Back to the Future: Consistency-based Trajectory Tracking

James Kurien
Computational Sciences Division
NASA Ames Research Center, MS 269-3
Moffett Field, CA 94035
jkurien@arc.nasa.gov

P. Pandurang Nayak
RIACS
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94035
nayak@cs.stanford.edu

Abstract

Given a model of a physical process and a sequence of commands and observations received over time, the task of an autonomous controller is to determine the likely states of the process and the actions required to move the process to a desired configuration. We introduce a representation and algorithms for incrementally generating approximate belief states for a restricted but relevant class of partially observable Markov decision processes with very large state spaces. The algorithm presented incrementally generates, rather than revises, an approximate belief state at any point by abstracting and summarizing segments of the likely trajectories of the process. This enables applications to efficiently maintain a partial belief state when it remains consistent with observations and revisit past assumptions about the process' evolution when the belief state is ruled out. The system presented has been implemented and results on examples from the domain of spacecraft control are presented.

Introduction

Given a model of a physical system and a sequence of commands and observations received over time, the task of an autonomous controller is to determine the likely states of the system and the actions required to move the system to a desired configuration. Focusing on the state identification question, a *belief state* is a probability distribution over the possible states of a system. If the system has the Markov property, then the influence of a new command and observation upon the belief state can be integrated via Bayes' rule. The updated belief state is a sufficient statistic, capturing within a single distribution all knowledge about the current state of the system contained within a history of commands and observations. The controller then makes use of the updated belief state in selecting an action.

Example 1 Consider the spacecraft propulsion subsystem of Figure 1. The helium tank pressurizes the two propellant tanks. When a propellant path to either engine is open, the pressurized tanks force fuel and oxidizer into the engine, producing thrust. Not shown are valve drivers that control the the latch valves and a set of flow, pressure and acceleration sensors that provide

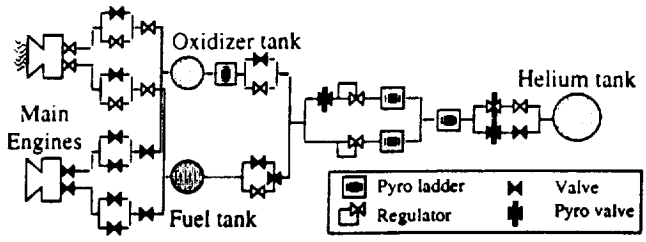

Figure 1: Propulsion system schematic.

partial observability. A model of a system specifies the modes of each component (e.g. a valve may be open, closed, stuck closed, and so on), behavior in each mode (e.g. a closed valve prevents flow), mode transitions (valves usually open when commanded, but stick closed with probability p) and connections between components (fuel flow into an engine is equal to the flow out of the attached fuel valve).

Consider the problem of determining the likelihood of the possible states of this subsystem. Unfortunately, computing a belief state in general requires enumeration of the state space. The propulsion subsystem has 38 components with an average of 3 states each. More complete spacecraft models capture 150 or more components averaging 4 states, yielding a state space of 2^{300} or more and making complete enumeration is implausible. One alternative is to track an approximation whose computation does not require enumeration of the state space, ideally enumerating only the most likely portion of the belief state at each point in time. *Livingstone* (Williams & Nayak 1996) tracks n approximately most likely states of the system by transitioning a small number of tracked states by the transitions that are most likely, given only the current observations. This approximation is extremely efficient and well suited to the problem of tracking the internal state of a machine, where the likelihood of the nominal or expected transition dominates and immediate observations often rule out the nominal trajectory when a failure occurs. The task then becomes one of diagnosing the most likely system transition, chosen from combi-

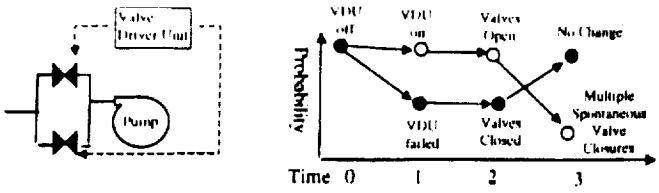

Figure 2: Evolution of a Valve Driver Unit and Valves

nations of component transitions, that would be consistent with the unexpected observations. Using this technique, *Livingstone* is able to perform approximate state identification and reconfiguration of systems with hundreds of state variables. It has been applied to the control of a number of systems within NASA and is an integral part of the Remote Agent architecture demonstrated in-flight on the Deep Space 1 spacecraft in 1999 (Bernard *et al.* 1998). Unfortunately, the true trajectory may not be among the most likely given only the current observations. Consider the following example.

Example 2 Figure 2 illustrates a small system and two possible trajectories. The pump pressurizes the system and the valves, if open, allow a fluid flow. The valve driver unit commands the two valves in parallel via the data bus represented by dashed lines. The graph to the right represents the probability of two possible trajectories. The filled circles represent the true state of the system. At time 0 the VDU is off, the valves are closed and the pump is off. At time 0 the VDU is commanded on. For the sake of illustration, consider an approximate belief state of size 1. The state wherein the VDU is on is placed into the belief state. The true state wherein the VDU is failed is discarded. At time 1, the VDU is commanded to open its valves. Since the only state in the belief state assumes the VDU is on, the single state in the updated belief state has the VDU on and all valves open. In the true, untracked state the valves are closed, as they never received a command. At time 2, the pump is turned on. Pressure is observed at the outlet of the pump, while no pressure is observed downstream of the valves. Failure of the pump alone has zero probability, given the observations. Failure of the VDU in the current time step has no effect on the valves. Thus, the most likely next state consistent with the observations requires that all valves spontaneously and independently shut. Regardless of the number of valves and the unlikeliness of spontaneously closure, this transition must be taken if it exists. If it does not exist, the belief state approximation becomes empty.

In general, as the true state evolves, the tracked subset of states may need to undergo arbitrarily unlikely transitions in order to remain consistent with the observations. While only one trajectory is tracked in this example, for any fraction of the trajectories that are tracked, an example can be constructed wherein the actual state of the system falls outside the tracked fraction and the error in the approximation may become arbitrarily large. We propose an alternative to comitting to a subset of the current belief state or maintaining an approximation of the entire belief state. We propose to maintain the information necessary to begin incrementally generating the current belief state in best-first order at any point in time. Since we do not update the entire belief state, we do not have a sufficient statistic, so a history must be maintained. We introduce a variable to represent every state variable, command and observation at every point in time and an algorithm for incrementally generating the exact belief state at any point. Requiring a set of variables that grows linearly with the history of the system seems impractical except for short duration tasks. We apply two approximations motivated by our experience modeling physical systems for *Livingstone*. First we introduce variables only when close interaction requires. This approximation is conservative in that it may admit impossible trajectories, which are likely to be eliminated with further observations, but it does not eliminate any possible trajectories. We then summarize possible trajectory segments that are consistent with extended periods of system evolution. We are then able to generate an approximate belief state using a constant number of variables. The variables represent an exact model of system evolution over the recent past, an approximate model over the intermediate past, and a gross summarization over the more distant past. This allows assignment of the most likely past transitions to be revisited as new observations become available. The fewest variables, and thus the least flexibility, are allocated to segments of the system trajectory that have remained consistent with the system's observed evolution for the longest time.

In the following sections of the paper, we give the complete history representation followed by a simple, exact, and intractable algorithm for enumerating the belief state. We introduce several optimizations and approximations in order to gain tractability while maintaining the ability to revise assesments of past system evolution. Finally we describe the results of running the algorithm on test scenarios developed applying *Livingstone* and describe upcoming demonstrations of the software on NASA spacecraft.

Transition Systems

We wish to represent the possible histories of a system composed of non-deterministic, concurrent automata given the commands issued to the automata and their output. We create a structure that allows incremental, best-first enumeration of all possible trajectories by extending the formalism of *Livingstone*. In order to compactly represent the trajectories, we add a set of transition variables that represent the non-deterministic transitions each automaton may make at each time step. Each assignment to a transition variable has a likelihood representing the prior probability of the corresponding non-deterministic transtion occurring. One trajectory of the system is thus an assignment to each transition variable, and given the appropriate indepe-

dence assumptions, the set of trajectories can be incrementally enumerated in order of likelihood. In order to capture the feasible behaviors of the automata, we introduce a set of formulae $\mathcal{M}_\Sigma$ describing the input/output mapping of the automata in each state, and a set of formulae $\mathcal{M}_\mathcal{T}$ for describing the feasible transitions of the automata.

Definition 1 A *transition system* $\mathcal{S}$ is a tuple $(\Pi, \mathcal{T}, \mathcal{D}, \mathcal{C}, \mathcal{M}_\Sigma, \mathcal{M}_\mathcal{T})$, where

- Π is a set of *state variables* representing the state of each automaton. Let n denote the number of automata and m denote the number of discrete, synchronous time steps over which the state is to be tracked. Π then contains $m \times n$ variables. Π_t will denote the set of state variables representing the state of the system at time step t. Each state variable y ranges over a finite domain denoted $\delta(y)$. The variable representing the assignment to y at time step t is denoted y_t.
- $\mathcal{D}$ is a finite set of *dependent variables*.
- $\mathcal{C}$ is a finite set of *command variables*.
- $\mathcal{T}$ is a set of *transition variables*. There is one transition variable for each state variable at each time point, denoted $\tau_{y,t}$. Each value in the domain of $\tau_{y,t}$ is is assigned a probability.
- State s_t is an assignment to $\Pi_t \cup \mathcal{T}_t \cup \mathcal{D}_t \cup \mathcal{C}_t$
- $\mathcal{M}_\Sigma$ is a propositional formula over Π_t and $\mathcal{D}_t$ that specifies the feasible subset of the state space. A state is feasible if it makes an assignment to $\Pi_t \cup \mathcal{D}_t$ that is consistent with $\mathcal{M}_\Sigma$.
- $\mathcal{M}_\mathcal{T}$ is a propositional formula Π_t, $\mathcal{D}_t$, $\mathcal{C}_t$, $\mathcal{T}_t$ and Π_{t+1} that specifies the feasible sequences of states. Specifically, $\mathcal{M}_\mathcal{T}$ is a conjunction of formulae of the form:

$$\phi_t \wedge (\tau_{y,t} = \tau^*) \Rightarrow y_{t+1} = y^*$$

where ϕ_t is a propositional formula over Π_t, $\mathcal{D}_t$ and $\mathcal{C}_t$, and $\tau^* \in \delta(\tau_{y,t})$.

Example 3 We introduce a transition system to model a VDU and two valves. The variables corresponding to the VDU consist of a state variable *vdu* representing the mode (*on*, *off*, or *failed*), the transition variable τ_{vdu}, a command variable *cmdin* representing commands to the VDU or its associated valves (*on*, *off*, *open*, *close*, *none*), and a dependent variable *cmdout* representing the command the VDU passes on to its valves (*open*, *close*, or *none*). The set of feasible states of the VDU is specified by the following formula

$$\begin{array}{lll} vdu = on & \Rightarrow & (cmdin = open \Rightarrow cmdout = open) \\ & & \wedge\ (cmdin = close \Rightarrow cmdout = close) \\ & & \wedge\ ((cmdin \neq open \wedge\ cmdin \neq close) \\ & & \quad \Rightarrow cmdout = none) \\ vdu = off & \Rightarrow & cmdout = none \\ vdu = failed & \Rightarrow & cmdout = none \end{array}$$

together with formulae like $(vdu \neq on) \vee (vdu \neq off) \vee (vdu \neq failed), \ldots$ that assert that variables have unique values. The time step subscript is omitted, indicating that all clauses refer to variables within the same time step. The valves $v1$ and $v2$ each have a state variable of domain (*open*, *closed*, or *stuckclosed*), a transition variable τ_{vi} and a dependent variable $flow_{vi}$ of domain (*zero*, *nonzero*). The feasible states of the $v1$ are specified by the formula below. The feasible states of $v2$ are specified similarly.

$$\begin{array}{rcl} v1 = open & \Rightarrow & flow_{v1} = nonzero \\ v1 = closed & \Rightarrow & flow_{v1} = zero \\ v1 = failed & \Rightarrow & flow_{v1} = zero \end{array}$$

$\mathcal{M}_\mathcal{T}$ for τ_{vdu} and τ_{v1} is as follows. τ_{v2} is as τ_{v1}.

$$\begin{array}{l} \tau_{vdu,t} = nominal \Rightarrow \\ \quad \begin{array}{lcl} vdu_t = off \wedge cmdin_t = on & \Rightarrow & vdu_{t+1} = on \\ vdu_t = off \wedge cmdin \neq on & \Rightarrow & vdu_{t+1} = off \\ vdu_t = on \wedge cmdin_t = off & \Rightarrow & vdu_{t+1} = off \\ vdu_t = on \wedge cmdin_t \neq off & \Rightarrow & vdu_{t+1} = on \\ vdu_t = failed & \Rightarrow & vdu_{t+1} = failed \end{array} \\ \tau_{vdu,t} = fail \Rightarrow vdu_{t+1} = failed \end{array}$$

$$\begin{array}{l} \tau_{v1,t} = nominal \Rightarrow \\ \quad \begin{array}{lcl} v1_t = closed \wedge cmdout_t = open & \Rightarrow & v1_{t+1} = open \\ v1_t = closed \wedge cmdout_t \neq open & \Rightarrow & v1_{t+1} = closed \\ v1_t = open \wedge cmdout_t = closed & \Rightarrow & v1_{t+1} = closed \\ v1_t = open \wedge cmdout_t \neq close & \Rightarrow & v1_{t+1} = open \\ v1_t = stuckclosed & \Rightarrow & v1_{t+1} = stuckclosed \end{array} \\ \tau_{v1,t} = stick \Rightarrow v1_{,t+1} = stuckclosed \end{array}$$

Infinitesimals

In general we must consider all trajectories when determining the likelihoods, or even relative likelihoods, of a set of states, as many unlikely trajectories may contribute probability mass to the same outcome state. The transition system representation will allow us to enumerate the most likely trajectories of a system in order. We would like to find a natural restriction on the form of prior probabilities of transitions such that there is a correspondence between the most likely trajectories we are able to identify and the most likely states in which we are interested. Our experience applying *Livingstone* was that an ad-hoc, order of magnitude probability scale was a sufficient representation for two reasons. First, the internal behavior of a machine is usually far less stochastic than its interaction with its environment. There is an expected or nominal behavior that a component will exhibit for a given state and input, with failure modes one or more orders of magnitude less likely. Second, precise estimates for these priors are often either inaccessible or unknown. However, the relative plausibility of each failure mode during operation can be elicited quite easily. In this work, we formalize and capitalize on these characteristics of the priors by making use of infinitesimals (Goldszmidt & Pearl 1992) to model the relative likelihoods of failures.

An infinitesimal probability is represented by an infintesimally small constant raised to an exponent referred to as the *rank*. The rank can be considered the degree of unbelievability. Intuitively, one would not consider a rank 2 infinitesimal believable unless all rank 0 and rank 1 possibilities had been eliminated. Composition of infinitesimals has many desirable properties.

If A and B are independent events, then

$$Rank(AB) = Rank(A) + Rank(B)$$
$$Rank(A \vee B) = min(Rank(A), Rank(B))$$

Thus an outcome that can occur through multiple independent events has rank i if one event has rank i and the remaining events, even if arbitrarily many, have ranks of i or more. This property allows us to consider only the most likely trajectories leading to a state: if a sequence of events of rank i ends in state s_j, then an arbitrary number of higher rank (i.e. less likely) trajectories leading to s_j will not change its rank. Similarly, if state s_j is reached by a trajectory of rank i, and no trajectory of rank i or less reaches s_k, then s_j is more likely than s_k, even if an arbitrary number of unlikely trajectories leading to s_k remain unconsidered. We frame our algorithms in terms most likely trajectories, knowing that there is a correct correspondence to most likely states given the infinitesimal interpretation of the priors.

Trajectory Identification

Definition 2 A *trajectory* for S is a sequence of states $s_0, s_1, \ldots s_m$ such that for all t such that $0 < t < m$ s_t is consistent with $\mathcal{M}_\Sigma$ and for all t such that $0 < t < (m-1)$ $s_t \cup s_{t+1}$ is consistent with $\mathcal{M}_\mathcal{T}$.

Consider the problem of determining the state of a physical process modeled by a transition system S at each point in a trajectory $s_0 \ldots s_m$. The subset of the dependent variables $\mathcal{D}$ whose assignment corresponds to a measurement from the process will be referred to as the observations, $\mathcal{O}$. We are given an assignment for the initial state, Π_0. In addition we are given assignments to commands $\mathcal{C}_t$ and observations $\mathcal{O}_t$ for all $0 < t < m$. The task is to choose assignments to $\tau_{y,t}$ for all y and t so as to ensure consistency with $\mathcal{M}_\Sigma$ and $\mathcal{M}_\mathcal{T}$ and maximize the likelihood of the trajectory. That is to say, given a starting state, a set of commands and a set of observations, we must find the most likely sequence of transitions such that each state is consistent with the state model $\mathcal{M}_\Sigma$ and the transitions are consistent with the transition model $\mathcal{M}_\mathcal{T}$. We define the likelihood of the trajectory to be:

$$\sum_{t=0}^{m} \sum_{y=1}^{n} Rank(\tau_{y,t})$$

This definition makes the assumption that the likelihood of assignments to $\tau_{y,t}$ are independent of $\tau_{x,t}$.

A Simple Tracking Algorithm

The transition-system formulation suggests an intuitive procedure to begin enumerating the belief state at any point. The transition system is initialized with $\mathcal{M}_\Sigma$ and a copy of all variables, representing the initial state. At time step t, we introduce a copy of $\mathcal{M}_\Sigma$ and a copy of all variables, representing the next state of the system, as well as a copy of $\mathcal{M}_\mathcal{T}$ representing the constraints between the current state and the next state. We assign $\mathcal{C}_t$ and $\mathcal{O}_{t+1}$ according to how the system was commanded and the observations that resulted.

Example 4 Below is an example trajectory-tracking problem. The command is *cmdin* and the observations are $flow_{v1}$ and $flow_{v2}$. These variables are assigned by the problem, as is the start state. The $\tau_{y,t}$ assignments must be chosen. The remaining variables will be constrained based upon these assignments. For all $\tau_{y,t}$ we will assume $Rank(\tau_{y,t} = nominal) = 0$ and $Rank(\tau_{y,t} \neq nominal) = 1$.

Variable	$t=0$	$t=1$	$t=2$
vdu_t	*off*		
$\tau_{vdu,t}$			
$cmdin_t$	*on*	*open*	*none*
$cmdout_t$			
$v1_t$	*closed*		
$\tau_{v1,t}$			
$flow_{v1,t}$	*zero*	*zero*	*zero*
$v2_t$	*closed*		
$\tau_{v2,t}$			
$flow_{v2,t}$	*zero*	*zero*	*zero*

Trajectories may be enumerated in order by enumerating assignments to all $\tau_{y,t}$ in order of the sum of the ranks, then testing for consistency with $\mathcal{M}_\mathcal{T}$ and $\mathcal{M}_\Sigma$. Conflict-directed, best-first search, or *CBFS* (Dressler & Struss 1992; de Kleer & Williams 1989; Williams & Nayak 1996) greatly focuses this process by using conflicts. A conflict that renders a candidate solution inconsistent is used to avoid generating any further candidate solutions that contain the same conflict.

Example 5 Below are two solutions to the above problem, representing a single failure of rank 1 at time 1 and a double failure of rank 2 at time 2, respectively.

Variable	$t=0$	$t=1$	$t=2$
vdu_t	off	*failed*	*failed*
$\tau_{vdu,t}$	fail	nominal	
$cmdin_t$	on	open	none
$cmdout_t$	*none*	*none*	*none*
$v1_t$	closed	*closed*	*closed*
$\tau_{v1,t}$	nominal	nominal	
$flow_{v1,t}$	zero	zero	zero
$v2_t$	closed	*closed*	*closed*
$\tau_{v2,t}$	nominal	nominal	
$flow_{v1,t}$	zero	zero	zero

Variable	$t=0$	$t=1$	$t=2$
vdu_t	*off*	*on*	*on*
$\tau_{vdu,t}$	nominal	nominal	
$cmdin_t$	*on*	*open*	*none*
$cmdout_t$	*none*	*open*	*none*
$v1_t$	*closed*	*closed*	*stuckclosed*
$\tau_{v1,t}$	nominal	stick	
$flow_{v1,t}$	*zero*	*zero*	*zero*
$v2_t$	*closed*	*closed*	*stuckclosed*
$\tau_{v2,t}$	nominal	stick	
$flow_{v2,t}$	*zero*	*zero*	*zero*

While applying CBFS to the full transition system exactly enumerates the most likely trajectories, and thus states, in order, problem size is a significant issue. Testing consistency of each candidate trajectory requires propositions representing the possible assignments to each variable at each time point. In addition, these propositions are constrained by a copy of

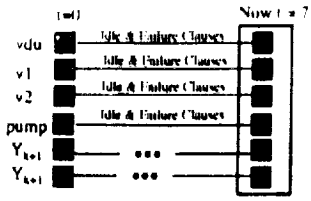

Figure 3: Evolution before commanding the valves

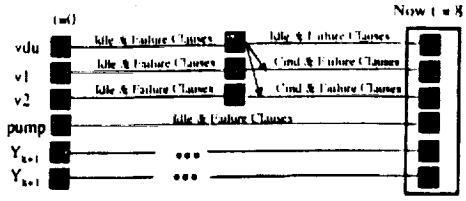

Figure 4: Evolution upon commanding the valves

$\mathcal{M}_\mathcal{T}$ and $\mathcal{M}_\Sigma$ at each time step. Let $|\Pi|_p$ denote the number of propositions needed to represent each possible value of each variable in Π. If we wish to track the system for m steps, checking a single trajectory candidate becomes a consistency problem of $m \times (|\mathcal{T}|_p + |\Pi|_p + |\mathcal{C}|_p + |\mathcal{D}|_p)$ propositions and $m \times (|\mathcal{M}_\mathcal{T}| + |\mathcal{M}_\Sigma|)$ clauses. Given sufficient time, m will outgrow the available computational resources.

Problem Size Reduction

In this section, we reduce the structure needed to represent the evolution of the system at a time point from a complete copy of the system model to a small number of variables and clauses. Intuitively, when a command is issued to the system, only a small number of components participate in transmitting that command through the system or transitioning in response to the command. Consider Figure 3. The squares represent state variables, the lines sets of constraints from $\mathcal{M}_\mathcal{T}$. As of time 7, the valves, pump and VDU have not been commanded nor have they interacted with other components by passing a command. If we did not detect a failure of any of these components, we represent the possibility that they remained idle or failed silently with a single set of variables and constraints as illustrated. At time 7 we command the valves on. We require variables $v1_8$ and $v2_8$ to represent the new states of the valves. $\mathcal{M}_\mathcal{T}$ suggests vdu_7, $v1_7$ and $v2_7$ will interact with $v1_8$ and $v2_8$. These variables, along with necessary transition variables $\tau_{vdu,7}$, $\tau_{v1,7}$ and, $\tau_{v2,7}$, are introduced to the system with the appropriate clauses from $\mathcal{M}_\mathcal{T}$. For all other variables, the variable representing y_7 is adequate to represent y_8. Figure 4 illustrates this process. In order to have a well-founded algorithm, we first we place a natural restriction on $\mathcal{M}_\mathcal{T}$ that does not impact correctness. Second we introduce an approximation that, importantly, does not rule out trajectories. Instead, some trajectories that are not consistent with past observations may be admitted, with the possibility that future observations will eliminate them.

Restricting $\mathcal{M}_\mathcal{T}$

We restrict $\mathcal{M}_\mathcal{T}$ as do *Livingstone* and *Burton* (Williams & Nayak 1997): a component moves to a failure state with equal probability from any state, and except for failures a component is kept in its current state by the *idle* command. $\mathcal{M}_\mathcal{T}$ is limited to the forms:

$$(\tau_{y,t} = \tau_{failure}) \Rightarrow y_{t+1} = y_{failure}$$

$$(\mathcal{C}_{y,t} = idle) \wedge (\tau_{y,t} = nominal) \Rightarrow y_{t+1} = y_t$$

$$(\mathcal{C}_{y,t} = C^*) \wedge \phi_t \wedge (\tau_{y,t} = nominal) \Rightarrow y_{t+1} = y^*$$

where ϕ_t is a propositional formula over $\Pi_t \cup \mathcal{D}_t$, $C^* \in \delta(\mathcal{C}_{y,t})$, $nominal \in \delta(\tau_{y,t})$ and $\tau_{fail} \in \delta(\tau_{y,t})$. Formulae of the first form model failures while formulae of the third form model nominal, commanded transitions. We next preprocess ϕ_t by replacing references to $\mathcal{D}_t$ with an implicate π_t that involves only Π_t. Intuitively, we replace a formula on the command a component c receives with a formula on the chain of components causing c to receive the command. We expect that for the type of clauses $\mathcal{M}_\mathcal{T}$ contains, growth will be proportional to the length of the component chain, l, which ranged from 1 to 5 in the spacecraft model of (Bernard *et al.* 1998). Our initial experience supports this hypothesis. Non-idle, non-failure clauses then take the following form, which does not depend upon $\mathcal{D}$.

$$(\mathcal{C}_{y,t} = C^*) \wedge \pi_t \wedge (\tau_{y,t} = nominal) \Rightarrow y_{t+1} = y^*$$

Intuitively, as far as $\mathcal{M}_\mathcal{T}$ is concerned, we need only introduce the variables of Π_t found in π_t if $\mathcal{C}_{y,t} \neq idle$.

Eliminating intermediate observations

$\mathcal{M}_\Sigma$ remains, and requires introduction of all variables in Π_t and $\mathcal{D}_t$ in order to check consistency against $\mathcal{O}_t$. We proceed by eliminating all variables $\mathcal{O}_t$ for values of t sufficiently far in the past. That is to say, transition choices are only constrained by consistency between the trajectories they imply and recent observations. As the system evolves, variables representing older observations, and the copies of $\mathcal{M}_\Sigma$ that constrain them, are discarded, or with relabeling, never introduced. For the portions of the trajectory where $\mathcal{M}_\Sigma$ is not introduced, we need only introduce the limited portion of Π_t required by $\mathcal{M}_\mathcal{T}$. This is of course an approximation. Note that even after observations are discarded, no partial assignment to $\mathcal{T}$ that was discovered to be in conflict with the observations will be reconsidered, as such conflicts are stored. However, if a partial assignment to $\mathcal{T}$ is in conflict with an observation, but is not considered until after the observation has been discarded, "imposter" trajectories containing the inconsistent assignment will be admitted. This is a conservative approximation in that no consistent trajectories are eliminated. Each imposter trajectory is checked against new observations and eliminated as soon as it fails to describe the future evolution of the system.

Selective Model Extension

Based upon these restrictions, the procedure *extend* introduces into time step t only the small fraction of the model involved with the evolution of the system due to the command $C_{y,t} = C^*$. This hinges upon Theorem 1.

Theorem 1 *Assume $C_{y,t} = C^*$, $C^* \neq idle$, and for all $x \neq y$, $C_{x,t} = idle$. Consider the formula of $\mathcal{M}_\mathcal{T}$*

$$(C_{y,t} = C^*) \wedge \pi_t \wedge (\tau_{y,t} = nominal) \Rightarrow y_{y+1} = y^*$$

For all state variables x_t, $x \neq y$, if $x_t \notin \pi_t$, then an equivalent consistency problem is formed by replacing x_t, $\tau_{x,t}$ and all formulae of $\mathcal{M}_\mathcal{T}$ involving these variables with a constraint between x_{t-1} and x_{t-1}.

Intuitively, $\mathcal{M}_\mathcal{T}$ and the assignment to C_t require x_{t+1} to be a failure or equal to x_{t-1} and prevent x_t from influence any other variables. Use of *extend* renders the problem size per time step proportional to $|\ \pi_t\ |$. Details appear in the eight-page version of this paper.

Conflict Coverage Search

The strengths of efficiently tracking a partial belief state are merged with the flexibility of incrementally enumerating belief states in the *CoverTrack* procedure of Figure 5. *CoverTrack* maintains a partial belief state of all consistent trajectories of rank γ. As a command and observations are received, trajectories are simply extended by the nominal, zero rank transition, and little computation is required. An extended trajectory that is inconsistent requires an additional failure, and will not have rank γ. These are discarded in keeping with the plausibility interpretation of infinitesimals. If no trajectories remain, a new belief state consisting of all trajectories of $\gamma + 1$ is generated, and tracking resumes. *CoverTrack* uses the *extend* algorithm to ensure the transition system is extended by a small number of variables at each time step. As trajectories are eliminated, the conflicts between partial assignments to $\mathcal{T}$ and observations are recorded. The *GenerateCover* algorithm generates all assignments to $\mathcal{T}$ of rank $\gamma + 1$ (or higher if none exist) that cover all known conflicts. Intuitively, we leave the $\tau_{y,t}$ at their zero rank values, introducing reassignment only to avoid conflicts, with a total cost of $\gamma + 1$. This is the *hitting set* problem. Details appear in the eight-page version of this paper.

Finite Horizons

While selective extension reduces the variables per time step, we still require an unbounded number of variables over time. Note that members of the belief state of *CoverTrack* contain initial transition assignments that have remained consistent with the system's evolution for an extended period. We make an additional approximation by comitting to these partial assignments. To operate over a fixed time horizon h, the most likely partial trajectories represented by assignments to $\tau_{y,t}$ for $0 < t < (m - h)$ are summarized by assignments to a single summary variable. The problem is reduced to a constant size, wherein the last variable assignment captures a choice of likely initial trajectories. Details appear in the eight-page version of the paper.

```
procedure CoverTrack()
  Conflicts = ∅; γ = 0;
  Variables = Π0 ∪ D0
  Assign Π0 to initial state;
  BeliefState = the empty trajectory;
  loop
    while BeliefState is not empty do
      Variables= extend(Variables, Ct), adding Tt
      Assign Ot+1 according to observations received;
      Assign Tt to nominal, 0 rank assignment.
      Survived = ∅;
      while BeliefState is not empty do
        Extension = pop(BeliefState) + Tt;
        if consistent(Extension) then
          push(Extension,Survived);
        else
          push(conflict in Extension,Conflicts);
        endif
      endwhile
      BeliefState= Survived; report BeliefState;
    endwhile
    BeliefState=GenerateCover(Variables,Conflicts,γ);
    γ = Rank(first(BeliefState)) + 1;
  endloop
```

Figure 5: Conflict Coverage Tracking Procedure

Results

The algorithms presented have been implemented and correctly track scenarios translated from *Livingstone* that confound partial belief state algorithms. Examples include silent failures whose impact propagates forward through time and multiple failure modes that are indistinguishable without future observations. We have largely translated *Livingstone*'s spacecraft models to begin performance analysis. Since *Livingstone* is roughly a Lisp implementation of *CBFS* with observation and summary horizons of 1, we expect to meet or exceed its generally high performance at those settings. Interestingly, increasing correctness by adding variables to admit failures in the recent past will not uniformly degrade performance. Removing a particularly critical failure in a complex *Livingstone* model has increased computation by orders of magnitude when the failure occurred, as not admitting the failure forces consideration of incorrect and potentially expensive hypotheses. An analysis with large models over a variety of horizons will appear in a longer version of this paper.

Related Work

The problem described is a partially observable Markov decision process, or POMDP, with focus placed upon belief revision. A large body of work exists addressing belief revision of exact and approximate belief states. Boyen and Koller (Boyen & Koller 1998), for example,

provide an approximate, factored belief state with a bounded error that can be updated without enumerating the state space. Intuitively, the error bound relies upon the stochasticity of the underlying system, parameterized by the problem's *mixing rate*, to continually smear both the approximate and true distributions, exponentially reducing rather than compounding errors over time. Unfortunately, the systems we consider have inadequate mixing rates. Intuitively, when monitoring the internal state of a complex device such as a spacecraft, the device may behave as if it were deterministic for long periods, then exhibit a failure, then return to apparent determinism. There is no process in place with sufficient stochasticity to quickly contract an arbitrary error introduced by a factored approximation.

Conclusions

This paper presents incremental belief state generation as an alternative to belief revision. Application of the described approximations creates a family of representations that track against a full model for a number of steps, then against a reduced model, then summarize over the most likely initial trajectories. Since the abstractions of the trajectory segments (full, minimal extension or summary) are represented uniformly, a single, simple search procedure may be employed. *CoverTrack* combines the efficiency of partial belief state propagation with the flexibilty of the transition system representation. The system will be evaluated on Earth-bound testbeds representing an interferometer and a Mars propellant plant. In addition, it will be flown as an experiment on the X-34 rocket plane in 2001 and the X-37 orbital vehicle in 2002.

References

Bernard, D. E.; Dorais, G. A.; Fry, C.; Jr., E. B. G.; Kanefsky, B.; Kurien, J.; Millar, W.; Muscettola, N.; Nayak, P. P.; Pell, B.; Rajan, K.; Rouquette, N.; Smith, B.; and Williams, B. C. 1998. Design of the remote agent experiment for spacecraft autonomy. In *IEEE Aerospace*.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *Proceedings UAI-98*.

de Kleer, J., and Williams, B. C. 1989. Diagnosis with behavioral modes. In *Proceedings of IJCAI-89*.

Dressler, O., and Struss, P. 1992. Back to defaults: Characterizing and computing diagnoses as coherent assumption sets. In *ECAI-92*.

Goldszmidt, M., and Pearl, J. 1992. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Proceedings of KR-92*, 661–672.

Williams, B. C., and Nayak, P. P. 1996. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI-96*, 971–978.

Williams, B. C., and Nayak, P. P. 1997. A reactive planner for a model-based executive. In *Proceedings of IJCAI-97*.