Controlling Wild Bodies Using Linear Temporal Logic

Leonardo Bobadilla Oscar Sanchez Justin Czarnowski
bobadil1@uiuc.edu sanche14@uiuc.edu jczarno2@uiuc.edu
Katrina Gossman Steven M. LaValle
kgossma2@uiuc.edu lavalle@uiuc.edu
Department of Computer Science
University of Illinois
Urbana, IL 61801 USA

(a) (b)

Fig. 1. a) Our vehicle of study is a $4 weasel ball; b) it consists entirely of a battery and slowly oscillating motor mounted to a plastic shell.

***Abstract*—There is substantial interest controlling a group of bodies from specifications of tasks given in a high-level, human-like language. This paper proposes a methodology that creates low-level hybrid controllers that guarantee that a group of bodies execute a high-level specified task without dynamical system modeling, precise state estimation or state feedback. We do this by exploiting the wild motions of very simple bodies in an environment connected by gates which serve as the system inputs, as opposed motors on the bodies. We present experiments using inexpensive hardware demonstrating the practical feasibility of our approach to solving tasks such as navigation, patrolling, and coverage.**

I. Introduction

A fundamental challenge in robotics is the construction of robot control strategies from specifications of tasks given in a high-level language. Ideally, we would like to describe tasks such as navigation, patrolling, coverage, and herding, and have them autonomously executed by a team of robots [2]. The low level details of the plan should be hidden. Furthermore, new plans must be efficiently constructed and guaranteed to be correct.

We propose an unusual paradigm as a thought-provoking step toward meeting this challenge. A common approach to many problems is to carefully design an autonomous vehicle, which involves steps such as system modeling and identification, implementing stable controllers, and installing sufficient sensing to ensure state feedback. In contrast, we propose to start with a "wildly behaving" body for which its precise equations of motion are unknown, it is far from stable, and has little or no sensing capabilities. Our main "vehicle" of study will be a $4 weasel ball (see Figure 1, which has no sensors, no computation, and one motor, which oscillates constantly at about 2Hz.

Although used throughout the experiments in this paper, the particular choice of body is not critical. We instead care only about its high-level motion properties. We informally consider a body to be *wild* if when placed into a bounded region $r \subset \mathbb{R}^2$, it moves along a trajectory that strikes every open interval along the boundary of r infinitely often. By *strike*, we mean that the body contacts the boundary with a non-tangential velocity. A well-studied family of systems that have this property is called *dynamical billiards* [35] (imagine a billiard ball that bounces off of the table sides forever). A strong system property that arises in that work and achieves our required wild behavior is *ergodicity*.[1] The idea of exploiting wild motions in robotics is reminiscent of the randomization work by Erdmann [11] and designing robot systems with ergodic dynamics by Shell et al. [33].

How do we control such systems? We are first inspired by the power of abstraction used in hybrid systems [6], [16], [18]. Following this, we are inspired by the family of work that converts high-level specifications into low-level control laws for the hybrid system [21], [30], [36]. In particular, our work uses the Linear Temporal Logic (LTL) framework that has been developed in several recent works [3], [13], [14], [17], [22], [23], [24], [25], [26], [27], [28], [34], [37].

Although we borrow the overall LTL framework, our method of control substantially differs. Whereas it is common in LTL implementations to derive state-feedback control laws within continuous regions [17], [22], [28], [34], we simply let our "vehicle" behave wildly. To control a wild body, we design *gates* that appear only along region boundaries and connect to other regions. When a body strikes a gate, the gate will induce our planned behavior, which might be to remain in the region or transition to another region. In

[1] In this context, ergodicity does not necessarily have anything to do with probabilities, as in the more commonly seen case of Markov chains.

this sense, we "gently guide" the body. This differs from previous LTL implementations because we do not require system identification, state feedback, or a state-feedback control law. Our approach instead draws inspiration from several areas, including *nonprehensile manipulation* [12], [19], [31] and vibrating plates [5], [32]. Even more closely related are designing *virtual fences* to control herds of cows [7] and designing fire evacuation strategies to safely "herd" humans out of a burning building [8].

Our approach consists of four steps. First, we propose discrete abstractions for the motion of one or more wild bodies. Second, we use a temporal logic to give descriptions of tasks. Third, we translate the temporal logic specification into a discrete plan. Finally, this discrete plan is converted into a policy that is executed in our setup involving simple gates and bodies.

The paper is organized as follows. Section II presents some preliminary concepts, including the interaction between the wild body, the gates, and the regions. Sections III and IV present our approach for the cases of a single and multiple bodies, respectively. Section V presents experiments and Section V concludes the paper.

II. The Overall Design

Regions and gates

Consider a planar workspace $E \subset \mathbb{R}^2$ that is partitioned into an obstacle set O and a finite set of bounded cells with connected open interior, each of which is either a *region* or a *gate*; Figure 2 shows a simple example. The following conditions are imposed: 1) No region shares a boundary with any other region, 2) No gate shares a boundary with any other gate; 3) Every region shares a boundary with at least one gate; 4) If a gate and a region share a boundary, then the boundary is a connected interval (rather than being a point or being disconnected). Let R denote the set of all regions and G denote the set of all gates. The union of all $r \in R$, all $g \in G$, and O yields E.

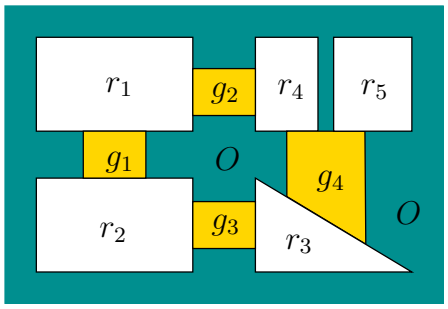


Fig. 2. An example arrangement of five regions and four gates.

Wild bodies

We now place a *body* b into the workspace. The body is assumed to be "small" with respect to the sizes of regions, gates, and their shared boundaries. It is therefore modeled geometrically as a point even though it may have complicated kinematics and dynamics. We assume that the body moves in a wild, uncontrollable way, but the trajectory satisfies the following high-level property: For any region $r \in R$, it is assumed that b moves on a trajectory that causes it to repeatedly strike every open interval in ∂r (the boundary of r), with non-zero, non-tangential velocities. We can now imagine that the body travels on a path through the bipartite graph shown in Figure 3(a), with transitions occurring only if specific gates allow it.

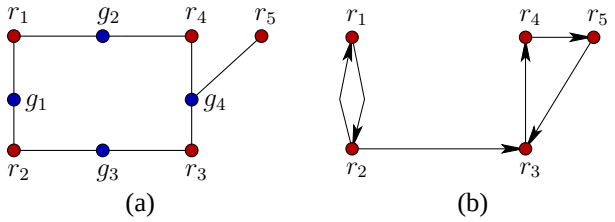


Fig. 3. a) A bipartite graph representation of the arrangement of regions and gates from Figure 2. b) A flow graph that corresponds to one particular composite mode. Each gate mode allows alternative possible flow directions between every pair of regions that are adjacent to the gate.

Guiding body flow with multimodal gates

Every gate $g \in G$ has an associated finite set of *modes* $M(g)$. At any moment in time, every gate $g \in G$ is in some current mode $m \in M(g)$. Let k be the total number of gates. Let M denote the composite mode space, obtained as the k-fold Cartesian product of $M(g)$ for every $g \in G$. The *composite mode* $(m_1, \ldots, m_k) \in M$ specifies the mode of every gate and can be considered as a discrete component of a hybrid system. The continuous component is provided by the body configuration or state $x \in X$. If there are multiple bodies, then the continuous component is the Cartesian product of all body state spaces. Thus, a hybrid system model can be obtained in which $Z = M \times X$ is the hybrid state space, and transition laws govern the mode and continuous state changes.

If a body strikes a gate g, then it is either blocked or allowed to pass, depending on the current mode $m \in M(g)$. Furthermore, a *mode transition equation* specifies the next mode m' for g, depending on m and the region r from which the striking body entered.

The behavior of the modes and their influence on a body can be nicely interpreted in terms of the bipartite graph shown in Figure 3(a). Let $N(g)$ denote the vertices adjacent to g. For each ordered pair (r, r'), such that $r, r' \in N(g)$, the mode $m \in M(g)$ could allow one of four behaviors:

1) Block all passage between r and r'
2) Allow passage only from r to r'
3) Allow passage only from r' to r
4) Allow bidirectional passage between r' and r

For each composite mode, we obtain a directed *flow graph* (see Figure 3(b)), in which the set of vertices is R and there is a directed edge from r to r' only if the mode allows passage from r to r'. Since the flow graph may have multiple out edges

per vertex, nondeterminism is allowed by the model; however, the policies developed in this paper will be deterministic.

Specifying tasks in LTL

We want to specify tasks in some high-level way, possibly starting from structured English or some simple logic. We chose Linear Temporal Logic (LTL) due to its increasing popularity and available toolkits; see [10]. The syntax includes a set Π of propositions, propositional logic symbols, and some temporal operators. Formulas are constructed from atoms $\pi \in \Pi$ using the grammar

$$\phi ::= \pi \mid \neg\phi \mid (\phi \vee \phi) \mid \phi \mathcal{U} \phi,$$

in which $\mathcal{U}$ is a temporal operator meaning *until*. Other operators and connectives can be derived from the grammar: *conjunction* ($\wedge$), *implication* ($\Rightarrow$), *equivalence* ($\Leftrightarrow$), *eventually* ($\diamondsuit$), and *always* ($\square$). For examples, suppose that π_i means that a robot is in r_i. Common task specifications are [26]:

- Navigation: $\diamondsuit \pi_1$
- Sequencing: $\diamondsuit(\pi_1 \wedge \diamondsuit(\pi_2 \wedge \diamondsuit(\pi_3 \wedge \cdots \diamondsuit \pi_k) \cdots)$
- Coverage: $\diamondsuit \pi_1 \wedge \diamondsuit \pi_2 \wedge \cdots \diamondsuit \pi_k$
- Avoiding regions: $\neg(\pi_1 \vee \pi_2 \cdots \vee \pi_k) \mathcal{U} \pi_{final}$
- Patrolling: $\square(\diamondsuit \pi_1 \wedge \diamondsuit \pi_2 \wedge \ldots \diamondsuit \pi_k)$.

An information-feedback plan

A *plan* or *control law* can generally be expressed as an information-feedback mapping $\pi : \mathcal{I} \rightarrow M$, in which M is the composite mode space and $\mathcal{I}$ is an *information space* that takes into account actuation histories and sensor observation histories (see Chapter 11 of [29]). Recall that for each composite mode, there is a corresponding flow graph. We can therefore imagine π as specifying a dynamic flow graph, which changes its flow as new information becomes available.

There are many possible choices for $\mathcal{I}$, depending on the kind of sensors and filters that are developed. We prefer to take a minimalist approach and use the weakest sensors and filters that can nevertheless accomplish the task. Therefore, we avoid the case in which $\mathcal{I} = Z = M \times X$, which would imply that state estimation is available and perfect state feedback can be performed. In coming sections, we will consider *time feedback*, for which $\mathcal{I} = T = [0, t]$, an interval of time. We will also use simple sensors that detect whether a body has passed in or out of a gate. If Y represents the set of all sensor outputs, then we will develop *sensor feedback* plans of the form $\pi : Y \rightarrow M$. In Section IV, a more complicated information space will appear, in which a filter keeps track of the number of bodies per region. This information will be used as feedback to define π.

III. Controlling One Wild Body

In this section a method for generating controllers for a single wild body is introduced. First, we present a discrete transition system that represents a time abstraction of the system, and allows a transformation of the problem from a continuous to a discrete domain. Second, we generate a discrete plan that satisfies a given formula in LTL. Finally, we show how to execute discrete plans using gates and limited sensing.

Discrete abstraction of motion

Given the set R of n regions in the environment, we define a set of Boolean propositions $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$. The proposition π_i is true if and only if the body is in r_i.

Note that only one π_i can be $True$ at any time. Our goal is to construct a plan so that the resulting body trajectory satisfies whatever LTL formula ϕ is given over the set of propositions Π.

When a body strikes a gate, it will experience an immediate transition to a region, according to our model. This corresponds to a discontinuous state jump through each gate area (recall Figure 2(b)). We could alternatively model continuous motions through gates; however, this is avoided in favor of simplicity and is not needed because we define propositions only over the regions. Let $\tilde{x} : [0, t] \rightarrow X$ (recall the hybrid system state space $Z = M \times X$) denote the body *trajectory*.

We now define a discrete transition system D_1 that simulates the original hybrid system. Let the state space of the discrete system be $Q = M \times R$ (recall that M is the composite mode space and R is the set of regions). The transition system is defined as

$$D_1 = (Q, q_0, \rightarrow_1), \tag{1}$$

in which $q_0 = (m_0, r_0)$ yields the initial composite mode m_0 and region r_0. The transition relation $q \rightarrow_1 q'$ is true if and only if, for $q = (m, r)$ and $q' = (m', r')$, whenever the body is in r and the composite mode is m, then it can strike a gate to arrive in r' and the composite mode changes to m'.

It is straightforward to show that D_1 is a simulation of the original hybrid system. Therefore, we can design a solution plan over D_1, thereby inducing the correct behavior of the original hybrid system. This is the standard approach to hybrid system control using a discrete abstraction [1], [23]. We can then apply standard model checking software, such as NuSMV [9] or SPIN [20] to find a trajectory $\tilde{q} = (q_0, q_1, \ldots)$ for D_1 that satisfies the formula $\tilde{q} \models \phi$ for a given LTL formula ϕ. The packages are quite fast in practice and have been used extensively for this purpose. The resulting trajectory $\tilde{q}$ could be finite or infinitely long (but expressed finitely).

To implement the plan on the original hybrid system, simple binary sensing is used to detect whether the body has transitioned through the gate so that the system can keep track of the region that currently contains the body. In this way, every transition from q_i to q_{i+1} in $\tilde{q}$ can be enforced.

A simple example

We will illustrate the ideas presented in this section with an environment that will be used for our experiments (Section V). In Figure 4, there are five regions $R = \{r_0, r_1, r_2, r_3, r_4\}$ and five gates $G = \{a, b, c, d, e, f\}$, shown in blue.

We request the body to visit the regions r_2, r_1, r_0, r_4 in that order. This is encoded in LTL as $\phi = \diamondsuit(\pi_2 \wedge \diamondsuit(\pi_1 \wedge \diamondsuit(\pi_0 \wedge$

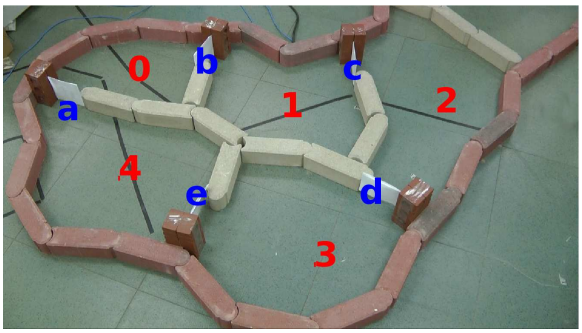


Fig. 4. An example of an arrangement of five regions and five gates.

$\Diamond\pi_4)))$.[2]

Running a model checker produces a sequence that sets the gate modes as follows:

1) Set gate c to allow passage from r_2 to r_1.
2) Set gate b to allow passage from r_1 to r_0.
3) Set gate a to allow passage from r_0 to r_4.

After each step, a sensor can be used to ensure that the body has passed the gate and transition to the next stage occurs. This example is actually so simple that the gate modes can be all set in advance and remain static during execution. This will occur for any trajectory $\tilde{q}$ that does not revisit any regions. In this case, the gates can be made from pieces of paper, as shown in Figure 4 and no sensing is even needed. More complicated examples, which require sensing and the greater expressive power of LTL are given in Section V.

IV. Controlling Multiple Wild Bodies

This section builds on the concepts of the previous sections and extends them to multiple wild bodies. We will achieve control of the bodies without assume any communication or coordination between the bodies or any central source. This is quite unusual for the control of multi-robot systems. We allow bodies to collide with each other, thereby eliminating collision avoidance overhead in terms of sensing and control. We do, however, assume that the bodies remain sufficient wild so that the region boundary is struck by at least one of them in finite time. Our experimental observations are that the wildness properties actually improve as more bodies are placed in the environment. Gates are struck more quickly and frequently.

Discrete abstraction for multiple bodies

Suppose that n identical, indistinguishable bodies are placed into the environment. The only information relevant for tasks will be the number of bodies per region at any instant. Once again, we will not care about the precise location of bodies with each region. Furthermore, bodies are interchangeable due to their indistinguishability.

Consider defining a discrete transition system

$$D_n = (Q_n, q_0, \rightarrow_n). \tag{2}$$

[2]Note this is nonstandard because our π_i represents the subset of Q that includes $q = (m, r_i)$ for all $m \in M$. By contrast, in [26], π_i corresponds to a singleton subset of Q because there are no gate modes.

Let $Q_n = M \times C$, in which M is once again the space of composite modes. We want C to correspond to set of possible body positions, recording only which region they are in. If the bodies were distinguishable, then C would be an n-fold Cartesian product of R, the set of regions. However, due to indistinguishability, C is defined as the set of all m dimensional vectors $c = (c_1, \ldots, c_j)$ for which each $c_i \in \mathbb{N} \cup \{0\}$ and $c_1 + \cdots + c_n = j$ and $j = |R|$. In other words, $c \in C$ encodes the number of bodies occupying each of the regions. The size of C is $\binom{j+n-1}{n}$, which from combinatorics is the number of ways to place n balls (bodies) into j boxes or urns (regions). Each $c \in C$ will be referred to as a *distribution* of balls.

In (2), $q_0 = (m_0, c_0)$, in which m_0 is the initial composite mode and c_0 is the initial body distribution. The transition relation $q \rightarrow_n q'$ is true if and only if, for $q = (m, c)$ and $q' = (m', c')$, whenever the body distribution is c and the composite mode is m, then when a body strikes a gate, the distribution changes to c' and the composite mode changes to m'.

Following by analogy to Section III, we have proved that D_n is a simulation of the original hybrid system of n bodies moving among regions and gates. It is assumed that the initial distribution of bodies is given. To express multi-body tasks in LTL, we define the set Π of propositions to correspond to every possible distribution in C. An LTL formula ϕ can then be defined to express any task that involves distributions of bodies across the regions. Furthermore, standard model checking software is once again applied to produce a trajectory $\tilde{q} = (q_0, q_1, \ldots, q_k)$ for D_n that satisfies $\tilde{q} \models \phi$. The trajectory is implemented by once again using simple binary sensors near the gates to ensure that each transition has occurred before changing the gate modes. A sequence of body distributions is obtained in practice that satisfies the desired LTL formula.

A simple example

Figure 5(a) shows an example that has three regions $R = \{r_1, r_2, r_3\}$ and three gates $G = \{a, b, c\}$. Suppose that each gate allows the bodies to transition in either direction, depending on its mode. The discrete transition system D_2 is given by (2) for $n = 2$. Any trajectory $\tilde{q}$ for D_2 corresponds to a walk through the graph shown in Figure 5(b). The gate that is crossed by a body is labeled on each edge.

Consider the following task. Suppose that both bodies are initially in r_1, as shown in Figure 5(a). The task is to bring them to r_3, then r_2, and then return to r_1. Suppose that propositions π_i means that both bodies are in r_i. The corresponding LTL formula is

$$\Diamond(\pi_1 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_2 \wedge \Diamond\pi_1))). \tag{3}$$

A possible solution trajectory for D_2 is depicted in Figure 6 as a sequence of body distributions for which transitions are caused by setting gate directions.

V. Experiments

We performed several experiments on low-cost hardware to illustrate the methodology and to show its practical feasibility.

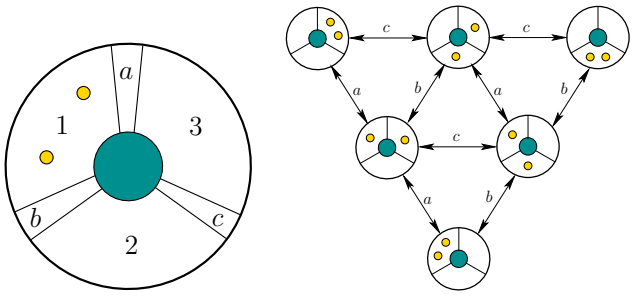


Fig. 5. a) An example with three regions, three gates, and two bodies; b) a graph that for which the vertices are C, the set of possible distributions, and the edges correspond to possible transitions.

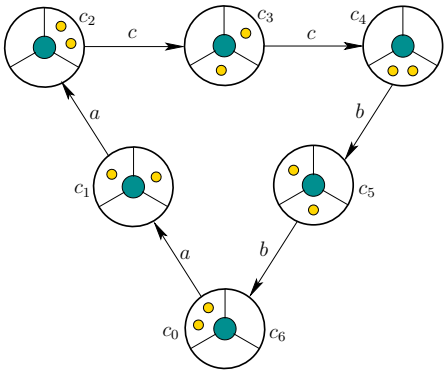


Fig. 6. An example trajectory that satisfies the LTL formula given in (3). A sequence $(c_0, \ldots, c_6)$ of distributions is visited.

The printed frames in this section do not due justice to the execution of the system. Full videos appear at

`http://msl.cs.uiuc.edu/rss11/`

The hardware

Recall from Figure 1 in Section I that we implement the wild bodies using weasel balls. Each one costs around $4 US and consists of a plastic ball of radius 8.5cm that has only a single offset motor inside that oscillates at about 2Hz We performed hundreds of experiments that consisted of placing one or more balls into regions and observing their motions. Without fail, they are easily able to strike our gates, which will be explained shortly. Therefore, we believe the sufficient satisfy our condition of wildness from Section II to be suitable for our experiments.

Now consider the design of gates. Various kinds are introduced in [4]. As mentioned at the end of Section III, some tasks do not require changing the gate mode during execution. For these cases, we can implement a *static gate* [4], which allows one-way motion from region to region and is fixed in advance.

A simple way to engineer a successful static gate is illustrated in Figure 7(a). A body moving from the bottom region to the top region can pass through the right side by bending the paper; a body moving in the other direction is blocked.

Fig. 7. a) A static, directional gate can be implemented making a flexible "door" from a stack of paper; in this case, the body can transition only from the bottom region to the top; b) this works much like a "doggie door".

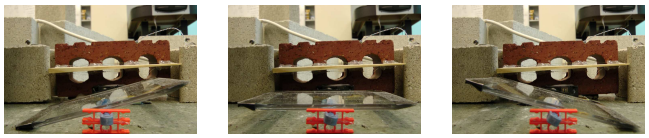


Fig. 8. The three gate configurations: a) the gate allows a body to cross in the left to right direction, b) the gate prevents bodies from crossing in either direction, and c) the gate allows an body to cross in the right to left direction.

This simple setup proved to be reliable in implementing the directional gate in some of our experiments.

For many experiments involving translating LTL formulas into low level controllers, static gates are insufficient. In these cases, we need be able to control the gate modes externally during execution based on sensor feedback. We will call this a *controllable gate*.

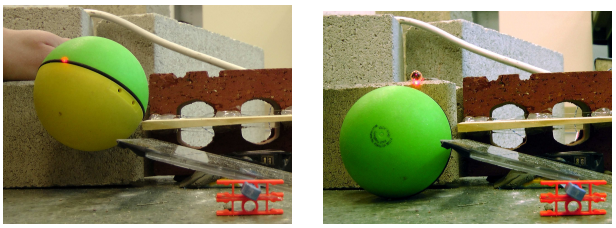
Our controllable gate is made from a piece of acrylic in the form of a *ramp*. By tilting the ramp, the direction of the gate is altered, and we can obtain three gate configurations to execute the gate actions, as seen in Figure 8.

The acrylic ramp element is attached to Futaba S3003 servo motors using standard servo horns. Servo motors were chosen for this application because they are inexpensive (around $8 US each) and allow precise control of output angle by the use of negative feedback. Additionally, the only control input required is a Pulse-Width Modulation (PWM) signal, which is easily generated by most microcontrollers.

Now consider the simple sensor feedback mentioned in the previous sections. Body crossing feedback is achieved through the use of optical emitter-detector pairs. Laser pointers were chosen because they are inexpensive (about $3 US each) and easily aimed. The laser pointers were modified to run on external battery packs and held in place by simple armature mounts (about $3 US each). Simple photodiodes (about $2 US each) were mounted on the opposite side to detect the laser beams.

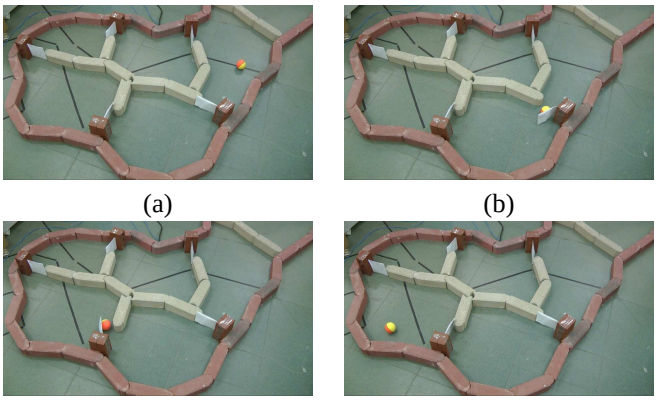
A change in voltage is observed when a body crosses the beam, thereby blocking the laser beam from reaching the photodetector. As can be seen in Figure 9, the laser beam/photodetector pairs are placed so that only an body which has just crossed a gate causes a beam crossing.

As previously mentioned, the ramp-type gates are imple-



(a) (b)

Fig. 9. a) A ball that has just crossed the gate interrupts the laser beam, while b) a body simply moving within a region does not interrupt the laser beam.



(a) (b)

(c) (d)

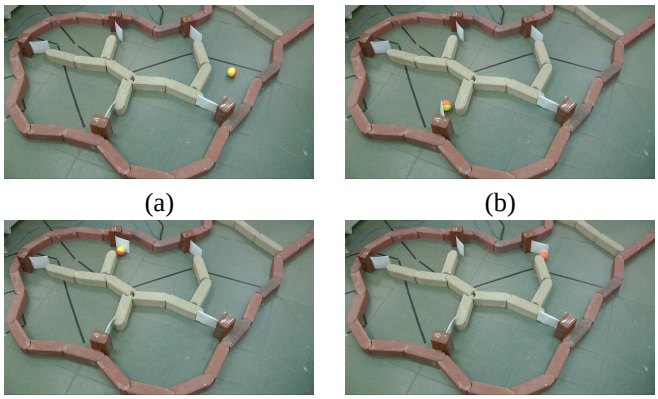
Fig. 10. "Starting in r_2, go to r_4": a) The weasel ball is placed initially in r_4 (leftmost); b) after 30 seconds strikes gate d and enters r_3; c) after 105 seconds it strikes gate e and d) moves into r_4, which completes the task.

mented using servo motors. The angular position of these servo motors is determined by the duty cycle of the PWM signal they receive. For this purpose, we used an Arduino Mega microcontroller board based on the Atmel ATmega1280 microcontroller. This platform was chosen as it is easy to configure and inexpensive (about $35 US), Additionally, Arduino documentation and code examples are plentiful.

Single body experiments

We show several experiments for a single weasel ball. We chose typical tasks specified using LTL, as mentioned in Section II and [26]. Even though all experiments can be easily implemented using controllable gates, we use static gates whenever possible to show the simplest implementation.

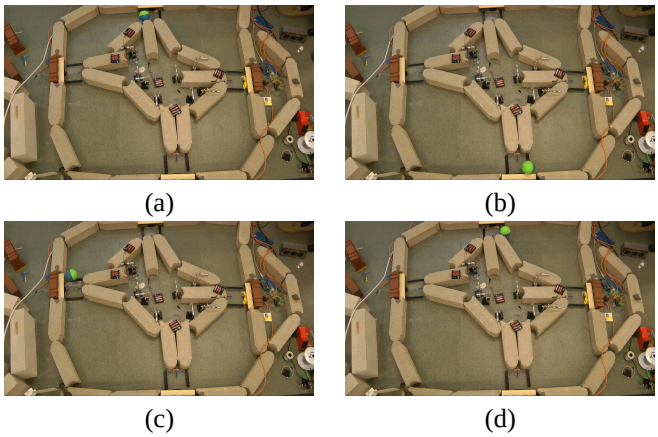
We implemented the navigation approach for a weasel ball in an environment of approximately 2 by 3 meters and five gates; see Figure 10. For the region and gate names, recall Figure 4. The specification of the task that we would like to achieve is: "Starting in r_2, go to r_4". An LTL formula that captures this specification is $\Diamond\pi_4$. We entered the discrete transition system and the LTL specification using the model checker NuSMV [9]. The output region sequence implies that gates d and e are enabled to allow transitions from r_2 to r_3 and from r_3 to r_4. The experimental execution is shown in Figure 10.



(a) (b)

(c) (d)

Fig. 11. "Patrol regions r_0, r_3 and r_1": a) The ball starts its route; b) after 107 seconds it has entered two new regions; c) after 212 seconds it has visited most regions; d) after 225 seconds, it completes a tour of all regions, and continues.



(a) (b)

(c) (d)

Fig. 12. A programmable coverage task: a) The body crosses into the upper-left region; b) after 15 seconds, the body crosses into the lower-right region, completing the coverage; c) after 50 seconds, the body crosses into the upper-left region on the return trip; d) after 240 seconds, the body returns to the upper-right region.

We also created an experiment to demonstrate patrolling. We defined the LTL formula

$$\Box(\Diamond\pi_0 \wedge \Diamond\pi_3 \wedge \Diamond\pi_1), \tag{4}$$

and an infinite trajectory was found by the model checker. A gate configuration that implements the sequence is shown in Figure 11 along with part of the actual execution. The ball visits attemps to visit the required regions infinitely often (in reality, its battery dies).

The next example uses programmable gates to implement sequencing. See Figure 12. Suppose that we want to visit regions in the following order: r_0 (upper right), r_1 (upper left), r_2 (lower left), r_3 (lower right), r_2, r_1, r_0.

The LTL formula to achieve this is

$$\Diamond(\pi_0 \wedge \Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_1 \wedge \Diamond\pi_0)))))). \tag{5}$$

The experiment for this example appears in Figure 12.

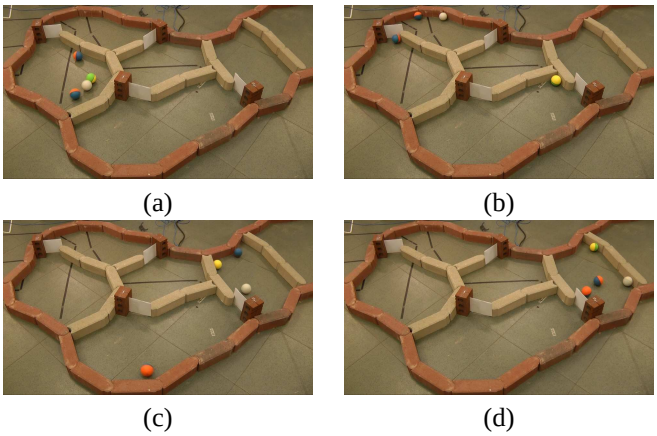


Fig. 13. Navigation with multiple balls: a) Four weasel balls are started in the left-most region; b) after 48 seconds some progress is made; c) after 262 seconds, all but one ball have arrived at the destination; after 270 seconds, all four balls have arrived.

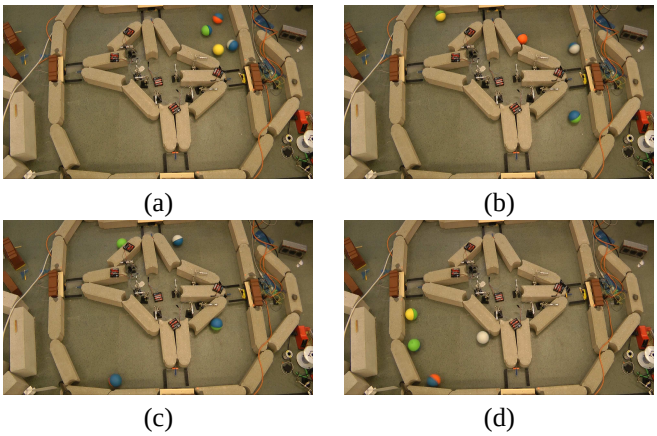


Fig. 15. A group splitting and coverage example: a) The 4 bodies begin together in the upper-right region; b) after 37 seconds the bodies begin to split; c) after 45 seconds bodies have split completely into independent regions; d) after 240 seconds bodies reconvene in the lower-left region.

Fig. 14. In this experiment, 50 weaselballs were successfully manipulated from a source region into a destination region.

Multiple bodies

The programmable get setup shown in Figures 8 and 9 is sufficient to implement any sequence of body distributions produced by a model checker. In cases, however, for which a cheaper setup of static gates suffices to satisfy the LTL formula, we implemented that instead. We solved a navigation task with 4 bodies using the specification: "Move all four bodies from r_4 (leftmost) to r_2 (rightmost)"; see Figure 13.

In another experiment, shown in Figure 14, we moved 50 weasel balls from from a starting to a goal region, in an environment with 6 regions and 6 gates. The regions are complicated shapes, some with interior obstacles, and the gates are narrow. It took around 40 minutes for all 50 balls to arrive in the goal due in part to a long tail distribution on arrivals.

Using the programmable gates, we implemented more complicated tasks, such as: "Starting with all four bodies in r_0 (upper-right), cover all four regions simultaneously and then meet again in r_3 (lower-right)"; see Figure 15.

Conclusions and Future Work

We have presented a methodology to translate Linear Temporal Logic formulas to low level controllers for simple bodies that achieve tasks such as navigation, patrolling, and coverage. A unique aspect of our approach is that the bodies behave wildly and cannot be directly controlled. The desired behavior is induced through the use of controllable gates that gently guide them from region to region. This avoids many traditional issues such as heaving sensing, state estimation, state feedback, system identification, communication, and coordination. The system was implemented using low-cost, widely available hardware and dozens of experiments were performed.

One direction for future research, which we have already begun to explore is the use of other platforms for implementing our methodology. We have performed some experiments in which the vibrating Hexbug Nano toy was controlled from region to region using simple directional gates. We have also developed a small differential drive robot for about $30 US that moves straight, contacts a boundary, rotates a random amount, and then moves straight again. This behavior again seems sufficiently wild to yield the desired performance. An important direction of future research is to analyze the time it takes to enter the gate for various motion models, region shapes, and gate widths. Can objective criteria be formulated for the motion and then optimized through a simple motion strategy for the body? Furthermore, statistical analysis might enable us to predict the expected time to completion for a task, which is currently a weakness of our approach.

To achieve more useful tasks, we envision enhancing the bodies with limited amounts of sensing, controllable actuation, and computation. As a step in this direction, we have equipped one weasel ball with a small WiFi module and microcontroller, allowing it to use Wi-Fi connections while wilding moving around. This enables more interesting tasks to be performed, such as Wi-Fi SLAM [15]. We imagine that a collection of wild bodies would be useful for exploration and mapping if

equipped with appropriate sensors for this purpose. As another task, we could equip each body with an Annoy-a-tron circuit board, which costs $13 US and emits a loud, piercing sound at irregular intervals, without warning. We could program the bodies to diffuse in a hostile indoor environment and then switch into an "annoy" mode during which the building inhabitants are constantly distracted by tiny devices stationed in unknown locations.

We are also considering other gate and sensor designs. Several gate varieties are shown in [4], including *pliant* gates with change their mode using only the energy from the ball. Very interesting behaviors can be prescribed in this way (imagine a compliant revolving door). Furthermore, there are many ways to make "virtual" gates, much in the same way that artificial walls can be set up when using the popular Roomba vacuum. We have performed some early experiments in which an iRobot Create equipped with a cheap color sensor can move over colored tape on the floor, deciding whether to "bounce" from the tape or pass through it, depending on the mode. The tape and color sensor simulate the gate.

Acknowledgments: This work was supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052. The authors are grateful for the helpful suggestions of Hadas Kress-Gazit and the anonymous reviewers.

REFERENCES

[1] R. Alur, T. A Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971984, 2002.

[2] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion [Grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):6170, 2007.

[3] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 2689–2696, 2010.

[4] L. Bobadilla, K. Gossman, and S. M. LaValle. Manipulating ergodic bodies through gentle guidance. In *RoMoCo 2011 : 8th Workshop on Robot Motion and Control*, 2011.

[5] K.-F. Böhringer, V. Bhatt, B. R. Donald, and K. Goldberg. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26:389–429, 2000.

[6] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[7] Z. Butler, P. Corke, R. Peterson, and D. Rus. Virtual fences for controlling cows. In *IEEE International Conference on Robotics and Automation*, page 44294436, 2004.

[8] L. G. Chalmet, R. L. Francis, and P. B. Saunders. Network models for building evacuation. *Fire Technology*, 18(1):90113, 1982.

[9] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, page 241268, 2002.

[10] E. A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, 8:9951072, 1990.

[11] M. A. Erdmann. Randomization in robot tasks. *International Journal of Robotics Research*, 11(5):399–436, October 1992.

[12] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics & Automation*, 4(4):369–379, August 1988.

[13] G. E. Fainekos. Revising temporal logic specifications for motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, 2011.

[14] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic mobile robots. *Automatica*, 45(2):343–352, February 2009.

[15] B. Ferris, D. Haehnel, and D. Fox. WiFi-SLAM using Gaussian process latent variable models. In *International Joint Conference on Artificial Intelligence*, 2007.

[16] R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 157–162, 2001.

[17] C. Finucane, G. Jing, and H. Kress-Gazit. LTLMoP: Experimenting with language, temporal logic and robot control. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1988–1993, 2010.

[18] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicles motion planning. Technical Report LIDS-P-2468, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1999.

[19] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[20] G. J. Holzmann. *The SPIN model checker: Primer and reference manual*. Addison Wesley Publishing Company, 2004.

[21] M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *Robotics, IEEE Transactions on*, 23(2):320330, 2007.

[22] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics and Automation*, 26(1):48–61, 2010.

[23] H. Kress-Gazit. *Transforming high level tasks to low level controllers*. PhD thesis, University of Pennsylvania, 2008.

[24] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.

[25] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics and Automation*, 25(6):1370–1381, December 2009.

[26] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.

[27] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Where's Waldo? sensor-based temporal logic motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, 2007.

[28] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3227–3232, 2010.

[29] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.

[30] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on ltl specifications. In *Proceedings IEEE Conference Decision and Control*, page 153158, 2005.

[31] M. T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, 2001.

[32] D. Reznik, E. Moshkoich, and J. Canny. Building a universal planar manipulator. In K. F. Bohringer and H. Choset, editors, *Distributed Manipulation*, page 147171. Kluwer, Norwell, MA, 2000.

[33] D. A. Shell, C. V. Jones, and M. J. Mataric. Ergodic dynamics by design: A route to predictable multirobot systems. In *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 291–297, 2005.

[34] S. L. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning under temporal logic constraints. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3288–3293, 2010.

[35] S. Tabachnikov. *Geometry and Billiards*. American Mathematical Society, Providence, Rhode Island, 2005.

[36] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *Automatic Control, IEEE Transactions on*, 51(12):18621877, 2006.

[37] M. Wu, G. Yan, Z. Lin, and Y. Lan. Synthesis of output feedback control for motion planning based on LTL specifications. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5071–5075, 2009.