Appears in IEEE Conference on Robotics and Automation
Nagoya, Japan, May 21-27, 1995

A Motion Description Language and a Hybrid Architecture for Motion Planning with Nonholonomic Robots *

Vikram Manikonda and P.S. Krishnaprasad

Dept. of Electrical Engineering &
Institute for Systems Research
University of Maryland
College Park, MD 20742

James Hendler

Dept. of Computer Science &
Institute for Systems Research
University of Maryland
College Park, MD 20742

Abstract

This paper puts forward a formal basis for behavior-based robotics, using techniques that have been successful in control-theory-based approaches for steering and stabilizing robots that are subject to nonholonomic constraints. In particular, behaviors for robots are formalized in terms of kinetic state machines, a motion description language, and the interaction of the kinetic state machine with real-time information from (limited range) sensors. This formalization allows us to create a mathematical basis for the study of such systems, including techniques for integrating sets of behaviors. In addition we suggest optimality criteria for comparing both atomic and compound behaviors in various environments. A hybrid architecture for the implementation of path planners that uses the motion description language is also presented.

1 Introduction

Traditional robot motion planning and obstacle avoidance concentrates on determining a path in the presence of holonomic or integrable, equality and inequality constraints on the configuration space. In practice however many robotic systems include constraints that are not holonomic. Such kinematic constraints cannot be reduced to equivalent constraints on the configuration variables which are explicit functions of position variables (e.g. a front wheel drive car, object manipulation by rolling with a robotic hand).

Often such drift-free (completely) nonholonomic systems, *where the number of controls is less than the number of states*, are controllable. Papers [1, 2, 3] present analytical tools based on Lie algebras to generate control sequences to steer such systems. The use of sinusoids in such problems is already anticipated in the work of Brockett on Singular Riemannian Geometry [4]. As these nonholonomic, drift-free systems do not satisfy Brockett's necessary condition for smooth stabilization [5], these systems cannot be stabilized to the origin in state space by using smooth time-invariant state feedback. This reinforces the need for alternatives such as piecewise smooth feedback controllers [6], time-varying periodic controllers [7] and explicit control design to generate time-varying stabilizable control laws [8].

While a majority of the above research on steering and stabilization of nonholonomic systems [9] assumes an obstacle-free world, we note that the problem of autonomous path planning and obstacle avoidance with nonholonomic robots is a nontrivial one. Modeling obstacles as constraints in the configuration space and then designing control laws is a complex problem. In addition deriving control laws for limited range sensors with imperfect and uncertain information poses additional problems.

In the AI literature there is growing interest in using reactive, sensor-based behaviors to solve path planning problems given little or no *a priori* information, inspired by the work of Rodney Brooks [10]. The idea here is to rely on the direct coupling of sensory information and actuators. This approach in contrast with methods based on control theory, has resisted mathematical formalization and is not amenable to

*This research was supported in parts by grants from the National Science Foundation's Engineering Research Centers Program: NSFD CDR 8803012, the AFOSR University Research Initiative Program, under grant AFOSR-90-0105, and AFOSR-F49620-92-J-0500, from NSF(IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039 and by ARI (MDA-903-92-R-0035, subcontract through Microelectronics and Design, Inc.)

tests for optimality. Arguably, a better understanding of the properties of nonholonomic systems, would enable one to exploit the underlying geometry along with real-time sensor information for path planning and obstacle avoidance. Hence there is a need for a framework that can capture and integrate features of both modern control-theoretic techniques and reactive planning methods.

In this paper, we introduce a motion description language which we call **MDLe** (to denote its relationship as an extension of Brockett's motion description language, **MDL**, [11, 12]), that can encode and integrate aspects of modern control theory approaches to steering nonholonomic robots with those of reactive-planning systems that rely on the direct coupling of sensory information and actuators. This is done by introducing sensor-driven trigger functions into **MDL** atoms. This motion description language also gives us the means to formalize concepts such as "behavior", "plan" etc. used in the path planning literature. We also introduce a hybrid architecture for path planning and obstacle avoidance that utilizes the formalism of this paper. For other related work inspired by Brockett's **MDL**, see [13].

In section 2 we present details of **MDLe**, examples of obstacle avoidance problems modeled in the framework of the language and suggest some optimality criteria. The hybrid architecture is discussed in section 3. A brief outline of a planner that generates "behaviors" for path planning and obstacle avoidance is also given. Section 4 includes final remarks and future research directions.

2 Language for Motion Planning

We treat a nonholonomic robot as a kinetic state machine (following Brockett [11]) which can be thought of as a continuous analog of a finite automaton. In the framework of **MDLe** these kinetic state machines are governed by differential equations of the form

$$\dot{x} = \sum_{i=1}^{m} b_i(x) u_i; \quad y = h(x) \in \mathbb{R}^p \tag{1}$$

where

$$x(\cdot) : \mathbb{R}^+ = [0, \infty) \to \mathbb{R}^n$$

$$\begin{array}{rccc} u_i : & \mathbb{R}^+ \times \mathbb{R}^p & \to & \mathbb{R} \\ & (t, y(t)) & \mapsto & u_i(t, y(t)) \end{array}$$

Further each b_i is a vector field in $\mathbb{R}^n$.

We now define the *atoms* of the motion language as triples of the form (U, ξ, T) where

$$U = (u_1, \cdots u_m)'$$

where u_i is as defined earlier,

$$\begin{array}{rccc} \xi : & \mathbb{R}^k & \to & \{0,1\} \\ & s(t) & \mapsto & \xi(s(t)) \end{array}$$

is a boolean function, $T \in \mathbb{R}^+$ and $s(\cdot) : [0, T] \to \mathbb{R}^k$ is a k dimensional signal that represents the output of the k sensors. ξ can be interpreted as an interrupt or trigger to the system which is activated in a case of emergency, e.g. the robot gets too close to an obstacle. Let us denote by $\widehat{T}$, (measured with respect to the initiation of the atom) the time at which an interrupt was received i.e. ξ changes state from 1 to 0. The definition of an atom here can be compared with that in **MDL** [11] where Brockett seems to treat time-outs in T, instead of giving explicit status to triggers.

If at time t_0 the kinetic state machine receives an input atom (U, ξ, T) the state will evolve governed by the differential equation (1), as

$$\dot{x} = B(x)U, \quad \forall\, t,\ t_0 \le t < \ t_0 + min[\widehat{T}, T].$$

If the kinetic state machine receives an input string $(U_1, \xi_1, T_1) \cdots (U_n, \xi_n, T_n)$ then the state x will evolve according to

$$\begin{array}{rcll} \dot{x} & = & B(x)U_1, & t_0 \le t < \ t_0 + min[\widehat{T_1}, T_1]. \\ \vdots & & & \\ \dot{x} & = & B(x)U_n, & t_0 + \cdots + min[\widehat{T_{n-1}}, T_{n-1}] \\ & & & \le t < \ t_0 + \cdots + min[\widehat{T_n}, T_n]. \end{array} \tag{2}$$

Hence we may denote a kinetic state machine as a seven-tuple $(\mathcal{U}, \mathcal{X}, \mathcal{Y}, \mathcal{S}, B, h, \xi)$, where

$\mathcal{U} = C^\infty(\mathbb{R}^+ \times \mathbb{R}^p; \mathbb{R}^m)$ is an input (control) space,
$\mathcal{X} = \mathbb{R}^n$ is the state space,
$\mathcal{Y} = \mathbb{R}^p$ is an output space,
$\mathcal{S} \subset \mathbb{R}^k$ is the sensor signal space,
B is an $\mathbb{R}^{n \times m}$ matrix (kinematic constraints matrix),
$h : \mathcal{X} \to \mathcal{Y}$ maps the state space to the output space and
$\xi : \mathcal{S} \to \{0,1\}$ is an interrupt.

As another point of departure from **MDL**, we find it useful to bring input scaling into the picture. This provides considerable flexibility as in the examples of section 2.1.

Definition: Given an atom, (U, ξ, T), define $(\alpha U, \xi, \beta T)$, $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}^+$ as the corresponding *scaled atom* and denote it as $(\alpha, \beta)(U, \xi, T)$.

Definition: An *alphabet* Σ is a finite set of atoms, i.e (U, ξ, T) triples. Thus $\Sigma = \{(U_1, \xi_1, T_1), \cdots, (U_n, \xi_n, T_n)\}$ for some finite n or equivalently $\Sigma = \{\sigma_1, \cdots, \sigma_n\}$ where σ_i denotes the triple (U_i, ξ_i, T_i), such that $\sigma_i \neq (\alpha, \beta)(\sigma_j)$ $\alpha \in \mathbb{R}, \beta \in \mathbb{R}^+$ and $i = 1, \cdots n$, $j = 1 \cdots n$.

We find it very useful to formalize scaling into the language and hence introduce the notions of extended alphabet and language.

To simplify notation in the rest of the discussion we denote the scaled atom $(1,1)\sigma_i$ simply by σ_i.

Definition: An *extended alphabet* Σ_e is the infinite set of scaled atoms, i.e. triples $(\alpha U, \xi, \beta T)$ derived from the alphabet Σ.

Definition: A *language* Σ^* (respectively Σ_e^*) is defined as the set of all strings over the fixed alphabet Σ (respectively extended alphabet Σ_e).

Definition: A *behavior* π is an element (i.e. word) of the extended language Σ_e^*. For example, given an alphabet $\Sigma = \{\sigma_1, \sigma_2\}$, a behavior π could be the string $(\alpha_1, \beta_1)\sigma_1(\alpha_2, \beta_2)\ \sigma_2(\alpha_3, \beta_3)\sigma_1$.

Remark: To account for constraints one might limit behaviors to lie in a sublanguage $B \subset \Sigma_e^*$. This will be explored in future work.

Definition: The *length* of a behavior denoted by $|\pi|$ is the number of atoms (or scaled atoms) in the behavior.

Definition: The *duration* $T(\pi)$ of a behavior

$$\pi = (\alpha_{i1}, \beta_{i1})(U_{i1}, \xi_{i1}, T_{i1}) \cdots (\alpha_{il}, \beta_{il})(U_{il}, \xi_{il}, T_{il})$$

executed beginning at time t_0 is the sum of the time intervals for which each of the atoms in the behavior was executed. That is,

$$T(\pi) = t_0 + min[\widehat{T_1}, \beta_{i1}T_{i1}] + \cdots + min[\widehat{T_n}, \beta_{il}T_{il}] \quad (3)$$

Definition: Given a kinetic state machine and a world-model, a *plan* Γ is defined as an ordered sequence of behaviors, which when executed achieves the given goal. For example a plan $\Gamma = \{\pi_3\pi_1\pi_n \cdots\}$ could be generated from a given language where each behavior is executed in the order in which they appear in the plan. The length of a plan Γ is given by $|\Gamma| = \sum |\pi_i|$ and the duration of the plan is given by $T(\Gamma) = \sum T(\pi_i)$. In a particular context there may be more than one plan that achieves a given goal.

Remark: Since each atom when executed by a kinetic state machine, combines in general both open loop and feedback controls, one could argue that our definition of behavior captures some aspects of the essence of locomotion behavior c.f. [14], as well as the sense in which the term is used by Brooks [10]. Further the passage from atoms to behaviors to plans suggests a layered architecture as we shall see below.

We now state a proposition that to some extent answers the question of the existence of an alphabet Σ (respectively Σ_e) which can be used to generate behaviors and hence plans to achieve the required goal.

Proposition 1 *Given an obstacle-free environment and a kinetic state machine that is governed by the differential equation*

$$\dot{x} = \sum_{i=1}^{m} b_i(x)u_i;\ x \in \mathbb{R}^n, u \in \mathbb{R}^m \quad (4)$$

such that the control Lie algebra (i.e. the vector space spanned at any point by all the Lie brackets of the vector fields b_i) has rank n, then there exists an alphabet Σ (respectively Σ_e) which can be used to generate behaviors (and hence plans) to steer the system from a given initial state x_o to a final state x_f.

Proof: From Chow's [15] theorem we know that if the control Lie Algebra has rank n then the system is controllable. This implies there there exist piecewise constant controls $u : [0, T] \rightarrow \mathbb{R}^m, T \geq 0$ that steer the system from any initial state $x_o(0)$ to any final state $x_f(T)$.

A simple alphabet that can be used to generate behaviors consists of m triples of the form $(U_1, 1, 1), \cdots, (U_m, 1, 1)$ $\alpha \in \mathbb{R}, \beta \in \mathbb{R}^+$ where

$$\begin{aligned} U_1 &= (1, 0, 0, 0, \cdots, 0)' \\ U_2 &= (0, 1, 0, 0, \cdots, 0)' \\ &\vdots \\ U_m &= (0, 0, 0, \cdots, 0, 1)' \qquad \square \end{aligned}$$

Example 1. Consider the problem of path planning with a unicycle, with a single sensor, that wanders around in a given environment without colliding into obstacles (analogous to the idea of the first level of competence in Brooks [10]). Let us assume the task of the robot (unicycle) in this case is to wander till it senses an obstacle. If it senses an obstacle it avoids the obstacle and continues to wander around. We now formulate and solve this problem treating the unicycle with its sensor as a kinetic state machine and find a plan that solves the problem. The differential equations governing the kinetic state machine are

$$\dot{x} = v_1 \cos\theta \quad (5)$$

$$\dot{y} = v_1 \sin\theta \quad (6)$$

$$\dot{\theta} = v_2 \quad (7)$$

where $(x, y) \in \mathbb{R}^2$ denotes the position of the unicycle w.r.t some inertial frame, $\theta \in S^1$ denotes the orientation of the unicycle relative to the horizontal axis, v_1 and v_2, the velocity of the unicycle and the angular velocity respectively are the inputs to the kinetic state machine. With reference to the standard notation (1), we identify $u_1 = v_1$, $u_2 = v_2$, $b_1 = (\cos\theta, \sin\theta, 0)'$ and $b_2 = (0, 0, 1)'$.

To generate the "wander behavior" (wander in a given environment without colliding into obstacles) let us consider the following atoms:
$\sigma_1 = (U_1, \xi_1, T_1)$ where

$$U_1 = (1, 0)'$$
$$\xi_1 = \begin{cases} 1 & \text{if } \rho > 10 \\ 0 & \text{if } \rho \leq 10 \end{cases}$$
$$T_1 \in (0, \infty)$$

where ρ is the distance between the robot and the obstacle that is returned by the sensor.
$\sigma_2 = (U_2, \xi_2, T_2)$ where

$$U_2 = (0, 1)'$$
$$\xi_2 = \begin{cases} 0 & \text{if } \rho > 10 \\ 1 & \text{if } \rho \leq 10 \end{cases}$$
$$T_2 \in (0, \infty)$$

$\sigma_3 = (U_3, \xi_3, T_3)$ where

$$U_3 = (0, 1)'$$
$$\xi_3 \equiv 1$$
$$T_3 \in (0, \infty)$$

Let $\alpha \in [\alpha_{min}, \alpha_{max}]$ and $\beta \in [0, \infty]$. Now consider the following atomic behaviors

$$\pi_1 = (\alpha_1^1, \beta_1^1)(U_1, \xi_1, 1)$$

$$\pi_2 = (\alpha_1^2, \beta_1^2)(U_2, \xi_2, 1)$$

$$\pi_3 = (\alpha_1^3, \beta_1^3)(U_3, \xi_3, 1)$$

Based on the equations of this robot, the behavior π_1 is interpreted as "move forward" with a velocity of α_1^1 units/sec for β_1^1 seconds, and behaviors π_2 and π_3 can be interpreted as "turn" with a velocity of α_1^i deg/sec for maximum of β_1^i seconds (here $i = 1, 2, 3$) i.e., turn counter clockwise by a maximum of β_1^i degrees, unless interrupted. As explained earlier the atoms of each behavior will only execute as long as their respective ξ functions are 1 and the time of execution is less than T. Hence, once π_3 begins executing it continues until $t = \beta_1^3$ since $\xi_3 = 1$ in the entire interval, $[0, \beta_1^3]$.

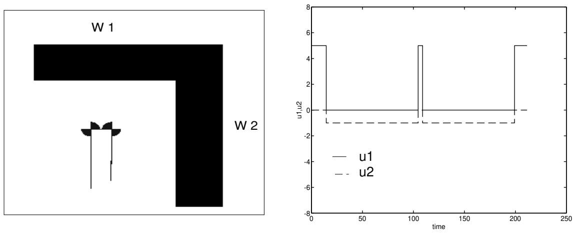

Figure 1: Trajectory and Inputs Generated by the Plan Γ_1

Consider the plan $\Gamma_1 = ((5, 100)\pi_1(-1, 90)\pi_3)^*$ i.e.

$$\Gamma_1 = \{(\alpha_1^1, \beta_1^1)\pi_1 \ (\alpha_1^3, \beta_1^3)\pi_3 \ (\alpha_1^1, \beta_1^1)\pi_1 \ (\alpha_1^3, \beta_1^3)\pi_3 \cdots\}$$

Observe that, if this plan is executed in the environment (with walls W1 and W2) as shown in the Fig. 1, the robot will move forward for time t, $t_0 \leq t < t_0 + 100$, where t_0 is the time at which the behavior was started, when ξ_1 will interrupt it (too close to W1). Let us assume that the interrupt was received at $t_0 + \widehat{T_1}$. The execution of behavior π_1 is then inhibited, behavior π_3 is picked up from the queue and is executed. As $\xi_3 = 1$ in the entire interval $t \in [t_0 + \widehat{T_1}, t_0 + \widehat{T_1} + 90)$ the robot will then turn clockwise by 90 degrees and then it will move forward (execute behavior π_1). But again after some finite time wall W2 (see Fig. 1) will cause $\xi_1 = 0$ and hence interrupt the move forward behavior. Behavior π_3 is executed as earlier i.e. the robot turns clockwise by 90 degrees, and now continues to move forward. If it does not detect an obstacle at the end of 100 seconds since it began moving forward, it will stop, turn clockwise by 90 degrees, and continue to repeat the sequence of actions.

Now consider the plan $\Gamma_2 = ((50, 2)\pi_1(-20, 5)\pi_2)^*$. Observe that, if this plan is executed in the same environment (see Fig. 2), then while executing the "move forward" i.e. π_1, in the time interval $t_0 \leq t < t_0 + 2$ the robot realizes that the obstacle is at a distance less than 10 units from it and hence ξ_1 interrupts the "move forward " and the robot begins to execute "turn right". Due to the choice of the interrupt function ξ_2 the robot will now switch between "turn right" and and "move forward" (a condition reffered to as chattering) and trace a trajectory as shown in the figure. Hence depending on the choice of the alphabet one can generate different plans to achieve the same task.

The question of how to generate a plan given an alphabet and a kinetic state machine, is an open one and it largely depends on the task and the planner. In section 3 we describe a path planner for nonholonomic

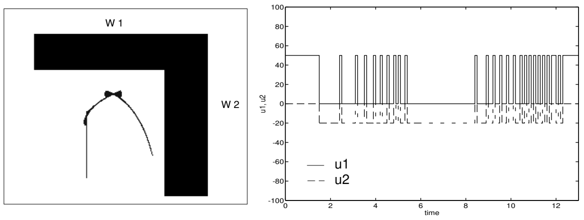

Figure 2: Trajectory and Inputs Generated by the Plan Γ_2

robots. Before we discuss the features of the planner we introduce some more definitions that help formalize measures to evaluate the performance of a plan.

2.1 Performance Measure of a Plan

At first, it appears that, to generate a plan to steer a system from a given initial state x_0 to a final state x_f requires complete *a priori* information of the world, which is not available in many instances of path planning. In the absence of such complete *a priori* information about the world $\mathcal{W}$, then the planning system has to generate a sequence of plans based on the limited information about $\mathcal{W}$ that it has, which when concatenated will achieve the required goal. We refer to these plans that are generated on limited information to achieve some subgoal as *partial plans* Γ^p. The plan to steer the system from a given initial state x_o to a final state x_f is then determined after the system has reached the final state and is: $\Gamma = \widehat{\Gamma_1}^p \widehat{\Gamma_2}^p \cdots \widehat{\Gamma_n}^p$ where $\widehat{\Gamma_i}^p$ is the partial plan consisting of only those behaviors and atoms in each behavior that have been executed for $t > 0$.

Remark: As a partial plan is generated with limited information of the world, not all the behaviors and not every atom in a behavior generated by the partial plan may be executed at run time for the following reasons:
(i) Let us consider a behavior $\pi_i = \sigma_3\sigma_1\sigma_4\cdots\sigma_n$. Let us assume that the atom σ_3 is interrupted by ξ_3 at $\hat{t}$. Now as explained earlier σ_1 will begin to execute. But if $\xi_3 = \xi_1$ the atom σ_1 will not be executed and depending on ξ_4, σ_4 will begin to execute.
(ii) For practical reasons we introduce a hierarchy of interrupts. While a specific behavior π_i of a partial plan is being executed, if a **level 0** interrupt is received, the execution of that particular atom is inhibited and the next atom in that behavior is executed just as explained in (i) above. If a **level 1** interrupt is received while a behavior is being executed the execution of that behavior is now inhibited and the next behavior in the partial plan is executed. Finally if a **level 2** interrupt is received the execution of the remainder of the current partial plan is stopped and a new partial plan is executed.

The length of a plan is given by $|\Gamma| = \sum_{i=1}^{n} |\pi_i|$ and the time of execution of the plan is given by $T(\Gamma) = \sum_{i=1}^{n} T(\pi_i)$.

With these formal definitions, we can start discussing the performance of an algorithm that uses these behaviors and analyzing some earlier algorithms for nonholonomic motion planning.

Given an algorithm that generates a plan Γ we define a candidate measure of performance $\Theta(\Gamma)$ of the plan as

$$\Theta(\Gamma) = T(\Gamma) + \tau|\Gamma| \qquad (8)$$

where τ is a normalizing factor having the units of time. (One need not limit oneself to such additive combinations although this is the only case used here.)

Defining a performance measure for a path planner is a difficult task as it is dependent on the goal the robot seeks to achieve. Some path planners use the total time to achieve the goal as a measure of performance. In many situations one might be interested in not only the time but also on the smoothness of the path traversed or the number of times switching between different controls was necessary. For example consider the task of parallel parking of a car. One might be able to achieve the goal by using only open-loop controls but switching between them at regular intervals, hence possibly reducing the time to achieve the goal but compromising on the smoothness of the path. On the other hand if one uses a time dependent feedback law, the same task could be achieved, possibly by moving along a smooth trajectory at the risk of taking longer time to achieve the goal. This indicates a trade-off between two competing requirements which is captured by the performance measure (8).
We now define the optimal performance of a plan as

$$\Theta(\Gamma)_{optimal} = min\{T(\Gamma) + \tau|\Gamma|\}. \qquad (9)$$

Here the minimization is performed over the subset of plans generated by the subset B of admissible behaviors. Depending on the kinetic state machine and the choice of the planner one can now place bounds on the optimal performance and hence compare the performance of different planners given the same language or that of the planner given a new language. This is illustrated in the example given below.

Example 2: Consider the problem of steering the unicycle from a given initial location z_o to z_f. The equations of the unicycle are given in example 1. Let us assume that the language is based on the following atoms. $\sigma_1 = (U_1, \xi_1, 1)$, $\sigma_2 = (U_2, \xi_2, 1)$ where U_1, ξ_1, U_2, ξ_2 are as defined in example 1. Let $\alpha \in [-5, +5]$ and $\beta \in (0, \infty)$.

Let us also assume that the planner did not have complete information about the world and had to generate n partial plans to achieve the goal. Each partial plan consists of steering the unicycle from z_i to z_j (see Fig 3) such that there are no obstacles in some small neighborhood of the line segment joining these two locations. Let us make a further assumption that the planner uses $\alpha_i \in \{1, -1\}$ as the scaling factor while generating partial plans.

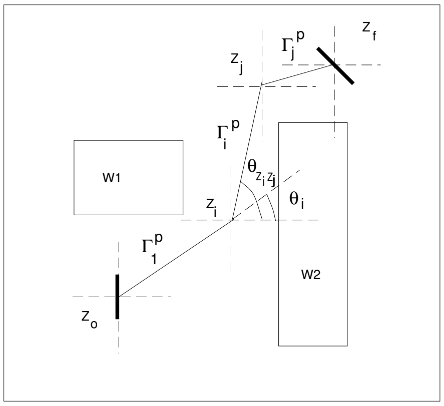

Figure 3: Partial Plan Generation

From the kinematic equations of the unicycle we know that a simple partial plan to steer a unicycle from $z_i = (x_i, y_i, \theta_i)$ to $z_j = (x_j, y_j, \theta_j)$ would be :

(i) turn by $(\theta_{z_i z_j} - \theta_i)$,

(ii) move by a distance d_i and

(iii) finally turn by $(\theta_j - \theta_{z_i z_j})$,

where $z_i z_j$ is the vector in $\mathbb{R}^2$ joining (x_i, y_i) and (x_j, y_j), $d_i = \|z_i z_j\|_2$ and $\theta_{z_i z_j}$ is the orientation of the vector w.r.t. to the x-axis.

We can rewrite this simple algorithm as a partial plan derived from the language using the atomic behaviors $\pi_1 = \sigma_1$ and $\pi_2 = \sigma_2$,

$$\Gamma^p{}_i = \{(\theta_{i1}/|\theta_{i1}|, |\theta_{i1}|)\sigma_2 \ (1, d_i)\sigma_1 \ (\theta_{i2}/|\theta_{i2}|, |\theta_{i2}|)\sigma_2\}$$

where θ_{i1} and θ_{i2} are the angles of the two turns as described above of the ith partial plan. Hence the plan to steer the system from z_o to z_f is given by

$$\Gamma = \{\Gamma^p{}_1 \Gamma^p{}_2 \cdots \Gamma^p{}_n\}.$$

Given a plan we now illustrate how bounds can be placed on the optimal performance based on the knowledge of the kinetic state machine and the language. Let $d_{max} = max\, \|z_i z_j\|$.

$$T_{max}(\Gamma^p{}_i) \ \leq \ 2\pi + d_{max} + 2\pi \ \leq \ 4\pi + d_{max}$$

and $|\Gamma^p{}_i|_{max} \leq 3$. Hence,

$$0 \leq \Theta(\Gamma) \leq 3n + n(4\pi + d_{max}).$$

However, as we are using only open-loop controls, we know from the kinematics of the system that given an initial state x_o and a final state x_f both the behaviors $(\alpha_i, \beta_i)\sigma_i$ and $(k\alpha_i, \beta_i/k)\sigma_i$ would steer the kinetic state machine from the initial state to the final state. Hence we could replace $(\alpha_i, \beta_i)\sigma_i$ by $(k\alpha_i, \beta_i/k)\sigma_i$.

Observe that in the generation of the above partial plan and in the calculation of the performance measure we restricted α_i to $\{-1, 1\}$ (in some sense placed bounds on the speed of the unicycle) because the planner did not have complete information about the world. But since the language permits $\alpha_i \in [-5, 5]$, we have

$$0 \leq \Theta(\Gamma)_{optimal} \leq \frac{n(4\pi + d_{max})}{5} + 3n.$$

Having placed bounds on a plan generated by one set of behaviors we can now compare the performance of another set of behaviors (for example, one using periodic functions to steer the robot) against these bounds.

In the above examples we have used very simple controls in our alphabet. But one should note that depending on the application, a wide variety of controls (open loop and closed loop) could be included in the alphabet and some examples of such controls can be found in [1, 2, 6, 7, 8, 16]

The question of how to generate a plan given an alphabet and a kinetic state machine, is an open one and it largely depends on the task. In the next section we describe a control architecture and a few details of a path planner for nonholonomic robots.

3 Hybrid Architecture

As we seek to attain higher levels of autonomy in robots, the need for hierarchical and distributed control schemes becomes apparent. Motivated in part by the hierarchical structure of neuromuscular control systems [14] we present a control architecture (see Fig. 4), to generate and execute plans to achieve a given task. (To avoid clutter, only one level of interrupts is given.) The lowest level is the kinetic state machine with sensors, where the sensors are used in a low-level feedback loop. The planner could be interpreted as the higher end of the system where sensory information is processed to generate goal-related trajectory information. The layered and distributed nature of the control becomes apparent when one observes that once a plan has been generated, each level and even various modules at the same level (e.g. cleanup and plan) continue to execute independently.

The task of the planner is to use the limited-range sensor information, to generate partial plans that result in collision-free feasible trajectories. Planning is done at two levels - global and local. For local planning, collision-free (non)feasible paths are generated using potential functions assuming that the robot is holonomic. A partial plan (feasible path) is then generated that obeys the constraints in the configuration variables. As feasible trajectories are only approximations to the trajectories generated using potential functions, collision with obstacles could occur while tracing them. While the robot is in motion, collisions are avoided by using the sensor information to trigger interrupts as described previously.

At a global level, heuristics along with the world map generated while the robot is *en route* to the goal are used to solve the problem of cycles.

Once a partial plan has been generated it is executed as explained in section 2. Let us assume that an atom (U_i, ξ_i, T_i) is executed at time $t = t_0$. In Fig. 4, T is a timer whose output is 1 (active high) while $t_0 \leq t < T_i$ and is 0 (active low) if $t \geq t_0 + T_i$. $\xi_i(s(t))$ returns an interrupt (active low) to the system when conditions defined by $\xi_i(s(t))$ are satisfied. Observe here that the interrupt could be of level 2,1 or 0. Hence the functioning of the AND gates in the kinetic state machine can be interpreted as follows - if either the robot receives an interrupt or $t \geq t_0 + T_i$ the input to gate II is an active low and hence the input to the kinetic state machine is inhibited i.e. the current atom/behavior/partial plan (depending on the interrupt level) is stopped and the next atom/behavior/partial plan in the queue is executed (see Remark (ii) in section 2.1).

Fig. 5 shows a simulation of the path generated by the planner. As the partial plans are generated based on local information (denoted by obstacle free disks in Fig. 5)[1], the paths initially generated are locally optimal. As the knowledge of the environment increases (e.g. after having reached the goal using a non-optimal path), the performance of the system improves when the same task is repeated. The bold solid lines denote the new trajectories (partial plans) generated after partial knowledge of the world has been gained. Observe that the length of the plan has decreased by less than a third. More details on the planner can be found in [17]. Some other examples of path planners for nonholonomic mobile robots can be found in [18, 19]

[1]It should be pointed out here that the obstacle-free disks generated by the planner are not entirely obstacle free, but this is because in the simulator we have used only sensors of the 'eye' to generate obstacle-free disks. For now, those obstacles that are not detected by the sensors are treated as being in the blind spots of the "robot, (c.f. [17]).

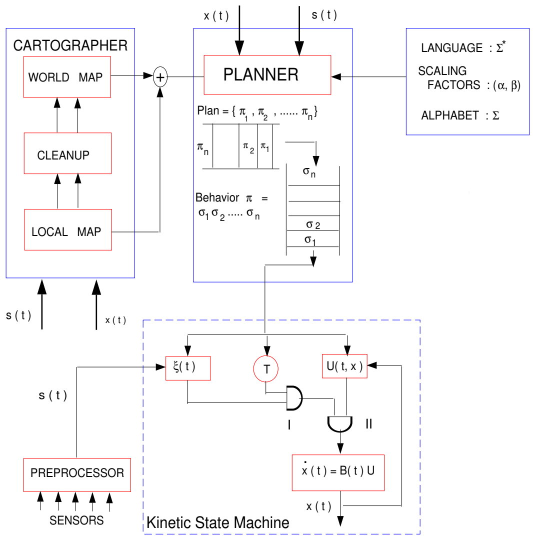

Figure 4: Hybrid Control Architecture

4 Final Remarks

The motion description language along with the control architecture serves as an abstraction between continuous and discrete time control strategies. Current directions of research include continuing formalization of behavior-based robotics and some experimental work involving the implementation of the language and planner on a car-like robot. This includes expanding the language to include multiple kinetic state machines and communication protocols between these machines. We are also working on extending the technique used by the planner to generate plans in the presence of moving obstacles.

References

[1] R. M. Murray and S. S. Sastry. Steering non-

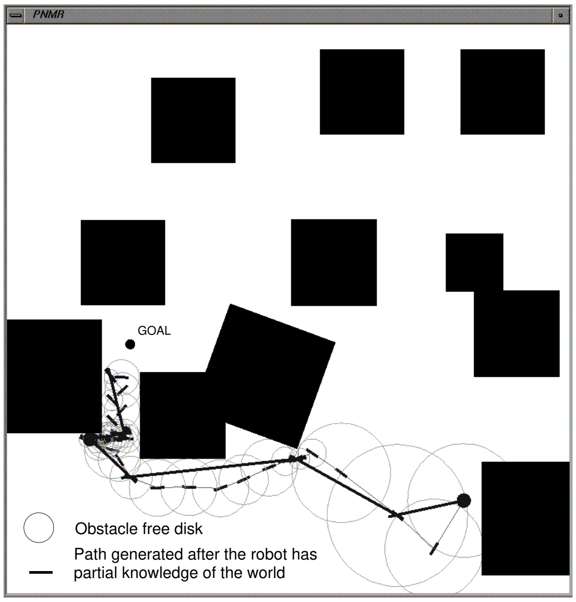

Figure 5: Paths Generated by the Planner

holonomic systems using sinusoids. In *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 2097–2101, Honolulu,HI, December 1990.

[2] H.J. Sussmann. Local controllability and motion planning for some classes of systems without drift. In *Proceedings of the 30th Conference on Decision and Control*, pages 1110–1114, Brighton, England, December 1991. IEEE.

[3] C. Fernandes, L. Gurvits, and Z.X. Li. Foundations of nonholonomic motion planning. In Z. X. Li and J. F. Canny, editors, *Nonholonomic Motion Planning*. Kluwer Academic, 1993.

[4] R. W. Brockett. Control theory and singular Riemannian geometry. In *New Directions in Applied Mathematics*, pages 13–27. Springer-Verlag, 1982.

[5] R. W. Brockett. Asymptotic stability and feedback stabilization. In *Differential Geometric Control Theory*, pages 181–191. Birkhauser, 1983.

[6] C. Canudas de Wit and O.J. Sordalen. Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Transactions on Automatic Control*, 37(11):1791–1797, November 1992.

[7] J.-M. Coron. Global asymptotic stabilization for controllable systems. *Mathematics of Control, Signals and Systems*, 5(3), 1992.

[8] J.B. Pomet. Explicit design of time-varying stabilizing control laws for a class of controllable systems without drift. *Systems and Control letters*, 18:147–158, 1992.

[9] R.M. Murray, Z. Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[10] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[11] R.W. Brockett. Formal languages for motion description and map making. In *Robotics*, pages 181–193. American Mathematical Society, 1990.

[12] R. W. Brockett. Hybrid models for motion control. In H. Trentelman and J. C. Willems, editors, *Perspectives in Control*, pages 29–51. Birkhauser Verlag, 1993.

[13] R.M. Murray, D.C. Deno, K.S.J. Pister, and S.S. Sastry. Control primitives for robot systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(1):183–193, January/February 1992.

[14] N. Bernstein. *The Co-ordination and Regulation of Movement*. Pergamon Press, Oxford, 1967.

[15] Robert Hermann. Accessibility problems for path systems. In *Differential Geometry and the Calculus of Variations*, chapter 18, pages 241–257. Math Sci Press, Brookline, MA, 1968. 2nd Edition.

[16] Naomi E. Leonard and P.S. Krishnaprasad. Averaging for attitude control and motion planning. In *Proceedings for the 32nd Conference on Decision and Control*, pages 3098–3104, Dec 1993.

[17] Vikram Manikonda. A hybrid control strategy for path planning and obstacle avoidance with nonholonomic robots. Master's thesis, University of Maryland, College Park, 1994 (also Institute for Systems Research Thesis Report M.S. 94-8).

[18] J.Barraquand and J.C.Latombe. Robot motion planning: A distributed representation approach. Technical Report STAN-CS-89-1257, Stanford University, May 1989.

[19] J.-P. Laumond, P.E. Jacobs, M.Taix, and R.M. Murray. A motion planner for nonholonomic robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593, Oct 1994.