2012 Int'l Conference on Robotics and Automation (ICRA)
http://www.cds.caltech.edu/~murray/papers/cltbm12-icra.html

Towards Formal Synthesis of Reactive Controllers for Dexterous Robotic Manipulation

Sandeep Chinchali, Scott C. Livingston, Ufuk Topcu, Joel W. Burdick, and Richard M. Murray

***Abstract*— In robotic finger gaiting, fingers continuously manipulate an object until joint limitations or mechanical limitations periodically force a switch of grasp. Current approaches to gait planning and control are slow, lack formal guarantees on correctness, and are generally not reactive to changes in object geometry. To address these issues, we apply advances in formal methods to model a gait subject to external perturbations as a two-player game between a finger controller and its adversarial environment. High-level specifications are expressed in linear temporal logic (LTL) and low-level control primitives are designed for continuous kinematics. Simulations of planar manipulation with our synthesized correct-by-construction gait controller demonstrate the benefits of this approach.**

I. Introduction

A fundamental challenge in autonomous robotics is the design of robotic controllers that can manipulate objects as well as the human hand. Fine manipulation is especially difficult amidst uncertainties in tactile sensing, object geometry and pose, and dynamic external perturbations. Even if a robust controller can be developed for a specific task, it may not generalize to a wider class of problems.

Motivated by such concerns in the DARPA Urban Challenge for autonomous cars, Wongpiromsarn applies formal methods to design reactive controllers [1]. Given linear dynamics, control input constraints, and bounded perturbations, a reachability analysis in an *abstraction* procedure discretizes the state space. Linear temporal logic (LTL) is used to specify a two-player game between a control system and its adversarial environment, and a correct-by-construction discrete controller in the form of an automaton is synthesized. The *abstraction* procedure ensures that transitions in the automaton can always be made given system dynamics. The TuLiP Toolbox [2] performs abstraction and employs JTLV [3] for controller synthesis from LTL specifications. Both Kress-Gazit *et al.* [4] and Wongpiromsarn [1] use LTL to synthesize controllers for sensor-based motion planning. To our knowledge, this paper is the first work applying formal methods as in [1]–[4] to dexterous manipulation.

Fearing [5] first demonstrated a finger gait, defined by Hong *et al.* [6] as manipulation of a grasped object until fingers are impeded by mechanical limitations such as reaching workspace boundaries. A limit is addressed by forming a stable grasp, often a force closure grasp, and repositioning the limited finger to resume unconstrained manipulation. Hong *et al.* construct feasible gaits involving three and four fingers [6], while Han *et al.* propose two primitive motions of substitution and rewind [7]. We consider a robotic hand with three fingers as in Figure 1. Only two grasping fingers are needed for manipulation while the third *auxiliary* finger is off the object. In finger substitution, a secure three-finger grasp is formed with the *auxiliary* finger so that a single constrained finger can be repositioned [7] (Figures 4a–4d). In finger rewind, multiple fingers can reach limits simultaneously, prompting each limit to be addressed sequentially.

All authors are with the California Institute of Technology, Pasadena, CA. Address concerns to sandeepc@caltech.edu

Xu *et al.* construct a hybrid controller for finger substitution to model continuous manipulations punctuated by discrete grasp transitions [8]. They also use rapidly-exploring random trees (RRTs), a sampling-based motion planning algorithm, to identify a sequence of grasps and manipulations for a gait [9]. Such RRT-based gaiting methods are slow, lasting on average about 2.93 hours for each of 20 simulations of simply rotating a circle. Furthermore, the plan may be infeasible due to unforeseen disturbances such as finger slip.

Due to the benefit of formal guarantees and natural modelling of grasp transitions with LTL, we formulate the gaiting problem in the framework of reactive synthesis. Specifically, we consider an adversarial environment that dynamically imposes finger constraints. We first synthesize a *Gait Controller* to indefinitely manipulate an object while addressing limitations as they arise. We also synthesize a *Gait Planner*, which is essentially a *Gait Controller* that plans between specific configurations of interest. The benefits of using reactive synthesis are:

Closer to Implementation – An actual controller must react in real time to slip, limitations, and potential link collisions.

Robust – In a motion planning approach, we must know with certainty at which configurations each finger reaches a limit. By abstracting away such interferences as environmentally induced, we need not track exactly when such limits will occur as long as we can adequately respond. This leads to reactiveness, since a large class of similar disturbances can be aggregated into a partially unknown environment.

Faster Planning – Motion planning methods operating in a high dimensional configuration space are slow since they must plan a long sequence of finger trajectories. In discrete synthesis, system variables in each state trigger short horizon planning so that we only consider a manipulation or grasp transition when explicitly needed.

Systematic – LTL can be used to automatically design complex strategies with guarantees on correctness.

II. Problem Setup

In Figure 1, three two-link planar arms rotate an ellipse at angle ϕ with demanded body velocity $V_b = \begin{bmatrix} 0 & 0 & \dot{\phi} \end{bmatrix}^T$. Palm P and finger base frames S_i remain fixed while the two

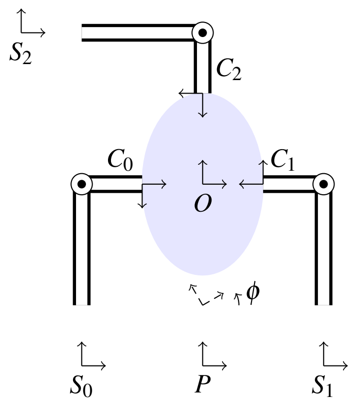


Fig. 1: Palm P, base S_i, and contact C_i frames for a gait.

grasping finger contact frames C_i and C_j move along with ϕ. A limit is declared if any links are about to collide with the ellipse or a finger approaches a kinematic singularity.

We follow [10] to describe manipulation kinematics. A homogenous transformation from frame a to b is represented by g_{ab}, while the adjoint operator $Ad_{g_{ab}}$ maps velocities between these frames. Spatial Jacobian $J^s_{s_if_i}$ maps joint angle velocities to fingertip spatial velocities. We compose finger $i's$ joint angle vector $\vec{\theta}_i = \begin{bmatrix}\theta_{1,i} & \theta_{2,i}\end{bmatrix}^T$ from the first and second joint angles $\theta_{1,i}$ and $\theta_{2,i}$. Joint angle vector $\vec{\theta} = \begin{bmatrix}\vec{\theta}_i & \vec{\theta}_j\end{bmatrix}^T$ describes a grasp with fingers i and j.

We transform contact coordinate forces $\begin{bmatrix}f_x & f_y\end{bmatrix}^T$ to wrenches $\begin{bmatrix}f_x & f_y & 0\end{bmatrix}^T$ via wrench basis B for a point contact with friction model,

$$B = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}^T. \tag{1}$$

All finger contact wrenches are aggregated into a resultant net wrench in the object frame via grasp map

$$G = \begin{bmatrix}Ad^T_{g^{-1}_{s_ic_i}} B & Ad^T_{g^{-1}_{s_jc_j}} B\end{bmatrix}. \tag{2}$$

Grasp map G and wrench basis B form the hand Jacobian

$$J_h(\vec{\theta}) = \begin{bmatrix}B^T Ad^{-1}_{g_{s_ic_i}} J^s_{s_if_i}(\theta_i) & 0\\ 0 & B^T Ad^{-1}_{g_{s_jc_j}} J^s_{s_jf_j}(\theta_j)\end{bmatrix}, \tag{3}$$

which models finger motion during manipulation by linking $\dot{\vec{\theta}}$ to body velocity V_b as

$$J_h(\vec{\theta})\dot{\vec{\theta}} = G^T V_b. \tag{4}$$

During a grasp transition, a finger is moved onto or off the object with kinematics governed by the spatial Jacobian.

Proposition 1 (Thm 5.6, pg 232 of [10]): A two-finger antipodal grasp is force closure (FC) if and only if the line bridging the contacts is within both friction cones, the sets of all forces possible with friction coefficient μ.

We adapt the approach in [11] to determine all antipodal grasps of a planar object:

1) Divide the object boundary ∂O into two regions parameterized by u_1, u_2 for antipodal point contacts.

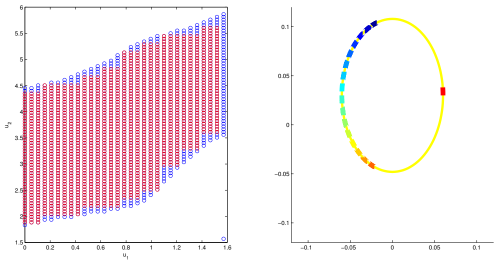


Fig. 2: (Left) FC inequality region for antipodal grasps of an ellipse with approximation in red. (Right) The colored spectrum shows target grasps for the red interval on the right.

2) Determine all FC grasps (FC inequality region) by plotting the intersection of the following four inequalities on the u_1, u_2 plane, which satisfy Proposition 1:

$$\begin{aligned}(\vec{x}_2 - \vec{x}_1)\times(\vec{x}_1 + \vec{n}_1 - \mu\,\vec{t}_l) &\geq 0\\ (\vec{x}_2 - \vec{x}_1)\times(\vec{x}_1 + \vec{n}_1 + \mu\,\vec{t}_l) &\leq 0\\ (\vec{x}_1 - \vec{x}_2)\times(\vec{x}_2 + \vec{n}_2 - \mu\,\vec{t}_2) &\geq 0\\ (\vec{x}_1 - \vec{x}_2)\times(\vec{x}_2 + \vec{n}_2 + \mu\,\vec{t}_2) &\leq 0\end{aligned}$$

where $\vec{x}_i(u_i)$ is a point in region i of ∂O, $\vec{n}_i$ is the inward normal, and $\vec{t}_i$ is the tangent to ∂O at $\vec{x}_i$.

3) Discretize ∂O into N subintervals. For each, determine the k 'target' intervals such that any two points in the start and target intervals always form an FC grasp. This amounts to approximating the FC inequality region with inscribed rectangles. Store targets in a *grasp database*.

A high-level plan must be designed to trigger grasp transitions during a gait. To handle many object geometries, the controller must expect variation in the configurations where workspace limits and link collisions occur. We address uncontrolled configuration variations by modelling them as adversarial environmental actions.

The nonlinear kinematics of manipulation render abstraction as in [1] inapplicable, and feedback linearization approaches impractical. Accordingly, we construct low-level control primitives to handle continuous kinematics independent of the high-level plan. Primitives are designed to essentially replicate *abstraction*. As discussed in Sections VI and VII, rare circumstances may arise where execution will not follow the high-level plan since continuous-time kinematics were unaccounted for in synthesis.

III. Preliminaries

We now follow [1] to introduce reactive synthesis, the methodology used to solve the informal synthesis problem described earlier. We consider a turn-based game where an adversarial environment acts first to control a finite set of variables E to which the system responds with a disjoint set of controlled variables S. Let $V = S \cup E$.

Definition 1: The domain of possible valuations of variables in set V are denoted by $\text{dom}(V)$. A *state* is simply a specific $v \in \text{dom}(V)$.

Definition 2: An atomic proposition is a Boolean variable from set V that can either be *True* or *False* in a state v.

Definition 3: An *execution* σ of a discrete-time system is defined as an infinite sequence of states $\sigma = v_0 v_1 v_2 \dots$. The set of all executions of a system is denoted Σ.

LTL is a formal language used to succintly describe behavior of such executions. Starting from atomic propositions, we use propositional connectives "and" ($\wedge$), "or" ($\vee$), "not" ($\neg$), "implies" ($\Longrightarrow$), and "equivalence" ($\Longleftrightarrow$) to obtain a *propositional formula*. Temporal operators include "next" ($\bigcirc$), "always" ($\Box$), and "eventually" ($\Diamond$) to express evolution of the system with time.

We now find executions that satisfy an LTL formula φ on variables from V. At state v_i, φ can hold at the next step $i+1$ ($\bigcirc\ \varphi$), for some $j \geq i$ ($\Diamond\ \varphi$), or all times $j \geq i$ ($\Box\ \varphi$) in the future. In formal synthesis, we desire guarantees that a formula φ is valid for the set of all possible executions Σ.

Definition 4: The expression $\sigma \models \varphi$ indicates execution σ satisfies formula φ. A system is termed *correct* with respect to specification φ if φ is satisfied by all executions $\sigma \in \Sigma$, in which case we write $\Sigma \models \varphi$.

To allow polynomial time computation of a control strategy, we consider LTL formulae of the Generalized Reactivity (1) or GR(1) subclass [12]. The assume-guarantee format (below) ensures if the environment satisfies assumptions described by φ_e, the system is guaranteed to react according to φ_s. The winning strategy is a synthesized finite automaton that guarantees a system response s for every possible environment action e to eventually achieve system goals. A GR(1) specification is of the form

$$\left(\varphi_{e,\text{init}} \wedge \bigwedge_{j \in J_{\text{safety}}} \Box\ \varphi^j_{e,t} \wedge \bigwedge_{j \in J_{\text{goal}}} \Box \Diamond\ \varphi^j_{e,\text{goal}}\right)$$
$$\Longrightarrow \left(\varphi_{s,\text{init}} \wedge \bigwedge_{i \in I_{\text{safety}}} \Box\ \varphi^i_{s,t} \wedge \bigwedge_{i \in I_{\text{goal}}} \Box \Diamond\ \varphi^i_{s,\text{goal}}\right)$$

All propositional formulae can involve variables from $S \cup E$. The propositional formulae $\varphi_{e,\text{init}}$ and $\varphi_{s,\text{init}}$ describe environment and system initial conditions. Environment and system transition formulae $\varphi^j_{e,t}$ and $\varphi^j_{s,t}$ include primed variables of $\bigcirc E$ and $\bigcirc S \cup \bigcirc E$, respectively. Primed variables refer to the state of variables in the *next* state. Since the environment acts first, $\varphi^j_{e,t}$ are formulae that cannot involve any system variables in $\bigcirc S$ (i.e., system variables in the next time step). Since the system subsequently responds to the environment, the formula $\varphi^i_{s,t}$ can involve any variables in $\bigcirc S \cup \bigcirc E$. Environment goals $\varphi^j_{e,\text{goal}}$ and system goals $\varphi^j_{s,\text{goal}}$ are propositional formulae, so describe sets of states that must be reached infinitely often.

IV. Application of Discrete Synthesis to Gaits

For the ith finger, system variable p_i indicates contact with the object while environment variable l_i indicates a limit. We always require at least one minimal grasp min_i, one for which lifting even a single finger loses FC. More complex grasps are simply superpositions of minimal grasps. In our example, all two-finger antipodal grasps are minimal, while the only superposition is a three-finger grasp.

For the *Gait Planner*, rotation angle ϕ is discretized into N sectors from 0 to 2π, each represented with a proposition t_i. Two *Gait Planners* were synthesized for a system with $N = 8$ sectors denoted by t_0 to t_7. *Gait Planner 1* begins in t_0 with goal of reaching t_3. To illustrate more complex behavior naturally described by LTL, *Gait Planner 2* begins in t_0 and must alternate between t_1 and t_3 infinitely often while never visiting t_4, forcing the rotation direction to switch.

Right and left torque system variables τ_r and τ_l allow rotations between adjacent sectors in the absence of limits. We assume a gait in t_i repositions the grasp suitably well to allow for unconstrained manipulation until t_{i-1} or t_{i+1}. This assumption prevents the environment from persistently posing limitations that impede progress from sector t_i. Specifically, a *flag* system variable indicates a gait has just ended, preventing limits immediately afterwards.

A. LTL specifications

In the following specifications, index $i \in \{1, \dots, M\}$ represents a specific finger in an M finger gait. Index $k \in \{1, \dots, N\}$ represents the k^{th} of N sectors in the subdivision of ϕ. We indicate limits on all fingers in an M finger gait by $l_{all} = \bigwedge_i l_i$ and some finger limit by $l_{some} = \bigvee_i l_i$. The subscript *con* indicates specifications shared by the *Gait Planner* and simpler *Controller*, while the subscript *plan* applies to the *Planners* only. Thus, a *Planner's* GR(1) formula is $(\varphi_{e,con} \wedge \varphi_{e,plan}) \Longrightarrow (\varphi_{s,con} \wedge \varphi_{s,plan})$, while a *Controller's* formula lacks conjunctions with any terms with a *plan* subscript. In our example with $M = 3$,

$$\varphi_{e,\text{init},con} \wedge \varphi_{s,\text{init},con} = (\neg\ l_{some} \wedge (p_0 \wedge p_1 \wedge \neg\ p_2))$$

indicates no finger is limited at the initial grasp formed by fingers 0 and 1.

Environment Assumptions: $\varphi_{e,con}$ is a conjunction of A1-4.

(A1) A finger remains limited if it stays on the object:

$$\bigwedge_i \Box\ \big((p_i \wedge l_i) \Longrightarrow \bigcirc\ l_i\big).$$

(A2) Lifting and repositioning a finger clears the limit:

$$\bigwedge_i \Box\ \big(\neg\ p_i \Longrightarrow \bigcirc\ \neg\ l_i\big).$$

(A3) All fingers cannot be limited simultaneously:

$$\bigwedge_i \Box\ \big(\bigcirc\ \neg\ l_{all}\big).$$

(A4) New limits do not spontaneously arise during a gait:

$$\bigwedge_i \Box\ \big((l_{some} \wedge \neg\ l_i) \Longrightarrow \bigcirc\ \neg\ l_i\big).$$

(A5) $\varphi_{e,plan}$ indicates limits cannot arise right after gaits:

$$\Box\ (flag \Longrightarrow \bigcirc\ \neg\ l_{some}).$$

System Properties: $\varphi_{s,con}$ is a conjunction of P1-P3.

(P1) Always ensure at least one minimal grasp for FC:

$$\Box\ \bigcirc\ \big(min_{some}\big).$$

(P2) At least one original minimal grasp must follow a superposition of minimal grasps:

$$\square \big((min_i \wedge \cdots \wedge min_k) \implies \bigcirc \, (min_i \vee \cdots \vee min_k)\big).$$

(P3) All minimal grasps for a three-finger gait are:

$$\bigwedge_i \square \Big((\neg \, p_i \wedge \bigwedge_{j \in \{1,\ldots,M\} \setminus i} p_j) \iff min_i\Big),$$

The goal $\varphi_{g,con} = \square \diamond \neg \, l_{some}$, added conjunctively into $\varphi_{s,con}$, is to freely manipulate the object without any limits. ***Gait Planner (GP) Properties***: $\varphi_{init,plan} = t_0 \wedge \neg flag$ indicates starting in t_0 without the flag to prevent limits. Conjunction of GP1-GP7 forms $\varphi_{s,plan}$.

(GP1) Without a limit, a torque shifts ϕ by one sector:

$$\bigwedge_k \square \big((t_k \wedge \tau_r \wedge \bigcirc \, \neg l_{some}) \implies \bigcirc \, t_{k+1}\big),$$

$$\bigwedge_k \square \big((t_k \wedge \tau_l \wedge \bigcirc \, \neg l_{some}) \implies \bigcirc \, t_{k-1}\big).$$

(GP2) Applying a torque does not work if a limit exists:

$$\bigwedge_k \square \Big(\big(t_k \wedge (\tau_l \vee \tau_r) \wedge \bigcirc \, l_{some}\big) \implies \bigcirc \, t_k\Big).$$

(GP3) Do not apply a torque if we already have a limit:

$$\square \, \bigcirc \, \neg\big(l_{some} \wedge (\tau_r \vee \tau_l)\big).$$

(GP4) The sector can not change if a torque is not applied:

$$\bigwedge_k \square \Big(\big(t_k \wedge \neg(\tau_l \vee \tau_r)\big) \implies \bigcirc \, t_k\Big).$$

(GP5) Raise a flag if a limit was just cleared by a gait:

$$\square \big((l_{some} \wedge \bigcirc \, \neg l_{some}) \implies \bigcirc \, flag\big).$$

(GP6) Disallow raising the flag when limits can occur:

$$\square \big((\tau_l \vee \tau_r) \implies \bigcirc \, \neg flag\big).$$

$$\square \big((\neg l_{some} \wedge \bigcirc \, \neg l_{some}) \implies \bigcirc \, \neg flag\big).$$

(GP7) Do not apply a torque when a grasp is being switched:

$$\square \, \bigcirc \, \neg\big((\tau_l \vee \tau_r) \wedge (p_0 \wedge p_1 \wedge p_2)\big).$$

(GP8) *Planner 2* must also switch directions to avoid t_4:

$$\square \, \bigcirc \, \neg t_4.$$

Goals $\varphi_{g,plan_1} = \square \diamond t_3$ and $\varphi_{g,plan_2} = (\square \diamond t_1) \wedge (\square \diamond t_3)$ are incorporated conjunctively into each *Planner's* $\varphi_{s,plan}$.

B. Control Strategy

Consider the TuLiP-synthesized *Gait Controller* of Figure 3. In any of the three *free manipulation* states, indicated by a self connection, the two grasping fingers p_i and p_j and absence of any limit l indicates unconstrained rotation. The three edges emanating from *free manipulation* nodes indicate possible environment limits, prompting the automaton to place the appropriate *auxiliary* finger on the object, indicated by the presence of all p. The absence of p_i at the next step indicates finger i is lifted off the object to address limitation l_i. The system satisfies its goal by countering limits to always arrive at a new *free manipulation* node. The synthesized strategies for *Gait Planners* 1 and 2 comprise 64 and 106 states respectively.

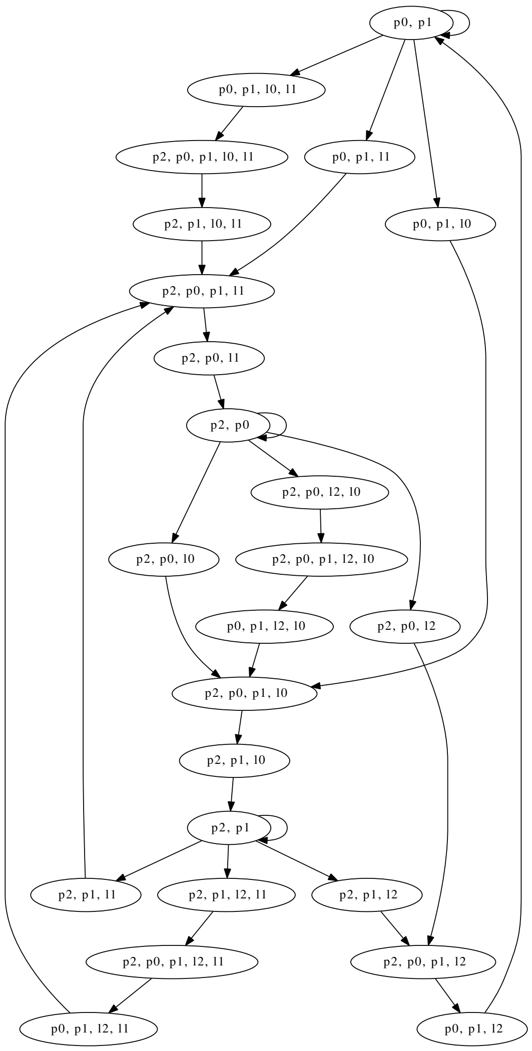


Fig. 3: The *Gait Controller* moves fingers p in each state to react to all possible limits l in finger rewind. If a variable name is not included in a graph node label, then it is false at that node.

TABLE I: Mapping Function

$p_i \wedge p_j \wedge \neg \; p_k \wedge \neg \; l_{some}$	Manipulate
$p_i \wedge p_j \wedge \neg \; p_k \wedge l_{some}$	Grasp Select
$p_i \wedge p_j \wedge p_k \wedge l_{some}$	Finger Move (on *then* lift)

V. System Architecture

To realize a controller, variables active in each state must be translated using a *Mapping Function* to control primitives, executable code that produces output such as moving the fingers. In a *Mapping Function* such as in Table I, states described by propositional formulae in the left column are linked to corresponding primitives in the right column. We developed a parser to search through a TuLiP automaton and insert in each state the appropriate control primitive indicated by the *Mapping Function* to create a Matlab state machine. This generic parser is a significant step in

TABLE II: Results for Gaiting Elliptical Objects. Friction μ and ellipse axes a and b are varied to measure the resulting net rotation $\Delta\phi$, simulation time Δt, and success of the gait.

μ (°)	$a(m)$	$b(m)$	$\Delta\phi$ (°)	Δt (s)	Works?
45	.06	1.3a	318.22	14.8	*Yes*
40, 35, 30	.07	1.1a	643.3	29.3, 28.6, 26.2	*Yes*
35, 30	.07	1.1a	643.3	28.6, 26.2	*Yes*, *Yes*
45	.08	1.1a	523.9	28.2	*Yes*
45, 35	.065	1.25a	628.5, 234.9	28.3, n/a	*Yes*, *No*
30, 25, 20	.08	1.0a	612.4, 225.4, 225.4	26.4, n/a, n/a	*Yes*, *No*, *No*

Algorithm 1 Manipulate Primitive

```
procedure MANIPULATE ($\theta_{init}$, $l_0$, $l_1$, $l_2$, $\phi_{init}$)
    while not ($l_0$ or $l_1$ or $l_2$) do
        $J_h(\theta)\dot{\theta} \leftarrow G^T V_b$
        $l_i \leftarrow$ Collision_Detect ($\theta_i$, $\phi$) or WS_limit ($\theta_i$)
        $l_j \leftarrow$ Collision_Detect ($\theta_j$, $\phi$) or WS_limit ($\theta_j$)
    end while
    return $\theta_f$ , $l_0$ , $l_1$, $l_2$, $\phi_f$
end procedure

function COLLISION_DETECT ($i$, $\theta_i$, $\phi$)
    $\vec{x}_{link_i} \leftarrow$ Discretize Links ($len_1$, $len_2$, $S_i$, $\theta_i$)
    for all ($\vec{x} \in \vec{x}_{link_i}$) do
        if (Relative_Location ($\vec{x}$, $\phi$) == inside) then
            $collision \leftarrow$ true
            break
        end if
    end for
    return $collision$
end function

function WS_LIMIT ($\theta_i$, $l_0$, $l_1$, $l_2$, $i$)
    if ($\|\theta_2^i\| < \theta_{preset}$) then
        $l_i \leftarrow$ true
    end if
    return $l_0$, $l_1$, $l_2$
end function
```

automation since it allows for systematic translation from high-level specifications to executable code.

We designed the MANIPULATE, FINGER_MOVE, and GRASP_SELECT primitives (Algorithms 1–3). MANIPULATE simulates kinematics with J_h until COLLISION_DETECT or WS_LIMIT ("workspace limit") stop numerical integration. FINGER_MOVE uses Corke's Matlab Robotics Toolbox [13] to move the *auxiliary* finger along a Cartesian space trajectory to form or break a grasp. Finger *i's* trajectory $traj$ from $\theta_{init,i}$ to final configuration $\theta_{mv,i}$ clears limit l_i if the finger is lifted off the object. This is determined by RELATIVE_LOCATION, which indicates the position of point x_{body} relative to an ellipse with axes $2a$ and $2b$ at angle ϕ.

Once fingers reach limits, GRASP_SELECT computes a new three-finger grasp. Since a finger will be lifted next, the original finger that remains, indicated by *stay*, must form a FC grasp with *auxiliary* finger *aux*. The *grasp database* from Section II yields many 'target' grasps for finger *aux*, but some trajectories to these points may cause link collisions. Only 'target' grasps leading to collision free trajectories are added to a feasible list. GET_FEASIBLE uses a heuristic to select a grasp from the feasible list to ensure a long manipulation before new limits. The direction of rotation indicated by demanded velocity $\omega_{dem} = \dot{\phi}$ prompts selection of either the right or leftmost point in the feasible list.

Algorithm 2 Finger_Move Primitive

```
procedure FINGER_MOVE ($i$, $\theta_{mv,i}$, $l_0$, $l_1$, $l_2$, $\theta_{init,i}$, $\phi$)
    $traj \leftarrow$ Cartesian_Trajectory ($\theta_{init,i}$, $\theta_{mv,i}$)
    if (Relative_Location ($\vec{x}_{move}$, $\phi$) == outside) then
        $l_i \leftarrow$ false
    end if
    return $\theta_{mv,i}$ , $l_0$ , $l_1$, $l_2$, $traj$
end procedure

function RELATIVE_LOCATION ($\vec{x}_{body}$, $\phi$)
    $\Omega \leftarrow \frac{x_{body}^2}{a^2} + \frac{y_{body}^2}{b^2} - 1$
    if ($\Omega > 0$) then return outside
    else if ($\Omega < 0$) then return inside
    else return on
    end if
end function
```

VI. RESULTS AND FUTURE WORK

We simulated the Matlab controller of Section V gaiting an ellipse with fingers of link lengths $len_1 = .02$ m and $len_2 = .01$ m. To experimentally demonstrate robustness, we varied semi–minor axis a, semi–major axis b, and mutual friction coefficient μ as in Table II. Since the controller was able to rotate ellipses virtually indefinitely, we manually terminated the simulation to measure net rotation $\Delta\phi$ and simulation time Δt. The latter quantity was only measured if the simulation was successful. Figures 4a–4f illustrate the gait in the first row of Table II. The short horizon plans enacted by the gait controller allow gait planning and execution on the order of seconds as compared to hours for RRT methods [9]. The ability to gait ellipses of varied geometries despite different configurations at which limits occur justifies both the claim of reactiveness and the design choice of abstracting limits as environmentally induced. Decreasing μ makes it harder to grasp an object since fewer antipodal grasps can be formed. Though results indicate success with coefficients as low as $\mu = 30°$, three failures occurred when FINGER_MOVE could not find a simple collision-free trajectory to any of the small set of next target grasps allowed by the lower μ. Since the discrete strategy assumes primitives can always find a feasible next grasp, this mismatch halts progress as discussed at the end of this section.

Gait Planner 1 successfully gaited a circle between t_0 and

Algorithm 3 Grasp_Select Primitive

```
procedure GRASP_SELECT ($\theta_{aux}$, grasp database, $\phi$, $l_0$, $l_1$, $l_2$)
    if $l_i$ == true and $l_j$ == true then
        stay ← $max_i$ ($l_i$ == true )
    else
        stay ← get indx ($l$ == false )
    end if
    DB indx ← get index ($\vec{x}_{stay,body}$)
    for targ ID = 1 → num targets (DB index) do
        [$\vec{x}_{start}$, $\vec{x}_{end}$] ← grasp database (DB indx, targ ID)
        for $\vec{x}_{final} \in [\vec{x}_{start}, \vec{x}_{end}]$ do
            $traj$ ← Cartesian_Trajectory ($\theta_{aux}$, $\theta_{final}$)
            if ($\forall \vec{x} \in traj$ : Rel._Loc. ($\vec{x}$, $\phi$) ≠ inside) then
                append (feasible list, $\vec{x}_{final}$)
            end if
        end for
    end for
    $\vec{x}_{grasp}$ ← Get_Feasible (feasible list, $\omega_{dem}$)
    return $\vec{x}_{grasp}$
end procedure

function GET_FEASIBLE (feasible list, $\omega_{dem}$)
    if ( $\omega_{dem} > 0$ ) then
        $\vec{x}_{grasp}$ ← leftmost (feasible list)
    else
        $\vec{x}_{grasp}$ ← rightmost (feasible list)
    end if
    return $\vec{x}_{grasp}$
end function
```

t_3. *Gait Planner 2* manipulated a circle from t_0 to t_3, switched direction, revisited t_1, switched direction again, and made progress until it reached a limitation in t_2. The controller had just finished a gait and, by assumption, did not expect a limit to immediately impede progress again. This assumption violation caused the system to not react to the limit, stagnate in t_2, and not satisfy its goal. Analysis of system execution until the error (which spanned 27 states with 6 grasp switches) indicated selection of 'suboptimal' grasps which quickly led to limits after only minimal rotation. Such lack of progress compounds until the object can not be rotated as far as required by the high-level plan. Figures 4j–4m illustrate a suboptimal sequence. The system should quickly re-compute a plan to gait the object again as it had done successfully multiple times before the nominal assumption violation. In future work we will apply the patching algorithm of [14] to extensions of our finger gait planner.

Though robustness is clearly demonstrated, few, potentially preventable failures arise due to mismatches between assumptions of the discrete controller and capabilities of the continuous primitives. For example, in the case where feasible grasps are limited by μ, FINGER_MOVE could plan more complex trajectories to target grasps. For *Gait Planner 2's* error, we could meet environmental assumptions by increasing the number of sectors N to allow easier progress before reaching limits. Future work is directed towards formal synthesis that incorporates nonlinear continuous kinematics and provides guarantees on correctness in terms of parameters such as μ, a, b, and sector size N.

VII. Conclusion

The long term goal of our research is to enable more sophisticated robotic behaviors, for which the complexity of intertwining high-level reasoning, actuation, and sensor fusion renders ad-hoc design of provably correct controllers virtually infeasible. As illustrated in this paper, reactive synthesis potentially provides a powerful design approach, rooted in underlying formal guarantees, to design the next generation of autonomous manipulation platforms.

Acknowledgments

This work is partially supported by the Boeing Corporation.

References

[1] T. Wongpiromsarn, "Formal methods for design and verification of embedded control systems : application to an autonomous vehicle," Ph.D. dissertation, California Institute of Technology, 2010.

[2] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *Proc. Hybrid Systems: Computation and Control*, 2011.

[3] A. Pnueli, Y. Sa'ar, and L. Zuck, "JTLV: A framework for developing verification algorithms," in *Computer Aided Verification*, vol. 6174, 2010, pp. 171–174, associated tool available at `http://jtlv.ysaar.net/`.

[4] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2007, pp. 3116–3121.

[5] R. Fearing, "Implementing a force strategy for object re-orientation," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1986, pp. 96–102.

[6] J. Hong, G. Lafferriere, B. Mishra, and X. Tan, "Fine manipulation with multifinger hands," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1990, pp. 1568–1573.

[7] L. Han and J. Trinkle, "The instantaneous kinematics of manipulation," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1998, pp. 1944–1949.

[8] J. Xu and Z. Li, "A kinematic model of finger gaits by multifingered hand as hybrid automaton," *IEEE Trans. on Automation Science and Engineering*, pp. 467–479, 2008.

[9] J. Xu, T.-K. Koo, and Z. Li, "Sampling-based finger gaits planning for multifingered robotic hand," *Autonomous Robots*, vol. 28, pp. 385–402, 2010.

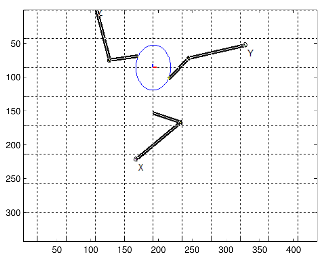
[10] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[11] B. Faverjon and J. Ponce, "On computing two-finger force-closure grasps of curved 2d objects," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1991, pp. 424–429.

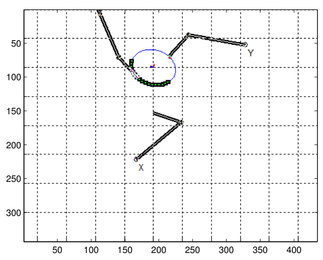
[12] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[13] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.

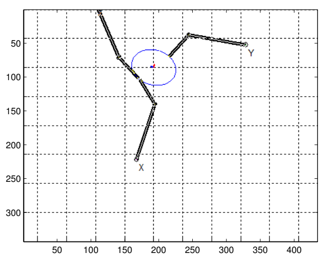
[14] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2012, in press.



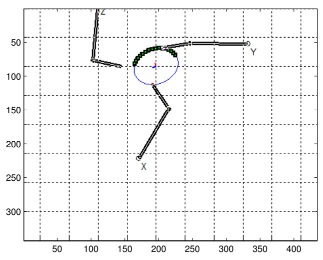
(a) Start



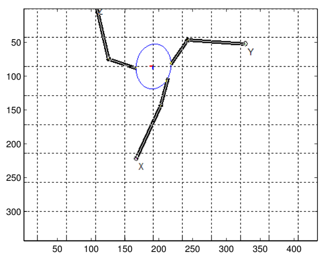
(b) End of *Manipulate*. Grasp Targets in green.



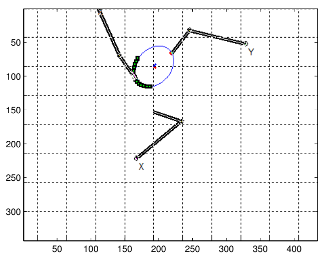
(c) Switch of grasp



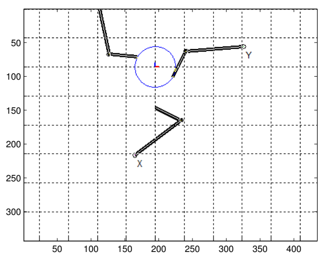
(d) End of *Manipulate*



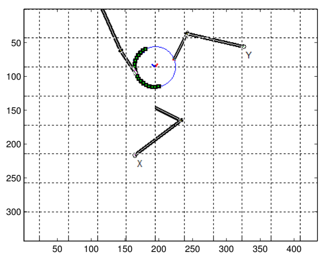
(e) Switch of grasp



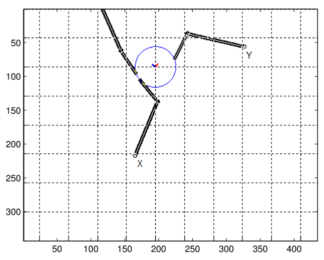
(f) Final Configuration



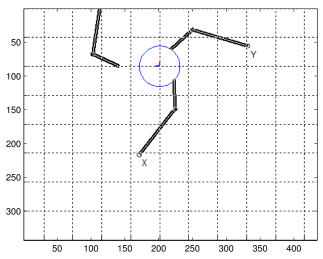
(g) Start gait of circle



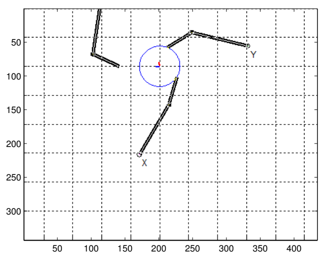
(h) Finger reaches limit



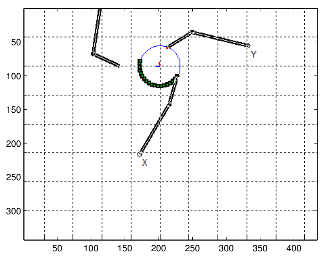
(i) Switch of grasp



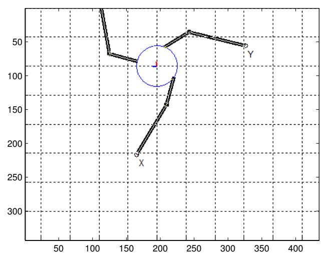
(j) A suboptimal grasp.



(k) *Manipulate* only rotates the object minimally until the bottom finger reaches a limit.



(l) The *auxiliary finger* can only reach the leftmost grasp targets in green.



(m) The resulting grasp is suboptimal again, as the top finger will quickly reach a limitation with minimal rotation, illustrating how problems compound.

Fig. 4: Subfigures *a* - *f* show a successful gait of an ellipse, *g* - *i* show frames from a gait of a circle, and *j* - *m* explain problematic grasps from *Gait Planner 2*.