis of Sensor-Driven Hybrid Automata

5.1 Introduction

The verification of control programs for autonomous robotic systems would be incomplete if the effect of state estimation uncertainty was not discussed. Autonomous systems see themselves and their environment through their sensors and estimators; all the state variables that drive the branching of the control program are, in fact, estimations of the true states. Because the control system is designed and verified for the actual states, an analysis of what may happen when the estimators are wrong is an important verification tool.

This chapter deals with the analysis of linear hybrid systems that have invariants and transition conditions that depend on the states of several uncontrollable and passively constrained state variables. Each uncontrollable state variable, which describes the environment or the health of the system (or anything that the system does not actively control), can reach some countable number of discrete states (or discrete sets of states) and the transitions between these states can be modeled as a stationary Markov process. These state variables' states are estimated by the system and these estimates of state are used by the linear hybrid automaton (LHA) to drive the discrete switching between locations. For a given LHA, there may be some combination of states and locations that are unsafe. In the perfect knowledge case (the estimator is always correct) when the LHA has been verified by a model checker, these unsafe conditions will never be met. However, when the accuracy of the estimator is not perfect, there exists some probability of reaching the unsafe condition. This chapter describes a method to calculate this failure probability for different types of LHA control systems.

It must be stated that the addition of uncertainty due to the estimation of the state variables driving the discrete transitions in the hybrid system adds some stochasticity to the problem. However, the problem is not effectively described as a stochastic hybrid system; as described later, it is better to think of the problem as an LHA with deterministic transitions and uncertainty. Stochastic hybrid systems include uncertainty in the transitions of the hybrid automaton as probabilistic transition conditions. Many methods to verify stochastic hybrid systems exist, some of which are referenced in Chapter 1. The method described here somewhat abstracts the transitions of the LHA, reducing the complexity of the problem dramatically.

The methods described in this chapter allow the analysis of the safety of a control program against the given unsafe set when the estimation of important state variables is not perfect. If these states were known exactly, a traditional hybrid system verification would be a sufficient test of the safety of this system. However, a full analysis of this system must include the calculation of the failure probability due to the estimation uncertainty. Section 5.2 sets up the failure probability calculation problem while Section 5.3 describes the steps of the calculation. Variations on this calculation procedure are discussed in Section 5.4. Because this method is very sensitive to problem complexity, reduction techniques are described in Section 5.5, followed by a discussion on methods to find an approximation of the failure probability in Section 5.6. Section 5.7 summarizes the contributions of this chapter.

5.2 Problem Definition

5.2.1 Automata Specification and Models

The architecture of the LHA control system involves a string of high-level completion tasks, such as driving a robot to a series of waypoints, that are executed in parallel with several minor tasks, such as maintaining the temperature of an instrument, limiting the robot's overall speed, or monitoring a sensor's health value. Thus, the locations in the hybrid automaton, v_i, can be sorted into disjoint groups, $V_1, V_2, ..., V_K$, based on which of the K high-level completion tasks that the location is trying to achieve. Then, each of the locations in a group V_k describes one method or tactic to complete the kth high-level task and all subtasks, and these tactics are chosen based on the states of the environment or uncontrollable states of the autonomous system that may affect the completion of the task. For example, the high-level task may be to drive the robot to a waypoint. One of the concurrent low-level tasks may be to maintain the temperature of the wheel motors above a

certain level through the use of two redundant strings of heaters. So, there are at least two tactics to accomplish the set of tasks; the first uses the primary set of heaters and the second uses the back-up heaters. The transition between these two locations is driven by the health of the primary string of heaters.

The flow of the linear hybrid automaton is

$$\psi_{i_f}(t_f)\tau_{i_f i_{f-1}}...\psi_{i_2}(t_2)\tau_{i_2 i_1}\psi_{i_1}(t_1)X_0 \tag{5.1}$$

where X_0 is the set of initial conditions on the controlled state variables, $\psi_{i_n}(t_n)$ is the flow associated with location v_{i_n} for t_n time steps, and $\tau_{i_n i_{n-1}}$ is the transition from location v_{i_n} to $v_{i_{n-1}}$. The flow of a location is based on the high-level and minor tasks that the location is trying to achieve. These dynamical equations are the continuous control actions that a particular tactic use in the completion of the appropriate task. Discrete control actions in the form of resets are grouped with the entry transition for a given location.

There are two types of transitions between the locations of an LHA with this group structure. First, transitions from a location, $v_i \in V_k$, to a location in the following group, $v_j \in V_{k+1}$, are called completion transitions, $\tau^c_{ji,k}$. The transition conditions in this case are based both on the state of the uncontrollable state variables of the system and on the completion of the high-level task that defines the group, V_k. The second type of transitions are transitions between locations in the same group, $v_i, v_j \in V_k$, and these are called failure transitions, $\tau^f_{ji,k}$. These transition conditions are between different tactics achieving the same high-level task and are based solely on the states of the uncontrollable state variables. Sometimes, the state of the system becomes such that there is no way to safely continue achieving a task; in that case, the automaton can transition from a location v_i to a special location called Safing and these failure transitions are labeled $\tau^f_{Si,k}$. All failure transition conditions must be entirely state-based; they cannot be based on the order of tactics attempted except in special circumstances described later. This restriction is not a serious one; in general, completely deterministic state-based transitions are a characteristic of more robust control programs. First, the definition of the complete system state of the uncertain state variables is given. Each of the uncontrollable state variables that is involved in failure transitions in the kth group of locations is called an uncertain state variable, $\chi \in \mathcal{U}_k$, where $\mathcal{U}_k$ is the set of all uncertain state variables in V_k.

Definition 5.2.1. A *complete system state*, s_j, is both the estimated and actual state values of each

uncertain state variable $\chi_i \in \mathcal{U}_k$ at a point in time in a possible execution of the LHA control system. The set of all possible complete system states is S. Each complete system state has two functions associated with it.

1. est$(s_j, \chi_i) \in \Lambda_i$ returns the estimated value of χ_i in s_j.

2. act$(s_j, \chi_i) \in \Lambda_i$ returns the actual value of χ_i in s_j.

Definition 5.2.2. For a hybrid automaton with *completely deterministic state-based transitions*, all states in a location's initial set, $s_i^\xi \in$ init(v_j), must satisfy the location's invariant, est$(s_i^\xi) \models$ inv(v_j). No transition $\tau_{jl,k}$ originating from the location can be satisfied by any states satisfying the location's invariant,

$$\text{est}(s_i^\xi) \models \text{inv}(v_j) \Rightarrow \text{est}(s_i^\xi) \not\models \tau_{jl,k},$$

for all $v_l \in V$, $v_l \neq v_j$. Also, for any location $v_j \in V_k$, there is only one transition $\tau_{jl,k}$ such that est$(s_i^\xi) \models \tau_{jl,k}$ for all $s_i^\xi \in \Xi_k$.

It is assumed that certain statistical information is known about the system. For each of the uncertain state variables, there must be a way to model the propagation of the state variable as a stationary Markov process; since these state variables are not controlled, oftentimes this approximation is a good one. Each state variable $\chi_i \in \mathcal{U}_k$ has a set $\Lambda_i = \{\lambda_1^i, \lambda_2^i, ..., \lambda_{n_i}^i\}$ of discrete state values (or discrete sets of state values) that can achieved by χ_i. The actual value of the state variable χ_i can be accessed by using its associated val$(\chi_i) \in \Lambda_i$ function. The probability of these state transitions, ρ_i are given by the following equation:

$$\rho_i(l, j) = P(\text{val}(\chi_i)[\kappa] = \lambda_j^i | \text{val}(\chi_i)[\kappa - 1] = \lambda_l^i), \tag{5.2}$$

for all $\chi_i \in \mathcal{U}_k$ and for all $\lambda_j^i, \lambda_l^i \in \Lambda_i$. The stationary probabilities give the probability that the state variable has a certain value in the steady state model, and these probabilities, α_i, are

$$\alpha_i(j) = P(\text{val}(\chi_i) = \lambda_j^i), \tag{5.3}$$

for all $\chi_i \in \mathcal{U}_k$ and for all $\lambda_j^i \in \Lambda_i$.

For each uncertain state variable, $\chi_i \in \mathcal{U}_k$, there exists an estimator for that state variable that has some non-zero amount of uncertainty. This uncertainty can be stated as a probability of

correctness of the estimated value. Furthermore, this probability can be broken up for each state value $\lambda_j^i \in \Lambda_i$ into the probability that if the value of the actual state, val(χ_i) is λ_l^i, then the value of the estimated state, val($\hat{\chi}_i$) is λ_j^i, for all $\lambda_j^i, \lambda_l^i \in \Lambda_i$. This probability, ρ_i^e, is

$$\rho_i^e(l, j) = P(\text{val}(\hat{\chi}_i)[\kappa] = \lambda_j^i | \text{val}(\chi_i)[\kappa] = \lambda_l^i). \tag{5.4}$$

The problems of measuring the uncertainty of an estimator and finding the probability of which state is estimated given an actual state are very real in practice. Methods such as simulation and testing can estimate these values; since the final failure probability should be used as a relative value, these probability values do not have to be exact. Much can be learned about the system by using these values as parameters that can be varied in the failure probability calculation.

5.2.2 Unsafe System States

When a LHA executes, that execution follows some path through the different locations based on, in this case, only the states of the system and time. For the automata described above, within a group the location that is executing is chosen based on the estimated system state alone; time only affects transitions out of groups. The execution path within a group, then, consists of a list of the estimated system state values and their associated locations at each time point; the length of the path is related to the completion time of the group, which will be defined in Section 5.2.3.

The conditions (states and locations) that should never occur in conjunction are called the unsafe set.

Definition 5.2.3. The *unsafe set* is a set of conditions $Z = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ that should never be reached in an execution of the LHA. Each condition ζ_j contains a set of constraints on the uncontrollable state variables' values and a set of locations loc(ζ_j) $\subset V$ in which the set of constraints should not hold. This information can be accessed using the following functions:

1. plc(ζ_j, χ_i) $\subseteq \Lambda_i$ returns a set of discrete values that the state variable, χ_i, is constrained to be, val(χ_i) $\in$ plc(ζ_j, χ_i), to satisfy the unsafe condition. If χ_i does not affect this unsafe condition, plc(ζ_j, χ_i) $= \Lambda_i$.

2. loc(ζ_j, v_i) returns true if $v_i \in$ loc(ζ_j) and false otherwise.

In a verified system, the unsafe set is unreachable when the estimated system state is accurate; failure can occur, however, when state estimation uncertainty is added. While the execution path for

a group of locations depends only on the estimated system state at each time point, determining if an execution path reaches the unsafe set requires both the estimated and actual system states. The estimated system state still determines which location will be executing at a given time, and the combination of the executing location and the actual system state dictate entrance into the unsafe set. The lemma that follows states exactly this.

Definition 5.2.4. Each location $v_n \in V_k$ has a function associated with it that returns the state values that each uncertain state variable, $\chi_i \in \mathcal{U}_k$, must take in order for that location to be executed, $ucons(v_n, \chi_i) \subseteq \Lambda_i$. This is the location's passive invariant in the state-based transitions case. If a state variable χ_i is unconstrained in a location v_n, $\text{ucons}(v_n, \chi_i) = \Lambda_i$.

Lemma 5.2.5. *Let $\Omega_k \subset S$ be the set of complete system states that drive the automaton execution from group V_k into the unsafe set Z. For a complete system state $s^\omega \in \mathcal{S}$, $s^\omega \in \Omega_k$ if and only if there exists $\zeta_j \in Z$ and $v_n \in V_k$ such that*

$$\left(\bigwedge_{\chi_i \in \mathcal{U}_k} act(s^\omega, \chi_i) \in plc(\zeta_j, \chi_i) \right) \wedge \left(\bigwedge_{\chi_i \in \mathcal{U}_k} est(s^\omega, \chi_i) \in ucons(v_n, \chi_i) \right) \wedge loc(\zeta_j, v_n) \quad (5.5)$$

is true.

Proof. Let $s^\omega \in \Omega_k$ but assume there is no v_n that satisfies (5.5). By the definition of the unsafe set, there must exist some $\zeta_j \in Z$ such that

$$\bigwedge_{\chi_i \in \mathcal{U}_k} \text{act}(s^\omega, \chi_i) \in \text{plc}(\zeta_j, \chi_i)$$

is true, since entrance into the unsafe set is always driven by the actual system state. Since the unsafe set specifies the total system state, including the location in the automaton, there must exist some v_n such that $\text{loc}(\zeta_j, v_n)$ is true. To enter location v_n and thus the unsafe set from complete system state s^ω,

$$\bigwedge \text{est}(s^\omega, \chi_i) \in \text{ucons}(v_n, \chi_i)$$

must be true since the transitions in the automaton are state driven by definition and because the estimated state drives the transitions in the hybrid automaton execution. Therefore, v_n does satisfy (5.5). The other direction of the proof is obvious by the definition of the unsafe set. □

Complete system states that cause the automaton to transition from a location $v_i \in V_k$ to Safing

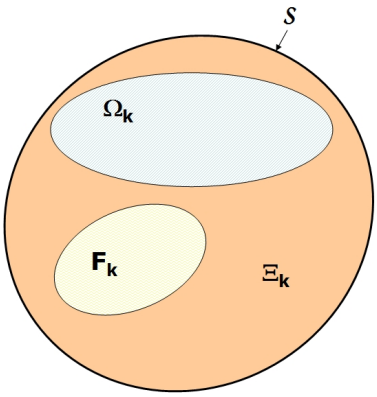

Figure 5.1: A representation of the classifications of complete system states. The nominal set of states is Ξ_k, which is $S_k \setminus (\Omega_k \cup F_k)$, where F_k is the set of Safing states and Ω_k is the set of unsafe states.

are elements of the safing set for group V_k, $s^f \in F_k$. Complete system states that allow the execution of the group to occur normally are elements of the nominal set, $s^\xi \in \Xi_k$. For each group of locations V_k,

$$S = \Xi_k \cup F_k \cup \Omega_k, \tag{5.6}$$

where S is the set of all complete system states, and the sets Ξ_k, F_k, and Ω_k are disjoint. Figure 5.1 illustrates this.

5.2.3 Failure Path Specification

The completion time of a group V_k depends on the completion task that defines the group. The type of completion time (uniform or non-uniform) depends on the presence of rate limiting tasks that affect the completion task in the group.

Definition 5.2.6. A *nominal execution path* of a group V_k is a path $s_1^\xi s_2^\xi ... s_r^\xi \in N_k$ in which only nominal complete system states are visited before the group is exited and execution continues in group V_{k+1}. The set of all nominal execution paths in group V_k is N_k.

Definition 5.2.7. Given the set of nominal execution paths N_k for group V_k, the *completion time*, c_k, is the minimum length of nominal execution path,

$$c_k = \min_{\nu \in N_k} \text{length}(\nu). \tag{5.7}$$

The completion time of a group is the time it takes to achieve the group's completion task at the fastest constrained rate.

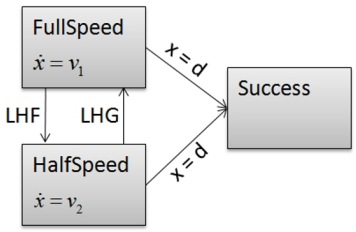

Figure 5.2: Hybrid control system for speed limit example

Definition 5.2.8. In a *uniform completion* group, V_k,

$$c_k = \min_{\nu \in N_k} \text{length}(\nu) = \max_{\nu \in N_k} \text{length}(\nu). \tag{5.8}$$

The uniform completion case holds in groups that have only one rate of completion of the task; in this case, all tactics contribute the same amount towards the completion of the task. Another way to define the uniform completion case is that the contribution values of each location in the group are the same.

Definition 5.2.9. The *contribution value* of a location $v_i \in V_k$, $\text{cval}(v_i) \in \mathbb{R}$, is the normalized contribution towards the achievement of the completion task in V_k that location v_i gives each time step. In uniform completion groups, for each $v_i \in V_k$, $\text{cval}(v_i) = 1$.

Definition 5.2.10. A *non-uniform completion* group is one in which

$$\min_{\nu \in N_k} \text{length}(\nu) \neq \max_{\nu \in N_k} \text{length}(\nu). \tag{5.9}$$

In the non-uniform completion, for each location $v_i \in V_k$, $\text{cval}(v_i) \leq 1$. In this case, the group would have more than one rate that constrains the achievement of the completion task. An example of non-uniform completion would be a rover that is assigned to reach a certain waypoint, but whose maximum velocity is related to the laser sensor's health value. One time step in a location would drive the rover a distance that is different than the distance achieved in a different location that constrains the rover to a different maximum velocity. Figure 5.2 shows the simple hybrid system for this example; assuming that $v_1 = 2v_2$, the contribution value of the first location would be 1 and the contribution of the second location would be $\frac{1}{2}$.

The definition of failure path is now given.

Definition 5.2.11. A *failure path* in group V_k is a sequence of nominal complete system states, s_i^{ξ},

$i = 0, ..., n$, followed by an unsafe system state, s^{ω}. The number of nominal complete system states is $n = 0, ..., r-1$, where $r = c_k$ in the uniform completion case and depends on the completion time and the contribution values of the locations visited along the path for the non-uniform completion case.

From Definition 5.2.2 of completely deterministic state-based transitions, it is clear that there can be a function cloc$(s^{\xi}, k) \in V_k$ that returns the location associated with the nominal complete system state s^{ξ} in group V_k.

Lemma 5.2.12. *For every failure path $\pi = s_1^{\xi} s_2^{\xi} ... s_{r-1}^{\xi} s^{\omega} \in \Pi_k$, where Π_k is the set of all failure paths in group V_k,*

$$\sum_{i=1}^{r-1} cval(cloc(s_i^{\xi}, k)) < c_k. \tag{5.10}$$

Proof. The proof of this lemma is simple; if the sum of the contribution values of the nominal states visited in a failure path equals or exceeds c_k, the execution continues into the next group by the definition of completion time. In order for the path to lead to the unsafe set within a group, the sum of the contribution values of the nominal states must ensure that the path is fully contained within the group; entrance into the unsafe set is always the last state transition in a failure path. □

Each failure path has some probability of occurring during an execution of that group of the hybrid automaton. This is called the failure path probability, and the sum of these over all possible failure paths is the failure probability of a group. Details of these calculations will be described in the next section.

5.3 Probability Calculations

The failure probability is calculated from the sum of all the failure path probabilities in a group; the paths in a group depend on the completion time, and the procedure for finding all the failure paths in a group depends on whether it is a uniform or non-uniform completion group. The failure paths for the uniform completion case are easy to find; the procedure for the non-uniform completion case is more involved. Both will be described in this section.

First, a combined Markov process-like transition probability matrix and a set of initial probabilities are calculated for each complete system state. The stationary probabilities of the individual Markov chains are used to calculate initial condition probabilities of each complete system state in

a group. By having this information, the separate group failure probabilities may be combined into a failure probability for the entire hybrid automaton, as will be shown later.

For a given complete system state, the initial probability, $P(s)$, is given by the following:

$$P(s) = \prod_{\chi_i \in \mathcal{U}_k} P(\text{val}(\chi_i) = \text{act}(s, \chi_i)) P(\text{val}(\hat{\chi}_i) = \text{est}(s, \chi_i) | \text{val}(\chi_i) = \text{act}(s, \chi_i)). \quad (5.11)$$

The transition probability of going from one complete system state to another, $P(s_l|s_j)$, is given by the following:

$$P(s_l|s_j) = \prod_{\chi_i \in \mathcal{U}_k} P(\text{val}(\chi_i)[\kappa] = \text{act}(s_l, \chi_i) | \text{val}(\chi_i)[\kappa - 1] = \text{act}(s_j, \chi_i)) \times$$
$$P(\text{val}(\hat{\chi}_i)[\kappa] = \text{est}(s_l, \chi_i) | \text{val}(\chi_i)[\kappa] = \text{act}(s_l, \chi_i)). \quad (5.12)$$

5.3.1 Uniform Completion Case

For the uniform completion case, collections of stationary and transition probabilities between nominal and unsafe system states can be created. First, the probability of executing a failure path of length one in group V_k (i.e., starting in the unsafe set) is

$$a_k = \sum_{s_j^\omega \in \Omega_k} P(s_j^\omega). \quad (5.13)$$

Likewise, let W_k be a vector of probabilities whose elements are the initial probabilities of each nominal system state, $s_j^\xi \in \Xi_k$. Thus, the jth element of vector W_k is

$$W_k(j) = P(s_j^\xi). \quad (5.14)$$

Next, transition probability constructs are defined. The matrix of transition probabilities between all nominal complete system states is Q_k, where the probability of a transition between s_i^ξ to s_j^ξ is

$$Q_k(i, j) = P(s_j^\xi | s_i^\xi). \quad (5.15)$$

Since all unsafe complete system states are accepting states, only the transitions into these states from the nominal complete system states are considered. The vector $W_{u,k}$ contains the probabilities

of transitions from each nominal complete system state to every unsafe complete system state,

$$W_{u,k}(j) = \sum_{s_i^\omega \in \Omega_k} P(s_i^\omega | s_j^\xi). \tag{5.16}$$

Proposition 5.3.1. *The failure probability for the uniform completion case in group V_k can be calculated using the following formula, for $c_k \in [2, \infty)$,*

$$W_s(k) = a_k + W_k \cdot (\sum_{i=0}^{c_k-2} Q_k^i) W_{u,k}. \tag{5.17}$$

When $c_k \to \infty$, the equation becomes

$$W_s(k) = a_k + W_k \cdot (I - Q_k)^{-1} W_{u,k}. \tag{5.18}$$

Proof. The failure probability is the sum of all the failure path probabilities; for the uniform completion case and the definitions of a_k, W_k, Q_k, and $W_{u,k}$ given above, Eq. (5.17) sums the path probabilities of all failure paths of length one to length c_k. If a path has length $c_k + 1$, c_k of the path elements must be nominal states; because for the uniform completion case, cval$(s^\xi) = 1$ for all $s^\xi \in \Xi_k$ and by Lemma 5.2.12, a path of length $c_k + 1$ is not possible in group V_k. Therefore, Eq. (5.17) is the sum of all possible failure paths in V_k. Using

$$\sum_{i=0}^{\infty} Q^i = (I - Q)^{-1}, \tag{5.19}$$

one can derive (5.18) from (5.17). □

In the infinite time case, the failure probability approaches $1 - W_f(k)$, where $W_f(k)$ is the probability of entering the Safing location from group V_k.

5.3.2 Non-Uniform Completion Case

In the uniform completion case, the number of failure paths considered was greatly reduced by the creation of the probabilistic transition matrix and vectors. Since all locations contributed the same amount to the completion of the group, the path length did not depend on which individual locations were visited. This is not the case in the non-uniform completion case, where the execution

path length does depend on which locations are executed and in what order. Like the uniform completion case, there is a way to reduce the number of failure paths that must be considered by grouping together locations by contribution values.

Let $B_k = \{b | b = \text{cval}(v_i), \forall v_i \in V_k\}$ be the set of contribution values in a group, where all $b \in B_k$ are unique (for all $b_i, b_j \in B_k, b_i \neq b_j$) and ordered ($B_k = \{b_1, b_2, ..., b_n\}$ such that $b_1 > b_2 > ... > b_n$). Since cval(v_i) is the rate of location $v_i \in V_k$ normalized by the maximum rate in group V_k, $b_1 = 1$. Now, let there be n sets β_i, where

$$\beta_i = \{s_j^\xi | \text{cloc}(s_j^\xi, k) = v_i \wedge \text{cval}(v_i) = b_i, \forall s_j^\xi \in \Xi_k\} \tag{5.20}$$

where each β_i is the set of locations that have a contribution value b_i. Then, failure paths can be created using β_i instead of using the individual nominal system states, s^ξ, where for failure path $\beta_{i_1}\beta_{i_2}...\beta_{i_{r-1}}s_r^\omega$,

$$\sum_{j=1}^{r-1} b_{i_j} < c_k. \tag{5.21}$$

All possible failure paths can be found using a simple algorithm that is based on a breadth-first search. A branch of the search tree is complete when adding any set β_i, $i = 1, ..., n$, to the path, π, would cause the sum of the contribution values of the path elements to equal or exceed the completion time,

$$\sum_{\beta_i \in \pi} b_i \geq c_k - b_n. \tag{5.22}$$

Also, the paths as they stand at each level of the search are added to the set Π of all potential failure paths. The algorithm is outlined below:

1. Initialize the search tree with the initial failure path placeholder, $\beta_\emptyset$. The initial failure path is the failure path with no nominal transitions. The placeholder, $\beta_\emptyset$, has no contribution value and is ignored in any path containing other β sets. Add this path to set Π_1. Set the level counter $l = 1$.

2. For each branch, π_i^l, on level l, compute the sum of the contribution values,

$$\text{cval}(\pi_i^l) = \sum_{\beta_j \in \pi_i^l} b_j. \tag{5.23}$$

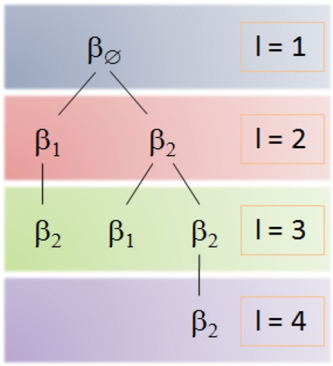

Figure 5.3: The search tree of potential failure paths for the speed limit example

(a) For each $b_j \in B_k$, if $\text{cval}(\pi_i^l) + b_j < c_k$, append β_j to the path, $\pi_i^l + \beta_j \in \Pi_{l+1}$.

(b) If $b_n = 0$, for each path π_i^l whose last set is β_n, β_n may not be added to that path.

3. Increment l.

4. Repeat steps 2 and 3 until $\Pi_l = \emptyset$.

5. Create the set of all potential failure paths,

$$\Pi = \bigcup_{i=1}^{l-1} \Pi_i. \tag{5.24}$$

This algorithm can be demonstrated with the simple speed limit example shown in Figure 5.2. Let $c_k = 2$ and $B = \{1, \frac{1}{2}\}$, where the location with the higher speed limit has a velocity constraint twice the other location's. The breadth-first search tree can be found in Figure 5.3 with the search levels denoted. The final set of potential failure paths is $\Pi = \{\beta_\emptyset, \beta_1, \beta_2, \beta_1\beta_2, \beta_2\beta_1, \beta_2\beta_2, \beta_2\beta_2\beta_2\}$. Each path in Π would need to be followed by a transition into the unsafe set for it to become a failure path; the element $\beta_\emptyset$ would simply be replaced by an initial condition in the unsafe set.

In the case that $b_n = 0$, the failure path algorithm has an exception. Because the execution can remain in the zero rate locations for an infinite number of time steps, once β_n is added to a path, the next set added to the path, β_j, must have a contribution value $b_j > 0$. This step avoids the path creation algorithm from becoming an infinite loop; the infinite number of failure paths is accounted for in the path probability calculation step.

The calculation of failure path probabilities is similar to the method described for the uniform completion case. The scalar value a_k is again the probability of starting in the unsafe set. Instead

of one vector of initial probabilities for the nominal states, W_k, there are n vectors, one for each set β_i, $i = 1, ..., n$. Each vector W_k^i has an initial probability for each nominal complete system state $s_j^\xi \in \beta_i$, calculated in (5.14). The Q_k transition matrix is broken up into several smaller matrices to account for all possible transitions between the β sets. For all β_i, β_j, $i, j = 1, ..., n$, in a group, there exists a matrix $Q_k^{i,j}$ that contains the transition probabilities from each nominal complete system state $s_l^\xi \in \beta_i$ to each $s_m^\xi \in \beta_j$. Finally, the vector of transition probabilities from nominal complete system states to the unsafe set, $W_{u,k}$ is also broken up into n vectors $W_{u,k}^i$, $i = 1, ..., n$, that are associated with the β sets.

The failure path probabilities are calculated using the initial and transition vectors and matrices described above and then all path probabilities are summed to find the group's failure probability. For the speed limit example, there are seven failure paths whose individual path probabilities must be calculated and summed to find the group's failure probability, as follows:

$$W_s(1) = a_1 + W_1^1 \cdot W_{u,1}^1 + W_1^2 \cdot W_{u,1}^2 + W_1^1 \cdot (Q_1^{1,2} W_{u,1}^2) + W_1^2 \cdot (Q_1^{2,1} W_{u,1}^1) + W_1^2 \cdot (Q_1^{2,2} W_{u,1}^2) + W_1^2 \cdot (Q_1^{2,2} Q_1^{2,2} W_{u,1}^2). \quad (5.25)$$

In the case that $b_n = 0$, each time a transition into β_n is encountered in a failure path, it is replaced by an infinite series of paths with increasing numbers of transitions into that set. Whereas a single transition into set β_j, $b_j > 0$ from set β_i is generally indicated in the failure path probability calculation by $Q_k^{i,j}$, a transition from β_i to β_n, $b_n = 0$ would be represented by

$$Q_k^{i,n} \left(\sum_{x=0}^{\infty} \left(Q_k^{n,n} \right)^x \right). \quad (5.26)$$

By (5.19), this becomes $Q_k^{i,n} \left(I - Q_k^{n,n} \right)^{-1}$. Likewise, if β_n is the first set of locations visited, its probability value is represented by $W_k^n \cdot \left(I - Q_k^{n,n} \right)^{-1}$.

5.3.3 System Failure Probability

The overall hybrid automata failure probability can be calculated by summing the probabilities of all the failure paths through the automata. To do this, the probability of each group reaching the Safing location, $W_f(k)$, must be calculated. The procedure for doing this is the same as for finding the failure probability, however, $s_i^f \in F_k$ are used as the accepting states instead of $s_i^\omega \in \Omega_k$. The

probability of traversing group V_k nominally is then

$$W_n(k) = 1 - W_s(k) - W_f(k). \tag{5.27}$$

Proposition 5.3.2. *The failure probability of the system of $K > 1$ groups is given by*

$$W_s = W_s(1) + \sum_{i=2}^{K}(\prod_{j=1}^{i-1} W_n(j))W_s(i). \tag{5.28}$$

Proof. The failure probability of the hybrid system is the sum of the failure path probabilities through the system. Since the failure paths can only consist of zero to $K-1$ nominal group transitions followed by a failure, Eq. (5.28) gives all failure paths through the hybrid system. Any entrance into Safing removes the execution from the hybrid system and precludes failure in the future, and so is excluded from the failure probability calculation. □

5.4 Variations on the Failure Probability Problem

5.4.1 Subgroups

Some groups may have two or more disjoint sets of locations; once execution enters one of the sets of locations in group V_k, the execution can only exit that set of locations to go to Safing or the next group, V_{k+1}. These disjoint sets of locations are called subgroups. Each subgroup, $V_{k,h} \subset V_k$, has a set $I_{k,h} \subset \Xi_k$ of complete system states that cause the execution of the automaton to enter the subgroup $V_{k,h}$ upon entering the group V_k. The initial transition into a subgroup allows only a subset of all nominal complete system states, however once in a subgroup, every complete system state continues the execution in that subgroup. Therefore, the W_k probability vector must be broken up into separate, disjoint $W_{k,h}$ vectors for each subgroup. Once the execution enters a subgroup, the unsafe set of complete system states may be different between the subgroups and from the initial set of unsafe complete system states; therefore, each subgroup has its own $F_{k,h}$, $\Xi_{k,h}$, and $\Omega_{k,h}$ for the non-initial transitions within the subgroup. This triggers separate transition probability matrices and nominal to unsafe transition vectors for each subgroup as well.

An example of this subgroup structure can be found in the following example. Suppose that there is a rover that must follow a path to a point, but the path branches and the choice of the final point is based on the rover's system health. Figure 5.4 shows the task and Figure 5.5 gives

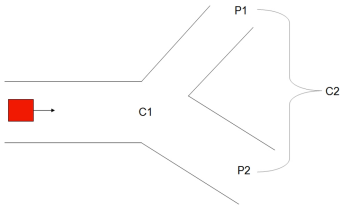

Figure 5.4: Depiction of the path for the simple rover task

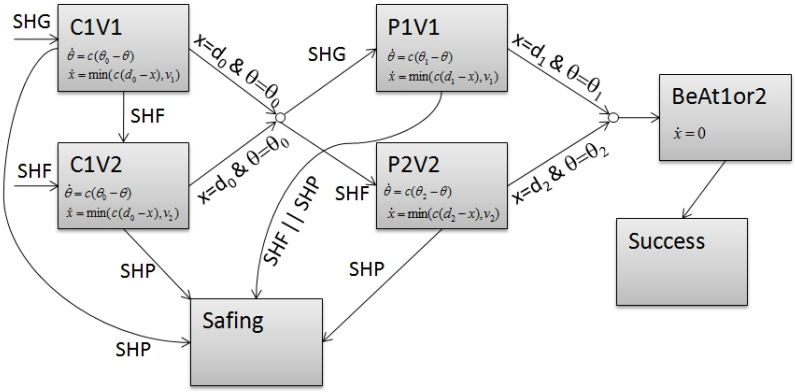

Figure 5.5: Hybrid control system for the simple rover task

the hybrid control system. Once the robot reaches point C1 and makes the choice to go to P1 or P2, this choice cannot be reversed. The set of all complete system states for this task is $S = \{\texttt{GG}, \texttt{GF}, \texttt{GP}, \texttt{FG}, \texttt{FF}, \texttt{FP}, \texttt{PG}, \texttt{PF}, \texttt{PP}\}$ (actual then estimated system health state), and the unsafe set is as follows:

1. $\dot{x} > 0$ and `SystemHealth` is `POOR`, and
2. $\dot{x} > v_2$ and `SystemHealth` is `FAIR`.

The initial unsafe set is $\Omega_2 = \{\texttt{FG}, \texttt{PG}, \texttt{PF}\}$. The initial set that allows transitions into the top location (going to P1) is $I_{2,1} = \{\texttt{GG}\}$; the initial set for the other location (going to P2) is $I_{2,2} = \{\texttt{GF}, \texttt{FF}\}$. The remaining complete system states are initially Safing, $F_2 = \{\texttt{GP}, \texttt{FP}, \texttt{PP}\}$.

Once the execution transitions into the first subgroup, all complete system states can be reached, and the unsafe, safing, and nominal sets for this subgroup are as follows: $\Omega_{2,1} = \{\texttt{FG}, \texttt{PG}\}$, $F_{2,1} = \{\texttt{GF}, \texttt{GP}, \texttt{FF}, \texttt{FP}, \texttt{PF}, \texttt{PP}\}$, and $\Xi_{2,1} = \{\texttt{GG}\}$. For the second subgroup, these sets are $\Omega_{2,2} = \{\texttt{PG}, \texttt{PF}\}$, $F_{2,2} = \{\texttt{GP}, \texttt{FP}, \texttt{PP}\}$, and $\Xi_{2,2} = \{\texttt{GG}, \texttt{GF}, \texttt{FG}, \texttt{FF}\}$.

Though a group may be have non-uniform completion, each subgroup within that group may be uniform or non-uniform. In the example above, group V_2 as a whole is a non-uniform completion group, but each subgroup, $V_{2,1}$ and $V_{2,2}$, is a uniform completion group. Once the initial transition

into a subgroup is taken, the subgroup can be treated like any other group when calculating the failure probability within the subgroup. These subgroup failure probabilities, $W_s(k,h)$, are calculated in the same way as the group failure probability for a connected group except that there is no initial failure probability (a_k) in the failure path probability sum. Then, the overall group failure probability for a group that has H subgroups is the sum of the initial failure probability and all the subgroup failure probabilities,

$$W_s(k) = a_k + \sum_{h=1}^{H} W_s(k,h). \tag{5.29}$$

The safing probability, $W_f(k)$, for a group V_k with subgroups is calculated in the same way; the nominal probability is $W_n(k) = 1 - (W_s(k) + W_f(k))$. These group probabilities can then be used to calculate the system failure probability as described in Eq. (5.28).

5.4.2 Completion Time Uncertainty

It may not be possible to exactly know the completion time for a group. If there is a probability distribution over a finite number of possible completion times, the failure probability for that group can be calculated by finding the failure probability for each possible completion time. The total failure probability for the group is the sum of the failure probabilities for each possible completion time multiplied by the completion time probability. This procedure works for both the uniform and non-uniform completion cases.

Let H be a hybrid automaton with stochastic differential equations,

$$dx_c = l(v, x_c)dt + \sigma(v, x_c)dw \tag{5.30}$$

that dictate the controllable state evolution for the completion task in each location. The drift, $l : V \times X_c \rightarrow \mathbb{R}$ is the linear flow condition for the continuous completion state variable and $\sigma : V \times X_c \rightarrow \mathbb{R}^{1\times p}$ is a dispersion vector for the $\mathbb{R}^p$-valued Wiener process $w(t)$. It is assumed that for any $v_i, v_j \in V_k$, $\sigma(v_i, x_c) = \sigma(v_j, x_c)$. For each group, V_k, let $L_k = \{l(v, x_c)|\forall v \in V_k\}$, the set of all completion rates in a group. Let $l_k = \min_{l_i \in L_k} l_i$; the completion time,

$$c_k = \frac{|\text{inv}(v, x_c)|}{l_k}, \tag{5.31}$$

is the invariant distance that the completion state variable must travel in order for the execution to move onto the next group. By definition, for all $v_i, v_j \in V_k$, inv$(v_i, x_c) =$ inv(v_j, x_c). However, with the addition of the Wiener process, $w(t)$, the actual completion rate of the location, $\tilde{l}(v, x_c)$, now is a normally-distributed random variable with mean $l(v, x_c)$. Since the invariant is strictly deterministic in this formulation, the completion time for the group, $\tilde{c}_k$, is also a normally-distributed random variable with mean c_k.

It is possible to approximate the failure probability of a group with this completion time uncertainty to varying degrees of accuracy. For uniform completion groups, the failure probability can be computed for a range of completion times centered about c_k, $C_k^n = \{c^i | c^i \in \mathbb{Z}^+ \wedge c^i \in [c_k - n\sigma, c_k + n\sigma]\}$. Each potential completion time $c^i \in C_k^n$ has an associated adjusted probability of occurring,

$$p(c^i) = \int_{c^i-1}^{c^i} \rho(y)dy - \rho(c^i - 1) \tag{5.32}$$

where ρ is the probability density function of $\tilde{c}_k$. Each potential completion time also has an associated group failure probability, $\tilde{W}_s(k, c^i)$, which is calculated according to equation (5.17).

With this information, one can calculate the estimated group failure probability,

$$\hat{W}_s^n(k) = \sum_{c^i \in C_k^n} p(c^i)\tilde{W}_s(k, c^i). \tag{5.33}$$

This is only a lower bound on the true group failure probability; as $n \to \infty$, $\hat{W}_s^n(k) \to W_s(k)$. However, the lower end of the completion time distribution can be overestimated by taking

$$p(c_{min}^i) = \int_{-\infty}^{c^i} \rho(y)dy \tag{5.34}$$

where

$$c_{min}^i = \min_{c^i \in C_k^n} c^i.$$

The failure probability $\tilde{W}_s(k, c^i)$ increases as c^i increases, so a similar overapproximation is not possible with c_{max}^i.

For the non-uniform completion case, the same process as described for the uniform completion case can be used, however, this assumes that the uncertainty in the different contribution values does not affect the failure paths. For contribution values that are well separated, for example, $\frac{1}{2}$ and $\frac{1}{3}$, this assumption is good for sufficiently small σ. However, as the differences between the contribution

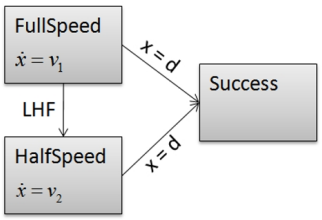

Figure 5.6: Hybrid automaton for the missing state transition example

values get smaller or the uncertainty gets larger, this assumption may no longer apply. In that case, the estimated failure probability will still be a lower bound on the actual failure probability, but increasing n will not make the estimated failure probability converge to the actual failure probability.

5.4.3 Missing State Transitions

The failure probability procedure works for LHA that have discrete transitions based solely on the state of the system. It can be adapted for certain small problems that may have transitions that are also based on the order in which the locations are visited; an example of this is shown in Figure 5.6. This is a similar velocity-controlled rover driving task to the one shown in Figure 5.2, however, once the execution enters into the HalfSpeed location, it must continue there until the task is completed. This extra restriction causes there to be no transition from the HalfSpeed location to the FullSpeed location upon the `LaserHealth` becoming `GOOD`; thus, this is called a missing state transition case.

The complication that arises in missing state transition cases is that assigning each complete system state to just one location becomes impossible. In the example introduced in Figure 5.6, the set of complete system states can be represented by $S = \{\texttt{GG}, \texttt{GF}, \texttt{FG}, \texttt{FF}\}$. While the complete system states with an estimated state of `FAIR` can only occur in the HalfSpeed location, due to the missing transition, the complete system states with estimated values that are `GOOD` are possible in both locations (though they are only possible in the FullSpeed location initially). In order to accommodate this, copies of the ambiguous complete system states must be added to S. If $\text{cloc}(s^\xi, k) = \{v_j | \bigwedge \text{est}(s^\xi, \chi_i) \in \text{ucons}(v_j, \chi_i)\}$ has $n > 1$ members, then each $s^\xi \in S$ must be replaced by n copies identified by location, $\tilde{s}^\xi = \{(s^\xi)^{v_j} | v_j \in \text{cloc}(s^\xi, k)\}$. Then, the adjusted state space becomes $\tilde{S} = S \cup \tilde{s}^\xi$ for all adjusted s^ξ.

In addition to augmenting S, the initial and transition probability vectors and matrices must also

be modified. Let s_i^ξ have an augmented set $\tilde{s}_i^\xi$ and let $v_j \in \text{cloc}(s_i^\xi, k)$ be the location such that $\text{est}(s_i^\xi) \in \text{init}(v_j)$. Then, the initial probability for each $(s_i^\xi)^{v_l} \in \tilde{s}_i^\xi$ is

$$P((s_i^\xi)^{v_l}) = \begin{cases} P(s_i^\xi) & l = j \\ 0 & l \neq j. \end{cases} \tag{5.35}$$

Likewise, the conditional probabilities must be adjusted. Let $e_{jl} \in E$ exist if and only if there exists a valid transition from location v_j to v_l.

Lemma 5.4.1. *For all $v_j, v_l \in cloc(s_i^\xi, k)$, $v_j \neq v_l$, $P((s_i^\xi)^{v_l} | (s_i^\xi)^{v_j}) = 0$.*

Proof. Assume that $\text{est}(s_i^\xi) \models \tau_{jl,k}$ and $v_j \neq v_l$. Since both $v_j, v_l \in \text{cloc}(s_i^\xi, k)$, $s_i^\xi \models \text{inv}(v_j) \wedge s_i^\xi \models \text{inv}(v_l)$. Since the transition scheme is still deterministic and except for the missing transitions, it is state-based, so if the transition does exist, by definition $s_i^\xi \not\models \text{inv}(v_j)$. So, the only location reachable from v_j with a state s_i^ξ is $v_l = v_j$, which negates the starting assumption. □

Lemma 5.4.2. *For each $s_m^\xi \in \tilde{S}$, there exists a unique $(s_i^\xi)^{v_j} \in \tilde{s}_i^\xi$ such that there exists an edge $e_{nj} \in E$ with an associated transition condition $\tau_{nj,k}$ such that est$(s_m^\xi) \models \tau_{nj,k}$, where $v_n \in cloc(s_m^\xi, k)$.*

Proof. If the unadjusted state $s_m^\xi \models \text{inv}(v_j)$, then $v_n = v_j$ and there are no appropriate transition conditions $\tau_{nj,k}$ such that $\text{est}(s_m^\xi) \models \tau_{nj,k}$ by the definition of state-based transitions. If $s_m^\xi \not\models \text{inv}(v_j)$, then there exists some transition edge e_{nj} and condition $\tau_{nj,k}$ that is satisfied by $\text{est}((s_i^\xi)^{v_j})$ and by the definition of state-based transitions, this transition is unique. □

In summary, each complete system state s_i^ξ can transition to only one copy of s_j^ξ with set $\tilde{s}_j^\xi$ because of the overlapping of invariant sets of the locations. In the example, the set $\tilde{S} = \{\texttt{GG}^1, \texttt{GG}^2, \texttt{GF}, \texttt{FG}^1, \texttt{FG}^2, \texttt{FF}\}$. If the original transition matrix for the set $S = \{\texttt{GG}, \texttt{GF}, \texttt{FG}, \texttt{FF}\}$ for the similar example shown in Figure 5.2 looked like this,

$$\mathcal{T} = \begin{bmatrix} 0.5 & 0.05 & 0.05 & 0.4 \\ 0.4 & 0.3 & 0.05 & 0.25 \\ 0.25 & 0.05 & 0.3 & 0.4 \\ 0.4 & 0.05 & 0.05 & 0.5 \end{bmatrix}, \tag{5.36}$$

then the adjusted matrix for set $\tilde{S}$ for the current example looks like this,

$$\tilde{T} = \begin{bmatrix} 0.5 & 0 & 0.05 & 0.05 & 0 & 0.4 \\ 0 & 0.5 & 0.05 & 0 & 0.05 & 0.4 \\ 0 & 0.4 & 0.3 & 0 & 0.05 & 0.25 \\ 0.25 & 0 & 0.05 & 0.3 & 0 & 0.4 \\ 0 & 0.25 & 0.05 & 0 & 0.3 & 0.4 \\ 0 & 0.4 & 0.05 & 0 & 0.05 & 0.5 \end{bmatrix}. \tag{5.37}$$

From this point, the failure probability calculation follows the same procedure for both the uniform and non-uniform completion cases as described previously.

5.5 Problem Complexity and Reduction Techniques

5.5.1 Problem Complexity

The failure probability found using the method described here is exact in the sense that it is not an approximation of or an upper bound on the actual failure probability given the initial information. Although the initial information, like the estimator uncertainty measure and the stationary Markov processes that describe state propagation, are generally approximations, a powerful use for this method is to understand how the failure probability is affected by changes in this initial information.

Unfortunately, the complexity of the failure probability calculation method is exponential in the number of uncertain state variables. Let $y(\chi_i)$ be the number of discrete states in Λ_i for each uncertain state variable $\chi_i \in \mathcal{U}_k$. Then, the number of complete system states is

$$\prod_{\chi_i \in \mathcal{U}_k} y(\chi_i)^2. \tag{5.38}$$

To simplify the problem, let each of the n uncertain state variables, $\chi_i \in \mathcal{U}_k$, $i = 1, ..., n$, have y discrete states in Λ_i. Then, the number of complete system states is y^{2n}. The number of complete system states affects the size of the transition matrices and vectors. The classification of each system state, the calculation of its stationary probability, and the creation of the transition probability matrices and vectors have been automated, however, which allows larger problems to be explored.

Another contributing factor to problem complexity in the non-uniform completion case is the number of distinct contribution values in a group. In general, the number and size of the contribution

values and the completion time affect the number of failure path groups that are possible. The number of failure path groups increases as the number of contribution values and the completion time increase, and decreases as the actual contribution values increase. However, the failure path creation algorithm is very efficient and can handle finding the path groups with little problem. The difficulty then becomes the number of math operations needed to find the failure probability, which is based on the number of complete system states and the number of failure path groups.

5.5.2 Complete System State Reduction Techniques

Because the complexity of the failure probability calculation depends on the number of uncertain state variables and states, techniques to reduce that number are important. One such reduction method is the introduction of derived state variables. A derived state variable is a non-physical state variable whose state propagation completely depends on two or more uncertain state variables. Let $\bar{\chi} \subset \mathcal{U}_k$ be a set of two or more uncertain state variables. Let $\bar{\Lambda}$ be the set of all combinations of discrete states of these state variables,

$$\bar{\Lambda} = \prod_{\chi_i \in \bar{\chi}} \Lambda_i.$$

In some cases, the control of the hybrid automaton may be based on collections of states, $\bar{\lambda}_i \subset \bar{\Lambda}$. If this is the case, a new derived state variable, δ, may be created with discrete sets of states, $\bar{\Lambda}_\delta = \{\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_m\}$, where

$$m < \prod_{\chi_i \in \bar{\chi}} n_i,$$

the number of individual states in $\bar{\Lambda}$. Therefore the contribution to the problem complexity of the derived state variable would be

$$m^2 < \prod_{\chi_i \in \bar{\chi}} n_i^2. \tag{5.39}$$

An example of this is the `SystemHealth` state variable that is modeled on three sensor health state variables (IMU, GPS, and LADAR), each having two discrete states (`GOOD` and `POOR`). The model of the `SystemHealth` state variable, with three states, and its corresponding hybrid control system is shown in Figure 5.7. In this example, $m = 3$ and $\prod n_i = 8$, so the complexity reduction, $9 < 64$ is significant.

Another way to reduce the complexity of the failure probability calculation is to leverage the state models of composite uncertain state variables. Unlike derived state variables, these uncer-

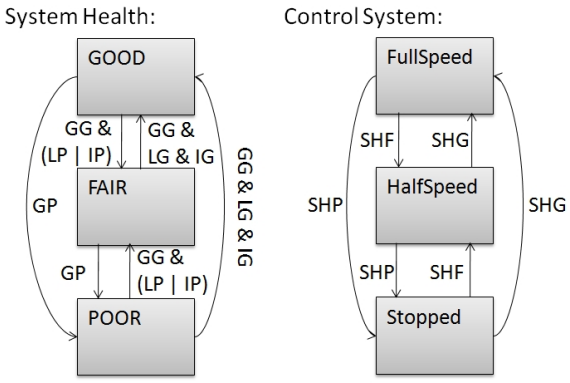

Figure 5.7: Model for derived system health state variable and corresponding hybrid automaton control system

tain state variables are physical and may be adequately described by a stationary Markov process. However, a better model for the state propagation of these state variables may be based on other uncertain state variables. For example, assume that the `LADARHealth` state variable (`LH`) for a robotic system depends on the position of the sun relative to the sensor and the amount of dust on the sensor. This state variable's state propagation can be adequately modeled as a stationary Markov process; however, if the relative position of the sun (`SP`) independently affects the hybrid control system, some reduction in the number of complete system states may be possible. For example, assume that $\texttt{LH} = \chi_1$ has the set of discrete states $\Lambda_1 = \{\texttt{GOOD}^1, \texttt{FAIR}^1, \texttt{POOR}^1\}$ and $\texttt{SP} = \chi_2$ has the set $\Lambda_2 = \{\texttt{DIRECT}^2, \texttt{INDIRECT}^2\}$. Assume also that the model of state propagation of `LH` dictates that if $\text{val}(\chi_2) = \texttt{DIRECT}^2$ then $\text{val}(\chi_1) = \texttt{POOR}^1$. In this case, there can be no complete system state, s, such that

$$(\text{act}(s, \chi_1) \neq \texttt{POOR}^1 \wedge \text{act}(s, \chi_2) = \texttt{DIRECT}^2) \vee (\text{est}(s, \chi_1) \neq \texttt{POOR}^1 \wedge \text{est}(s, \chi_2) = \texttt{DIRECT}^2)$$

is true. This knowledge reduces the total number of complete system states.

5.6 Approximate Methods

5.6.1 Stochastic Hybrid Model Verification

Since methods for verifying certain classes of stochastic hybrid systems exist, it is worth some effort to attempt to construct a suitable stochastic hybrid model for the type of problem solved in this chapter. The hybrid systems treated here assume discrete time execution, so the same assump-

tion will apply to the stochastic hybrid model. The definition for discrete-time switching diffusion processes used in this section is given in Definition 2.3.1.

The hybrid control systems without estimator uncertainty can easily be converted into a type of stochastic hybrid model with probabilistic transitions and deterministic flow equations. The locations, edges, resets, continuous state space and flow equations are one to one between the original hybrid system and the stochastic system; $\mathcal{V}_n = V_o$, $\mathcal{E}_n = E_o$, $X_n = X_o$, and $\phi(X_n, \mathcal{V}_n) = \psi(X_o, V_o)$, where the subscripts stand for "new" and "old," respectively. The transition conditions of the original automaton were based on the discrete states of environment state variables whose state propagation could be modeled by a stationary Markov process. In the conversion, these transition conditions become the transition probabilities between the environment states that satisfy the originating location's invariant and the states that satisfy the accepting location's invariant. For example, let there exist an edge e_{ij} between locations v_i and v_j and let $\Gamma_i = \{s \in S | s \models \text{inv}(v_i)\}$ and $\Gamma_j = \{s \in S | s \models \text{inv}(v_j)\}$. Since the transition condition associated with edge e_{ij} is

$$\tau_{ij} = \bigwedge_{s \in \Gamma_j} s$$

and since for the perfect knowledge case, $\text{act}(s) = \text{est}(s)$, let the transition probability associated with edge e_{ij} be

$$\mu_{ij} = \sum_{s_n \in \Gamma_i} \sum_{s_m \in \Gamma_j} \prod_{\chi_l \in \mathcal{U}_k} P(\text{val}(\chi_l)[\kappa] = \text{act}(s_m, \chi_l) | \text{val}(\chi_l)[\kappa - 1] = \text{act}(s_n, \chi_l)). \tag{5.40}$$

How to add the estimation uncertainty to the stochastic models of the environment variables is not as obvious. Because the hybrid system was verified against the unsafe set in the perfect knowledge case, it is important to distinguish between the actual and estimated states of the system since failure can only occur when these are different. In the uncertain system, the actual and estimated system states have different jobs; the estimated state drives the transitions between the locations of the control system and the actual state in a location can cause the system to reach an unsafe state. In order to differentiate between executing a location nominally and in an unsafe way, a new location, v_u, must be created in each group to account for the unsafe states. So, in the uncertain case, $\mathcal{V}_k = V_k \cup \{v_u\}$.

The transition probabilities between the locations that are not unsafe would also depend on estimation uncertainty in addition to the state propagation probability models. An edge e_{iu} would

be added to each location v_i from which a direct transition into the unsafe set is possible; the transition probability associated with that edge would be the sum of the transition probabilities from each nominal execution state whose estimated state values satisfies the original location's invariant, $s^\xi \models \text{inv}(v_i)$, to each unsafe execution state, s^ω. Let $\Gamma_i^\xi = \{s^\xi \in \Xi_k | \text{ est}(s^\xi) \models \text{inv}(v_i)\}$, then

$$\mu_{ij} = \sum_{s_n^\xi \in \Gamma_i^\xi} \sum_{s_m^\xi \in \Gamma_j^\xi} P(s_m^\xi | s_n^\xi) \tag{5.41}$$

and

$$\mu_{iu} = \sum_{s_n^\xi \in \Gamma_i^\xi} W_{u,k}(n). \tag{5.42}$$

The overall group failure probability would be the probability of reaching the unsafe location. An upper bound of this probability could be found using a variety of existing stochastic hybrid system verification methods.

The problem constructed in this way gives few advantages over the method described previously. The transition probabilities between the locations would need to be calculated for the stochastic hybrid system in much the same way as described previously. Depending on the verification method used, paths through the group may not need to be found explicitly, which may reduce the problem complexity slightly, but the number of complete system states continues to drive the problem complexity. Also, though the number of locations is basically hidden in the previously described failure probability calculation, it is important and even increased slightly in the stochastic hybrid model formulation. Many of the stochastic verification methods available are affected by the number of reachable locations. Finally, most stochastic hybrid system verification methods can only find an approximation of the failure probability, whereas the original method presented here is exact; however, because of the complexity issues with this failure probability calculation, Markov Chain Monte Carlo simulation is a natural next step.

5.6.2 Markov Chain Monte Carlo Simulation

Monte Carlo simulation is a useful way of approximating the failure probability of systems that are too large to reason about using the method described here. Stochastic hybrid systems like those described in the previous section are set up for Monte Carlo simulation; however, getting a system into that form may take more time and/or memory than one has available. The complexity of this problem is exponential in the number of uncertain state variables.

Some systems can be approximated even more. There is an automatic way to enumerate each of the complete system states for a problem and even to sort these states into the appropriate set (Ξ_k, Ω_k, or F_k) for each group V_k, $k = 1, ..., K$. However, calculating the individual transition probabilities for each complete system state can be difficult for large systems. Instead, if the stationary Markov chains modeling the uncertain state variables converge quickly, the equilibrium probability of each complete system state can be automatically calculated and summed over the sets (Ξ_k, Ω_k, and F_k) up to a specified accuracy. For the nominal set, the probability could be broken down further based on the different contribution values.

Let the state propagation models for each uncertain state variable, $\chi_i \in \mathcal{U}_k$, be stationary, ergodic, finite-state Markov processes whose transition matrices, P_i, satisfy

$$P_i = \lim_{n\to\infty} P_i^n. \tag{5.43}$$

Let the estimation uncertainty matrix, $E_{\chi,i}$, for each uncertain state variable be symmetric; also, let the probability of estimating the correct state be the same for each possible state. Finally, let the augmented probability matrix for each state variable, $P_{\chi,i}$, be the matrix that gives the transition probability between complete states of the individual uncertain state variable. By abuse of notation, let $s \in \Lambda_i \times \Lambda_i$ be a complete state of single uncertain state variable χ_i, and let $\text{act}(s) \in \Lambda_i$ and $\text{est}(s) \in \Lambda_i$. For any two states $s_j, s_l \in \Lambda_i \times \Lambda_i$, the augmented transition probability is

$$\begin{aligned} P_{\chi,i}(j,l) = P(\text{val}(\chi_i)[\kappa] = \text{act}(s_l) | \text{val}(\chi_i)[\kappa-1] = \text{act}(s_j)) \times \\ P(\text{val}(\hat{\chi}_i)[\kappa] = \text{est}(s_l) | \text{val}(\chi_i)[\kappa] = \text{act}(s_l)). \end{aligned} \tag{5.44}$$

Proposition 5.6.1. *The augmented transition probability matrix, $P_{\chi,i}$, satisfies*

$$P_{\chi,i} = \lim_{n\to\infty} P_{\chi,i}^n \tag{5.45}$$

if the original transition probability matrix P_i also satisfies the same equation.

Proof. Because P_i satisfies Eq. (5.43), $n_i \times n_i$ matrix is a column vector n_i $1 \times n_i$ vectors, π,

$$P_i = \begin{bmatrix} \pi \\ \vdots \\ \pi \end{bmatrix}, \tag{5.46}$$

where π is the stationary distribution of P_i,

$$\pi = \pi P_i. \tag{5.47}$$

Therefore, the jth column of P_i is a $n_i \times 1$ vector of value $\pi(j)$. Since the estimation probability is based on the actual value of the state variable at time κ instead of at time $\kappa - 1$, the augmented matrix is created as follows. The jth column of $P_{\chi,i}$ corresponding to a state s such that act$(s) = \lambda_j$ and est$(s) = \lambda_l$ is

$$P_{\chi,i}(j) = \vec{\pi}(j) P(\text{val}(\hat{\chi}_i) = \lambda_l | \text{val}(\chi_i) = \lambda_j) \tag{5.48}$$

where $\vec{\pi}(j)$ is a column vector of n_i $\pi(j)$ values. Since the constant row vector is multiplied by a constant, the resulting column vector of $P_{\chi,i}$ is also a constant vector. This is true for all columns of $P_{\chi,i}$; therefore,

$$P_{\chi,i} = \begin{bmatrix} \pi_\chi \\ \vdots \\ \pi_\chi \end{bmatrix}, \tag{5.49}$$

where π_χ is the stationary distribution of the augmented transition probability matrix and $P_{\chi,i}$ satisfies Eq. (5.45). □

Proposition 5.6.2. *The composition of augmented matrices, $\tilde{P}_\chi = P_{\chi,1} \circ P_{\chi,2} \circ ... \circ P_{\chi,N_k}$, satisfies*

$$\tilde{P}_\chi = \lim_{n \to \infty} \tilde{P}_\chi^n. \tag{5.50}$$

Proof. The proof is by construction and is similar to the proof of Proposition 5.6.1. Since the column vectors of each of the augmented matrices are constant vectors, the column vectors of $\tilde{P}_\chi$ are also constant vectors. Therefore,

$$\tilde{P}_\chi = \begin{bmatrix} \tilde{\pi} \\ \vdots \\ \tilde{\pi} \end{bmatrix} \tag{5.51}$$

where $\tilde{\pi}$ is the stationary distribution of $\tilde{P}_\chi$ and $\tilde{P}_\chi$ satisfies Eq. (5.50). □

These two well-known results show that manipulating the stationary Markov chain in the given ways does not change its desired properties. By Proposition 5.6.2, the Markov chain that controls the transitions between complete system states has reached its stationary or equilibrium distribution

initially if each uncertain state variable's stationary Markov chain modeling its state propagation also starts out at the stationary distribution. Thus, the mixing time (time to reach the stationary distribution within a given error) is zero and the stationary probabilities can safely be used in the failure probability estimation. There is an automatic algorithm that can find each complete system state, calculate its stationary probability, and place it into the appropriate set (Ξ_k, Ω_k, and F_k). Then, the stationary probability of each set can be calculated from the sum of the stationary probabilities of its elements. Since there may be many complete system states with a negligible stationary probability, the algorithm sorts the elements so that those with greater probability values are placed into the sets first, and once the sum of the set probabilities reaches a pre-determined value, the algorithm aborts. The pre-determined value must be chosen so that the remaining probability can be assigned to the unsafe set without too much conservatism.

In a uniform completion case, the paths could easily be found and an approximation of the failure probability calculated. For more complicated examples, including non-uniform completion cases and cases with uncertain completion times, Markov Chain Monte Carlo simulation is a useful way to find the failure probability approximation.

5.7 Conclusion

A formal method for calculating the probability of a verifiable sensor-driven hybrid system entering into a specified unsafe set due to estimation uncertainty was presented. The calculation of the failure probability of this system gives the designer some information about the control system. If the failure probability of a given system is too high for the design requirements, several changes could be made. First, the estimator for the state variable could be improved; for some cases, a better sensor could be used to reduce the probability of failure; and finally, the control system could be designed to depend less on a relatively unknown state variable. The verification of control systems in the presence of different forms of uncertainty, including estimation uncertainty, is an important problem, and this approach seems promising as a design tool for hybrid control systems with state-based transitions. These techniques have been applied on two significant examples which are described in the next chapter.