Resilient Task Planning and Execution for Reactive Soft Robots

Scott Hamill, John Whitehead, Peter Ferenz, Robert F. Shepherd, and Hadas Kress-Gazit[1]

***Abstract*— Soft robots utilize compliant materials to perform motions and behaviors not typically achievable by rigid bodied systems. These materials and soft actuator fabrication methods have been leveraged to create multigait walking soft robots. However, soft materials are prone to failure, restricting the ability of soft robots to accomplish tasks. In this work we address the problem of generating reactive controllers for multigait walking soft robots that are resilient to actuator failure by applying methods of formal synthesis. We present a sensing-based abstraction for actuator performance, provide a framework for encoding multigait behavior and actuator failure in Linear Temporal Logic (LTL), and demonstrate synthesized controllers on a physical soft robot.**

I. Background and Introduction

Biological systems often rely on compliant and soft tissues to perform complex motions and interact with the environment in ways that would otherwise be impossible with rigid bodied systems. Soft robotic systems seek to emulate these behaviors through the use of flexible, continuously deformable materials. Incorporating these compliant materials into soft actuators and chassis components allows soft robots to perform tasks and operate in unstructured environments inaccessible to rigid robotic systems.

In this work, we are interested in the behavior of fluidic soft actuators - those in which deformation of a compliant actuator is driven by pressurization of internal chambers. Methods used to fabricate these actuators, such as casting silicone elastomers [1]–[5] and lithographic processes with photopolymers [6]–[8], provide an expansive design space that can be exploited to create novel actuator designs and robots capable of unique methods of locomotion [9]–[11].

Previous research has leveraged this expansive design space to create walking soft robotic systems. Walking soft robots achieve motion by pressurizing a set of soft actuators in contact with a ground surface in a sequence, which we refer to as a gait. Walking soft robots have been developed that demonstrate multigait behavior - having more than one available pressurization sequence that propels the robot. The authors of [3] and [6] both present walking soft robots for which multiple gaits have been developed, and in [5] we present a method for automatically synthesizing gaits for arbitrarily constructed, modular soft robot systems.

However, the compliant materials used in soft actuators are prone to failure, either by rupturing or by developing leaks due to small tears in the material. Ruptures and leaks may prevent complete actuation, thereby restricting the ability of the robot to move through a workspace and preventing the robot from achieving its tasks. The performance of soft actuators is difficult to predict as compliant, soft materials are difficult to model. However, methods have been developed for measuring the deformation of soft actuators [12], [13]. The ability to monitor soft actuator performance allows a multigait robot to detect and react to actuator failure.

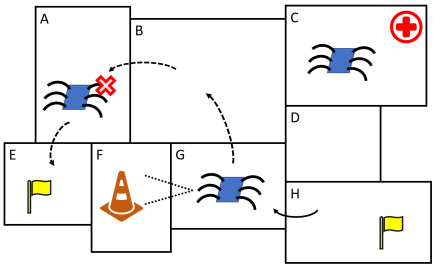

Fig. 1. Example: A multigait robot patrols between regions E and H. The robot experiences an actuator failure in A as it moves around the hazard in F on the way to E. The robot must utilize different gaits in order to reach the current goal in region E as well as replacement components in C.

In this work, we generate high-level controllers that allow soft robotic systems to react to environmental events such as actuator failure. Specifically, we address the following problem: given an environment, multigait robot, and a mission specification, generate a controller that is *resilient* to actuator failure - that is, generate a controller that allows the robot to leverage gait redundancy to fulfill the mission specification despite actuator failure.

Example: Consider a scenario in which a multigait, legged robot is tasked with continuously patrolling between the regions E and H in the environment shown in Fig. 1. The legs of the robot are replaceable and spare components are available in region C. While moving toward region E, the robot observes a hazard in region F and must take a more circuitous route through B and A. The robot experiences an actuator failure in A and must subsequently utilize different gaits in such a way as to reach region E while still being able to return to C for a replacement actuator.

Previous research has addressed the issue of gait resilience with respect to walking robots [14], [15] as well as soft robots [16], but these methods do not provide guarantees of robot performance. Here, we address the problem of generating resilient controllers by utilizing methods of formal synthesis. Recent research has addressed the problem of generating controllers for high-level robot behaviors [17]. These methods use discrete abstractions of robot and adversarial environment behaviors, capture these behaviors in Linear Temporal Logic (LTL), and provide techniques for synthesizing correct-by-construction controllers that guarantee the ability of the robot to accomplish its tasks. Within the context of

[1] Sibley School of Mechanical and Aerospace Engineering, Cornell University, {sbh92,jdw268,pmf58,rfs247,hadaskg}@cornell.edu. This work was supported by NSF CMMI-1745139 and EFMA-1830924.

resilience, similar techniques have been applied to different domains such as vehicle power system management [18], [19]. These works address synthesizing high-level controllers that guarantee system performance despite generator failure, but utilize different system abstractions.

High-level abstractions and synthesis methods provide an expressive and flexible framework for encoding robot and environmental behaviors. As shown in Sections V and VI, variations in actuator reliability and environmental conditions such as hazards are easily encoded, and these differences produce markedly different, nuanced robot behaviors.

In this work, we present three contributions: 1) a sensing-based abstraction of actuator health and failure, 2) a framework for encoding multigait, reactive behavior in LTL, and 3) a demonstration of synthesized, correct-by-construction controllers for a soft robotic system.

II. Preliminaries

The following section provides a brief overview of Linear Temporal Logic (LTL) synthesis for high-level robot control. The reader is referred to [17] for a more in depth discussion.

A. High-Level Control

We address the issue of generating a controller for high-level robot behavior by posing the problem as a two player game between the *system* (the robot) and an adversarial *environment*. The environment controls the sensors of the robot, i.e. how the system perceives the state of the world, and the robot reacts by performing actions. The goal is to synthesize a strategy for the system such that the robot is guaranteed to be able to satisfy some behavioral requirement.

We define two sets of Boolean propositions: those controlled by the environment, $\mathcal{X}$, and those controlled by the system, $\mathcal{Y}$. The set of all atomic propositions is $AP = \mathcal{X} \cup \mathcal{Y}$.

B. Linear Temporal Logic Syntax and Semantics

Linear Temporal Logic [20] is a formal language that contains the Boolean operators $\neg$ ("not") and $\wedge$ ("and"), as well as the temporal operators $\bigcirc$ ("next") and $\mathcal{U}$ ("until"). LTL formulas, defined over AP, are defined recursively as:

$$\varphi ::= \pi \in AP \mid true \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \quad (1)$$

The operators $\vee$ ("or"), $\rightarrow$ ("implies"), and $\leftrightarrow$ ("bi-implication") are derived from $\wedge$ and $\neg$. The operators $\Box$ ("always") and $\Diamond$ ("eventually") are derived from $\bigcirc$ and $\mathcal{U}$.

LTL formulas are evaluated over an infinite sequence of truth assignments, $\sigma = \sigma_0\sigma_1\sigma_2...$, such that $\sigma_i \in 2^{AP}$. The expression σ_i denotes the propositions in AP that are true at position i.

The formula $\Box\varphi$ is true at position i if the formula is true at all positions $j \geq i$ in σ. The formula $\Diamond\varphi$ is true at position i if the formula is true in at least one position $j \geq i$ in σ. The formula $\bigcirc\varphi$ is true at position i if the formula is true in the next position $i + 1$ in σ.

Mission Specification: The LTL specification used in this work is of the following form [21]:

$$\varphi = (\varphi_i^e \wedge \varphi_t^e \wedge \varphi_g^e) \rightarrow (\varphi_i^s \wedge \varphi_t^s \wedge \varphi_g^s) \quad (2)$$

The specifications φ_i^e and φ_i^s are Boolean formulas and are the initial conditions of the environment and system, respectively. The specifications φ_t^e and φ_t^s are the environment assumptions and system guarantees. These are of the form $\wedge_{m\in M}\Box C_m$, where C_m are Boolean formulas that may contain the $\bigcirc$ operator. The specifications φ_g^e and φ_g^s are the environment liveness assumptions and system liveness guarantees (the environment and system goals), respectively. These specifications are of the form $\wedge_{o\in O}\Box\Diamond D_o$, where D_o are Boolean formulas.

Controller Synthesis: The synthesis process is presented in detail in [21]. The resulting controller is a finite state automaton (FSA) of the form: $A = (\mathcal{X}, \mathcal{Y}, Q, Q_0, \delta, L)$ where:

- $\mathcal{X}$ is the set of environment propositions
- $\mathcal{Y}$ is the set of system propositions
- Q is the set of states
- $Q_0 \subseteq Q$ is the set of initial states
- $\delta : Q \times 2^{\mathcal{X}} \rightarrow Q$ is a transition function
- $L : Q \rightarrow 2^{\mathcal{Y}}$ is the labeling function of the states

The function δ is the system transition relation that maps a current state and environment proposition values to the next state. The function L maps a state to the system propositions that are true in that state. The specification is said to be realizable if there exists a solution such that the specification is satisfied regardless of the behavior of the environment. In this work we use the synthesis tool SLUGS [22].

III. Approach

A. Abstracting Multigait Behavior

Environment: We assume the robot workspace consists of a finite number of regions, $Reg = \{R_1, ..., R_p\}$. The location of the robot is represented by a set of p Boolean propositions, $reg = \{r_1, ..., r_p\} \subset \mathcal{Y}$. The robot may only occupy one region at a time. The adjacency relation, $\tau \subseteq Reg \times Reg$ gives the possible region transitions.

Robot Gaits: A multigait robot is able to select a gait from a set of m different gaits, $G = \{G_1, ..., G_m\}$. Gait selection is represented by the set of propositions $gaits = \{g_1, ..., g_m\} \subset \mathcal{Y}$, such that the proposition g_i is true if and only if the robot is using gait G_i. Gaits are mutually exclusive as the robot may only utilize one gait at a time. Each gait is directional and propels the robot in a given direction with respect to a local reference frame. In order to capture the directionality of each gait, we define a mapping function, $\mathcal{D}_T : \tau \rightarrow 2^G$ that maps a region transition to the gaits capable of making the transition. In order to avoid a situation in which no valid gait is available, we add a proposition *noGait* to the set $gaits$. In this work we abuse notation slightly by interchangeably using G_i and g_i as well as R_i and r_i for clarity.

B. Abstracting Actuator Sensing and Health

Actuators: The set of all n leg actuators used by the robot is denoted by the set $Legs = \{L_1, ..., L_n\}$. Not all gaits utilize all actuators. As such, we define a mapping function, $\mathcal{L}_G : G \rightarrow 2^{Legs}$, that maps gait G_i to a subset of $Legs$ used by that gait. These are the actuators that are part of the pressurization cycle for gait G_i. In addition, we define a mapping function, $\mathcal{G}_L : Legs \rightarrow 2^G$, that maps each actuator L_i to the subset of gaits in G that utilize that actuator.

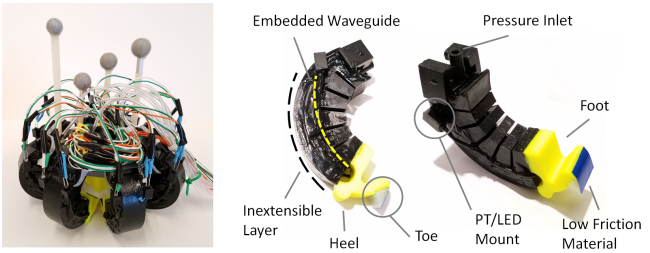

Fig. 2. Left: physical system. Eight actuators are arranged in a radial pattern around a chassis. The silver spheres are markers for a Vicon motion capture system. Right: actuator design.

Actuator Sensors: Each actuator is associated with an integer number of *health states*. The set of health states for each of the n actuators is denoted by the set $H = \{h_1, ..., h_n\}$. For each actuator i, we define a set of Boolean propositions, P_i, that describes its health state: $P_i = \{\cup_{(j=1,...,h_i)} p_{i,j}\} \subset \mathcal{X}$. The propositions in each set P_i are mutually exclusive. Actuator health stages degrade (change state) in a consecutive manner, e.g. if proposition p_{i,h_i} is true in one state and degradation is sensed, at the next time step proposition p_{i,h_i-1} is true and p_{i,h_i} is false. If proposition p_{i,h_i} is true, the actuator is considered to be in perfect working condition, i.e. "healthy." If the proposition $p_{i,1}$ is true, the actuator is considered to be inoperable. We assume that the health of an actuator may degrade any time the robot uses that actuator to make a transition between two regions.

Cache Regions: We specify certain regions of the workspace, $Reg_c \subseteq Reg$, as *cache regions* where the robot is able to access new components, thereby allowing the robot to exchange damaged actuators. When the robot reaches one of these regions, the status of each actuator is reset to "healthy." The set Reg_c may be empty, which may affect the ability of the robot to satisfy a mission specification.

IV. Robot Design and Gaits

A. Robot and Actuator Design

The robot design is shown in Fig. 2 (left). The robot is composed of eight actuators mounted radially to a rigid polyurethane (Carbon® RPU 70) plastic frame printed on a Carbon® M1 printer. Each actuator is attached to a tab on the chassis and is easily replaced. Embedded in each actuator is a pressure chamber, flow to which is controlled by a solenoid valve. All pressure generation and control is off-board.

Each actuator, one of which is shown in Fig. 2 (right), is comprised of two parts: a body and a "foot." The components of the body, printed on the same Carbon® printer as the chassis, are elastomeric polyurethane (Carbon® EPU 40). The pressure chamber of the body is comprised of a bellowed surface attached to a thick backing surface. The foot is further comprised of three components: a rigid, ABS plastic 3D printed foot, a "toe" made from silicone rubber (Smooth-On Dragon Skin® 20), and a low friction material adhered to one side of the toe.

B. Actuator Motion and Gaits

Each actuator is connected via a pneumatic channel and solenoid valve to a common pressure rail. The thickness of the backing side of the actuator is significantly greater than that of the bellowed section, in effect creating an inextensible layer (Fig. 2). As each actuator is pressurized, the bellowed section expands and the inextensible material in the outer layer causes the actuator to unfurl. As the actuator unfurls, the higher-friction silicon material contacts the ground, imparting a force on the ground surface. The resulting chassis motion is shown in Fig. 3. The low friction patch of material on one side of the toe allows the actuator to return to its resting position without further disrupting the chassis. The yellow ABS protrusion, the "heel," provides a low friction contact point when each actuator is at rest, allowing the chassis to slide as other actuators are pressurized.

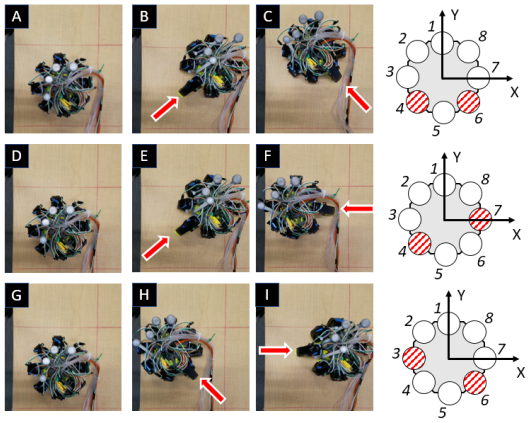

Fig. 3. Sequences A-C, D-F, and G-I demonstrate three different gaits propelling the robot in the positive Y direction in the local frame. Shown right are the actuators used in each gait and the corresponding indices.

As previously discussed, a gait is an actuator pressurization sequence - a subset of the actuators on the robot are pressurized according to a predetermined timing that produces chassis motion. The octopod robot used in this work is capable of three different gait motion primitives, developed empirically, depicted in Fig. 3. Each gait utilizes two actuators pressurized sequentially, and the rail pressure and solenoid timing were hand tuned. For example, as shown in Fig. 3 D-F, the robot uses actuator 4 to push the chassis diagonally in the positive X/positive Y direction, and then uses actuator 7 to push the robot in the negative X direction back toward the center of the image. The net displacement is in the positive Y direction. As the robot is radially symmetric, these gait primitives are able to push the robot in four directions in a body-fixed reference frame. In total there are twelve gaits available to the robot. Note - adjusting the timing of each primitive slightly and actuating both actuators simultaneously causes the chassis to rotate in place. This adjustment cycle is solely used to correct orientation errors, and it is not part of the standard gait selection as it is assumed that no actuators fail during this cycle.

C. Detecting Curvature and Actuator Degradation

Adhered to the side of each actuator is an optical waveguide, marked in Fig. 2 by a dotted yellow line. We leverage [13] to provide actuator performance sensing; there, optical waveguides were embedded in the fingers of a soft pros-

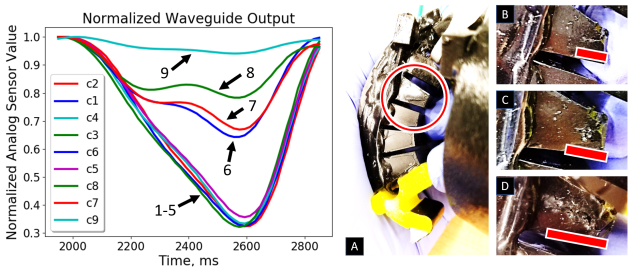

Fig. 4. Left: normalized waveguide data. Right: **A**-**D** detail induced, progressive damage to an actuator (imparted by an X-ACTO blade).

thetic hand. An infrared LED acts as a light source and a phototransistor (PT) detects light intensity. As the hand deforms, the waveguides experience strain and the light transmissivity decreases, providing the ability to quantify actuator deformation. We utilize the same technique in this work. Attached to the body of the actuator are two mounting locations for an infrared LED and a phototransistor (PT) for detecting waveguide deformation.

The plot in Fig. 4 (left) shows the recorded PT intensity values for a waveguide on an actuator during several actuation cycles. The intensity analog values are filtered and normalized with respect to the at-rest sensor output value for each individual cycle. Degradation and failure are determined by the net change in the PT analog signal during actuation.

The plotted curves depict the reduction in performance of an increasingly damaged actuator. Starting with a healthy actuator, the robot executed the same gait several times while curvature data was recorded. After each gait sequence, an X-ACTO blade was used to deliberately damage one of the bellowed sections of the actuator body, as shown in Fig. 4 (right), A-D. The curvature behaviors exhibited during the first five cycles are all similar - these data correspond approximately with Fig. 4 B. A small cut in the material caused an audible air leak, but ultimately no significant degradation in performance was observed. Cycles 6, 7, and 8 roughly correspond with the amount of damage depicted in Fig. 4 C. In this case, the cut was expanded and enlarged. Although the actuator was observed to be less effective during these event cycles, the actuator was still capable of moving the robot. Cycle 9 corresponds with the damage shown in Fig. 4 D. In this case, the cut was lengthened and expanded significantly. The corresponding waveguide output plot clearly shows a distinguishable degradation in performance and the actuator was observed to be ineffectual in propelling the robot.

In this specific case, the plots of actuator degradation can be grouped into three sets - cycles 1-5, for which no degradation was detected, cycles 6-8, for which slight degradation was detected, and cycle 9, for which the actuator was deemed inoperable. As such, this particular actuator experiencing the damage shown in Fig. 4 (B-D) could be considered to have 3 actuator health states, $h_i = 3$. This data represents the behavior of one actuator with respect to a very specific type of damage - thoroughly validating this method for different actuators in different operating conditions will require further testing.

V. Encoding Robot Behavior

A. Region Transitions and Gait Selection

We encode region transition and gait selection behavior by appending the following LTL formulas to φ_t^s. Formulas encoding proposition mutual exclusivity are omitted for brevity.

Region Transitions: Each gait propels the robot in a certain direction and allows the robot to make certain region transitions. We encode the region transition behavior in the following way:

$$\bigwedge_{r_s \in reg} \square \left(r_s \rightarrow \bigcirc \left(\bigvee_{\{r_e | (r_s, r_e) \in \tau\}} \quad \bigvee_{g \in \mathcal{D}_T(r_s, r_e)} r_e \wedge g \right) \right) \tag{3}$$

If the robot is in region r_s, at the next time step the robot must transition to one of the regions allowed by the transition relation τ, and must choose a gait that is compatible with that direction of motion. If there are no available gaits for a given transition in τ, the robot may not move to that region. Self transitions, associated with *noGait*, are included in τ.

Gait Restrictions: For any inoperable actuator, the robot is restricted from selecting any gait that utilizes that actuator. These restrictions are encoded in the following way:

$$\bigwedge_{L_i \in Legs} \square \left(\bigcirc p_{i,1} \rightarrow \left(\bigwedge_{g \in \mathcal{G}_L(L_i)} \neg \bigcirc g \right) \right) \tag{4}$$

B. Actuator Health

We encode actuator degradation and replacement by appending the following LTL formulas to φ_t^e.

Actuator Degradation: The environment controls the state of each actuator but is only allowed to influence the state of an actuator used by the currently selected gait as the others are not pressurized. This behavior is captured in the following way:

$$\bigwedge_{g \in gaits} \square \left(\left(g \wedge \neg \left(\bigvee_{r \in reg_c} r \right) \right) \rightarrow \left(\bigwedge_{L_i \in \mathcal{L}_G(g)} \varphi_{D_i} \wedge \bigwedge_{L_k \notin \mathcal{L}_G(g)} \varphi_{D_k} \right) \right) \tag{5}$$

where φ_{D_i}, and φ_{D_k} are defined as:

$$\varphi_{D_i} = \left(\bigwedge_{j=h_i, h_i-1, \ldots, 2} (p_{i,j} \rightarrow (\bigcirc p_{i,j} \vee \bigcirc p_{i,j-1})) \right) \wedge (p_{i,1} \rightarrow \bigcirc p_{i,1})$$

$$\varphi_{D_k} = \bigwedge_{j=h_k, h_k-1, \ldots, 1} (p_{k,j} \leftrightarrow \bigcirc p_{k,j})$$

For those actuators that are associated with the currently selected gait, assuming the robot is not in a cache region, the environment may only alter the states of the actuator health propositions according to the rule of consecutive degradation previously discussed.

Cache Regions: If the robot moves into a cache region, in the next time step the status of all actuators is set to "healthy," indicating that any failed actuators have been replaced:

$$\square \left(\left(\bigvee_{r \in reg_c} r \right) \rightarrow \left(\bigwedge_{i=1,\ldots,n} \bigcirc p_{i,h_i} \right) \right) \tag{6}$$

Appending these statements to the mission specification captures the high-level, gait switching behavior of a multigait

robot and, if the specification is realizable, ensures the resilience of the robot to actuator failure.

C. Example Scenario

Fig. 5 (left) depicts the example scenario we address in this work. The robot operates in a grid and is able to move "north," "south," "east," or "west" at each time step. The mission specification is a patrolling task - the robot must repeatedly visit two of the regions in the map, regions 11 and 12, depicted with yellow diamonds. This is encoded by appending the following to φ_g^s:

$$(\square\diamond r_{11}) \wedge (\square\diamond r_{12}) \tag{7}$$

Actuator caches are located in regions 3 and 14 and are marked with green circles.

In this example we show how one can encode more complex environment behaviors and tasks. We specify sets of regions of the workspace that contain hazards that are detectable by the robot and may be "activated" or "deactivated" by the environment. If the robot moves into an active hazard region, all of the actuators associated with the currently selected gait are immediately rendered inoperable regardless of health state.

Formally, we specify v sets of hazard regions by the set $T = \{T_1, ..., T_v\}$. We define a mapping function $\mathcal{F}_T : T \to 2^{Reg}$, that maps each hazard set to a set of workspace regions. The hazard sets are associated with a set of Boolean propositions, $t = \{t_1, ..., t_v\} \subset \mathcal{X}$. If the proposition t_i is true, the robot senses active hazards in each of the regions in $\mathcal{F}_T(T_i)$. We assume the propositions in t are mutually exclusive and the set of hazard regions and set of cache regions are disjoint.

In order to capture actuator damage due to hazards in the specification, $(\wedge\neg\varphi_H)$ is added to the antecedent in Eq. 5:

$$\varphi_H = \left(\bigvee_{t_k \in t} t_k \wedge \left(\bigvee_{r \in \mathcal{F}_T(t_k)} r\right)\right) \tag{8}$$

and the statement:

$$\bigwedge_{g \in gaits} \square\left(g \wedge \neg\left(\bigvee_{r \in reg_c} r\right) \wedge \varphi_H\right) \rightarrow \left(\left(\bigwedge_{L_i \in \mathcal{L}_G(g)} \bigcirc p_{i,1}\right) \wedge \bigwedge_{L_k \notin \mathcal{L}_G(g)} \varphi_{D_k}\right) \tag{9}$$

is added to φ_i^e, where φ_{D_k} is defined as before.

In this example, there are two sets of hazard regions, marked in Fig. 5 with blue checkers (regions 7, 9, 10) and red stripes (regions 2, 5, 6, 15).

VI. Demonstration

We present two different robots, **A** and **B**, with different assignments of actuator health states as shown in Fig. 5 (right). The actuators of robot **A** : $H_A = \{3, 3, 3, 3, 3, 3, 3, 3\}$, have equal levels of reliability. Those of **B** : $H_B = \{5, 2, 3, 5, 1, 5, 1, 4\}$ have asymmetrical actuator reliability. Here, two actuators only have one state, meaning that they are perpetually considered inoperable, are not affected by cache regions, and may not be used for any gait.

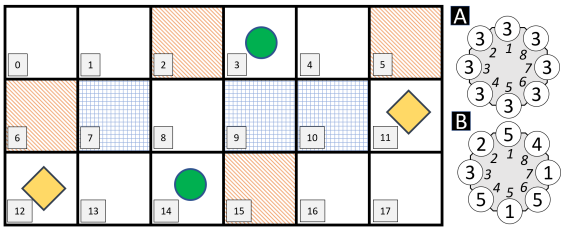

Fig. 5. Left: example grid environment. Right: actuator reliability for example robots A and B, values marked inside the circles. Italicized numbers are actuator indices.

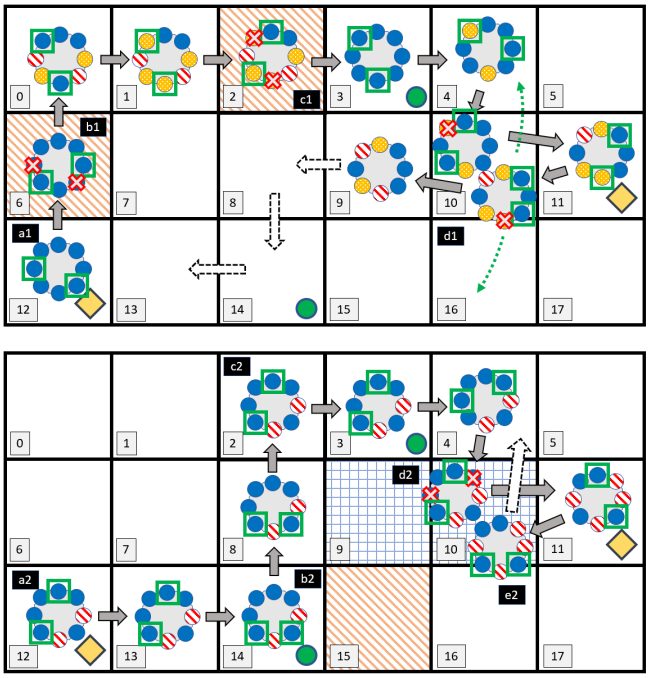

Fig. 6. Example scenario robot behavior, robot **A** shown at top, **B** bottom. The blue and red patterns depict the two sets of (active) hazard regions.

A. Robot Behaviors

The synthesis algorithm produces markedly different strategies for each robot that take advantage of the strengths and weaknesses of the robot. Robot **A** has a greater level of gait redundancy than robot **B**, meaning the robot may use different gaits with independent sets of actuators in all directions. Robot **B**, in contrast, does not have the same level of directional redundancy due to the two permanently failed actuators. However, several of the actuators of robot **B** (1, 4, and 6) have a higher number of actuator health states and, as a result, robot **B** has a greater reachable space than robot **A**, i.e. robot **B**, unimpeded by hazard regions, may travel farther than robot **A** before requiring actuator replacement.

B. Example Scenario Results and Physical Demonstration

The specifications for robots **A** and **B** are both realizable. Runs of the synthesized automata are depicted graphically in Fig. 6, and the corresponding behavior of the physical system is shown in Fig. 7. In this scenario, both robots start in region 12. In both figures, hazard regions are highlighted if they are active as the robot approaches the region (other activated hazard regions are not highlighted for clarity). In

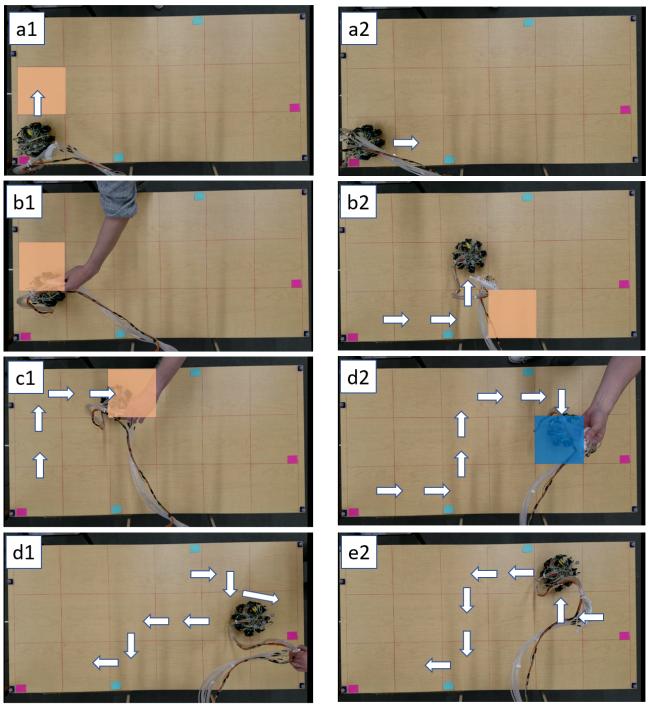

Fig. 7. Physical demonstration, Robot **A** shown left, robot **B** shown right. Image labels and active hazard highlighting correspond to instances in Fig. 6. Images *b1*, *c1*, *d1*, and *d2* show the crimping of the pneumatic lines.

the graphical depiction, active actuators are boxed in green, healthy actuators are colored blue, actuators that fail during a transition are marked with a red *X*, failed actuators are subsequently marked with red stripes, and actuators that the environment has caused to degrade are shown in yellow.

In the images of the physical demonstration, goal regions are marked with pink tags and cache regions with teal tags. In each run of the physical demonstration, the degradations sensed by the robot were simulated by manually setting the proposition values, and actuator failures were simulated by physically crimping the pneumatic channels, significantly reducing the motion of the affected actuator. No actuators were physically harmed during these demonstrations. In both demonstrations the robot successfully sensed all actuation "failure" events and reacted according to the synthesized controller. Localization was provided by a Vicon motion capture system.

Robot A: Initially, robot **A** starts in region 12, moves up to region 0, and then over to the cache in region 3 ($a1$, $b1$, and $c1$). In this scenario, the environment activates all possible hazard regions and causes actuator degradations at every possible transition, ultimately causing actuators 3, 6, 2, and 5 to fail, yet the robot takes the same route regardless as it is able to leverage gait redundancy.

The robot subsequently moves through regions 4 and 10 to the goal in 11 before moving back toward the goal in region 12. At this point, there is a bifurcation in the route depending on the state of the actuators. As the robot moves from region 11 back into region 10 (where the hazard is inactive), actuator 5, previously in a degraded state, is caused to fail ($d1$). In this instance, the robot senses the hazard in region 9 is inactive, and proceeds back to region 11 via 8 and 14 (shown as white, dashed arrows in Fig. 6). However, had the actuator not failed, the robot would have proceeded to 11 along the bottom of the map via region 16 ignoring the potential hazard in 15 (again, leveraging increased gait redundancy), and had the hazard in 10 been active, the robot would have been forced to take a more circuitous route back through the cache in region 3. Both alternate paths are shown as dashed green arrows in Fig. 6.

Robot B: In the absence of activated hazards, robot **B** would be able take a direct route to region 11. However, the robot must avoid hazards wherever possible due to the two permanently failed actuators. The robot senses an active hazard in region 15, and subsequently takes a conservative route through both cache regions ($a2$, $b2$, and $c2$).

In moving from region 3 toward region 11 however, the robot moves through region 10 despite the active status of the hazard ($d2$). As actuators 3 and 8 fail as a result, the robot must return to the cache in 3 before moving back toward the goal in 12, denoted by a large dashed arrow ($e2$), again conforming to a more conservative route.

VII. Discussion and Conclusion

In this work we addressed the problem of generating controllers for multigait, walking soft robots that are resilient to actuator failure. We presented an abstraction of multigait behavior and actuator sensing, provided a framework for encoding gait selection and actuator failure in LTL, and demonstrated resulting synthesized automata on a physical soft robot system operating in an adversarial environment.

The demonstrations show only a small part of the complex behaviors of synthesized automata for different robot configurations, but highlight the effectiveness of the synthesis algorithm. The multigait abstraction and straightforward, easily altered behavioral encoding in LTL produced nuanced controllers for both robots. The flexibility of the behavioral encoding allows specifications to be rapidly edited to address different robots - in this case the only difference was the integer health values specified in the set H. This concept could easily be extended to include further robot behaviors. For example, if actuators on the robot were able to grasp an object as well as to contribute to a gait, holding or carrying objects may restrict the robot from using associated gaits. Further, encoding robot behaviors in more complex environments rather than the simple, two dimensional, obstacle-free environment presented here (e.g. different gaits for different terrains) can be done with the same framework. Encoding these this behaviors and restrictions would involve simple additions to the specification.

Future work on resilient, multigait behavior will focus on two issues: 1) generating a configuration for a given set of actuators with varying health levels that will satisfy a specification, and 2) determining the minimum number of health states required to satisfy given specifications.

ACKNOWLEDGMENT

We thank Anand Mishra for his contributions to waveguide development and sensing.

References

[1] F. Ilievski, A. D. Mazzeo, R. F. Shepherd, X. Chen, and G. M. Whitesides, "Soft robotics for chemists," *Angewandte Chemie - International Edition*, vol. 50, no. 8, pp. 1890–1895, 2011.

[2] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced Functional Materials*, vol. 24, no. 15, pp. 2163–2170, 2014.

[3] R. F. Shepherd, F. Ilievski, W. Choi, S. a. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[4] H. Zhao, Y. Li, A. Elsamadisi, and R. Shepherd, "Scalable manufacturing of high force wearable soft actuators," *Extreme Mechanics Letters*, vol. 3, pp. 89–104, 2015.

[5] S. Hamill, B. Peele, P. Ferenz, M. Westwater, R. F. Shepherd, and H. Kress-Gazit, "Gait Synthesis for Modular Soft Robots," in *International Symposium on Experimental Robotics*. Springer, Cham, oct 2016, pp. 669–678.

[6] D. Drotman, S. Jadhav, M. Karimi, P. DeZonia, and M. T. Tolley, "3D printed soft actuators for a legged robot capable of navigating unstructured terrain," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5532–5538.

[7] B. N. Peele, T. J. Wallin, H. Zhao, and R. Shepherd, "3D printing antagonistic systems of artificial muscle using projection stereolithography," *Bioinspiration & Biomimetics*, vol. 10, no. 5, p. 055003, 2015.

[8] E. Cohen, V. Vikas, B. A. Trimmer, and S. P. Mccarthy, "Design Methodologies for Soft-Material Robots Through Additive Manufacturing , From Prototyping to Locomotion," *Volume 5B: 39th Mechanisms and Robotics Conference*, no. July 2016, pp. 1–9, 2015.

[9] H.-T. Lin, G. G. Leisk, and B. Trimmer, "GoQBot: a caterpillar-inspired soft-bodied rolling robot." *Bioinspiration & biomimetics*, vol. 6, no. 2, p. 026007, 2011.

[10] A. D. Marchese, C. D. Onal, and D. Rus, "Autonomous Soft Robotic Fish Capable of Escape Maneuvers Using Fluidic Elastomer Actuators," *Soft Robotics*, vol. 1, no. 1, pp. 75–87, 2014.

[11] C. D. Onal, X. Chen, G. M. Whitesides, and D. Rus, "Soft mobile robots with on-board chemical pressure generation," in *Springer Tracts in Advanced Robotics*, vol. 100. Springer, Cham, 2017, pp. 525–540.

[12] W. Felt, K. Y. Chin, and C. D. Remy, "Smart Braid Feedback for the Closed-Loop Control of Soft Robotic Systems," *Soft Robotics*, vol. 00, no. 00, p. soro.2016.0056, 2017.

[13] H. Zhao, K. O 'brien, S. Li, and R. F. Shepherd, "Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides," *Science Robotics*, vol. 7529, no. 1, p. eaai7529, 2016.

[14] A. Cully, J. Clune, D. Tarapore, and J. B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[15] S. Koos, A. Cully, and J. B. Mouret, "Fast damage recovery in robotics with the T-resilience algorithm," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1700–1723, 2013.

[16] V. Vikas, P. Grover, and B. Trimmer, "Model-free control framework for multi-limb soft robots," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem. IEEE, 2015, pp. 1111–1116.

[17] H. Kress-Gazit, L. Morteza, and V. Raman, "Synthesis for Robots: Guarantees and Feedback for Robot Behavior," *Annu. Rev. Control Robot. Auton. Syst. 2018*, vol. 1, pp. 211–247, 2017.

[18] H. Xu, U. Topcu, and R. M. Murray, "Specification and synthesis of reactive protocols for aircraft electric power distribution," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 193–203, 2015.

[19] N. Ozay, U. Topcu, and R. M. Murray, "Distributed power allocation for vehicle management systems," *IEEE Conference on Decision and Control and European Control Conference*, pp. 4841–4848, 2011.

[20] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, 1977, pp. 46–57.

[21] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'Ar, "Synthesis of Reactive(1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[22] R. Ehlers and V. Raman, "Slugs: Extensible GR(1) synthesis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9780. Springer, Cham, 2016, pp. 333–339.