L*: An Intelligent Path Planning Algorithm based on Renormalized Measure of Probabilistic Regular Languages

Ishanu Chattopadhyay  
ixc128@psu.edu

Goutham Mallapragada  
grm150@psu.edu

Asok Ray  
axr2@psu.edu

The Pennsylvania State University
University Park, PA 16802, USA

Abstract—A novel path planning algorithm L* is introduced that reduces the problem to optimization of a probabilistic finite state machine and applies the rigorous theory of language-measure-theoretic optimal control to compute \( \gamma \)-optimal paths to the specified goal. It is shown that although the underlying navigation model is probabilistic, the proposed algorithm computes plans that can be executed in a deterministic sense with automated optimal trade-off between path length and robustness under dynamic uncertainty. The algorithm has been validated on mobile robotic platforms in a laboratory environment.

Index Terms—Language Measure; Finite Markov Chains; Discrete Event Systems; Supervisory Control

1. INTRODUCTION & MOTIVATION

The field of trajectory and motion planning is enormous, with applications in such diverse areas as industrial robots, mobile robot navigation, spacecraft reentry, video games and even drug design. In the context of planning for mobile robots and manipulators much of the literature on path and motion planning is concerned with finding collision-free trajectories [7]. Graph-based techniques have been used very successfully in many wheeled ground robot path planning problems and have been used for some UAV planning problems, typically radar evasion [1]. These approaches suffer from exponential growth in complexity with both graph size and dimension. To circumvent the complexity associated with graph-based planning, sampling based planning methods [2] such as probabilistic roadmaps have become widespread. These approaches are probabilistically complete (i.e. if a feasible solution exists it will be found, given enough time) but there is no guarantee of finding a solution within a specified time. This paper introduces a novel approach to path-planning 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. The key advantages are as follows:

1) \( L^* \) pre-processing is cheap: Conventional cellular approaches decompose the set of free configurations into simple non-overlapping regions [7][8] with the adjacency relations and reachability constraints represented in a connectivity graph which is subsequently searched for a path. Construction of the connectivity graph that faithfully reflects the continuous C-Space constraints (e.g. obstacles) is expensive and often intractable for high dimensional problems. The decomposition required by \( L^* \) is simple and computationally cheap. The cells are mapped to PFSA states which are defined to have identical connectivity via symbolic inter-state transitions. Desirable or "good" states assigned relatively positive weights to undesirable or "bad" states. A change in the C-Space constraints therefore requires updating the 1D state weight vector only; recomputing the decomposition is unnecessary.

2) \( L^* \) is fundamentally different from a search: \( L^* \) replaces the "search" problem to solution of a sequence of linear systems (i.e. matrix inversions). On completion of cellular decompostion, \( L^* \) optimizes the resultant PFSA via a iterative sequence of combinatorial operations which elementwise maximizes the language measure vector [4][5]. Note that although \( L^* \) involves probabilistic reasoning, the final waypoint sequence obtained is deterministic.

3) Computational efficiency: The intensive step in \( L^* \) is the inversion of certain specialized matrices to compute the language measure. 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 [2]. However, \( L^* \) 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 \( L^* \).

The paper is organized in five sections including the present one. Section 2 briefly explains the language-theoretic models considered in this paper and reviews the language-measure-theoretic optimal control of probabilistic finite state machines. Section 3 presents the basic problem formulation with Section 4 deriving a decision-theoretic solution to the path-planning problem. 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 [9]. Let \( G = \langle Q, \Sigma, \delta, q_0, Q_W \rangle \) be a trim (i.e., accessible and co-accessible) finite-state automaton model that represents the discrete-event dynamics of a physical plant, where \( Q = \{q_k : k \in I_Q\} \) is the set of states and \( I_Q = \{1, 2, \cdots, n \} \) is the index set of states; the automaton starts with the initial state \( q_0 \); the alphabet of events is \( \Sigma = \{\sigma_k : k \in I_\Sigma\} \), having \( \Sigma \cap I_Q = \emptyset \) and \( I_\Sigma = \{1, 2, \cdots, \ell \} \) is the index set of events; \( \delta : Q \times \Sigma \rightarrow Q \)
is the (possibly partial) function of state transitions; and 
\( Q_m \equiv \{ q_{m1}, q_{m2}, \ldots, q_{ml} \} \subseteq Q \) is the set of marked (i.e., accepted) states with \( q_{mj} = q_j \) for some \( j \in I_Q \). Let \( \Sigma' \) be the Kleene closure of \( \Sigma \), i.e., the set of all finite-length strings made of the events belonging to \( \Sigma \) as well as the empty string \( e \) that is viewed as the identity of the monoid \( \Sigma' \) under the operation of string concatenation, i.e., \( es = s = se \).

The state transition map \( \delta \) is recursively extended to its reflexive and transitive closure \( \delta : Q \times \Sigma' \to Q \) by defining \( \forall q_j \in Q, \delta(q_j, e) = q_j \) and \( \forall q_j \in Q, o \in \Sigma, s \in \Sigma^* \), \( \delta(q_j, os) = \delta(\delta(q_j, o), s) \).

**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: \( L_m(q_i) = \{ s \in \Sigma^+ \mid \delta^*(q_i, s) \in Q_m \} \).

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

The formal language measure is first defined for terminating plants \([6]\) 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 \( \pi : \Sigma^* \times \rightarrow [0, 1] \) such that \( \forall q_j \in Q, \forall o_j \in \Sigma, s_j \in \Sigma^* \),

1. \( \pi(o_j, q_j) \equiv \pi_{o_j} \notin [0, 1], \sum_{o_j} \pi_{o_j} = 1 - \theta, \) with \( \theta \in (0, 1) \);
2. \( \pi(o_j, q_j) = 0 \) if \( \delta(q_j, o) \) is undefined, \( \pi(e, q_j) = 1 \);
3. \( \pi(o_j, s_j) = (\pi_{o_j, q_j}) \forall (o_j, s_j, q_j) \).

The \( n \times f \) event cost matrix is defined as: \( \Pi_{ij} = \pi(q_i, o_j) \).

**Definition 2.4:** The state transition probability matrix is defined as: \( \Pi_{ik} = \pi(q_i, o_j) \).

The set \( Q_m \) of marked states is partitioned into \( Q_{m1} \) and \( Q_{m2} \), i.e., \( Q_m = Q_{m1} \cup Q_{m2} \) and \( Q_{m1} \cap Q_{m2} = \emptyset \), where \( Q_{m1} \) contains all good marked states that we desire to reach, and \( Q_{m2} \) 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 real value based on its 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 \( L(q_i, q_j) \) is defined as:

\[
\forall q_j \in Q, \quad \chi(q_j) = \begin{cases} 
1, & q_j \in Q_{m1} \\
0, & q_j \notin Q_m \\
-1, & q_j \in Q_{m2}
\end{cases}
\]

The state weighting vector, denoted by \( \chi = [\chi_1 \chi_2 \cdots \chi_n]^T \), where \( \chi_j \equiv \chi(q_j) \forall j \in I_Q \), is called the \( \chi \)-vector. The \( j \)-th element \( \chi_j \) of \( \chi \)-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 \( Q_{m1} \) and \( Q_{m2} \) respectively. Any event string belonging to the language \( L_m = L(q_i) - L_m(q_i) \) 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:

\[
L(q_i) = \bigcup_{q_{mk} \in Q_m} L_{ik} \quad \text{and} \quad L_m(q_i) = \bigcup_{q_{mk} \in Q_m} L_{ik} \cup L^0
\]

where the sublanguage \( L_{ik} \subseteq G_i \) having the initial state \( q_i \) uniquely labelled by the terminal state \( q_j \), \( j \in I_Q \), and \( L_{ij} \cap L_{ik} = \emptyset \) if \( j \neq k \), and \( L_{im} \equiv \bigcup_{q_{mk} \in Q_m} L_{ik} \) and \( L_{m} \equiv \bigcup_{q_{mk} \in Q_m} L_{ik} \) are good and bad sublanguages of \( L_m(q_i) \), respectively. Then, \( L^0 = \bigcup_{q_{mk} \in Q_m} L_{ik} \) and \( L(q_i) = L^0 \cup L^+_m \cup L_m^- \).

A signed real measure \( \mu : 2^{I_Q} \to R \equiv (-\infty, +\infty) \) is constructed on the \( \sigma \)-algebra \( 2^{I_Q} \) for any \( i \in I_Q \); interested readers are referred to [9] for the details of measure-theoretic definitions and results. With the choice of this \( \sigma \)-algebra, every singleton set made of an event string \( s \in L(q_i) \) is a measurable set. By Hahn Decomposition Theorem [10], each of these measurable sets qualifies itself to have a numerical value based on the above state-based decomposition of \( L(q_i) \) into \( L_0^\{\text{null}^\} \), \( L^\{\text{positive}^\} \), and \( L^\{\text{negative}^\} \) sublanguages.

**Definition 2.6:** Let \( \omega \in L(q_i, q_j) \subseteq 2^{I_Q} \). The signed real measure \( \mu' : 2^\{\text{singleton}^\} \to R \equiv (-\infty, +\infty) \) is constructed on the \( \sigma \)-algebra \( 2^\{\text{singleton}^\} \) for any \( i \in I_Q \); interested readers are referred to [9] for the details of measure-theoretic definitions and results. With the choice of this \( \sigma \)-algebra, every singleton set made of an event string \( s \in L(q_i) \) is a measurable set. By Hahn Decomposition Theorem [10], each of these measurable sets qualifies itself to have a numerical value based on the above state-based decomposition of \( L(q_i) \) into \( L_0^\{\text{null}^\} \), \( L^\{\text{positive}^\} \), and \( L^\{\text{negative}^\} \) sublanguages.

**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 \( \chi \), the models can be completely specified by a sextuple as: \( G = (Q, \Sigma, \delta, \Pi, \chi, \epsilon) \).

**Definition 2.8:** (Control Philosophy) If \( q_i \to q_0 \) and the event \( \sigma \) is disabled at state \( q_0 \), then the supervisory action is to prevent the plant from making a transition to the state \( q_0 \) by forcing it to stay at the original state \( q_i \). Thus disabling any transition \( \sigma \) 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 \( \sigma \) 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 \( \epsilon \). Note controllability is state-based.

**B. The Optimal Supervision Problem: Formulation & Solution**

A supervisor disables a subset of the set \( \epsilon \) of controllable transitions and hence there is a bijection between the set of all possible supervision policies and the power set \( 2^\epsilon \). That is, there exists \( 2^\epsilon \) possible supervisors and each supervisor is uniquely identifiable with a subset of \( \epsilon \) and the language measure \( \nu \) allows a quantitative comparison of different policies.

**Definition 2.9:** For an unsupervised plant \( G = (Q, \Sigma, \delta, \Pi, \chi, \epsilon) \), let \( G^\epsilon \) and \( G^\epsilon \) be the supervised plants with sets of disabled transitions, \( \mathcal{P}^\epsilon \subseteq \epsilon \) and \( \mathcal{P}^\epsilon \subseteq \epsilon \), respectively, whose measures are \( \nu^\epsilon \) and \( \nu^\epsilon \). Then, the supervisor that disables \( \mathcal{P}^\epsilon \) is defined to be superior to the supervisor that disables \( \mathcal{P}^\epsilon \) if \( \nu^\epsilon \geq (\text{Elementwise}) \nu^\epsilon \) and strictly superior if \( \nu^\epsilon > (\text{Elementwise}) \nu^\epsilon \).

**Definition 2.10:** (Optimal Supervision Problem) Given a (non-terminating) plant \( G = (Q, \Sigma, \delta, \Pi, \chi, \epsilon) \), the problem is to compute a supervisor that disables a subset \( \mathcal{P}^\epsilon \subseteq \epsilon \), such that \( \nu^\epsilon \geq (\text{Elementwise}) \nu^\epsilon \) \( \forall \mathcal{P}^\epsilon \subseteq \epsilon \) where \( \nu^\epsilon \) and \( \nu^\epsilon \) are the measure vectors of the supervised plants \( G^\epsilon \) and \( G^\epsilon \) under \( \mathcal{P}^\epsilon \) and \( \mathcal{P}^\epsilon \), respectively.

**Remark 2.1:** The solution to the optimal supervision problem is obtained in [5], [3] by designing an optimal policy for a terminating plant [6] with a sub-stochastic transition probability matrix \((1 - \theta)\Pi \) with \( \theta \in (0, 1) \). To ensure that the computed optimal policy coincides with the one for \( \theta = 0 \), the suggested algorithm chooses a small value for \( \theta \) in each iteration step of the design algorithm. However, choosing \( \theta \) too small may cause numerical problems in convergence. Algorithms reported in [5], [3] computes how small a \( \theta \) is actually required, i.e., computes the critical lower bound \( \theta_{\text{cr}} \), thus solving the optimal supervision problem for a generic PFSA. It is further shown that the solution obtained is optimal and unique and can be computed by an effective algorithm.

**Definition 2.11:** Following Remark 2.1, we note that algorithms reported in [5], [3] 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 \( \theta_{\text{min}} = \min_i \theta^{(i)} \) where \( \theta^{(i)} \) is the termination probability computed in the \( i^{\text{th}} \) iteration.

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


![Fig. 1](image_url) (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. (c) shows uncontrollable transitions "u" from states corresponding to blocked grid locations to "q0".

We consider a 2D workspace for the mobile agents. This restriction on workspace dimensionality serves to simplify the exposition and can be easily relaxed. To set up the problem, the workspace is first discretized into a finite grid and hence the approach developed in this paper falls under the generic category of discrete planning. 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 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 boundaries are handled by removing the moves that take the robot out of the workspace. 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 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 \( G_{\text{NAV}} = (Q, \Sigma, \delta, \Pi, \chi) \) be a Probabilistic Finite State Automaton (PFSA). The state set \( Q \) consists of states that correspond to grid locations and one extra state denoted...
by \( q_0 \). The necessity of this special state \( q_0 \) is explained in the sequel. The grid squares are numbered in a predetermined scheme such that each \( q_i \in Q \setminus \{q_0\} \) denotes a specific square in the discretized workspace. The particular numbering scheme chosen is irrelevant. In the absence of dynamic uncertainties and state estimation errors, the alphabet contains one uncontrollable event \( i.e. \Sigma = \Sigma_c \cup \{u\} \) such that \( \Sigma_c \) 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 \( q_0 \) 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 \( q_0 \) which is physically interpreted to be a collision. We further assume that the robot fails to recover from collisions which is reflected by making \( q_0 \) a deadlock state. We note that \( q_0 \) does not correspond to any physical particular state.

\[ \delta^\#(q_j, u) = q_0 \quad (\text{if defined}) \]  
\[ \delta^\#(q_0, u) = q_0 \quad (\text{Self-loop}) \] (7)

The navigation model is defined to have a \( \text{PFSA} \) based navigation model allows us to compute:

[Events is uniform over the set of moves defined at any of the unblocked states and no controllable events are removed from the blocked states. Thus, if a robot moves 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 elementwise 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 \( L^* \) 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 \( G_{\text{Nav}} \) be the unsupervised navigation automaton and \( G^*_{\text{Nav}} \) be the optimally supervisedPFSA obtained by \( L^* \). We note that \( v^i_\theta \) is the normalized probability of the terminating plant \( G^*_{\text{Nav}}(\Theta_{\min}) \) with substochastic event generation probability matrix \( \Pi^\# \). Denoting the event generation function (See Definition 2.3) for \( G_{\text{Nav}} \) and \( G^*_{\text{Nav}}(\Theta_{\min}) \) as \( \tilde{\pi} : Q \times \Sigma \rightarrow [0,1] \) and the characteristic state-weight vector specified as \( \chi : Q \rightarrow [-1,1] \). The characteristic state-weight vector \( \chi \) assigns scalar weights to the PFSA states to capture the desirability of ending up in each state.

**Definition 3.1:** The characteristic weights are specified for the navigation automaton as follows:

\[ \chi(q_i) = \begin{cases} 
-1 & \text{if } q_i = q_0 \\
1 & \text{if } q_i \text{ is the goal} \\
0 & \text{otherwise} 
\end{cases} \] (3)

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 3.2:** Since there is no uncontrollable events defined at any of the unblocked states and no uncontrollable events defined at any of the blocked states, we have the following consistent specification of event generation probabilities: \( \forall q_i \in Q, \sigma_j \in \Sigma_c \)

\[ \tilde{\pi}(q_i, \sigma_j) = \begin{cases} 
\frac{1}{\text{No. of controllable events at } q_i} & \text{if } \sigma_j \in \Sigma_c \\
1 & \text{otherwise} 
\end{cases} \]

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

**Definition 3.3:** 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 \( \Sigma_c \) is defined from each of the unblocked states.

4. PROBLEM SOLUTION AS A DECISION-THEORETIC OPTIMIZATION OF PFSA

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 \( [5] \) described in Section 2. Keeping in line with nomenclature in the path-planning literature, we refer to the language-measure-theoretic algorithm as \( L^* \) 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 elementwise 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 \( L^* \) 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 \( G_{\text{Nav}} \) be the unsupervised navigation automaton and \( G^*_{\text{Nav}} \) be the optimally supervisedPFSA obtained by \( L^* \). We note that \( v^i_\theta \) is the normalized probability of the terminating plant \( G^*_{\text{Nav}}(\Theta_{\min}) \) with substochastic event generation probability matrix \( \Pi^\# \). Denoting the event generation function (See Definition 2.3) for \( G_{\text{Nav}} \) and \( G^*_{\text{Nav}}(\Theta_{\min}) \) as \( \tilde{\pi} : Q \times \Sigma \rightarrow [0,1] \) and \( \tilde{\pi}^\#: Q \times \Sigma \rightarrow [0,1] \) respectively, we have

\[ \tilde{\pi}^\#(q_i, e) = 1 \]

\[ v_i \in Q, \sigma_j \in \Sigma_c, \tilde{\pi}^\#(q_i, \sigma_j) = (1 - \Theta_{\min}) \tilde{\pi}(q_i, \sigma_j) \]

**Notation 4.1:** For notational simplicity, we use

\[ v_i^s(L(q_i)) = v_i(q_i) = v_i \]

where \( v_s = \Theta_{\min} [1 - (1 - \Theta_{\min}) \Pi^\#]^r \chi \)

**Definition 4.1:** \( (L^*-\text{path}) \) A \( L^* \)-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 \( \rho = \{q_i, \ldots, q_{n-1}, q_n\} \) with \( q_n \in Q \), \( \forall s \in \{1, \ldots, M\}, M \leq \text{CARD}(Q) \) such that

\[ q_1 = q_i \]

\[ q_j = q_i \]

\[ q_{j+1} \neq q_j \]

\[ q_{j+1} \neq q_i \]

\[ q_{j+1} \neq q_s \]

\[ q_j = q_i \]

\[ q_j = q_i \]

\[ q_j = q_i \]

\[ q_j = q_i \]

\[ q_j = q_i \]

\[ q_j = q_i \]

**Lemma 4.1:** There exists an enabled sequence of transitions from state \( q_i \in Q \setminus Q_{\text{OBSTACLE}} \) to \( q_j \in Q \setminus Q_{\text{OBSTACLE}} \) in \( G^*_{\text{Nav}} \) if and only if there exists a \( L^* \)-path \( \rho(q_i, q_j) \) in \( G^*_{\text{Nav}} \).

**Proof:** Let \( q_i \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \cdots \xrightarrow{\sigma_k} q_j \) be an enabled sequence of transitions in \( G^*_{\text{Nav}} \) where \( \ell \in \{1, \ldots, M\} \) for some \( M \subset \text{CARD}(Q) \). We note from the defined structure of the navigation automaton that there is no transition \( q_i \xrightarrow{\sigma} q_i \) where \( q_i \in Q_{\text{OBSTACLE}}, \sigma \neq q_i \). It follows that \( \forall i \in \{1, \ldots, M\}, q_i \neq q_{i+1} \). This in turn implies that each of the individual transitions \( q_{i+1} \xrightarrow{\sigma} q_{i+2} \) in the above transition sequence is controllable. Monotonic convergence of the optimization algorithm \( [5, 3] \) then implies \( v_s(q_1) \leq v_s(q_2) \leq \cdots \leq v_s(q_j) \) implying that \( \{q_1, q_2, \ldots, q_j\} \) is a \( L^* \)-path. The converse follows by a similar argument.

**Proposition 4.1:** For the optimally supervised navigation automaton \( G^*_{\text{Nav}} \), we have

\[ \forall q_i \in Q \setminus Q_{\text{OBSTACLE}}, L(q_i) \subseteq \Sigma_c^* \]

**Proof:** We proceed by the method of contradiction. Assume \( \exists s \in L(q_i) \) for some \( q_i \in Q \setminus Q_{\text{OBSTACLE}} \) such that \( s \notin \Sigma_c^* \).

Since \( u \in \Sigma \) is the only uncontrollable event and

\[ \delta^\#(q_i, u) = q_0 \]

(6)

\[ \delta^\#(q_0, u) = q_0 \] (Self-loop) (7)
it follows that $s = s_1 u^K$ where $s_1 \in \Sigma^*$ and $K \geq 1$. Since $u$ is only defined from $q_i \in Q_{OBSTACLE}$, it follows that there exists an enabled sequence of controllable transitions from $q_i$ to $q_j \in Q_{OBSTACLE} \setminus \{q_{GOAL}\}$ which in turn implies (See Lemma) that
\[ v_k(q_i) \leq v_k(q_j) \tag{8} \]

Since $L(q_i) = u^*$, it follows from Definition that
\[ v_k(q_i) = \min_{\omega \in \Sigma^*} \{ (1 - \theta_{min})^{i-1} - (\theta_{min} - \theta_k)^2 \} < 0 \tag{9} \]
\[ v_k(q_i) = v_k(\omega) = L(q_i) \setminus \{q_{GOAL}\} \]

Now, since only $q_{GOAL}$ has a negative characteristic weight (See Definition), we have $v_k(\omega) \in L(q_i) \setminus \{q_{GOAL}\} \geq 0$. On the other hand, $q_j \notin Q_{OBSTACLE}$ implies
\[ v_k(\omega) = L(q_i) \setminus \{q_{GOAL}\} \]
\[ \Rightarrow v_k(\omega) = \delta(q_i, \omega) \in Q_{OBSTACLE} \]
\[ \Rightarrow \exists \omega \in L(q_i) \setminus \{q_{GOAL}\} \therefore v_k(q_i) \leq v_k(q_j) \tag{11} \]

where $\alpha = \theta_{min} \sum_{\omega \in \Sigma^*} q_i^{\min} q_j^{\min}$. We observe that
\[ \alpha \in [0, 1] \text{ with the strict upper bound } \alpha < 1 \text{ following from the consideration of the worst case scenario where } \delta(q_i, \omega) \in Q_{OBSTACLE}, \forall \omega \in \Sigma^*, \text{in which case we have} \]
\[ \alpha = \theta_{min} \frac{1}{\text{CARD} \{1 - \theta_{min}\} \times \text{CARD} \{\Sigma^*\} = \theta_{min} - \theta_k^2 < 1} \]

Hence we conclude $v_k(q_i) \leq v_k(q_j)$ which contradicts Eq. (8).

**Corollary 4.1: (Obstacle Avoidance):** There exists no $L^*$-path from any unblocked state to any blocked state in the optimally supervised navigation automaton $G_{NAV}$.

**Proof:** Let $q_i \in Q \setminus Q_{OBSTACLE}$ and assume that there exists a $L^*$-path from $q_i$ to some $q_j \in Q \setminus Q_{OBSTACLE}$. It follows from Lemma 4.1 that there exists an enabled sequence of transitions from $q_i$ to $q_j$. Since the uncontrollable transition is defined from $q_i \in Q_{OBSTACLE}$ (See Definition), it follows that $L(q_i) \notin \Sigma^*$ which contradicts Proposition 4.1.

**Proposition 4.2: (Existence of $L^*$-paths):** There exists a $L^*$-path $p(q_i, q_{GOAL})$ from any state $q_i \in Q$ to the goal $q_{GOAL} \in Q$ if and only if $v_k(q_i) > 0$.

**Proof:** Partitioning $L(q_i)$ based on terminating states,
\[ v_k(q_i) = v_k(\omega : \delta(q_i, \omega) = q_{GOAL}) + v_k(\omega : \delta(q_i, \omega) \in Q \setminus \{q_{GOAL}\}) \]
\[ + v_k(\omega : \delta(q_i, \omega) \in Q_{OBSTACLE}) \]
\[ \Rightarrow \exists \omega \in L(q_i) \setminus \{q_{GOAL}\} \therefore v_k(q_i) \leq v_k(q_j) \tag{12} \]

Now, Corollary 4.1 implies $\omega : \delta(q_i, \omega) \in Q_{OBSTACLE} = \emptyset \Rightarrow v_k(\omega : \delta(q_i, \omega) \in Q_{OBSTACLE}) = 0$. Also, since $v_k(q_j) \in Q \setminus Q_{OBSTACLE} \setminus \{q_{GOAL}\}$, we have $v_k(\omega : \delta(q_i, \omega) \in Q \setminus Q_{OBSTACLE} \setminus \{q_{GOAL}\}) = 0$. Hence we conclude
\[ v_k(q_i) = v_k(\omega : \delta(q_i, \omega) = q_{GOAL}) \tag{13} \]

It follows from $\chi(q_{GOAL}) = 1$ that $\delta(q_i, \omega) = q_{GOAL} \Rightarrow v_k(\omega) > 0$ implying there exists an enabled sequence of controllable transitions from $q_i$ to $q_{GOAL}$ in $G_{NAV}^*$ if and only if $v_k(q_i) > 0$. The result then follows from Lemma 4.1.

**Corollary 4.2: (Absence of Local Maxima):** If there exists a $L^*$-path from $q_i \in Q$ to $q_j \in Q$ and a $L^*$-path from $q_i$ to $q_{GOAL}$ then there exists a $L^*$-path from $q_i$ to $q_{GOAL}$; i.e.,
\[ \forall q_i, q_j \in Q \exists p_1(q_i, q_{GOAL}) \cup \exists p_2(q_j, q_{GOAL}) \Rightarrow \exists p(q_i, q_{GOAL}) \]

**Proof:** Let $q_i, q_j \in Q$. We note
\[ \exists p_1(q_i, q_{GOAL}) \Rightarrow v_k(q_i) > 0 \tag{Proposition 4.2} \tag{14a} \]
\[ \exists p_2(q_j, q_{GOAL}) \Rightarrow v_k(q_j) \leq v_k(q_i) \tag{Definition 1} \tag{14b} \]

It follows that $v_k(q_j) > 0$ which implies that there exists a $L^*$-path from $q_j$ to $q_{GOAL}$ (Proposition 4.2).

**Algorithm 1: Computation of Next State**

```
1 Set M = -Inf;
2 for j = 1 to CARD($\Sigma^*$)
3   qj = $\delta(q_{CURRENT}$, $\nu$);
4   if v_k(qj) > M then
5      q_{NEXT} = qj;
6      M = v_k(qj);
7 end
```

**Fig. 2.** States $q_j, q_{GOAL}, q_{BDSTACLE}$ with $q_j \leq q_{BDSTACLE} \leq q_{GOAL}$ implying $q_j = q_{GOAL}$ is a $L^*$-path from $q_i$ to $q_{GOAL}$; smoothed path is $[q_j, q_{GOAL}]$.

**Remark 4.1:** (Smoothing computed $L^*$-paths): We note that given an arbitrary current state $q_j$, Algorithm 1 computes the next state $q_{NEXT}$ as the one which has maximal measure among all possible next states $q_{j'}$ satisfying $v_k(q_{j'}) \geq v_k(q_j)$. This precludes the possibility illustrated in Figure 2 implying that the computed plan is relatively smoother. However, such smoothing may result in a computed plan that is not necessarily the shortest path from the initial state to the goal. This is expected since $L^*$ is not meant to determine the shortest path, but compute a v-optimal path to the goal. In the sequel, we expound this optimal tradeoff that $L^*$ makes between path length and robust planning.

**Notation 4.2:** We denote the $L^*$-path computed by Algorithm 1 as a $L^*$-plan in the sequel.

**A. Optimal Tradeoff between Computed Path Length & Robustness to Dynamic Uncertainty**

Majority of reported path planning algorithms consider minimization of the computed feasible path length as the sole optimization objective. Mobile robotic platforms however suffer from varying degrees of dynamic and parametric uncertainties, implying that path length minimization is of lesser practical importance to computing plans that are robust under sensor noise, imperfect actuation and possibly accumulating odometry errors. Even with sophisticated signal processing techniques such errors cannot be eliminated. The $L^*$ algorithm addresses this issue by an optimal trade-off between path lengths and availability of feasible alternate routes in the event of unforeseen dynamic uncertainties. If $\omega$ is the shortest path to goal from state $q_i$, then the shortest path from state $q_i$ (with $q_i \rightarrow q_j$) is given by $q_j, \omega$. However, a larger number of feasible paths may be available from state.
The lower bound computed in Proposition 4.3 is not tight and if the alternate paths are longer or if there are multiple 'shortest' paths then the number of alternate routes required is significantly higher. Detailed examples can be easily presented to illustrate the situation where L* opts for a longer but more robust plan.

5. Summary & Future Research

A novel path planning algorithm L* is introduced that models the autonomous navigation as an optimization problem for probabilistic finite state machines and applies the rigorous theory of language-measure-theoretic optimal control to compute ν-optimal plan to the specified goal, with automated trade-off between path length and robustness of the plan under dynamic uncertainty.

A. Future Work

Future work will extend the language-measure theoretic planning algorithm to address the following problems:

1. Multi-robot coordinated planning: Run-time complexity grows exponentially with the number of agents if one attempts to solve the full Cartesian product problem. However L* can be potentially used to plan individually followed by an intelligent assembly of the plans to take interaction into account.

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; however care must be taken to ensure that the computed solution is not too far from the optimal one.

3. Handling uncontrollable dynamic events: In this paper the only uncontrollable event is the one modeling a collision when the robot attempts to move into a blocked state. Physical dynamic response of the robot may need to be modeled as uncontrolled transitions in highly uncertain environments and will be addressed in future publications.

References


