2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Optimal Path-planning Under Finite Memory Obstacle Dynamics Based on Probabilistic Finite State Automata Models[†]

Ishanu Chattopadhyay[‡] ixc128@psu.edu
Asok Ray[‡] axr2@psu.edu

***Abstract*— The $\nu^\star$-planning algorithm is generalized to handle finite memory obstacle dynamics. A sufficiently long observation sequence of obstacle dynamics is algorithmically compressed via Symbolic Dynamic Filtering to obtain a probabilistic finite state model which is subsequently integrated with the navigation automaton to generate an overall model reflecting both navigation constraints and obstacle dynamics. A $\nu^\star$-based solution then yields a deterministic plan that maximizes the difference of the probabilities of reaching the goal and of hitting an obstacle. The approach is validated by simulated solution of dynamic mazes.**

***Index Terms*— Language Measure; Probabilistic Finite State Machines; Robotics; Path Planning; Supervisory Control**

1. Introduction & Motivation

Recently, a novel path planning algorithm $\nu^\star$ was reported that models the navigation problem in the framework of Probabilistic Finite State Machines and computes robust optimal plans via optimization of the PFSA from a strictly control-theoretic viewpoint. In this paper, we present a significant improvement; $\nu^\star$ is generalized to handle finite memory obstacle dynamics. A sufficiently long observation sequence of obstacle movement in terms of an evolving obstacle map is symbolically compressed to obtain a finite state probabilistic model of obstacle dynamics which is then used to compute optimal plans in dynamic cluttered environments. It is important to note that the problem considered in this paper is different from the ones that modify plans on-the-fly (*e.g.* $D^\star$), to incorporate newly learnt information about obstacle locations; in this paper we propose an approach for optimally incorpoating apriori knowledge of the expected dynamical evolution of obstacles. The key advantages are:

1) **Pre-processing is cheap:** The cellular decomposition required by $\nu^\star$ is simple and computationally cheap. The cells are mapped to PFSA states which are defined to have identical connectivity via symbolic inter-state transitions.
2) **Fundamentally different from search:** $\nu^\star$ optimizes the resultant PFSA via a iterative sequence of combinatorial operations which elementwise maximizes the language measure vector [1][2].
3) **Computational efficiency:** The time complexity of each iteration step can be shown to be linear in problem size implying significant numerical advantage over search-based methods for high-dimensional problems.
4) **Global monotonicity:** The solution iterations are globally monotonic. The final waypoint sequence is generated essentially by following the measure gradient which is maximized at the goal. The measure gradient is reminiscent of potential field methods [3]. However, $\nu^\star$ automatically generates the measure gradient; no potential function is necessary. Furthermore, the potential function based planners often get trapped in local minimum which can be shown to be a mathematical impossibility for $\nu^\star$.

The paper is organized in five sections including the present one. Section 2 briefly explains the language-theoretic models considered in this paper, reviews the language-measure-theoretic optimal control of probabilistic finite state machines and presents the necessary details of the reported $\nu^\star$ algorithm. Section 3 presents the approach for symbolic compression of observed obstacle dynamics. Section 4 formulates the integration of the obstacle automaton and the navigation automaton to generate the overall model to be subsequently optimized via $\nu^\star$ planning. Simulation examples are included for illustration. The paper is summarized and concluded in Section 5 with recommendations for future work.

2. Language Measure-theoretic Optimization

This section summarizes the signed real measure of regular languages; the details are reported in [1]. Let $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m\rangle$ be a trim (i.e., accessible and co-accessible) finite-state automaton model that represents the discrete-event dynamics of a physical plant, where $Q = \{q_k : k \in \mathcal{I}_Q\}$ is the set of states and $\mathcal{I}_Q \equiv \{1, 2, \cdots, n\}$ is the index set of states; the automaton starts with the initial state q_i; the alphabet of events is $\Sigma = \{\sigma_k : k \in \mathcal{I}_\Sigma\}$, having $\Sigma \bigcap \mathcal{I}_Q = \emptyset$ and $\mathcal{I}_\Sigma \equiv \{1, 2, \cdots, \ell\}$ is the index set of events; $\delta : Q \times \Sigma \rightarrow Q$ is the (possibly partial) function of state transitions; and $Q_m \equiv \{q_{m_1}, q_{m_2}, \cdots, q_{m_l}\} \subseteq Q$ is the set of marked (i.e., accepted) states with $q_{m_k} = q_j$ for some $j \in \mathcal{I}_Q$. Let Σ^* be the Kleene closure of Σ, i.e., the set of all finite-length strings made of the events belonging to Σ as well as the empty string ϵ that is viewed as the identity of the monoid Σ^* under the operation of string concatenation, i.e., $\epsilon s = s = s\epsilon$. The state transition map δ is recursively extended to its reflexive and transitive closure $\delta : Q \times \Sigma^* \rightarrow Q$ by defining $\forall q_j \in Q$, $\delta(q_j, \epsilon) = q_j$ and $\forall q_j \in Q, \sigma \in \Sigma, s \in \Sigma^\star$, $\delta(q_i, \sigma s) = \delta(\delta(q_i, \sigma), s)$

Definition 2.1: *The language $L(q_i)$ generated by a DFSA G initialized at the state $q_i \in Q$ is defined as: $L(q_i) = \{s \in \Sigma^* \mid \delta^*(q_i, s) \in Q\}$ The language $L_m(q_i)$ marked by the DFSA G initialized at the state $q_i \in Q$ is defined as: $L_m(q_i) = \{s \in \Sigma^* \mid \delta^*(q_i, s) \in Q_m\}$*

Definition 2.2: *For every $q_j \in Q$, let $L(q_i, q_j)$ denote the set of all strings that, starting from the state q_i, terminate at the state q_j, i.e., $L_{i,j} = \{s \in \Sigma^* \mid \delta^*(q_i, s) = q_j \in Q\}$*

The formal language measure is first defined for terminating plants [4] with sub-stochastic event generation probabilities *i.e.* the event generation probabilities at each state summing to strictly less than unity.

Definition 2.3: *The event generation probabilities are specified by the function $\tilde{\pi} : \Sigma^\star \times Q \rightarrow [0,\ 1]$ such that $\forall q_j \in Q, \forall \sigma_k \in \Sigma, \forall s \in \Sigma^\star$,*

(1) $\tilde{\pi}(\sigma_k, q_j) \triangleq \tilde{\pi}_{jk} \in [0, 1)$; $\sum_k \tilde{\pi}_{jk} = 1 - \theta$, with $\theta \in (0, 1)$;
(2) $\tilde{\pi}(\sigma, q_j) = 0$ if $\delta(q_j, \sigma)$ is undefined; $\tilde{\pi}(\epsilon, q_j) = 1$;
(3) $\tilde{\pi}(\sigma_k s, q_j) = \tilde{\pi}(\sigma_k, q_j)\ \tilde{\pi}(s, \delta(q_j, \sigma_k))$.

The $n \times \ell$ event cost matrix is defined as: $\widetilde{\mathbf{\Pi}}|_{ij} = \tilde{\pi}(q_i, \sigma_j)$

Definition 2.4: *The state transition probability $\pi : Q \times Q \rightarrow [0, 1)$, of the DFSA G_i is defined as follows: $\forall q_i, q_j \in Q, \pi_{ij} = \sum_{\sigma \in \Sigma \text{ s.t. } \delta(q_i,\sigma)=q_j} \tilde{\pi}(\sigma, q_i)$ The $n \times n$ state transition probability matrix is defined as $\mathbf{\Pi}|_{jk} = \pi(q_i, q_j)$*

The set Q_m of marked states is partitioned into Q_m^+ and Q_m^-, i.e., $Q_m = Q_m^+ \cup Q_m^-$ and $Q_m^+ \cap Q_m^- = \emptyset$, where Q_m^+ contains all *good* marked states that we desire to reach, and Q_m^- contains all *bad* marked states that we want to avoid, although it may not always be possible to completely avoid the *bad* states while attempting to reach the *good* states. To characterize this, each marked state is assigned a

[†]This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant No. W911NF-07-1-0376 and by the Office of Naval Research under Grant No. N00014-08-1-380. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

[‡]The Pennsylvania State University, University Park, PA 16802, USA.

real value based on the designer's perception of its impact on the system performance.

Definition 2.5: *The characteristic function $\chi : Q \to [-1,1]$ that assigns a signed real weight to state-based sublanguages $L(q_i, q)$ is defined as: $\forall q \in Q$, $\chi(q) \in \begin{cases} [-1,0), & q \in Q_m^- \\ \{0\}, & q \notin Q_m \\ (0,1], & q \in Q_m^+ \end{cases}$ The state weighting vector, denoted by $\boldsymbol{\chi} = [\chi_1\ \chi_2\ \cdots\ \chi_n]^T$, where $\chi_j \equiv \chi(q_j)$ $\forall j \in \mathcal{I}_Q$, is called the $\boldsymbol{\chi}$-vector. The j-th element χ_j of $\boldsymbol{\chi}$-vector is the weight assigned to the corresponding terminal state q_j.*

In general, the marked language $L_m(q_i)$ consists of both good and bad event strings that, starting from the initial state q_i, lead to Q_m^+ and Q_m^- respectively. Any event string belonging to the language $L^0 = L(q_i) - L_m(q_i)$ leads to one of the non-marked states belonging to $Q - Q_m$ and L^0 does not contain any one of the good or bad strings. Based on the equivalence classes defined in the Myhill-Nerode Theorem, the regular languages $L(q_i)$ and $L_m(q_i)$ can be expressed as: $L(q_i) = \bigcup_{q_k \in Q} L_{i,k}$ and $L_m(q_i) = \bigcup_{q_k \in Q_m} L_{i,k} = L_m^+ \cup L_m^-$ where the sublanguage $L_{i,k} \subseteq G_i$ having the initial state q_i is uniquely labelled by the terminal state $q_k, k \in \mathcal{I}_Q$ and $L_{i,j} \cap L_{i,k} = \emptyset\ \forall j \neq k$; and $L_m^+ \equiv \bigcup_{q_k \in Q_m^+} L_{i,k}$ and $L_m^- \equiv \bigcup_{q_k \in Q_m^-} L_{i,k}$ are good and bad sublanguages of $L_m(q_i)$, respectively. Then, $L^0 = \bigcup_{q_k \notin Q_m} L_{i,k}$ and $L(q_i) = L^0 \cup L_m^+ \cup L_m^-$.

A signed real measure $\mu^i : 2^{L(q_i)} \to \mathbb{R} \equiv (-\infty, +\infty)$ is constructed on the σ-algebra $2^{L(q_i)}$ for any $i \in \mathcal{I}_Q$; interested readers are referred to [1] for the details of measure-theoretic definitions and results. With the choice of this σ-algebra, every singleton set made of an event string $s \in L(q_i)$ is a measurable set. By Hahn Decomposition Theorem [5], each of these measurable sets qualifies itself to have a numerical value based on the above state-based decomposition of $L(q_i)$ into L^0(null), L^+(positive), and L^-(negative) sublanguages.

Definition 2.6: *Let $\omega \in L(q_i, q_j) \subseteq 2^{L(q_i)}$. The signed real measure μ^i of every singleton string set $\{\omega\}$ is defined as: $\mu^i(\{\omega\}) \equiv \tilde{\pi}(\omega, q_i)\chi(q_j)$. The signed real measure of a sublanguage $L_{i,j} \subseteq L(q_i)$ is defined as: $\mu_{i,j} \equiv \mu^i(L(q_i, q_j)) = \left(\sum_{\omega \in L(q_i,q_j)} \tilde{\pi}[\omega, q_i]\right)\chi_j$*

Therefore, the signed real measure of the language of a DFSA G_i initialized at $q_i \in Q$, is defined as $\mu_i \equiv \mu^i(L(q_i)) = \sum_{j \in \mathcal{I}_Q} \mu^i(L_{i,j})$. It is shown in [1] that the language measure can be expressed as $\mu_i = \sum_{j \in \mathcal{I}_Q} \pi_{ij}\mu_j + \chi_i$. The language measure vector, denoted as $\boldsymbol{\mu} = [\mu_1\ \mu_2\ \cdots\ \mu_n]^T$, is called the $\boldsymbol{\mu}$-vector. In vector form, we have $\boldsymbol{\mu} = \boldsymbol{\Pi}\mu + \boldsymbol{\chi}$ whose solution is given by $\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Pi})^{-1}\boldsymbol{\chi}$ The inverse exists for terminating plant models [4] because $\boldsymbol{\Pi}$ is a contraction operator [1] due to the strict inequality $\sum_j \Pi_{ij} < 1$. The residual $\theta_i = 1 - \sum_j \Pi_{ij}$ is referred to as the termination probability for state $q_i \in Q$. We extend the analysis to non-terminating plants with stochastic transition probability matrices (*i.e.* with $\theta_i = 0$, $\forall q_i \in Q$) by renormalizing the language measure [1] with respect to the uniform termination probability of a limiting terminating model as described next.

Let $\widetilde{\Pi}$ and Π be the stochastic event generation and transition probability matrices for a non-terminating plant $G_i = \langle Q, \Sigma, \delta, q_i, Q_m \rangle$. We consider the terminating plant $G_i(\theta)$ with the same DFSA structure $\langle Q, \Sigma, \delta, q_i, Q_m \rangle$ such that the event generation probability matrix is given by $(1-\theta)\widetilde{\Pi}$ with $\theta \in (0,1)$ implying that the state transition probability matrix is $(1-\theta)\Pi$.

Definition 2.7: (Renormalized Measure:) *The renormalized measure $\nu_\theta^i : 2^{L(q_i(\theta))} \to [-1,1]$ for the θ-parametrized terminating plant $G_i(\theta)$ is defined as: $\forall \omega \in L(q_i(\theta))$, $\nu_\theta^i(\{\omega\}) = \theta\mu^i(\{\omega\})$ The corresponding matrix form is given by $\boldsymbol{\nu_\theta} = \theta\ \boldsymbol{\mu} = \theta\ [I - (1-\theta)\Pi]^{-1}\boldsymbol{\chi}$* with $\theta \in (0,1)$. *We note that the vector representation allows for the following notational simplification $\nu_\theta^i(L(q_i(\theta))) = \boldsymbol{\nu_\theta}\big|_i$ The renormalized measure for the non-terminating plant G_i is defined to be $\lim_{\theta \to o^+} \nu_\theta^i$.*

A. Event-driven Supervision of PFSA

Plant models considered in this paper are *deterministic* finite state automata (plant) with well-defined event occurrence *probabilities*. In other words, the occurrence of events is probabilistic, but the state at which the plant ends up, *given a particular event has occurred*, is deterministic. Since no emphasis is placed on the initial state and marked states are completely determined by χ, the models can be completely specified by a sextuple as: $G = (Q, \Sigma, \delta, \widetilde{\boldsymbol{\Pi}}, \chi, \mathscr{C})$

Definition 2.8: (Control Philosophy) *If $q_i \xrightarrow[\sigma]{} q_k$, and the event σ is disabled at state q_i, then the supervisory action is to prevent the plant from making a transition to the state q_k, by forcing it to stay at the original state q_i. Thus disabling any transition σ at a given state q results in deletion of the original transition and appearance of the self-loop $\delta(q,\sigma) = q$ with the occurrence probability of σ from the state q remaining unchanged in the supervised and unsupervised plants. For a given plant, transitions that can be disabled in the sense of Definition 2.8 are defined to be controllable transitions. The set of controllable transitions in a plant is denoted $\mathscr{C}$. Note controllability is state-based.*

B. The Optimal Supervision Problem: Formulation & Solution

A supervisor disables a subset of the set $\mathscr{C}$ of controllable transitions and hence there is a bijection between the set of all possible supervision policies and the power set $2^{\mathscr{C}}$. That is, there exists $2^{|\mathscr{C}|}$ possible supervisors and each supervisor is uniquely identifiable with a subset of $\mathscr{C}$ and the language measure ν allows a quantitative comparison of different policies.

Definition 2.9: *For an unsupervised plant $G = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C})$, let $G^\dagger$ and $G^\ddagger$ be the supervised plants with sets of disabled transitions, $\mathscr{D}^\dagger \subseteq \mathscr{C}$ and $\mathscr{D}^\ddagger \subseteq \mathscr{C}$, respectively, whose measures are $\boldsymbol{\nu}^\dagger$ and $\boldsymbol{\nu}^\ddagger$. Then, the supervisor that disables $\mathscr{D}^\dagger$ is defined to be superior to the supervisor that disables $\mathscr{D}^\ddagger$ if $\boldsymbol{\nu}^\dagger \geqq_{\texttt{(Elementwise)}} \boldsymbol{\nu}^\ddagger$ and strictly superior if $\boldsymbol{\nu}^\dagger >_{\texttt{(Elementwise)}} \boldsymbol{\nu}^\ddagger$.*

Definition 2.10: (Optimal Supervision Problem) *Given a (non-terminating) plant $G = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C})$, the problem is to compute a supervisor that disables a subset $\mathscr{D}^\star \subseteq \mathscr{C}$, such that $\boldsymbol{\nu}^\star \geqq_{\texttt{(Elementwise)}} \boldsymbol{\nu}^\dagger\quad \forall \mathscr{D}^\dagger \subseteq \mathscr{C}$ where $\boldsymbol{\nu}^\star$ and $\boldsymbol{\nu}^\dagger$ are the measure vectors of the supervised plants $G^\star$ and $G^\dagger$ under $\mathscr{D}^\star$ and $\mathscr{D}^\dagger$, respectively.*

Definition 2.11: *We note that algorithms reported in [2] compute a lower bound for the critical termination probability for each iteration of such that the disabling/enabling decisions for the terminating plant coincide with the given non-terminating model. We define $\theta_{min} = \min_k \theta_\star^{[k]}$ where $\theta_\star^{[k]}$ is the termination probability computed in the k^{th} iteration.*

Definition 2.12: *If G and $G^\star$ are the unsupervised and supervised PFSA respectively then we denote the renormalized measure of the terminating plant $G^\star(\theta_{min})$ as $\nu_\#^i : 2^{L(q_i)} \to [-1,1]$ (See Definition 2.7). Hence, in vector notation we have: $\boldsymbol{\nu}_\# = \theta_{min}[I - (1-\theta_{min})\Pi^\#]^{-1}\boldsymbol{\chi}$ where $\Pi^\#$ is the transition probability matrix of the supervised plant $G^\star$, we note that $\boldsymbol{\nu}_\# = \nu^{[K]}$ where K is the total number of iterations required for convergence.*

C. Formulating A PFSA Model of Autonomous Navigation

Purely for expositional simplicity, we consider a 2D workspace for the mobile agents. The workspace is first discretized into a finite grid. The underlying theory does not require the grid to be regular; however for the sake of clarity we shall present the formulation under the assumption of a regular grid. The obstacles are represented as blocked-off grid locations in the discretized workspace. We specify a particular location as the fixed goal and consider the problem of finding optimal and feasible paths from arbitrary initial grid locations in the workspace. Figure 1(a) illustrates the basic problem setup. We further assume that at any given time instant the robot occupies

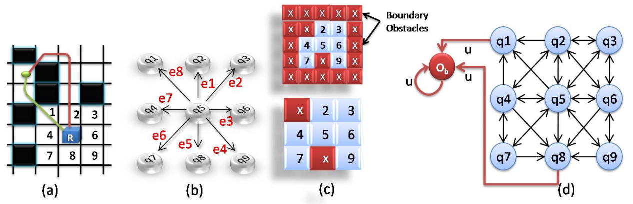


Fig. 1. **(a)** shows the vehicle (marked "R") with the obstacle positions shown as black squares. The green dot identifies the goal **(b)** shows the finite state representation of the possible one-step moves from the current position. **(d)** shows uncontrollable transitions "u" from blocked states

one particular location (*i.e*. a particular square in Figure 1(a)). As shown in Figure 1, the robot has eight possible moves from any interior location. The possible moves are modeled as controllable transitions between grid locations since the robot can "choose" to execute a particular move from the available set. We note that the number of possible moves (8 in this case) depends on the chosen fidelity of discretization of the robot motion and also on the intrinsic vehicle dynamics. The complexity results presented in this paper only assumes that the number of available moves is significantly smaller compared to the number of grid squares, *i.e*., the discretized position states. Specification of inter-grid transitions in this manner allows us to generate a finite state automaton (FSA) description of the navigation problem. Each square in the discretized workspace is modeled as a FSA state with the controllable transitions defining the corresponding state transition map. The formal description of the model is as follows:

Let $\mathbb{G}_{\text{Nav}} = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi)$ be a Probabilistic Finite State Automaton (PFSA). In the absence of dynamic uncertainties and state estimation errors, the alphabet contains one uncontrollable event *i.e*. $\Sigma = \Sigma_C \bigcup \{u\}$ such that Σ_C is the set of controllable events corresponding to the possible moves of the robot. The uncontrollable event u is defined from each of the blocked states and leads to $q_\ominus$ which is a deadlock state. All other transitions (*i.e*. moves) are removed from the blocked states. Thus, if a robot moves into a blocked state, it uncontrollably transitions to the deadlock state $q_\ominus$ which is physically interpreted to be a collision. We further assume that the robot fails to recover from collisions which is reflected by making $q_\ominus$ a deadlock state. We note that $q_\ominus$ does not correspond to any physical grid location. The set of blocked grid locations along with the obstacle state $q_\ominus$ is denoted as $Q_{\text{Obstacle}} \subsetneqq Q$. Figure 1 illustrates the navigation automaton for a nine state discretized workspace with two blocked squares. Next we augment the navigation FSA by specifying event generation probabilities defined by the map $\tilde{\pi} : Q \times \Sigma \rightarrow [0,1]$ and the characteristic state-weight vector specified as $\chi : Q \rightarrow [-1,1]$. The characteristic state-weight vector [2] assigns scalar weights to the PFSA states to capture the desirability of ending up in each state.

Definition 2.13: *The characteristic weights are specified for the navigation automaton as follows:* $\chi(q_i) = \begin{cases} -1 & \text{if } q_i \equiv q_\ominus \\ 1 & \text{if } q_i \text{ is the goal} \\ 0 & \text{otherwise} \end{cases}$

In the absence of dynamic constraints and state estimation uncertainties, the robot can "choose" the particular controllable transition to execute at any grid location. Hence we assume that the probability of generation of controllable events is uniform over the set of moves defined at any particular state.

Definition 2.14: *Since there is no uncontrollable events defined at any of the unblocked states and no controllable events defined at any of the blocked states, we have the following consistent specification of event generation probabilities:* $\forall q_i \in Q, \sigma_j \in \Sigma$, $\tilde{\pi}(q_i, \sigma_j) = \begin{cases} \frac{1}{\text{No. of controllable events at } q_i}, & \text{if } \sigma_j \in \Sigma_C \\ 1, & \text{otherwise} \end{cases}$

The boundaries are handled by "surrounding" the workspace with blocked position states shown as "boundary obstacles" in the upper part of Figure 1(c).

Definition 2.15: *The navigation model id defined to have identical connectivity as far as controllable transitions are concerned implying that every controllable transition or move (i.e. every element of Σ_C) is defined from each of the unblocked states.*

D. Problem Solution as PFSA Optimization

The above-described probabilistic finite state automaton (PFSA) based navigation model allows us to compute optimally feasible path plans via the language-measure-theoretic optimization algorithm [2] described in Section 2. We refer to the language-measure-theoretic algorithm as $\boldsymbol{\nu}^\star$ in the sequel. For the unsupervised model, the robot is free to execute any one of the defined controllable events from any given grid location (See Figure 1(b)). The optimization algorithm selectively disables controllable transitions to ensure that the formal measure vector of the navigation automaton is element-wise maximized. Physically, this implies that the supervised robot is constrained to choose among only the enabled moves at each state such that the probability of collision is minimized with the probability of reaching the goal simultaneously maximized. *Although $\boldsymbol{\nu}^\star$ is based on optimization of probabilistic finite state machines, it is shown that an optimal and feasible path plan can be obtained that is executable in a purely deterministic sense.*

Let $\mathbb{G}_{\text{Nav}}$ be the unsupervised navigation automaton and $\mathbb{G}^\star_{\text{Nav}}$ be the optimally supervised PFSA obtained by $\boldsymbol{\nu}^\star$. We note that $\nu^i_\#$ is the renormalized measure of the terminating plant $\mathbb{G}^\star_{\text{Nav}}(\theta_{min})$ with substochastic event generation probability matrix $\widetilde{\Pi}^{\theta_{min}} = (1-\theta_{min})\widetilde{\Pi}$. Denoting the event generating function (See Definition 2.3) for $\mathbb{G}^\star_{\text{Nav}}$ and $\mathbb{G}^\star_{\text{Nav}}(\theta_{min})$ as $\tilde{\pi} : Q \times \Sigma \rightarrow Q$ and $\tilde{\pi}^{\theta_{min}} : Q \times \Sigma \rightarrow Q$ respectively, we have $\tilde{\pi}^{\theta_{min}}(q_i, \epsilon) = 1$ and $\forall q_i \in Q, \sigma_j \in \Sigma, \tilde{\pi}^{\theta_{min}}(q_i, \sigma_j) = (1-\theta_{min})\tilde{\pi}(q_i, \sigma_j)$

Notation 2.1: *For notational simplicity, we use* $\nu^i_\#(L(q_i)) = \nu_\#(q_i) = \boldsymbol{\nu}_\#|_i$ where $\boldsymbol{\nu}_\# = \theta_{min}[I - (1-\theta_{min})\Pi^\#]^{-1}\chi$

Definition 2.16: ***($\boldsymbol{\nu}^\star$-path:)*** *A $\boldsymbol{\nu}^\star$-path $\rho(q_i, q_j)$ from state $q_i \in Q$ to state $q_j \in Q$ is defined to be an ordered set of PFSA states $\rho = \{q_{r_1}, \cdots, q_{r_M}\}$ with $q_{r_s} \in Q$, $\forall s \in \{1, \cdots, M\}, M \leq \text{Card}(Q)$ such that*

$$q_{r_1} = q_i, \; q_{r_M} = q_j, \; \forall i,j \in \{1,\cdots,M\}, \; q_{r_i} \neq q_{r_j} \tag{1a}$$

$$\forall s \in \{1,\cdots,M\}, \forall t \leqq s, \; \nu_\#(q_{r_t}) \leqq \nu_\#(q_{r_s}) \tag{1b}$$

We reproduce without proof the following key results pertaining to $\boldsymbol{\nu}^\star$- planning as reported in [6].

1) There exists an enabled sequence of transitions from state $q_i \in Q \setminus Q_{\text{Obstacle}}$ to $q_j \in Q \setminus \{q_\ominus\}$ in $\mathbb{G}^\star_{\text{Nav}}$ if and only if there exists a $\boldsymbol{\nu}^\star$-path $\rho(q_i, q_j)$ in $\mathbb{G}^\star_{\text{Nav}}$.
2) For the optimally supervised navigation automaton $\mathbb{G}^\star_{\text{Nav}}$, we have $\forall q_i \in Q \setminus Q_{\text{Obstacle}}, \; L(q_i) \subseteqq \Sigma^\star_C$
3) **(Obstacle Avoidance:)** There exists no $\boldsymbol{\nu}^\star$-path from any unblocked state to any blocked state in the optimally supervised navigation automaton $\mathbb{G}^\star_{\text{Nav}}$.
4) **(Existence of $\boldsymbol{\nu}^\star$-paths:)** There exists a $\boldsymbol{\nu}^\star$-path $\rho(q_i, q_{\text{Goal}})$ from any state $q_i \in Q$ to the goal $q_{\text{Goal}} \in Q$ if and only if $\nu_\#(q_i) > 0$.
5) **(Absence of Local Maxima:)** If there exists a $\boldsymbol{\nu}^\star$-path from $q_i \in Q$ to $q_j \in Q$ and a $\boldsymbol{\nu}^\star$-path from q_i to q_{Goal} then there exists a $\boldsymbol{\nu}^\star$-path from q_j to q_{Goal}, *i.e.*, $\forall q_i, q_j \in Q\Big(\exists \rho_1(q_i, q_{\text{Goal}}) \bigwedge \exists \rho_2(q_i, q_j) \Rightarrow \exists \rho(q_j, q_{\text{Goal}})\Big)$

3. Observation-driven Identification of Obstacle Dynamics

The $\boldsymbol{\nu}^\star$ algorithm assumes a known obstacle map to compute the optimal plan. In this paper, we investigate scenarios where the obstacle map varies with time and solve the problem under the assumption that there exists a finite state symbolic representation of the obstacle dynamics. This requires that the number of possible configurations for the obstacle locations be finitely bounded. Also, if each configuration is mapped to an unique abstract symbol, then the sequence obtained from a discrete observation sequence of the dynamic obstacles, must have bounded memory.

A. Data-driven PFSA Construction Algorithms

The Data-driven PFSA construction approach is built upon the concepts of Symbolic Dynamics [7], Probabilistic Finite State Automata [8], [9] and Pattern Recognition [10] as means to capture the information on fast scale dynamical behavior in terms of symbol sequences. Partitioning the space of process dynamics yields an alphabet which is subsequently used to obtain symbol sequences from time-series data. Among several papered algorithms [9], the following are relevant for this paper:

1) **Causal-State Splitting Reconstruction:** $CSSR$ [11] constructs PFSA which belong to the class of sofic shifts [7] and has an a priori unknown structure; it yields optimal pattern discovery in the sense of mutual information [12].
2) **Symbolic Dynamic Filtering:**SDF [13] constructs PFSA which belong to the more restricted class of shifts of finite type [7] and has an a priori known structure that can be freely chosen.

Both algorithms stated above exhibits remarkable insensitivity to spurious noise and exogenous disturbances due to:

- Inherent "coarse graining" [14] of the time series data in the process of space partitioning [15].
- Very small variance in the estimated parameters of probability distribution, which is a consequence of repeated recurrences of paths in the graph of the finite state machine with a relatively small number of states and a very large number of sample points [1].

It is important to mention here that conventional HMM-based approaches could be sensitive to variations in initial conditions especially if the resulting model is initially off-phase with the current behavior. Also, in Dynamic Bayesian Networks [16], the state machines are hand-coded and the probabilities are estimated with Expectation-Maximization algorithms that are computationally much slower than either SDF or $CSSR$ [9].

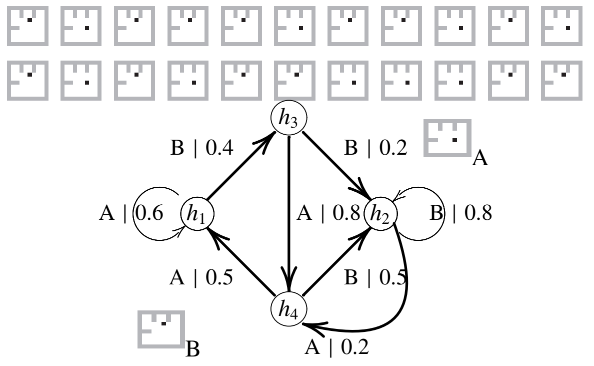


Fig. 2. Symbolic Obstacle dynamics model generation: The left plat illustrates the first few observed obstacle maps with the black spot denoting the moving obstacle and gray denoting static obstacles. White denotes open space. The plate on the right illustrates the constructed $\mathbb{H}_{\mathsf{Obs}}$ which is shown to have 4 states and a two event alphabet {A,B}, where the maps corresponding to A and B are also shown. Note $\mathcal{M}(h_1) = \mathcal{M}(h_4) = \{B\}$ and $\mathcal{M}(h_2) = \mathcal{M}(h_3) = \{A\}$

B. Symbolic Identification of Obstacle Dynamics

Let $\{\mathscr{M}_i\}_{i=1,\dots,N}$ denote the discrete observed sequence of obstacle maps with $\mathscr{M}_i$ being the map observed at the i^{th} observation instant. The observed sequence is symbolized by using Algorithm 1. Note that Algorithm 1 generates both the symbolic sequence $\omega_{\mathscr{M}}$

```
   input  : Model $\{\mathscr{M}_i\}_{i=1,\dots,N}$
   output: Symbolic Sequence $\omega_{\mathscr{M}}$, Alphabet $\Sigma_{\mathscr{M}}$
1  begin
2      Let $\Sigma = \sigma_0$;
3      Assign $\mathscr{M}_1 \mapsto \sigma_0$;
4      Set $k = 1$;
5      for $i = 1 : N$ do
6          for $r = 1 : k$ do
7              if $d(\mathscr{M}_i, \mathscr{M}_r) < T$ then
                   ; /* d: appropriate metric, T: threshold */
8                  Assign $\mathscr{M}_i \mapsto \sigma$ where $\mathscr{M}_r \mapsto \sigma$;
9              else
10                 $\Sigma_{\mathscr{M}} = \Sigma_M \cup \{\sigma_{k+1}\}$; /* Augment alphabet */
11                 k = k+1;
12                 Assign $\mathscr{M}_i \mapsto \sigma_k$;
13
14
15
16 end
```

Algorithm 1: Symbolic Observation Sequence Generation

and the appropriate alphabet $\Sigma_{\mathscr{M}}$. The size of the alphabet is dependent on the value of the classification threshold T (See Line 7 of Algorithm 1). Thus, if the exact configuration space is infinite, we obtain a finite state approximation by using a non-zero threshold over an appropriate metric d. The exact metric is unimportant; if the obstacle maps are represented as matrices, then d can be chosen to be the infinity norm of the matrix difference. The generated symbolic sequence can now be compressed to a PFSA. Note the assumption that $N < \infty$ guarantees that one can always construct a finite state model from $\omega_{\mathscr{M}}$; the trick, however, is to construct one that has a small number of states and yet captures the relevant obstacle dynamics. Here we assume that the sequence length N is large enough for convergence of the PFSA construction algorithms. In the sequel, we will assume that, inspite of the suboptimal output, SDF is used for this PFSA construction. This choice simplifies the approach to a considerable degree, as will be elucidated in the sequel.

Definition 3.1: *The probabilistic obstacle automaton $\mathbb{H}_{\mathsf{Obs}} = (Q_{\mathscr{M}}, \Sigma_{\mathscr{M}}, \delta_{\mathscr{M}}, \widetilde{\Pi}_{\mathscr{M}})$ is defined to be the PFSA obtained via Symbolic Dynamic Filtering (SDF) on the symbolic observation sequence $\omega_{\mathscr{M}}$. As before, $Q_{\mathscr{M}}$ is the set of states, $\Sigma_{\mathscr{M}}$ is the event alphabet, $\delta_{\mathscr{M}} : Q_{\mathscr{M}} \times \Sigma_{\mathscr{M}} \rightarrow Q_{\mathscr{M}}$ is the transition map and $\widetilde{\Pi}_{\mathscr{M}}$ is the event generation probability matrix.*

Lemma 3.1: *Given the comparison metric $d(\cdot,\cdot)$ and the classification threshold T, each state $q_i \in Q_{\mathscr{M}}$ of $\mathbb{H}_{\mathsf{Obs}}$ can be uniquely associated with a set of observed obstacle maps $\mathcal{M}(q_i)$ satisfying*

$$\forall \mathscr{M}_r, \mathscr{M}_s \in \mathcal{M}(q_i), \ d(\mathscr{M}_r, \mathscr{M}_s) < T \tag{2}$$

Proof: SDF output has a special structure which is genrally denoted as D-Markov PFSA models [13] or equivalently as Probabilistic Suffix Automata [9]. PFSA states are right invariant equivalence classes [17] of symbolic sequences; for D-Markov models we have

$$\forall q_i \in Q_{\mathscr{M}}, \exists s, \tau(q_i) \in \Sigma^{\star}_{\mathscr{M}}, \ s.t. \ q_i \equiv \{\omega \in \Sigma^{\star}_{\mathscr{M}} : \omega = s\tau(q_i)\} \tag{3}$$

where $\tau(q_i)$ is a unique string depending on q_i. It follows that each state q_i in $Q_{\mathscr{M}}$ can be uniquely associated with a symbol from $\Sigma_{\mathscr{M}}$, namely the terminating symbol for $\tau(q_i)$. The result then follows from noting that each symbol in $\Sigma_{\mathscr{M}}$ corresponds to a set of observed obstacle maps that satisfy Eq.(2) (See Algorithm 1). ❒

Each observed obstacle map $\mathscr{M}_i$ is a specification of the obstacle locations at the corresponding observation instant and therefore is a specification of the characteristic weights on the states of the navigation automaton (See Definition 2.13) at that instant. Let for each $\mathscr{M}_i$, the corresponding characteristic weight vector be denoted as $\chi^{\mathscr{M}_i}$. Once $\mathbb{H}_{\mathsf{Obs}}$ is constructed, we are required to assign the state characteristic weights, which, in light of Lemma 3.1, can be done easily as follows:

Definition 3.2: *The characteristic weights on the states of $\mathbb{H}_{\mathsf{OBS}}$ is defined as follows:*

$$\forall h_i \in Q_{\mathscr{M}}, \chi(h_i) = \frac{1}{\textsc{Card}(\mathcal{M}(h_i))} \sum_{\mathscr{M}_r \in \mathcal{M}(h_i)} \chi^{\mathscr{M}_r} \tag{4}$$

where $\mathcal{M}(h_i)$ is as defined in Eq. (2).
Thus, each $h_i \in Q_{\mathscr{M}}$ can be associated with a single symbol (Lemma 3.1), and the characteristic of h_i quantifies an average effect of all obstacle maps that are identified to this symbol by Algorithm 1. The procedure is illustrated in Figure 2, where we end up with a 4 state $\mathbb{H}_{\mathsf{OBS}}$ and $\mathcal{M}(h_1) = \mathcal{M}(h_4) = \{B\}$ and $\mathcal{M}(h_2) = \mathcal{M}(h_3) = \{A\}$, where A, B are the two distinct obstacle maps observed. Next we integrate the constructed $\mathbb{H}_{\mathsf{OBS}}$ and the navigation automaton $\mathbb{G}_{\mathsf{NAV}}$ to obtain the overall description of estimated obstacle dynamics and navigation constraints and transitions in the PFSA framework.

4. Integration of Navigation Automaton & Obstacle Automaton

We note that the event alphabet Σ defined by $\mathbb{G}_{\mathsf{NAV}}$ consists of controllable transition (except the collision transition u), while $\Sigma_{\mathscr{M}}$ in $\mathbb{H}_{\mathsf{OBS}}$ consists of uncontrollable transitions that capture the obstacle dynamics. Also, the generation probabilities (specified by $\widetilde{\Pi}_{\mathscr{M}}$) of the events in $\mathbb{H}_{\mathsf{OBS}}$ must be preserved in the integrated model, implying that a simple product construction would not suffice. We employ the a input-output automata [18] based approach to define the syntactics of the integration. Namely, we define the states of the integrated model to be of the type (q, h), where the first coordinate specifies the current state in the underlying navigation model. The second coordinate relates to the state of the obstacle dynamics. However, we need two distinct states $(q, h), (q, h^\circ)$ for the same underlying navigation state q and the same state h of the obstacle dynamics to capture the required segregation of navigation events and obstacle-dynamics events. Thus for the state set $Q^{\otimes}$ of the integrated model, we have $\textsc{Card}(Q^{\otimes}) = 2\textsc{Card}(Q)\textsc{Card}(Q_{\mathscr{M}})$. It is important to note that the obstacle locations in the modified model are specified by the individual obstacle maps which evolve according to $\mathbb{H}_{\mathsf{OBS}}$. Hence, the navigation automaton $\mathbb{G}_{\mathsf{NAV}} = (Q, \Sigma, \delta, \widetilde{\Pi}, \chi, \mathscr{C})$ no longer needs to have the obstacle state $q_{\ominus}$. Hence, in the sequel, $\mathbb{G}_{\mathsf{NAV}}$ is assumed to represent purely the navigation constraints and the uncontrollable collision event u is eliminated from Σ. Also, note that this implies every event in Σ is now controllable. The formal consruction is presented next:

Definition 4.1: *The integrated model denoted as $\mathbb{G}_{\mathsf{NAV}} \otimes \mathbb{H}_{\mathsf{OBS}} \triangleq (Q^{\otimes}, \Sigma^{\otimes}, \delta^{\otimes}, \widetilde{\Pi}^{\otimes}, \chi^{\otimes}, \mathscr{C}^{\otimes})$ is constructed from the navigation automaton $\mathbb{G}_{\mathsf{NAV}}$ and the obstacle automaton $\mathbb{H}_{\mathsf{OBS}}$ by the following relations. Note that $Q^\circ_{\mathscr{M}}$ is an isomorphic copy of $Q_{\mathscr{M}}$ and h°_j is the image of h_j in the isomorphic copy.*

(1) $Q^{\otimes} = (Q \times Q_{\mathscr{M}}) \bigcup \left(Q \times Q^\circ_{\mathscr{M}}\right)$

(2) $\Sigma^{\otimes} = \Sigma \bigcup \Sigma_{\mathscr{M}}$

(3) $\forall q_i \in Q, h_j \in Q_{\mathscr{M}},\ \delta^{\otimes}((q_i, h_j), \sigma) = \begin{cases} (\delta(q_i, \sigma), h^\circ_j) & \text{, if } \sigma \in \Sigma \\ \text{Undefnd.} & \text{, otherwise} \end{cases}$

(4) $\forall q_i \in Q, h^\circ_j \in Q^\circ_{\mathscr{M}}$,
$\delta^{\otimes}((q_i, h^\circ_j), \sigma) = \begin{cases} (q_i, \delta_{\mathscr{M}}(h_j, \sigma)) & \text{, if } \sigma \in \Sigma_{\mathscr{M}} \\ \text{Undefnd.} & \text{, otherwise} \end{cases}$

(5) $\forall q_i \in Q, h_j \in Q_{\mathscr{M}},\ \widetilde{\Pi}^{\otimes}((q_i, h_j), \sigma) = \begin{cases} \widetilde{\Pi}(q_i, \sigma) & \text{, if } \sigma \in \Sigma \\ 0 & \text{, otherwise} \end{cases}$

(6) $\forall q_i \in Q, h^\circ_j \in Q^\circ_{\mathscr{M}}, \widetilde{\Pi}^{\otimes}((q_i, h^\circ_j), \sigma) = \begin{cases} \widetilde{\Pi}_{\mathscr{M}}(h_j, \sigma) & \text{, if } \sigma \in \Sigma_{\mathscr{M}} \\ 0 & \text{, otherwise} \end{cases}$

(7) $\forall q_i \in Q, h_j \in Q_{\mathscr{M}},\ \chi^{\otimes}((q_i, h_j)) = 0$

(8) $\forall q_i \in Q, h^\circ_j \in Q^\circ_{\mathscr{M}},\ \chi^{\otimes}((q_i, h^\circ_j)) = \xi(\chi_{\mathscr{M}}(h_j), \chi(q_i))$

where $\xi : [0, 1]^2 \to [0, 1]$ is defined as $\xi(a, b) = \begin{cases} a & \text{, if } b = 0 \\ b & \text{, otherwise} \end{cases}$

(9) *$\sigma \in \Sigma$ are controllable; $\sigma \in \Sigma_{\mathscr{M}}$ are uncontrollable*

It follows that $\chi^{\otimes}(q, h^\circ)$ is -1 if the location denoted by q is blocked in the obstacle map currently active, and $\chi^{\otimes}(q, h^\circ)$ is 1, *i.e.*, (q, h°) is a goal if q is the goal in the underlying navigation automaton and it is not currently blocked. It is interesting to note that in the trivial case of $\textsc{Card}(Q_{\mathscr{M}}) = 1$, *i.e.*, where the obstacles are static, we obtain twice the number of states as compared to $\mathbb{G}_{\mathsf{NAV}}$. However, it can be easily shown, that the integrated model obtained in this special case is in fact a non-minimal realization [17] of $\mathbb{G}_{\mathsf{NAV}}$.

A. $\nu^\star$-Optimization of the Integrated PFSA

The integrated PFSA model $\mathbb{G}_{\mathsf{NAV}} \otimes \mathbb{H}_{\mathsf{OBS}}$ can be optimized to obtain optimal plans in a straightforward manner by using the $\nu^\star$ planning algorithm. The semantics of the plan plan execution is however, somewhat different. Algorithm 2 summarizes the overall approach. We note that the computed optimal measure vector $\nu_{\#}$ induces

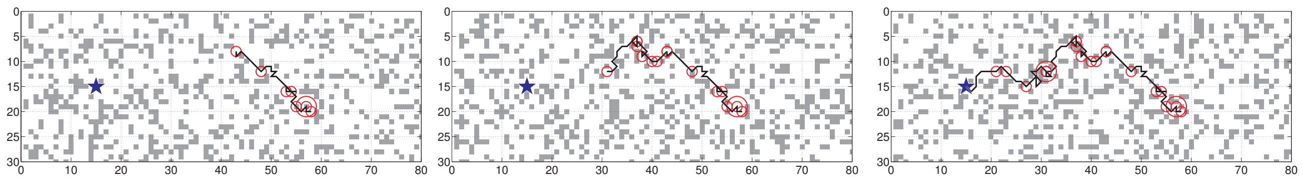
```
input : Observation sequence $\{\mathscr{M}_i\}_{i=1,\dots,N}$
output: Optimal plan and execution
1  begin
2      Compute $\mathbb{H}_{\mathsf{OBS}}$, $\mathbb{G}_{\mathsf{NAV}}$, $\mathbb{G}_{\mathsf{NAV}} \otimes \mathbb{H}_{\mathsf{OBS}}$;
3      $\nu^\star$-Optimize $\mathbb{G}_{\mathsf{NAV}} \otimes \mathbb{H}_{\mathsf{OBS}}$ and obtain $\nu_{\#}$;
4      Set initial state $(q, h)$ ;
5      while GOAL_NOT_REACHED do
6          Compute $W$ = {set of reachable states from $(q, h)$};
7          Compute $(q_{next}, h^\circ) = \underset{w \in W}{argmax}\, \nu_{\#}(w \in W)$;
8          Move to $(q_{next}, h^\circ)$; /* Controllable Move          */
9          Set $q = q_{next}$;
10         Estimate current $\mathbb{H}_{\mathsf{OBS}}$ state $h_{next}$;
11         Move to $(q, h_{next})$; /* Uncontrollable Move          */
12         set $h = h_{next}$;
13
14 end
```

Algorithm 2: Plan computation & Execution

a gradient on the state space $Q^{\otimes}$, which the robot can follow to reach the goal. Thus the computed plan is not probabilistic; given the current state, the plan deterministically specifies the next move. In the case of static obstacles [6], it was shown that the planned path never collides. However, in the generalized situation, this is not guaranteed deterministically. Due to the symbolic compression of the obstacle dynamics, there is always a small probability that the obstacle appears where it is not generally expected, resulting in a small probability of collision. The key point to be noted in Algorithm 2, is the alternating sequence of controllable and uncontrollable moves. The robot can decide which controllable move to execute; but not how the obstacle map will evolve. However, incorporation of the symbolic dynamic model of obstacle evolution in the planning phase implies that the robot can decide upon the best controllable move at any point. Also, there is the implicit assumption that the time scales of the obstacle dynamics is slower compared to the move execution time scale for the robot. This is critical, since, if the obstacle map switches faster than than the robot can move, then the state estimates will be erroneous leading to incorrect execution. We have the following result:

Proposition 4.1: *The proposed planning and execution approach solves the following optimization problem: Maximize $\Psi = (\wp_{\textsc{Goal}} - \wp_{\textsc{Obstacle}})$ under the navigation constraints and observed history of obstacle dynamics, where $\wp_{GOAL}$ is the probability of reaching the goal and $\wp_{OBS}$ is the probability of hitting an obstacle.*

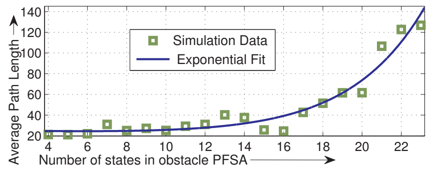
Proof: We recall that the language-measure-theoretic optimization of PFSA accomplishes the maximization of $\wp^T\chi$ where $\wp$ is the stationary probability vector on the automaton states [2]. Since $\chi(\textsc{Goal}) = 1$ and $\chi(\textsc{Obstacle}) = -1$ and all other states have zero characteristic, it follows that Ψ gets maximized in the optimization. ❐



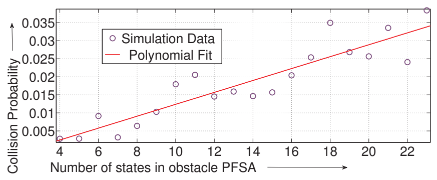
(a) (b) (c)

Fig. 3. Simulated execution on a 30×80 dynamic maze. The plates (a) - (c) are snapshots at different times. Note that the obstacle map is different in the three cases. The red circles indicate locations where the simulated robot decided to wait for one or more ticks as the obstacles continued to evolve

B. Simulation Examples on Dynamic Mazes



(a)



(b)

Fig. 4. Effect of increasing complexity of obstacle dynamics: (a) shows the exponential increase in average path lengths to reach goal, (b) shows linear increase of collision probability

The proposed algorithm is validated on simulated models. Figure 3 illustrates one such simulation run on a 30×80 grid. Plates (a) - (c) capture snapshots of the plan execution at different times. Note that the obstacle map (gray for blocked locations; white for open spaces) is different in the three plates. Also note, that the executed path is significantly rough. However, the optimization result of Proposition 4.1 guarantees that the executed path indeed is optimal. The obstacle automaton in this case has 10 events and 100 states. The red circles in Figure 3 indicates where the simulated robot waited for one or more ticks, *i.e.*, decided not to execute any controllable move. Note that while it is futile to "wait" at any location for a static obstacle map; it is often the optimal course of action when the obstacles are dynamic.

We simulate the effect of increasing complexity of obstacle dynamics by considering $\mathbb{H}_{\textrm{Obs}}$ with larger and larger number of states. The results are shown in Figure 4(a)-(b). Note that while the collision probability increases only linearly with increased complexity of the obstacle dynamics, the average path length required to reach the goal increases exponentially.

5. Summary & Future Research

The $\nu^\star$ planning algorithm is augmented to handle dynamic obstacles. The obstacle dynamics is first encoded symbolically from a finite but sufficiently long observation sequence as a probabilistic finite state machine. This symbolic dynamic model of the obstacles is then integrated with the navigation automaton and is subsequently optimized by the language-measure theoretic approach. It is shown that the proposed approach maximizes the difference in the probability of reaching the goal and the probability of hitting an obstacle and is optimal in that sense. Future work will extend the language-measure theoretic planning algorithm to address the following problems:

1) **Multi-robot coordinated planning:** Future work will address multi-robot scenarios, with each robot treating the remaining group as moving obstacles.
2) **Hierarchical implementation to handle very large workspaces:** Large workspaces can be solved more efficiently if planning is done when needed rather than solving the whole problem at once.
3) **Handling partially observable dynamic events:** Physical errors and onboard sensor failures may need to be modeled as unobservable transitions and will be addressed in future publications.

References

[1] I. Chattopadhyay and A. Ray, "Renormalized measure of regular languages," *Int. J. Control*, vol. 79, no. 9, pp. 1107–1117, 2006.

[2] I. Chattopadhyay and A. Ray, "Language-measure-theoretic optimal control of probabilistic finite-state systems," *Int. J. Control*, August,2007.

[3] J. Barraquand, B. Langlois, and J.-C. Latombe, *Robot motion planning with many degrees of freedom and dynamic constraints*. Cambridge, MA, USA: MIT Press, 1990.

[4] V. Garg, "An algebraic approach to modeling probabilistic discrete event systems," *Proceedings of 1992 IEEE Conference on Decision and Control*, pp. 2348–2353, Tucson, AZ, December 1992.

[5] W. Rudin, *Real and Complex Analysis, 3rd ed.* McGraw Hill, New York, 1988.

[6] I. Chattopadhyay, G. Mallapragada, and A. Ray, "$\nu^\star$: a robot path planning algorithm based on renormalized measure of probabilistic regular languages," *International Journal of Control*, in press.

[7] D. Lind and M. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, United Kingdom, 1995.

[8] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation, 2nd ed.* Addison-Wesley, 2001.

[9] K. Murphy, "Passively learning finite automata," 1996.

[10] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley & Sons Inc., 2001.

[11] C. R. Shalizi, K. L. Shalizi, and J. P. Crutchfield, "An algorithm for pattern discovery in time series," *Technical Report, Santa Fe Institute*, October 2002.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, New York, 1991.

[13] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, vol. 84, no. 7, pp. 1115–1130, 2004.

[14] C. Beck and F. Schlogl, *Thermodynamics of chaotic systems: an introduction*. Cambridge University Press, United Kingdom, 1993.

[15] V. Rajagopalan and A.Ray, "Symbolic time series analysis via wavelet-based partitioning," *Signal Processing*, vol. 86, no. 11, pp. 3309–3320, 2006.

[16] E. Denis, N.; Jones, "Spatio-temporal pattern detection using dynamic bayesian networks," *In Proceedings of 42nd IEEE Conference on Decision and Control*, vol. 5, pp. 4533–4538, 9-12 Dec. 2003.

[17] I. Chattopadhyay and A. Ray, "Structural transformations of probabilistic finite state machines," *International Journal of Control*, vol. 81, pp. 820–835, May 2008.

[18] N. Lynch and M. Tuttle, "An introduction to input/output automata," Technical Memo MIT/LCS/TM-373, Massachusetts Institute of Technology, November 1988.