Bottom-up symbolic control: attractor-based planning and behavior synthesis

Jie Fu and Herbert G. Tanner

***Abstract*—A class of hybrid systems with convergent continuous dynamics is abstracted to a special type of finite automata which operate on an infinite alphabet. The abstraction is predicate-based, enabled by the convergence properties of the continuous dynamics of the hybrid system, and encompasses existing low-level controllers rather than replacing them during synthesis. The abstract models are finite yet capable of storing and manipulating continuous data expressing attributes of the concrete hybrid model. The latter is shown to weakly simulate its discrete abstraction, and thus behaviors planned using the abstraction are always implementable on the concrete system. A case study illustrates how this methodology can be put into practice, orchestrating a temporal sequence of continuous controllers that allows the hybrid system to achieve a performance objective.**

***Index Terms*—Hybrid systems, abstraction, symbolic control.**

I. Introduction

This paper presents a method for synthesizing control plans that combines existing low-level control designs in a way that allows the system to perform tasks which its low-level designs cannot complete on their own. In this approach, a machine is viewed as a hybrid dynamical system capable of switching between operating modes with well defined pre-conditions and post-conditions. Pre-conditions determine when a certain mode can be activated, and post-conditions describe the guaranteed steady-state behavior. The synthesis problem can then be lifted to a discrete domain through appropriate abstractions.

The proposed abstraction process hinges on the convergence properties of the continuous dynamics. There are many instances of systems with specialized continuous controllers that are well suited for the particular application, as in the case of interaction with human subjects [1], [2]. It may also be the case that some validated and trusted low-level continuous control loops might be off-limits to a higher-level controller; examples can be found in process control [3] or automotive applications [4]. Then a synthesis methodology that works *with*, rather than in lieu of, existing control loops, can be useful. The focus of this paper is therefore not on control design for the continuous dynamics, but rather on abstraction and planning at the discrete level, knowing that the plan can always be implemented at the concrete level.

For this purpose, we propose discrete abstractions for a specific class of hybrid systems. Concrete system and abstraction are linked through a weak simulation relation. The size of the abstraction is not dependent on the discretization resolution on the continuous state space. Continuous states are grouped together according to the convergence properties of the trajectries that start from them. Convergence supports a bottom-up design where existing low-level control loops (control modes) are not replaced, but are parameterized and then *combined* into a sequence that is ensured to achieve the desired objective.

Undoubtably, there are performance trade-offs when the continuous dynamics in each control mode are required to be always convergent. On the other hand, convergence of the continuous vector fields enables to transfer the planning problem to the discrete domain, where it may be more amenable to analytical treatment and algorithm development.

Abstraction allows planning and control design for hybrid systems with smaller analytical and computational cost. In principle, plans can be designed using a finite-state abstraction and then refined into controllers on the infinite-state system. For this refinement to be possible, the concrete system and its abstraction need to be related formally.

To this end, simulation or bisimulation relations [5] can be used, inducing partitions of the continuous state space, where each block contains states which behave in a "similar" manner. Instead of tracking the behavior of the system trajectories from each individual continuous state, one can then consider the behavior from whole blocks. The dynamics of trajectories originating from these blocks is captured by a discrete model which serves as an abstraction of the original system.

Computing bisimulation relations for general classes of hybrid systems is known to be difficult [6]. In general, for an algorithm that computes such partitions to be decidable, one needs to restrict either the discrete, or the continuous dynamics of a hybrid system [7], [8]. In this paper we abstract a hybrid system into one where all dynamics evolve at the discrete level. We do so by restricting the continuous dynamics: we confine analysis to a particular class of hybrid systems having convergent continuous dynamics with parametrized domains and positive limit sets. Convergence supports an abstraction in which only the asymptotic behavior of the continuous dynamics is preserved.

We can broadly categorize existing approaches to discrete abstraction of hybrid systems into two classes: one that is based on simulation, bisimulation [9], or approximate bisimulation [10], [11]; and another that is primarily partition-based [12]–[17]. Approaches can also be hybrid [18], using state-quantization methods and approximate bisimulations. The one utilized in this paper is a hybrid one, in the sense that it is based on (*a priori*) partitioning the continuous state space, but also establishes some relation between the states within each partition block. Earlier versions of this abstraction construction

Jie Fu and Herbert Tanner are with the Mechanical Engineering Department at the University of Delaware, Newark DE 19716. {jiefu,btanner}@udel.edu. This work is supported by NSF under award # 0907003.

can be traced to [19]; here, we abstract to a new type of automata that preserve some continuous information from the concrete hybrid system, and focus on how to use this abstraction for planning. The type of partitioning used in this paper is reminiscent of predicate abstraction [20].

In this paper, the states of the finite system are labeled with Boolean vectors, which express the valuation of logical propositions at the particular block of continuous states, control parameters, and exogenous (environment) variables. The set of these logical propositions is used to express the pre-conditions and post-conditions of the continuous controllers of the hybrid system.

Control is exercised through the selection of the controller parameters, which inform the controllers as to where to steer the continuous system trajectories. The abstraction does not have continuous dynamics; rather, it models possible sequences of continuous controllers in the input sequences that it accepts. The input alphabet of the abstraction consists of the (discrete) indices of these controllers, along with a set of (continuous) variables that parameterize them. Our abstract model is a variant of a machine known as a register automaton [21], [22], which is a finite state machine equipped with a finite number of registers, which are essentially storage units. The input to this machine is not simply a string of discrete symbols, but rather a *data word*, a sequence of pairs consisted of a discrete symbol together with a continuous data value. Although it manipulates continuous data, the model is still discrete since there is no dynamics on this data.

Alternative *discrete* computational models cannot manipulate continuous data. As a result, optimal control schemes either have to operate on the continuous domain [23], or depend on the resolution of the discretization imposed on the continuous state space [18], [24]. In this paper, the discrete abstractions do not require a discretization of the continuous domains: continuous states are grouped in blocks, and controller parameter space is left intact. In principle, feasible solutions to the planning problem can be obtained using these models, and even optimal ones if the underlying continuous dynamics and cost functions are simple enough [25].

In Section II, notation and terminology is introduced. Section III presents the new abstraction and establishes a (weak) simulation relation between hybrid and discrete model. Section IV outlines a new control synthesis framework, using this abstraction and a novel variation of dynamic programming (DP)—a straightforward application of DP will not work. The simulation relation guarantees that the plans devised using the abstraction are implementable on the concrete system. Section V presents a numerical case study, with a mobile manipulator tasked with grasping and delivering an object through a sequence of maneuvers implemented through given control laws. Section VI concludes the paper with a summary of the results and thoughts for future extensions.

II. The Models

Let Σ be a finite set of symbols, and $D \subseteq \mathbb{R}^k$. Pairs of the form $w_i = (\sigma_i, d_i) \in \Sigma \times D$ are called *atoms*, and concatenations of atoms form finite sequences $w = w_1 \cdots w_n$ over $\Sigma \times D$ called *data words*. A data word v is a *prefix* of another data word w if there exists a sequence $x \in (\Sigma \times D)^*$ such that $w = vx$. The *length* of a word w is denoted $|w|$, and $\mathrm{dom}(w)$ is the index set $\{1, \ldots, |w|\}$ of the positions of the atoms $w_i = (\sigma_i, d_i)$ in w. For $i \in \mathrm{dom}(w)$, the *data projection* $\mathrm{val}_w(i) = d_i$ gives the data value associated with the symbol σ_i. Similarly, the *string projection* $\mathrm{str}_w(i) = \sigma_i$ gives the symbol associated with a data value in atom w_i. The following machine operates on strings $w \in (\Sigma \times D)^*$.

Definition 1 (Register Automaton cf. [21], [26][1]): A nondeterministic two-way register automaton is a tuple $\mathbf{R} = \langle \mathcal{Q}, q_0, F, \Sigma, D, k, \tau, \Delta \rangle$, in which

$\mathcal{Q}$	a finite set of states;
$q_0 \in \mathcal{Q}$	the initial state;
$F \subseteq \mathcal{Q}$	the set of final states;
Σ	a finite alphabet;
D	a set of data values;
$k \in \mathbb{N}$	the number of registers;
$\tau : \{1, \ldots, k\} \to D \cup \{\varnothing\}$	the register assignment;[a]
Δ	a finite set of *read*,[b] or *write*[c] transitions.

[a] When $\tau(i) = \varnothing$, this means that register i is empty. The initial register assignment is denoted τ_0. Given $(\sigma, d) \in \Sigma \times D$, a register can perform a test in the form of a first-order logical formula φ, constructed using the grammar $\varphi ::= d \leq \tau(i) \mid d < \tau(i) \mid \neg\varphi \mid \varphi \wedge \varphi$. The set of all such formulae is denoted $\mathrm{Test}(\tau)$.

[b] Read transitions are of the form $(i, q, \varphi_r) \xrightarrow{\sigma} (q', \delta)$, where q and q' belong in $\mathcal{Q}$, i ranges in $\{1, \ldots, k\}$, σ in Σ, φ_r in $\mathrm{Test}(\tau)$, and $\delta \in \{\mathrm{right,stay,left}\}$, respectively.

[c] Write transitions are of the form $(q, \varphi_w) \xrightarrow{\sigma} (q', i, \delta)$, where $\varphi_w \in \mathrm{Test}(\tau)$, q and q' are in $\mathcal{Q}$, and all other elements range in the same sets as in read transitions.

Given a data word w, a *configuration* γ of $\mathbf{R}$ is a tuple $[j, q, \tau]$, where j is a position in the input data word, q is a state, and τ the current register content. Configurations $\gamma = [1, q_0, \tau_0]$ and $\gamma = [j, q, \tau]$ with $q \in F$, are *initial* and *final*, respectively. Given $\gamma = [j, q, \tau]$ and input (σ_j, d_j), the transition $(i, p, \varphi_r) \xrightarrow{\sigma_j} (p', \delta)$ *applies* to γ if, and only if, $p = q$ and φ_r is true, while $(p, \varphi_w) \xrightarrow{\sigma_j} (p', i, \delta)$ *applies* to γ if, and only if, $p = q$ and φ_w is true.

The semantics of this machine is as follows. At configuration $[j, q, \tau]$, the machine is in state q, the input read head is at position j in the data word, and the contents of the registers are expressed by τ. Upon reading $w_j = (\sigma_j, d_j)$, if φ_r is true and $(i, q, \varphi_r) \xrightarrow{\sigma_j} (q', \delta) \in \Delta$, then $\mathbf{R}$ enters state q' and the read head moves in the direction of δ, i.e., $j' = j + 1, j' = j, j' = j - 1$ for $\delta \in \{\mathrm{right,stay,left}\}$. The configuration is now $[j', q', \tau]$. If φ_w is true and $(q, \varphi_w) \xrightarrow{\sigma_j} (q', i, \delta) \in \Delta$, then $\mathbf{R}$ enters state q', d_j is copied to register i, and the read head moves in the direction δ (in this order). The configuration is now $[j', q', \tau']$, where the updated register

[1] In [21] the set of logical tests that the machine can perform does not explicitly appear in the definition, because they are assumed to be only equality tests. In [26], however, the definition of register automata explicitly includes a set of register tests in the form of logical propositions.

assignment τ' is such that for $\kappa = 1, \ldots, i-1, i+1, \ldots, k$, it is $\tau'(\kappa) = \tau(\kappa)$ and $\tau'(i) = d_j$. The automaton is deterministic if at each configuration for a given data atom there is at most one transition that applies. If there are no left-transitions it is called *one-way*. An example is given in Appendix A.

Our special class of hybrid systems is a *hybrid agent*:

Definition 2 (Hybrid Agent): The hybrid agent is a tuple $\mathbf{H} = \langle \mathcal{Z}, \Sigma, \iota, \mathcal{P}, \pi_i, \mathcal{AP}, f, \textsc{Pre}, \textsc{Post}, s, T \rangle$ with

$\mathcal{Z} = \mathcal{X} \times \mathcal{L}$	a set of continuous and Boolean states;[a]
Σ	a finite set of control modes;[b]
$\iota : \Sigma \to \{1, \ldots, k\}$	indices for the elements of Σ;
$\mathcal{P} \subseteq \mathbb{R}^m$	a vector of control parameters;
$\pi_i : \mathbb{R}^m \to \mathbb{R}^{m_i}$	a set of canonical projections;[c]
$\mathcal{AP}$	a set of atomic propositions over $\mathcal{Z} \times \mathcal{P}$; [d]
$f_\sigma : \mathcal{X} \times \mathcal{L} \times \mathcal{P} \to T\mathcal{X}$	a finite set of parameterized vector fields;[e]
$\textsc{Pre}$: $\Sigma \to \mathcal{C}$	the pre-condition of $\sigma \in \Sigma$ and $\mathcal{C}$ is defined next;[f]
$\textsc{Post}$: $\Sigma \to \mathcal{C}$	the post-condition of $\sigma \in \Sigma$;[g]
$s : \mathcal{Z} \times \mathcal{P} \to 2^{\mathcal{P}}$	is the parameter reset map;[h]
$T : \mathcal{Z} \times \mathcal{P} \times \Sigma \to \mathcal{Z} \times \mathcal{P} \times \Sigma$	is the transition map.[i]

[a] Here, $\mathcal{X} \subset \mathbb{R}^n$ is a compact set, and $\mathcal{L} \subseteq \{\mathbf{0}, \mathbf{1}\}^r$, with $n, r \in \mathbb{N}$. A state $z \in \mathcal{Z}$ is called a *composite* state.

[b] The symbols in Σ label the different closed-loop continuous dynamics.

[c] For $i = 1, \ldots, k$, we write $p = (\pi_1(p)^\intercal, \ldots, \pi_k(p)^\intercal)^\intercal$.

[d] $\mathcal{AP}$ is a set of atomic propositions, denoted α. A *literal* β is defined to be either α or $\neg\alpha$, for some $\alpha \in \mathcal{AP}$. Set $\mathcal{C}$ is a set of *logical sentences*, each of which is a conjunction of literals, i.e., $\mathcal{C} = \{c = \beta_1 \wedge \beta_2 \ldots \wedge \beta_n \mid (\exists \alpha \in \mathcal{AP})[\beta_i = \alpha \vee \beta_i = \neg\alpha]\}$ and for any $c \in \mathcal{C}$, a proposition in $\mathcal{AP}$ appears at most once [27].

[e] For each $\sigma \in \Sigma$, f_σ is parametrized by $p \in \mathcal{P}$ and $\ell \in \mathcal{L}$. The set $\mathcal{X}$ is positively invariant [28] under f_σ. Due to the compactness and invariance of $\mathcal{X}$, each f_σ has a compact, attractive limit set parametrized by $p \in \mathcal{P}$, denoted $L^+(p, \sigma)$ [28].

[f] $\textsc{Pre}(\sigma)$ maps mode σ to a logical sentence over $\mathcal{Z} \times \mathcal{P}$ that needs to be satisfied whenever the machine switches to mode σ from any other mode.

[g] $\textsc{Post}(\sigma)$ maps mode σ to a logical sentence over $\mathcal{Z} \times \mathcal{P}$ that is satisfied when the trajectories of f_σ reach an ε-neighborhood of its limit set.

[h] The reset map assigns $(z, p) \in \mathcal{Z} \times \mathcal{P}$ to a subset of $\mathcal{P}$ which contains parameter values p' for which there is a mode σ, with pre-condition $\textsc{Pre}(\sigma)$ satisfied by (z, p').

[i] The transition map sends (z, p, σ) to (z, p', σ') if (z, p) satisfies $\textsc{Post}(\sigma)$ and (z, p') satisfies $\textsc{Pre}(\sigma')$ with $p' \in s(z, p)$.

The *configuration* of $\mathbf{H}$ is a tuple $[z, p, \sigma]$. A transition from σ_i to σ_{i+1} (if any) is forced and can occur once the trajectory of $f_{\sigma_i}(z, p)$ hits an ε-neighborhood of its limit set.[2]

[2]Written $L^+(p, \sigma) \oplus \mathcal{B}_\varepsilon(0)$, where $\oplus$ denotes the Minkovski (set) sum and $\mathcal{B}_\varepsilon(x)$ is the open ball of radius ε centering at x.

This model describes a continuous dynamical system that switches between different control laws based on some discrete logic. The discrete logic is a formal system consisting of the atomic propositions in $\mathcal{AP}$ together with the logical connectives $\neg$ and $\wedge$. The semantics of the set of logical sentences $\mathcal{C}$ generated, expresses the convergence guarantees available for each component vector field f_σ in the form $\textsc{Pre}(\sigma) \implies \textsc{Post}(\sigma)$. (Formally, it is $\textsc{Pre}(\sigma) \Diamond \textsc{Post}(\sigma)$, where $\Diamond$ is the temporal logic symbol for *eventually*; however time here is abstracted away.) The switching conditions, on the other hand, depend not only on the continuous variables, but also on the discrete control modes: a transition may, or may not be triggered, depending on which mode the hybrid system is in. Control over $\mathbf{H}$ is exercised by selecting a particular sequence of parametrized control modes. Resetting in the parameters of the system, activates transitions to specific modes, which in turn steer the continuous variables toward predetermined limit sets.

Compared to the definition for a hybrid system given in [29], $\mathbf{H}$ is special because it does not involve jumps in the continuous states, its discrete transitions are forced, and the continuous vector fields converge. The model for $\mathbf{H}$, however, also allows the system evolution to be influenced by, possibly externally set, continuous and discrete variables (p and ℓ). In addition, initial and final states are not explicitly marked, allowing the machine to accept a *family* of input languages instead of a single one as that of [29].

Let us now describe more formally the limit sets of the continuous dynamics f_σ, and highlight their link to the $\textsc{Pre}$ and $\textsc{Post}$ conditions of each mode. To this end, let $\phi_\sigma(t; x_0, \ell, p)$ denote the flow of vector field $f_\sigma(x; \ell, p)$ passing from x_0 at time $t = 0$. The positive limit set in control mode σ, when parametrized by p is expressed as

$$L^+(p, \sigma) = \big\{ y \mid \exists \{t_n\} : \lim_{n \to \infty} t_n = \infty, \\ \phi_\sigma(t_n; x_0, \ell, p) \to y \text{ as } n \to \infty, \, \forall\, x_0 \in \Omega(p, \sigma) \big\} \ ,$$

where $\Omega(p, \sigma) \subseteq \mathcal{X}$ is the attraction region of control mode σ parametrized by p. We assume that $L^+(p, \sigma)$, for a given σ and for all $p \in \mathcal{P}$, is path connected.[3] If it is not, and there are isolated components $L_i^+(p, \sigma)$ for $i = 1, \ldots, B(\sigma)$, one can refine a control mode σ into $\sigma^1, \ldots \sigma^{B(\sigma)}$, one for each $L_i^+(p, \sigma)$. For simplicity we assume that for $\mathbf{H}$ of Definition 2, Σ does not afford any further refinement. For each discrete location σ, the formulae $\textsc{Pre}(\sigma)$ and $\textsc{Post}(\sigma)$ are related to the limit sets and their attraction regions in that location as follows:[4]

$$(z, p) \equiv (x, \ell, p) \models \textsc{Post}(\sigma) \iff \\ (x, \ell, p) \in \big\{ (x, \ell, p) \mid x \in L^+(p, \sigma) \oplus \mathcal{B}_\varepsilon(0), \ell \in \mathcal{L} \big\}$$

$$(z, p) \equiv (x, \ell, p) \models \textsc{Pre}(\sigma) \iff \\ (x, \ell, p) \in \big\{ (x, \ell, p) \mid x \in \Omega(p, \sigma), \ell \in \mathcal{L} \big\} \ .$$

A state z, which together with some parameter p satisfy $\textsc{Pre}(\sigma)$, can *evolve* along $\phi_\sigma(t; z, p)$ to some other composite

[3]A set is path connected if any two points in the set can be connected with a path (a continuous map from the unit interval to the set) [30].

[4]Symbol $\models$ is read "satisfies," and we write $(z, p) \models c$ if the valuation of logical sentence $c \in \mathcal{C}$ over variables (z, p) is true.

state z' for which (z', p) satisfies $\textsc{Post}(\sigma)$, and we write $z \overset{\sigma[p]}{\hookrightarrow} z'$. A sequence of the form $(\sigma_1, p_1)\cdots(\sigma_N, p_N)$ is an *input* to $\mathbf{H}$, specifying how control modes are to be concatenated and parametrized in $\mathbf{H}$. The input sequence is a data word. We say that a data atom (σ_1, p_1) is *admissible* at the initial setting (z_0, p_0) of $\mathbf{H}$ if $p_1 = p_0$ and (z_0, p_1) satisfies $\textsc{Pre}(\sigma_1)$, or if $p_1 \in s(z_0, p_0)$ and (z_0, p_1) satisfies $\textsc{Pre}(\sigma_1)$. A data atom (σ', p') is *admissible* in $\mathbf{H}$ at configuration $[z, p, \sigma]$, if there is a $[z, p', \sigma'] \in \mathcal{Z}\times\mathcal{P}\times\Sigma$ such that $T([z, p, \sigma]) = [z, p', \sigma']$. A pair of data atoms $(\sigma_j, p_j)(\sigma_{j+1}, p_{j+1})$ is admissible at configuration $[z, p, \sigma]$ if (σ_j, p_j) is admissible at $[z, p, \sigma]$, and there is a composite state $z' \in \mathcal{Z}$ to which z evolves to under σ_j parameterized by p_j (i.e. $z \overset{\sigma_j[p_j]}{\hookrightarrow} z'$), giving a configuration $[z', p_j, \sigma_j]$ where the second input atom (σ_{j+1}, p_{j+1}) is also admissible. A data word w is admissible in $\mathbf{H}$ if every prefix of w is admissible.

The planning problem addressed in this paper is a reachability problem: for a given $\textsc{Spec} \in \mathcal{C}$, the goal is to design a control policy that drives the system from its initial configuration to a configuration where $\textsc{Spec}$ is satisfied.

Problem 1: Given a hybrid agent $\mathbf{H}$ at an initial configuration satisfying a formula $\textsc{Init} \in \mathcal{C}$, find an admissible sequence $(\sigma_1, p_1)\cdots(\sigma_N, p_N)$ so that the configuration of $\mathbf{H}$ after N transitions, for some $N \in \mathbb{N}$, satisfies $\textsc{Spec} \in \mathcal{C}$.

III. Discrete Abstractions

Each hybrid agent $\mathbf{H}$ can be associated to a special one-register automaton. Since we do not mark initial and final states in $\mathbf{H}$, the discrete system is a *semiautomaton*.[5] We say that this one-register semiautomaton is *induced* by $\mathbf{H}$. The relation between the state-parameter pairs of $\mathbf{H}$, and the states of the register semiautomaton is expressed by a map.

Definition 3 (Valuation map): The valuation map V_M: $\mathcal{Z}\times\mathcal{P} \to \mathcal{Q} \subseteq \{\mathbf{1}, \mathbf{0}\}^{|\mathcal{AP}|}$ is a function that maps a state-parameter pair (z, p), to a binary vector $q \in \mathcal{Q}$ of length $|\mathcal{AP}|$. The entry at position i in q, denoted $q[i]$, is $\mathbf{1}$ or $\mathbf{0}$ depending on whether α_i in $\mathcal{AP}$ evaluated at (z, p) is true or false, respectively. For $q \in \mathcal{Q}$, we denote this valuation $\alpha_i(z, p) = q[i]$,

With reference to $\mathbf{H}$ and $V_M(\cdot)$, a set valued map $\boldsymbol{\lambda} : \mathcal{P}\times\mathcal{Q}\times\mathcal{Q}\times\Sigma \to 2^{\mathcal{P}}$ is defined as

$$\boldsymbol{\lambda}(\tau; q, q', \sigma') \mapsto \big\{p' \mid (\forall z : V_M(z, \tau) = q) \\ \big[p' \in s(z, \tau) \wedge (z, p') \models \textsc{Pre}(\sigma') \wedge V_M(z, p') = q'\big]\big\}. \quad (1)$$

Note that $\boldsymbol{\lambda}$ may not be defined for every q, σ and q'.

The register semiautomaton $\mathbf{R}(\mathbf{H})$ which serves as an abstraction of $\mathbf{H}$ is now defined as follows.[6]

Definition 4 (Induced register semiautomaton): The deterministic finite one-way register semiautomaton induced by hybrid agent $\mathbf{H}$ (with reference to Definition 2), is a tuple $\mathbf{R}(\mathbf{H}) = \langle \mathcal{Q}, \Sigma, \mathcal{P}, 1, \tau, \Delta\rangle$, with

$\mathcal{Q}$	a finite set of states;[a]
Σ	the alphabet (same as that of $\mathbf{H}$);
$\mathcal{P}$	the data set (same as that of $\mathbf{H}$);
1	an m-dimensional *array* register;
$\tau : 1 \mapsto \mathcal{P}\cup\{\varnothing\}$	the register assignment;[b]
Δ	a finite set of read,[c] and write[d] transitions.

[a] The set of states is defined as

$$\mathcal{Q} = \big\{q \in \{\mathbf{0}, \mathbf{1}\}^{|\mathcal{AP}|} : \exists\,(z, p) \in \mathcal{Z}\times\mathcal{P} : V_M(z, p) = q\big\}.$$

[b] Given input data atom $(\sigma, p) \in \Sigma\times\mathcal{P}$, the set $\mathrm{Test}(\tau)$ consists of formulae defined by the grammar $\varphi ::= p = \tau \mid \pi_j(p) = \pi_j(\tau) \mid p \in \boldsymbol{\lambda}(\tau; q, q', \sigma) \mid \neg\varphi \mid \varphi \wedge \varphi$, where $q, q' \in \mathcal{Q}$ and $j \in \{1, \ldots, |\Sigma|\}$.

[c] A read transition $(q, \varphi_r) \overset{\sigma_j}{\to} (q', \mathrm{right})$ where φ_r is $\tau = p_j$, is defined if *for all* z such that $V_M(z, \tau) = q$, the pair (z, τ) satisfies $\textsc{Pre}(\sigma_j)$ and there exists a continuous evolution $z \overset{\sigma_j[p_j]}{\hookrightarrow} z'$ such that $V_M(z', p_j) = q'$.

[d] A write transition $(q, \varphi_w) \overset{\sigma_j}{\to} \big(q', \mathrm{stay}\big)$ where φ_w is $p \in \boldsymbol{\lambda}(\tau; q, q', \sigma_j) \wedge \neg\big[\pi_{\iota(\sigma_j)}(p_j) = \pi_{\iota(\sigma_j)}(\tau)\big]$, is defined if there exists a parameter p in $\mathcal{P}$ such as the set $\big\{p' \in \mathcal{P} \mid p' \in \boldsymbol{\lambda}(p; q, q', \sigma_j) \wedge \pi_{\iota(\sigma_j)}(p') \neq \pi_{\iota(\sigma_j)}(p)\big\}$ is not empty.

With the machine at configuration $[j, q, \tau]$, and upon receiving input $w_j = (\sigma_j, p_j)$, if $p_j = \tau$, the read transition $(q, \varphi_r) \overset{\sigma_j}{\to} (q', \mathrm{right})$ applies as long as it is in Δ. In this case, the machine moves to state q', and the input read head advances one position. If, on the other hand, $\pi_{\iota(\sigma_j)}(p_j) \neq \pi_{\iota(\sigma_j)}(\tau)$, while data value p_j belongs to the set $\boldsymbol{\lambda}(\tau; q, q', \sigma_j)$ for some $q' \in \mathcal{Q}$, then the write transition $(q, \varphi_w) \overset{\sigma_j}{\to} (q', \mathrm{stay})$ applies as long as it is in Δ. Then the machine reaches q' without moving the input read head, and overwrites the content of its register with p_j.

A data atom (σ, p) is *admissible* at configuration $[j, q, \tau]$ if there is transition in Δ that applies to $[j, q, \tau]$ when (σ, p) appears at the input. A pair of data atoms $(\sigma_1, p_1)(\sigma_2, p_2)$ is admissible if there is a transition in Δ that applies to some configuration $[j, q, \tau]$ on input (σ_1, p_1), taking $\mathbf{R}(\mathbf{H})$ to configuration $[j+1, q', \tau']$, where some other transition in Δ applies on input (σ_2, p_2). A data word w is admissible if every prefix of w is admissible.

For the special class of hybrid systems considered here, the registers of such a model turn out to be adequate for capturing the continuous behavior, up to the resolution allowed by the given set of atomic propositions. The only change we introduce to the standard register automaton model is the capacity to perform inequality tests on the data; however, given that the most basic logical operation is set inclusion, and in order for a machine to be able to do any equality test, it has to do so by means of a conjunction of inequalities. Thus, the extension we propose does not fundamentally require any additional computational power on the part of the machine. Compared to the register automaton of Definition 1, the construction of Definition 4 differs. First, there are no initial and final states—it is a semiautomaton—and second, there is a single register that stores an *array* rather than a single variable. Register tests, though, are performed element-wise

[5] A semiautomaton is like a typical finite state machine but without initial and final states. All its states can serve as both initial and final. Thus, it accepts a set of (regular) languages, but once one assigns specific initial and final states, then it becomes an automaton and only accepts one language.

[6] This machine has only one register, so to lighten notation we drop the argument from the current assignment of the register.

on the register.

We treat the write transitions in $\mathbf{R}(\mathbf{H})$ as *silent*, given that they do not advance the read head of the machine. Read transitions, on the other hand, are *observable*. A concatenation of any number of silent transitions with a single observable transition triggered by input atom w_j, taking the machine from state q to state q' is denoted $q \overset{w_j}{\rightsquigarrow} q'$, and we refer to this transition sequence as a *composite* transition. Since only one observable transition is taken in a composite transition, the read head advances only one step. A composite transition is called *maximal* if the machine cannot make another transition without reading a new data atom.

Proposition 1: Let $w = w_1 \cdots w_n$ be an admissible input sequence for $\mathbf{R}(\mathbf{H})$. Then any maximal composite transition from state q to state q' contains either a single read transition, or a write transition followed by a read transition.

Proof: Let $\mathbf{R}(\mathbf{H})$ be at configuration $[j, q, \tau]$. Suppose $\mathbf{R}(\mathbf{H})$ takes a composite transition, $q \overset{w_j}{\rightsquigarrow} q'$, where $w_j = (\sigma_j, p_j)$. If $p_j = \tau$ then the machine jumps from q to q' and advances the read head by one position. In this case, the configuration changes from $[j, q, \tau]$ to $[j+1, q', \tau]$. If $p_j \neq \tau$, no read transition applies, which means that a write transition must have taken place. Once this write transition is completed, τ has the value of p_j. The machine still reads $w_j = (\sigma_j, p_j)$ on the input tape, since the read head has not advanced. But now, upon reading w_j again, the machine finds $\tau = p_j$. A read transition is triggered and the read head advances one position forward. Configuration $[j, q, \tau]$ changes first to some intermediate $[j, q^t, \tau']$ after the write transition, and then to the final $[j+1, q', \tau']$ after the read transition. In any case, a composite transition either includes a single read transition or a write transition followed by a read transition—the latter referred to as a write-read transition pair. ∎

Proposition 1 implies that a write transition cannot occur without either immediately triggering a read transition or halting the machine. Before we establish the relation between $\mathbf{H}$ and $\mathbf{R}(\mathbf{H})$, we need to introduce some terminology.

Definition 5: A *labeled transition system* is a tuple $\mathbf{T} = \langle Q, \Sigma, \rightarrow \rangle$ with components

Q	a set of states;
Σ	a set of labels;
$\rightarrow \subseteq Q \times \Sigma \times Q$	a transition relation.

The transition $(q_1, \sigma, q_2) \in \rightarrow$ is commonly denoted $q_1 \overset{\sigma}{\rightarrow} q_2$.

In $\mathbf{T}$, we distinguish a subset of labels $\Sigma_\epsilon \subseteq \Sigma$, and we call a transition labeled with a label from that subset, silent; otherwise observable. We write $q \rightsquigarrow q'$ to denote that q' is reachable from q with an arbitrary number of silent transitions, and $q \overset{\sigma}{\rightsquigarrow} q'$ if q' is reachable from q a composite transition containing one observable transition labeled σ.

Definition 6 (Weak (observable) simulation [31]): Consider two (labeled) transition systems over the same input alphabet Σ: $\mathbf{T}_1 = (Q_1, \Sigma, \rightarrow_1)$ and $\mathbf{T}_2 = (Q_2, \Sigma, \rightarrow_2)$. Let $\Sigma_\epsilon \subset \Sigma$ be a set of labels for silent transitions. An ordered binary relation $\mathfrak{R} \subseteq Q_1 \times Q_2$ is a *weak (observable) simulation* if: (i) $\mathfrak{R}$ is total, i.e., for any $q_1 \in Q_1$ there exists a state $q_2 \in Q_2$ such that $(q_1, q_2) \in \mathfrak{R}$, and (ii) for every ordered pair $(q_1, q_2) \in \mathfrak{R}$, if there exists a state $q_1' \in Q_1$ which the machine can reach with a composite transition from q_1, i.e., $q_1 \overset{\sigma}{\rightsquigarrow}_1 q_1'$, then there also exists $q_2' \in Q_2$ that can be reached with a composite transition from q_2, i.e., $q_2 \overset{\sigma}{\rightsquigarrow}_2 q_2'$, and $(q_1', q_2') \in \mathfrak{R}$. Then $\mathbf{T}_2$ weakly simulates $\mathbf{T}_1$ and we write $\mathbf{T}_2 \gtrsim \mathbf{T}_1$.

In other words, $\mathbf{T}_2$ weakly simulates $\mathbf{T}_1$ if any input admissible in $\mathbf{T}_1$ is also admissible in $\mathbf{T}_2$. In that sense, a hybrid agent that weakly simulates its induced register semiautomaton can implement every input sequence admissible in the register semiautomaton. Indeed, this is always the case:

Theorem 1: Hybrid agent $\mathbf{H}$ weakly simulates its induced register semiautomaton $\mathbf{R}(\mathbf{H})$ in the sense that the ordered total binary relation $\mathfrak{R}$ defined as $(q, z) \in \mathfrak{R} \Leftrightarrow \exists\ p \in \mathcal{P}, V_M(z, p) = q$, satisfies

$$(q, z) \in \mathfrak{R} \text{ and } q \overset{w_j}{\rightsquigarrow} q' \text{ with } w_j = (\sigma_j, p_j) \implies \exists\, z' \in \mathcal{Z} : z \overset{\sigma_j[p_j]}{\hookrightarrow} z' \text{ with } \left(q', z'\right) \in \mathfrak{R}\ . \quad (2)$$

Proof: First note that relation $\mathfrak{R}$ is total by construction, since any state $q \in \mathcal{Q}$ is by definition the image under the valuation map V_M of some $(z, p) \in \mathcal{Z} \times \mathcal{P}$. To establish that $\mathfrak{R}$ is a weak simulation, let the register semiautomaton $\mathbf{R}(\mathbf{H})$ be at configuration $[j, q, \tau]$, with a state q for which we can find a state $z \in \mathcal{Z}$ in $\mathbf{H}$ to related it with: $(q, z) \in \mathfrak{R}$. Suppose now that $\mathbf{R}(\mathbf{H})$ takes a (composite) transition w_j; then according to Proposition 1, this composite transition consists of either a single read transition, $[j, q, \tau] \overset{w_j}{\rightarrow} \left[j+1, q', \tau\right]$, or a write-read pair: $[j, q, \tau] \overset{w_j}{\rightarrow} \left[j, q^t, \tau'\right] \overset{w_j}{\rightarrow} \left[j+1, q', \tau'\right]$. The mere existence of a transition originating from q on input (σ_j, p_j) ensures that for *any* z that satisfies $V_M(z, \tau) = q$, it holds that either (z, τ) satisfies $\textsc{Pre}(\sigma_j)$, with $\tau = p_j$ (if we have a single read transition), or that the $\iota(\sigma_j)$ components of the control parameter and register do not match, meaning $\pi_{\iota(\sigma_j)}(\tau) \neq \pi_{\iota(\sigma_j)}(p_j)$, and there is some $q^t \in \mathcal{Q}$, such that $V_M(z, p_j) = q^t$ and $p_j \in \boldsymbol{\lambda}(\tau; q, q^t, \sigma_j)$ (if we have a write-read transition pair). In the latter case, by the definition of $\boldsymbol{\lambda}$, we know that (z, p_j) must satisfy $\textsc{Pre}(\sigma_j)$. If w_j triggers a single read transition (the case $\tau = p_j$), then there must exist a continuous evolution in $\mathbf{H}$ in control mode σ_j parameterized by p_j, taking z to z' (namely, $z \overset{\sigma_j[p_j]}{\hookrightarrow} z'$) at which $V_M(z', p_j) = q'$; it follows that $(q', z') \in \mathfrak{R}$. If, instead, w_j triggers a write-read transition pair, then after updating its register by setting $\tau = p_j$, $\mathbf{R}(\mathbf{H})$ still reads (σ_j, p_j) as the input. Since (z, p_j) satisfies $\textsc{Pre}(\sigma_j)$ and now $\tau = p_j$, $\mathbf{R}(\mathbf{H})$ has to take a read transition to reach q'. The argument of the previous case applies and completes the proof. ∎

Theorem 1 suggests that while all admissible input sequences in $\mathbf{R}(\mathbf{H})$ will be also admissible in $\mathbf{H}$, it is also the case that a control policy that takes $\mathbf{H}$ from its present state into another that satisfies $\textsc{Spec}$ might not have a matching run in $\mathbf{R}(\mathbf{H})$. To ensure a matching run we need to strengthen the link between the two models. Theorem 2 gives sufficient conditions for a weak *bisimulation* to be established between $\mathbf{H}$ and $\mathbf{R}(\mathbf{H})$. Note, however, that simulation is sufficient for planning purposes.

Theorem 2: Given the hybrid agent $\mathbf{H}$ and its induced register semiautomaton $\mathbf{R}(\mathbf{H})$, the binary relation $\mathfrak{R}$ defined

as $(q,z) \in \mathfrak{R} \iff \exists p \in \mathcal{P}, V_M(z,p) = q$ is a weak bisimulation relation under the following conditions:

a) given $p \in \mathcal{P}$, for any two $z_1, z_2 \in \mathcal{Z}$, if $V_M(z_1,p) = V_M(z_2,p) = q$, then whenever (z_1,p) satisfies $\text{PRE}(\sigma)$ we have (z_2,p) also satisfying $\text{PRE}(\sigma)$. In addition, the parametrized control mode $\sigma[p]$ that takes z_1 to z_1', takes z_2 to some z_2' for which $V_M(z_1',p) = V_M(z_2',p)$.

b) given $p \in \mathcal{P}$, for any two $z_1, z_2 \in \mathcal{Z}$, for which $V_M(z_1,p) = V_M(z_2,p) = q$, if $p' \in s(z_1,p)$ and $V_M(z_1,p') = q'$, then $p' \in s(z_2,p)$ and $V_M(z_2,p') = q'$.

c) given $z \in \mathcal{Z}$, and any $p_1, p_2 \in \mathcal{P}$, if (z,p_1) satisfies $\text{PRE}(\sigma)$ and (z,p_2) does not satisfy $\text{PRE}(\sigma)$, then the $\iota(\sigma)$ components of p_1 and p_2 do not match: $\pi_{\iota(\sigma)}(p_1) \neq \pi_{\iota(\sigma)}(p_2)$.

Proof: Since we know from Theorem 1 that $\mathbf{H}$ weakly simulates $\mathbf{R}(\mathbf{H})$, we only need to show the implication (2) in the opposite direction: if the conditions above are satisfied, then given $(q,z) \in \mathfrak{R} \subseteq \mathcal{Q} \times \mathcal{Z}$, $z \overset{\sigma[p]}{\hookrightarrow} z' \implies \exists q' \in \mathcal{Q} : q \overset{(\sigma,p)}{\rightsquigarrow} q' \wedge (q',z') \in \mathfrak{R}$. To this end, select any $p^o \in \mathcal{P}$ such that $q = V_M(z,p^o)$, and examine the two possibilities:

Case 1, where $p^o = p$: The evolution $z \overset{\sigma[p]}{\hookrightarrow} z'$, implies that the pair (z,p) satisfies $\text{PRE}(\sigma)$. Given condition a), we know that any other z_1 such that $V_M(z_1,p) = q$ (and is therefore related to q), will also make a pair (z_1,p) that satisfies $\text{PRE}(\sigma)$. This means that any such z_1 will evolve to some z_1' in mode σ parameterized by p. We can collect all these limit points z_1' to a set $Z'(p) \triangleq \{z' \mid z \overset{\sigma[p]}{\hookrightarrow} z',$ for some $z : V_M(z,p) = q\}$. Condition a) also ensures that any $z_1', z_2' \in Z'(p)$, $V_M(z_1',p) = V_M(z_2',p) = q'$ for some state $q' \in \mathcal{Q}$. Based on Definition 4, there exists a read transition in $\mathbf{R}(\mathbf{H})$ taking q to q' upon input (σ,p). All $z' \in Z'(p)$ give $V_M(z',p) = q'$ and thus $(q',z') \in \mathfrak{R}$.

Case 2, where $p^o \neq p$: without loss of generality assume that (z,p^o) does not satisfy $\text{PRE}(\sigma)$; otherwise, we can have $z \overset{\sigma[p^o]}{\hookrightarrow} z'$, which reduces this case to Case 1. Condition c) then requires that $\pi_{\iota(\sigma)}(p^o) \neq \pi_{\iota(\sigma)}(p)$. Since we are given that $z \overset{\sigma[p]}{\hookrightarrow} z'$, we can conclude that (z,p) satisfies $\text{PRE}(\sigma)$. The definition of the reset map then suggests that $p \in s(z,p^o)$. Let $V_M(z,p) = q^t$. Condition b) ensures that for any state z_1 that makes a pair (z_1,p^o) such that $V_M(z_1,p^o) = V_M(z,p^o) = q$, it is $p \in s(z_1,p^o)$ and $V_M(z_1,p) = V_M(z,p) = q^t$. Let the set of all such states z_1 be Z_1. Since (z,p) satisfies $\text{PRE}(\sigma)$ and $z \in Z_1$, using condition a) we have that for all $z_1 \in Z_1$, the pair (z_1,p) satisfies $\text{PRE}(\sigma)$. Recall that $\boldsymbol{\lambda}(p^o;q,q^t,\sigma) = \{p \in \mathcal{P} \mid (\forall z : V_M(z,p^o) = q)[p \in s(z,p^o) \wedge (z,p) \models \text{PRE}(\sigma) \wedge V_M(z,p) = q^t]\}$, and note that this set is nonempty since it always contains p. Therefore, a write transition $(q,\varphi_w) \xrightarrow{\sigma} (q^t, \text{stay})$ applies on input atom (σ,p) with formula φ_w expressed as $p \in \boldsymbol{\lambda}(p^o;q,q^t,\sigma) \wedge \pi_{\iota(\sigma)}(p^o) \neq \pi_{\iota(\sigma)}(p)$. This write transition takes q to q^t and updates the the register with p. Now we have Case 1. ∎

Register automata do not have a standard graphical representation, so it is not clear how reachability analysis using graph search methods can be performed. A state may be reached either by a read, or a write transition; however, the nature of the incoming transition matters when it comes to reasoning as to what happens next. Configurations, on the other hand, cannot be enumerated due to the inclusion of the continuous data in τ. For these reasons, we suggest an embedding of $\mathbf{R}(\mathbf{H})$ into a labeled transition system, hereby referred to as the *transformation semiautomaton*, which brings out some information about register updates and the nature of transitions. The embedding is used here for computational convenience; there may be ways of working directly on $\mathbf{R}(\mathbf{H})$.

Definition 7: The transformation semiautomaton of $\mathbf{R}(\mathbf{H})$ is a tuple $\mathbf{TR}(\mathbf{H}) = \langle \bar{\mathcal{Q}}, \bar{\Sigma}, \bar{\Delta} \rangle$ consisting of:

$\bar{\mathcal{Q}} \subseteq \mathcal{Q} \times \{\mathfrak{p}, \mathfrak{p}'\}$	a finite set of states;[a]
$\bar{\Sigma} = \Sigma \cup \Lambda \cup \{\theta\}$	a set of transition labels;[b]
$\bar{\Delta}$	a set of transitions of four types.[c–f]

[a] $\bar{\mathcal{Q}}$ contains couples where the first element is a state of $\mathbf{R}(\mathbf{H})$ and the second element is a symbol, either $\mathfrak{p}$ or $\mathfrak{p}'$. Whenever a state in $\mathbf{R}(\mathbf{H})$ is reached with a write transition, its corresponding state in $\mathbf{TR}(\mathbf{H})$ is marked with a $\mathfrak{p}'$.

[b] Subset Λ contains labels indexing all different possible write transitions in $\mathbf{R}(\mathbf{H})$, each write transition assigned to a unique λ in Λ. The singleton $\{\theta\}$ contains an auxiliary label marking trivial write transitions (write self-loops) in $\mathbf{R}(\mathbf{H})$ which do not modify the register content.

[c] One type is $(q,\mathfrak{p}) \overset{\lambda_i}{\dashrightarrow} (q',\mathfrak{p}')$, defined if q' is accessible from q in $\mathbf{R}(\mathbf{H})$ via a write transition $(q,\varphi_w) \xrightarrow{\sigma} (q', \text{stay})$.

[d] Another type is $(q,\mathfrak{p}) \overset{\theta}{\dashrightarrow} (q,\mathfrak{p}')$, defined if q is accessible from *any* $q' \in \mathcal{Q}$ via a write transition.

[e] A third type is $(q,\mathfrak{p}) \xrightarrow{\sigma} (q',\mathfrak{p})$, defined if there exists a read transition $(q,\varphi_r) \xrightarrow{\sigma} (q', \text{right})$ and q is not accessible via a write transition from any other state in $\mathbf{R}(\mathbf{H})$.

[f] The last type is $(q,\mathfrak{p}') \xrightarrow{\sigma} (q',\mathfrak{p})$, defined if there exists a read transition $(q,\varphi_r) \xrightarrow{\sigma} (q', \text{right})$ and q is accessible via at least one write transition from some state in $\mathbf{R}(\mathbf{H})$.

We define the injective function $\boldsymbol{\Lambda} : \Lambda \to \mathcal{Q} \times \mathcal{Q}$ that singles out the transition of $\mathbf{TR}(\mathbf{H})$ that is labeled by the particular label in Λ. Consequently, $\boldsymbol{\lambda}(\tau;q,q',\sigma) \equiv \boldsymbol{\lambda}(\tau;\boldsymbol{\Lambda}(\lambda),\sigma)$.

It is straightforward to show that $\mathbf{TR}(\mathbf{H})$ and $\mathbf{R}(\mathbf{H})$ are weakly bisimilar; intuitively, one merges any pair of states of the form $(q,\mathfrak{p})$ and $(q,\mathfrak{p}')$ in $\mathbf{TR}(\mathbf{H})$.

Proposition 2: The transformation semiautomaton $\mathbf{TR}(\mathbf{H})$ and the induced register semiautomaton $\mathbf{R}(\mathbf{H})$ are weakly bisimilar: there exists an ordered binary relation $\mathfrak{R}$ on $\mathcal{Q} \times \bar{\mathcal{Q}}$ such that: (i) $\mathfrak{R}$ is total, and (ii) whenever[7] $(q,(q,*)) \in \mathfrak{R}$ there exists a read or write transition from q to some q' in $\mathbf{R}(\mathbf{H})$ for some $\sigma \in \Sigma$, then there exists a composite transition in $\mathbf{TR}(\mathbf{H})$, $(q,*) \overset{a}{\rightsquigarrow} (q',*)$ with $a \in \Sigma \cup \Lambda$ and $(q',(q',*)) \in \mathfrak{R}$. Conversely, if there is a transition in $\mathbf{TR}(\mathbf{H})$ taking $(q,*)$ to $(q',*)$, then there exists a composite transition in $\mathbf{R}(\mathbf{H})$ taking q to q' while $(q,(q,*)) \in \mathfrak{R}$ and $(q',(q',*)) \in \mathfrak{R}$.

Proof: Define $\mathfrak{R}$ implicitly as a partition on $\bar{\mathcal{Q}}$ in which $(q,\mathfrak{p})$ and $(q,\mathfrak{p}')$ belong in the same block and the equivalence class is labeled by q. Note that $\mathbf{TR}(\mathbf{H})$ is constructed in a way that guarantees $\mathfrak{R}$ to be total. First take the case of a read transition $(q,\varphi_r) \xrightarrow{\sigma} (q', \text{right})$ in $\mathbf{R}$. By construction,

[7] $*$ stands for either $\mathfrak{p}$ or $\mathfrak{p}'$.

$\mathbf{TR}(\mathbf{H})$ can either take $(q,\mathfrak{p}) \xrightarrow{\sigma} (q',\mathfrak{p})$ or $(q,\mathfrak{p}') \xrightarrow{\sigma} (q',\mathfrak{p})$, and obviously $(q',(q',\mathfrak{p})) \in \mathfrak{R}$. Any transition $(q,\varphi_w) \xrightarrow{\sigma} (q',\mathrm{stay})$ in $\mathbf{R}(\mathbf{H})$ can be matched by the transition $(q,\mathfrak{p}) \xrightarrow{\lambda} (q',\mathfrak{p}')$ in $\mathbf{TR}(\mathbf{H})$, and since $(q',(q',\mathfrak{p}')) \in \mathfrak{R}$, it follows that $\mathbf{TR}(\mathbf{H}) \gtrsim \mathbf{R}(\mathbf{H})$.

The other direction is shown as follows: consider any $(q,*) \overset{a}{\rightsquigarrow} (q',*)$, with $a \in \Sigma \cup \Lambda$. If $a \in \Sigma$, three possible cases arise: a) $(q,\mathfrak{p}) \xrightarrow{a} (q',\mathfrak{p})$, b) $(q,\mathfrak{p}') \xrightarrow{a} (q',\mathfrak{p})$, and c) $(q,\mathfrak{p}) \xrightarrow{\theta} (q,\mathfrak{p}') \xrightarrow{a} (q',\mathfrak{p})$, for some $\theta \in \Theta$. In all three cases, the *end* state $(q',\mathfrak{p})$ is related to q' via $\mathfrak{R}$ and there is always a transition of the form $(q,\varphi_r) \xrightarrow{a} (q',\mathrm{right})$ in $\mathbf{R}(\mathbf{H})$ by construction of $\mathbf{TR}(\mathbf{H})$. If $a \in \Lambda$, then by construction there is a transition in $\mathbf{R}(\mathbf{H})$: $(q,\varphi_w) \xrightarrow{\sigma} (q',\mathrm{stay})$ and since q' can be reached by a write transition, there exists $(q,\mathfrak{p}) \xrightarrow{a} (q',\mathfrak{p}') \in \bar{\Delta}$. Since both $(q,(q,\mathfrak{p}))$ and $(q',(q',\mathfrak{p}'))$ belong in $\mathfrak{R}$, we conclude that it is also the case that $\mathbf{R}(\mathbf{H}) \gtrsim \mathbf{TR}(\mathbf{H})$. ∎

IV. Time-Optimal Planning

Any transition in $\mathbf{R}(\mathbf{H})$ may incur a cost, but in this context we assume only observable transitions do so. The cost of an observable transition in $\mathbf{R}(\mathbf{H})$, corresponding to a continuous evolution in $\mathbf{H}$, is determined by the component continuous dynamics active during that time period, the initial conditions for the continuous states, and the assignment of parameters. The component dynamics when $\mathbf{H}$ is at control mode σ is expressed as $\dot{x} = f_\sigma(x,\ell,p)$, with $\sigma \in \Sigma$, $p \in \mathcal{P}$, $\ell \in \mathcal{L}$, and $x \in \mathcal{X}$. An incremental cost function $R : \mathcal{X} \times \mathbb{R}_+ \to \mathbb{R}_+$ is used to define the *atomic cost* $g_\sigma(x,\ell,p)$ for $\mathbf{H}$ evolving in control mode σ along flow $\phi_\sigma(t;x,\ell,p)$ for $t \in [t_0,t_f]$:

$$g_\sigma(x,\ell,p) \overset{\text{def}}{=} \int_{t_0}^{t_f} R\big(x(t)\big)\,\mathrm{d}t \ .$$

We define the incremental cost using the indicator function: $R(x(t)) \overset{\text{def}}{=} \mathbf{1}_{\{L^+(p,\sigma)\oplus\mathcal{B}_\varepsilon(0)\}^{\mathrm{c}}}\big(x(t)\big)$ where $\mathbf{1}_A$ denotes the indicator function of set A, $\oplus$ the Minkovski (set) sum, $\mathcal{B}_\varepsilon(x)$ is the open ball of radius ε of appropriate dimension centered at x, and $\{\cdot\}^{\mathrm{c}}$ denotes set complement. Other choices are of course possible; however, this choice of R yields an atomic cost g_σ which measures the time it takes the flow of vector field $f_\sigma(x,\ell,p)$ to hit an ε-neighborhood of $L^+(p,\sigma)$:

$$g_\sigma(x,\ell,p) = \int_0^\infty R\big(\phi_\sigma(t;x,\ell,p)\big)\,\mathrm{d}t \ . \tag{3}$$

In an admissible data word $w = (\sigma_1,p_1)\ldots(\sigma_N,p_N)$, for any σ_{i-1}, σ_i appearing consecutively in $\mathrm{str}(w)$, the data value p_{i-1} that comes along with σ_{i-1} should match with some $z \in \mathcal{Z}$, in a way that the pair (z,p_{i-1}) satisfies $\textsc{Post}(\sigma_{i-1})$. In addition, for that same z, (z,p_{i-1}) either also satisfies $\textsc{Pre}(\sigma_i)$, or its image under the reset map s contains some other $p_i \neq p_{i-1}$, for which (z,p_i) satisfies $\textsc{Pre}(\sigma_i)$.

We can thus eliminate the dependence of g_σ on $z = (x,\ell)$ in (3) by conservatively over-approximating the atomic cost for a transition $\sigma_i \in \mathrm{str}(w)$, using a function of parameters:

$$\bar{g}_{\sigma_i}\big(p_{i-1},p_i\big) \overset{\text{def}}{=} \max_{z:(z,p_{i-1})\in S} \int_0^\infty R\big(\phi_{\sigma_i}(t;z,p_i)\big)\,\mathrm{d}t \tag{4}$$

where

$$S \overset{\text{def}}{=} \begin{cases} \{z \mid (z,p_{i-1}) \models \textsc{Pre}(\sigma_i) \wedge \textsc{Post}(\sigma_{i-1})\}, & p_i = p_{i-1} \\ \{z \mid (z,p_{i-1}) \models \textsc{Post}(\sigma_{i-1}), & \\ \quad p_i \in s(z,p_{i-1}), (z,p_i) \models \textsc{Pre}(\sigma_i)\}, & \text{otherwise.} \end{cases}$$

The integral in (4) does not always have to be computed explicitly. This is because the time required for a continuous state $x \in \mathcal{X}$ to converge under controller σ to an ε neighborhood of $L^+(p,\sigma)$ can be over-approximated using Lyapunov-based techniques, discussed briefly in Appendix B.

The *accumulated cost* J_w for executing data word $w = (\sigma_1,p_1)\ldots(\sigma_N,p_N)$ from configuration $[z,p,\sigma]$, assuming that w is admissible at $[z,p,\sigma]$, is upper bounded by

$$\begin{aligned} J_w(z,p) &\le \bar{J}_w(z,p) \\ &\overset{\text{def}}{=} g_{\sigma_1}(z,p_1) + \sum_{i=2}^{N} \bar{g}_{\sigma_i}\big(\mathrm{val}_w(i-1),\mathrm{val}_w(i)\big) \ . \end{aligned}$$

The optimization problem can then be stated as follows:

Problem 2: With the hybrid agent $\mathbf{H}$ at an initial configuration $[z_0,p_0,\sigma_0]$, where (z_0,p_0) satisfies $\textsc{Init} \in \mathcal{C}$ and $\sigma_0 \in \Sigma$, find out of all admissible sequences $w = (\sigma_1,p_1)\cdots(\sigma_N,p_N)$ solving Problem 1, the one that achieves $\min_{\{p_j\}_{j=1}^N} \bar{J}_w(z_0,p_0)$.

Let us define a set of ternary vectors $\mathsf{q} \in \{\mathbf{0},\mathbf{1},*\}^{|\mathcal{AP}|}$, where the semantics of $*$ at location i within a vector q, is that atomic proposition α_i can be either true or false. In that sense, a ternary vector q can be identified with a *set* of binary vectors, and thus we may write $q \in \mathsf{q}$. Recalling formula $\textsc{Spec}$ in Problem 1, we represent the set of all binary vectors q for which a pair (z,p) with $V_M(z,p) = q$ satisfies $\textsc{Spec}$, by a single ternary vector $\mathsf{q}_{\textsc{Spec}}$. If $\mathsf{q}_{\textsc{Spec}}[i] = *$, this means that α_i does not appear in $\textsc{Spec}$.

Problem 2, defined on $\mathbf{H}$, can now be recast as a problem defined on $\mathbf{R}(\mathbf{H})$ and $\mathbf{TR}(\mathbf{H})$:

Problem 3: For a given $\textsc{Spec} \in \mathcal{C}$ and a pair (z_0,p_0) satisfying $\textsc{Init}$, and for any $q_f \in \mathsf{q}_{\textsc{Spec}}$, find a data word $w = (\sigma_1,p_1)\cdots(\sigma_N,p_N)$ for which

1) there exists a walk $\mathfrak{w}$ in $\mathbf{TR}(\mathbf{H})$ from $(q_0,\mathfrak{p})$ to $(q_f,\mathfrak{p})$ with $q_0 = V_M(z_0,p_0)$ and $q_f \in \mathsf{q}_{\textsc{Spec}}$, such that its projection to Σ, denoted $\mathfrak{w} \lfloor_\Sigma$, satisfies $\mathfrak{w} \lfloor_\Sigma = \mathrm{str}(w)$, and
2) $\bar{J}_w(z_0,p_0)$ is minimized with respect to $\{p_j\}_{j=1}^N$, where $p_N = p_f$ as specified in $\textsc{Spec}$.

Condition 2) restates the optimality requirement of Problem 2. Theorem 1 ensures that w is a solution to Problem 1.

At a first glance, the way that Problem 3 is stated seems to suggest a solution approach: path search for 1) and DP for 2). This intuition is basically correct, with the exception that there are technical complications in the implementation. For the first part, path search may not yield a solution—condition 1) calls for a *walk*, rather than a simple path; because of the restrictions imposed by the set-valued map $\boldsymbol{\lambda}$, the run might have to revisit some state in $\bar{\mathcal{Q}}$ a few times in order to bring the parameter to the value specified in $\textsc{Spec}$. In addition, due to the possible existence of cycles in a candidate walk $\mathfrak{w}$, a straightforward application of DP may fail. Finally, the decision sequence in

DP may not be finite, and an implication of this is that no "worst case" policy to improve on.

To solve Problem 3 we introduce modifications in the two methodological components: graph search and DP. To find the walks we augment $\mathbf{TR}(\mathbf{H})$ by adding the initial and desired final states based on Problem 3, and obtain a deterministic finite state automaton (DFA). Then we generate a regular expression (RE) of this DFA.[8] From this RE we can construct successively longer walks satisfying condition 1) of Problem 3, and then optimize them using a modified version of dynamic programming discussed next. With cycles allowed there is no theoretical bound on the length of admissible strings in the DFA, and thus we limit the number of walks that can be checked for optimality by setting an upper bound on the cost, based on an assumed maximum affordable cost, and the cost of the least expensive observable transition.

The modification of dynamic programming used for parameter optimization in candidate walks is based on reversing the direction of search for the optimal solution. Instead of taking the worst case and try to improve on it, we start with the best case—the shortest possible walk found—and test locally for optimality. If an optimal solution is encountered, the search stops. The details of these technical modifications are discussed in the following sections.

A. Finding walk candidates

With initial state $(q_0, \mathfrak{p})$ and final state $(q_f, \mathfrak{p})$, we obtain the DFA $\langle \mathbf{TR}(\mathbf{H}), (q_0, \mathfrak{p}), (q_f, \mathfrak{p}) \rangle = \langle \bar{\mathcal{Q}}, \bar{\Sigma}, \bar{T}, (q_0, \mathfrak{p}), (q_f, \mathfrak{p}) \rangle$ and find an RE, denoted $\mathbf{RE}(\mathbf{H})$, associated with this DFA using known methods [33]. Replacing every occurrence of the Kleene star $*$ in $\mathbf{RE}(\mathbf{H})$ with a natural number, gives a set $\mathfrak{W}(m)$ of all admissible walks of length m in the DFA, $\mathfrak{W}(m) \stackrel{\text{def}}{=} \{\mathfrak{w} | \mathfrak{w} \in \mathbf{RE}(\mathbf{H}), |\mathfrak{w}| = m\}$. Any walk in $\mathfrak{W}(m)$ has a matching admissible input data word on $\mathbf{R}(\mathbf{H})$ (Theorem 1). However, $\mathbf{TR}(\mathbf{H})$ has no information on specific register values, and thus the corresponding admissible data word may not comply with the requirement for p_0 and p_f. To remove inadmissible walks we develop a procedure for translating a walk in $\mathbf{TR}(\mathbf{H})$ to a family $\bar{w}$ of data words in $\mathbf{R}(\mathbf{H})$, in which all individual words $\bar{w}$ have the same symbol string $\mathrm{str}(w)$ but different the data value assignments. The domains of possible data value assignments is specified by a sequence of set-valued maps:

Given a walk $\mathfrak{w} = u_1 \cdots u_m$, set $i := 1$, $j := 1$, and for $1 \leq i \leq m$, distinguish three cases:

1) $u_i \in \Sigma$: then, set $\sigma_j := u_i$, $\bar{w}_j := (\sigma_j, p_j)$, $M_j(\cdot) := \mathrm{id}_P(\cdot)$, $j := j + 1$, $i := i + 1$;
2) $u_i \in \Lambda$: then, set $\sigma_j := u_{i+1}$, $M_j(\cdot) := \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(u_i), \sigma_j)$, $\bar{w}_j := (\sigma_j, p_j)$, $j := j + 1$, $i := i + 2$;
3) otherwise, set $\sigma_j := u_{i+1}$, $M_j(\cdot) := \mathrm{id}_P(\cdot)$, $\bar{w}_j := (\sigma_j, p_j)$, $j := j + 1$, $i := i + 2$.

In the above, $\mathrm{id}_P : p \mapsto p$ is the identity map on $\mathcal{P}$. A walk $\mathfrak{w} = u_1 \ldots u_m$ is thus translated into a family of data words $\bar{w} = (\sigma_1, p_1) \cdots (\sigma_N, p_N)$, and a sequence of set-valued maps $M_i(\cdot) : \mathcal{P} \to 2^{\mathcal{P}}$, for $i \in \{1, \ldots, N\}$.

To check whether a walk $\mathfrak{w}$ generates a data word $w \in \bar{w}$ that can match the parameter specifications, we use the sequence of set-valued maps $\{M_i(\cdot)\}_{i=1}^N$ constructed by this procedure and verify the *consistency* condition

$$\{p \in \mathcal{P} \,|\, \exists z \in \mathcal{Z} : V_M(z, p) = q_f\} \cap M_N \circ \cdots \circ M_1(p_0) \neq \varnothing. \quad (5)$$

B. A modified dynamic programming algorithm

Let the maximal allowable cost for any solution to Problem 3 be $J_{\max}$. Then, if the minimum cost of executing an observable transition labeled with $\sigma \in \Sigma$ is some $J_{\min} > 0$, an upper bound U on the length of data words translated from walks is $U \stackrel{\text{def}}{=} \left\lceil \frac{J_{\max}}{J_{\min}} \right\rceil$.

If (5) holds, then there exists a sequence of N parameter values $\{p_i\}_{i=1}^N$ with $|\mathfrak{w} \restriction_\Sigma| = N \leq m$ such that $w = (\sigma_1, p_1) \cdots (\sigma_N, p_N)$ is an admissible input for $\mathbf{R}(\mathbf{H})$. Input w takes the register semiautomaton from configuration $[1, q_0, p_0]$ to some configuration $\left[N + 1, q_f, p_f\right]$ where $q_f \in \mathfrak{q}_{\text{SPEC}}$. Among all walks which pass the test (5), we pick the shortest one as the candidate most likely to yield the optimal solution to Problem 3.

In the case a candidate walk is found, we modify the standard DP algorithm of [34], and apply it in its new form to obtain an optimal sequence of parameters. We first obtain a set of subsets of $\mathcal{P}$, denoted $\{\mathcal{P}_i\}_{i \in \mathrm{dom}(\bar{w})}$, where each $\mathcal{P}_i$ consists of all parameter values that can be used to parametrize a control mode (data atom) at stage i of execution of an input data word in $\bar{w}$, and is found as

$$\mathcal{P}_i \stackrel{\text{def}}{=} M_i \circ \ldots \circ M_1(p_0) \;\cap\; \left(M_N \circ \ldots \circ M_{i+1}\right)^{-1}(S) \;.$$

where $S = \{p \mid \exists z \in \mathcal{Z} : V_M(z, p) = q_f\}$. Closed-form expressions for the optimal values of parameters and the accumulated cost can be obtained in the special case where the continuous dynamics of each control mode associated with a data atom in the input $\bar{w}$ is linear, and the related atomic cost is quadratic. In more general (nonlinear) cases, sets $\mathcal{P}_i$, for $i \in \mathrm{dom}(\bar{w})$ may have to be discretized. Naturally, the resolution of this discretization affects the optimality of the solution obtained.

Assuming a general case where closed form solutions for the optimal parameters are impractical, consider a partition of $\mathcal{P}_i$ into K_i blocks, enumerate the blocks, and let $p_i[k]$ denote the representative of the parameter values belonging to block $k \in \{1, \ldots, K_i\}$. The DP algorithm selects the optimal sequence of parameter representatives $\{p_1{}^*, \ldots, p_N{}^*\}$ in the family $\bar{w}$ as follows.

Let $i = N$, and for each $p_{N-1}[k] \in \mathcal{P}_{N-1}$ for $k = 1, \ldots, K_{N-1}$, set

$$P_N{}^*(p_{N-1}[k]) := \arg \min_{p_N[j] \in \mathcal{P}_N} \bar{g}_{\sigma_N}\big(p_{N-1}[k], p_N[j]\big) \quad (6a)$$

$$\bar{J}_N^*(p_{N-1}[k]) := \bar{g}_{\sigma_N}\big(p_{N-1}[k], P_N{}^*(p_{N-1}[k])\big) \;. \quad (6b)$$

This process constructs two discrete maps on $\mathcal{P}_{N-1}$. The first map associates $p_{N-1}[k]$ to the value $P_N{}^*(p_{N-1}[k])$, which the parameter should be reset to in order to trigger the transition

[8] A regular expression is defined recursively as follows [32]: 1) empty string ϵ and all $\sigma \in \Sigma$ are REs 2) If r and s are REs, then rs (concatenation), $(r+s)$ (union) and r^*, s^* (Kleene-closure) are REs; 3) There are no RE other than those constructed by applying rules 1 and 2 above a finite number of times.

with the minimum cost. The second map associates a representative $p_{N-1}[k]$, assumed to be written at the register before the σ_N transition is triggered, to the minimum accumulated cost $\bar{J}_N^*(p_{N-1}[k])$ incurred during the σ_N transition.

For $i = N-1, \ldots, 2$ we repeat

$$P_i{}^*(p_{i-1}[k]) := \arg \min_{p_i[j] \in \mathcal{P}_i} \Big\{ \bar{g}_{\sigma_i}(p_{i-1}[k], p_i[j]) + \bar{J}_{i+1}^*(p_i[j]) \Big\} \tag{7a}$$

$$\bar{J}_i^*(p_{i-1}[k]) := \bar{g}_{\sigma_i}\big(p_{i-1}[k], P_i{}^*(p_{i-1}[k])\big) + \bar{J}_{i+1}^*\big(P_i{}^*(p_{i-1}[k])\big) \ . \tag{7b}$$

Finally, for $i = 1$ we finish by setting

$$P_1{}^*(p_0) := \arg \min_{p_1[j] \in \mathcal{P}_1} \big\{ g_{\sigma_1}(z_0, p_1[j]) + \bar{J}_2^*(p_1[j]) \big\} \tag{8a}$$

$$\bar{J}_1^*(z_0, p_0) := \bar{g}_{\sigma_1}\big(z_0, P_1{}^*(p_0)\big) + \bar{J}_2^*(P_1^*(p_0)) \ . \tag{8b}$$

Then the optimal sequence of parameter representatives $\{p_1{}^*, \ldots, p_N{}^*\}$ is obtained iteratively. This sequence identifies a particular member of the input word family $\bar{w}$ as the solution w^* to Problem 3. The (conservative) accumulated cost is given by $\bar{J}_1^*(z_0, p_0)$.

The time complexity of DP with discretized parameter space is polynomial $\mathcal{O}(NK^2)$, in which $N = |\bar{w}|$ and $K = \max_{i=1,\ldots,N} K_i$. To generate a candidate data word for DP, one checks (5), which in the worst case requires the enumeration of all data words of maximal length U.

The solution obtained is sub-optimal because: (i) in the case when a weak bisimulation cannot be established between $\mathbf{H}$ and $\mathbf{R}(\mathbf{H})$, there may exist sequences of parametrized control modes with lower cost that are only admissible in $\mathbf{H}$; (ii) the accumulated cost computed in $\mathbf{R}(\mathbf{H})$ over-approximates the time needed for executing w from (z, p) in $\mathbf{H}$, and it is conceivable that a word with a higher accumulated cost $\bar{J}_w$ might actually be executed faster; (iii) if the upper bound on the length of data words U is smaller than the length of the optimal solution, then the optimal solution is not analyzed; and (iv) the discretization on the parameter space introduces quantization errors.

V. Case study: Fetch my printout

We illustrate the method with an example. The problem to be solved is as follows: a mobile manipulator is instructed to fetch a document at the printer and deliver it to the user. The locations of the printer and the user is known.

The mobile manipulator consists of two subsystems: a wheeled mobile platform, and a two degree-of-freedom robotic arm moving on a vertical plane. The robot exhibits three different behaviors: (i) it can move from some initial position to a desired posture (position and orientation), (ii) it can reach out with its arm, grasp an object in the workspace and hold it, and (iii) it can reach out with its arm to a desired position and release an object held in its gripper. When the robot performs any one of these maneuvers we say that it is in a particular *control mode*, and these modes are labeled a, b, and c, respectively. The controller responsible for each of these behaviors is given to us *a priori* and no access to its low-level software is permitted. We have to determine the sequence and parameterization of the controllers to achieve the desired outcome: printout delivered to user.

One obvious (to a human) solution is to bring the robot to the vicinity of the printer, have it reach out and pick up the printout from the output tray, then navigate to the user and deliver the paper stack. However, it is not clear how such a plan can be generated automatically.

A. Control mode a: nonholonomic control

The mobile platform is modeled kinematically as a unicycle

$$\dot{x} = v \cos\vartheta \qquad \dot{y} = v \sin\vartheta \qquad \dot{\vartheta} = \omega$$

where v the velocity and ω the angular velocity are the control inputs. Control mode a steers the robot's posture $X_p \stackrel{\text{def}}{=} (x, y, \vartheta)^\intercal \in \mathbb{R}^2 \times \mathbb{S}^1$ from an initial configuration $X_{p0} = \big(x_0, y_0, \vartheta_0\big)^\intercal$ to a target $X_{pf} = \big(x_f, y_f, \vartheta_f\big)^\intercal$. A coordinate transformation naturally reduces this problem to steering the unicycle to the origin.[9]

The controller in mode a is designed based on [36]. Let $x_1 = \vartheta \mod (2\pi)$, $x_2 = x\cos\vartheta + y\sin\vartheta$, $x_3 = -2(x\sin\vartheta - y\cos\vartheta) + \big(\vartheta \mod (2\pi)\big)(x\cos\vartheta + y\sin\vartheta)$. Define

$$\omega = -k_1 x_1 + k_3 \frac{x_3^r}{x_2} \qquad v = -k_1 x_2 + 0.5(x_1 x_2 - x_3)\omega$$

where $k_1, k_3 > 0$ are control gains, $r = \frac{m}{n}$, and $m < n$ are odd naturals. The closed loop system is

$$\dot{x}_1 = -k_1 x_1 + k_3 \frac{x_3^r}{x_2} \quad \dot{x}_2 = -k_1 x_2 \quad \dot{x}_3 = -k_3 x_3^r \ . \tag{9}$$

Vector field f_a is defined by the right-hand sides of (9). It can be verified that with $C \stackrel{\text{def}}{=} x_3(0)^{1-r}$ and $T \stackrel{\text{def}}{=} \frac{C}{k_3(1-r)}$, when $t \le T$, $x_3(t) = \operatorname{sign}(C)\big||C| - k_3(1-r)t\big|^{\frac{1}{1-r}}$, and for $t \ge T$, $x_3(t) = 0$. Then for $t \ge T$, $x_1(t) = x_1(T)e^{k_1(t-T)}$, $x_2(t) = x_2(T)e^{k_1(t-T)}$, where $x_1(T) = \frac{x_1(0) + \frac{k_3}{x_2(0)} \int_0^T e^{2k_1 s}(C - k_3(1-r)s)^{\frac{r}{1-r}} \, \mathrm{d}s}{e^{k_1 T}}$ and $x_2(T) = x_2(0)e^{k_1 T}$. $\textsc{Post}(a)$ is defined as the area where x_1 and x_2 are in a ball of radius ε of the origin. It is guaranteed that $\textsc{Post}(a)$ is satisfied in time at most $\max\big\{T + \frac{\ln(\frac{\sqrt{2}x_1(T)}{2\varepsilon})}{k_1}, \frac{\ln(\frac{\sqrt{2}x_2(0)}{2\varepsilon})}{k_1}\big\}$ after switching to control mode a.

B. Control modes b and c: catch and release

In control modes b and c, the robot's arm maneuvers to pick, and release an object, respectively. The arm is mounted on the mobile platform at a height h_p. The lengths of the two arm links are l_1, l_2, and the corresponding joint angles are ψ_1, ψ_2. Let $\Psi \stackrel{\text{def}}{=} (\psi_1, \psi_2)^\intercal$. The workspace of the arm is the set of end-effector absolute positions $p_a = (p_{xa}, p_{ya}, p_{za})^\intercal \in \mathbb{R}^3$, reachable in the sense that given $X_p = (x, y, \vartheta)$ we have

$$\left.\begin{array}{c} \sqrt{(p_{xa}-x)^2 + (p_{ya}-y)^2 + (p_{za}-h_p)^2} \in [\,|l_1 - l_2|, |l_1 + l_2|\,] \\ \tan\vartheta = \frac{p_{ya}-y}{p_{xa}-x} \end{array}\right\} \tag{10}$$

[9] Here, the workspace is obstacle-free. If obstacles are present, one may replace the controller in mode a with one that can handle obstacles, such as [35]. The challenge comes in approximating convergence rate—for this, see Appendix B.

If (10) is true, we write $p_a \in W(X_p)$. The system is kinematically redundant: for a given p_a, many postures X_p can satisfy (10). Let the set of all these postures be $W^{-1}(p_a)$.

Inverse kinematics yields the joint angles $\Psi^d \stackrel{\text{def}}{=} (\psi_1^d, \psi_2^d)^\intercal$ that positions the end-effector to a desired p_a:

$$\psi_2^d = \cos^{-1}\frac{(p_{xa}-x)^2+(p_{ya}-y)^2+(p_{za}-h_p)^2-(l_1^2+l_2^2)}{2l_1l_2}$$
$$\psi_1^d = \tan^{-1}\frac{(p_{za}-h_p)(l_1+l_2\cos\psi_2^d)-l_2\sqrt{(p_{xa}-x)^2+(p_{ya}-y)^2}\sin\psi_2^d}{l_2(p_{za}-h_p)\sin\psi_2^d+\sqrt{(p_{xa}-x)^2+(p_{ya}-y)^2}(l_1+l_2\cos\psi_2^d)}.$$

Let $\Psi^h \stackrel{\text{def}}{=} (\psi_1^h, \psi_2^h)^\intercal$ denote the *center* of the arm's workspace, the joint angle combination for which the distance between the end-effector the workspace boundary is maximized. With the workspace being a compact set, the existence of this joint angle configuration is ensured.

The error in joint angles is written $E_\psi(t) \stackrel{\text{def}}{=} \Psi(t) - \Psi^d$. With direct joint angle control, and with steady state considered reached when $|\psi_1 - \psi_1^d| \leq \varepsilon$ and $|\psi_2 - \psi_2^d| \leq \varepsilon$, vector fields f_b and f_c are defined by the closed loop joint error dynamics $\dot{E}_\psi = -KE_\psi$, where $K \stackrel{\text{def}}{=} \begin{pmatrix} b_1 & 0 \\ 0 & b_2 \end{pmatrix}$. The difference between the two control modes is that while in mode b the arm's gripper is initially open and closes to grasp the object at the desired end-effector position, in mode c the originally closed gripper opens at the arm's desired configuration. With the arm anywhere within its workspace, the maximum time to complete a pick (b) or place (c) maneuver is $T_j = \max\left\{\frac{2\ln\left(\frac{|\psi_1^h-\psi_1^d|}{\varepsilon}\right)}{b_1}, \frac{2\ln\left(\frac{\psi_2^h-\psi_2^d}{\varepsilon}\right)}{b_2}\right\}$.

C. The system model

We model the robot as a hybrid agent $\mathbf{H} = \langle \mathcal{Z}, \mathcal{P}, \Sigma, \iota, \pi_i, \mathcal{AP}, f_\sigma, \textsc{Pre}, \textsc{Post}, s, T\rangle$ with components:

$\mathcal{Z} = \mathcal{X} \times \mathcal{L}$	set of composite states[a]
$\mathcal{P} = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^3$	set of control parameters[b]
$\Sigma = \{a, b, c\}$	set of control modes[c]
$\iota = \{(a,1),(b,2),(c,3)\}$	indexing bijection on Σ
π_i, $i \in \{1,2,3\}$	projection function on $p \in \mathcal{P}$ [d]
f_σ, $\sigma \in \Sigma$	parameterized vector fields[c]
$\mathcal{AP} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$	indexed atomic propositions[e]
$\textsc{Pre} : \Sigma \to \mathcal{C}$	precondition of mode $\sigma \in \Sigma$ [f]
$\textsc{Post} : \Sigma \to \mathcal{C}$	postcondition of mode $\sigma \in \Sigma$ [f]
$s : \mathcal{Z} \times \mathcal{P} \to 2^{\mathcal{P}}$	system parameter reset map[g]
$T : \mathcal{Z}\times\mathcal{P}\times\Sigma \to \mathcal{Z}\times\mathcal{P}\times\Sigma$	mode transition map.

[a] $\mathcal{X} = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^2 \times \mathbb{R}^3$ is the set of continuous variables describing the posture of the platform $X_p \in \mathbb{R}^2 \times \mathbb{S}^1$, the joint angles of the arm $\Psi \in \mathbb{S}^2$, and the position of the manipulated object $X_o \stackrel{\text{def}}{=} (x_o, y_o, z_o)^\intercal \in \mathbb{R}^3$. Here, $\mathcal{L} = \{\mathfrak{g}\}$ contains a single Boolean variable $\mathfrak{g}$ that expresses whether the gripper is closed ($\mathfrak{g} = \mathbf{1}$), or not ($\mathfrak{g} = \mathbf{0}$).

[b] The parameter vector $p = (p_p^\intercal, p_a^\intercal)^\intercal \in \mathcal{P}$ describes the desired posture $p_p \in \mathbb{R}^2 \times \mathbb{S}^1$ for the mobile platform and the absolute position reference $p_a \in \mathbb{R}^3$ for the arm's end-effector. Component $p_a \in \mathbb{R}^3$ parameterizes modes b, c.

[c] In control mode a, the mobile platform evolves according to f_a and converges to a desired posture p_p; in control mode b, the joint angles evolve under f_b, the arm picks up an object at X_o and holds it; in mode c, the joint angles evolve under f_c and the arm releases the object at p_a.

[d] Defined as $\pi_1(p) \stackrel{\text{def}}{=} p_p$, $\pi_2(p) \stackrel{\text{def}}{=} p_a$, $\pi_3(p) \stackrel{\text{def}}{=} p_a$.

[e] Proposition α_1 is $X_p \in p_p \oplus \mathcal{B}_\varepsilon(0)$ and when true, it means that the platform is ε-close to its reference position; α_2 is $X_o \in p_a \oplus \mathcal{B}_\varepsilon(0)$, and when true, the object is in an ε-neighborhood of position p_a; α_3 is $p_a \in W(p_p)$, and when true, it suggests given the platform being at p_p, the parameter component p_a specifying a reference location for the end-effector, is within the reachable workspace. Proposition α_4 is true iff $\mathfrak{g} = 1$.

[f] $\mathcal{C}$ is the set of logical sentences obtained with $\mathcal{AP}$. Table I summarizes the PRE and POST for each mode.

[g] For $p = (p_p^\intercal, p_a^\intercal)^\intercal$ and $p' = (p_p'^\intercal, p_a'^\intercal)^\intercal$, writing $p' \in s(z,p)$ implies that $p_a' \notin p_a + \mathcal{B}_\varepsilon(0)$ or $p_p' \notin p_p + \mathcal{B}_\varepsilon(0)$.

[h] Exactly as in Definition 2.

TABLE I
PRE AND POST MAPS FOR THE CONTROL MODES OF THE HYBRID AGENT.

	a	b	c
PRE	$\neg\alpha_1$	$\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge (\neg\alpha_4)$	$\alpha_1 \wedge (\neg\alpha_2) \wedge \alpha_3 \wedge \alpha_4$
POST	α_1	$\alpha_1 \wedge (\neg\alpha_2) \wedge \alpha_3 \wedge \alpha_4$	$\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge (\neg\alpha_4)$

The induced register semiautomaton for $\mathbf{H}$ is the tuple $\mathbf{R}(\mathbf{H}) = \langle \mathcal{Q}, \Sigma, \mathcal{P} \cup \{\varnothing\}, \tau, \Delta\rangle$, with:

$\mathcal{Q}$	set of states[α]
$\tau : 1 \to \mathcal{P} \cup \{\varnothing\}$	register assignment
Δ	transition relation.[β–γ]

[α] This set can be practically restricted to $\{\mathbf{0000}, \mathbf{1000}, \mathbf{1110}, \mathbf{1011}, \mathbf{1001}, \mathbf{0110}, \mathbf{0001}\}$. More states exist, but for this task of reaching q_f from q_0 (see Section V-D), the remaining states are either unreachable from q_0, or cannot reach q_f, and thus are ignored.

[β] The read transitions are the following:
$(\mathbf{0000}, \varphi_r) \xrightarrow{a} (\mathbf{1000}, \text{right})$, $(\mathbf{1110}, \varphi_r) \xrightarrow{b} (\mathbf{1011}, \text{right})$, $(\mathbf{1011}, \varphi_r) \xrightarrow{c} (\mathbf{1110}, \text{right})$, $(\mathbf{0001}, \varphi_r) \xrightarrow{a} (\mathbf{1001}, \text{right})$, $(\mathbf{0110}, \varphi_r) \xrightarrow{a} (\mathbf{1110}, \text{right})$.

[γ] The write transitions are the following:
$(\mathbf{1000}, \varphi_w) \xrightarrow{a} (\mathbf{0000}, \text{stay})$, $(\mathbf{1000}, \varphi_w) \xrightarrow{b} (\mathbf{1110}, \text{stay})$, $(\mathbf{1011}, \varphi_w) \xrightarrow{a} (\mathbf{0001}, \text{stay})$, $(\mathbf{1011}, \varphi_w) \xrightarrow{c} (\mathbf{1011}, \text{stay})$, $(\mathbf{1001}, \varphi_w) \xrightarrow{c} (\mathbf{1011}, \text{stay})$, $(\mathbf{1110}, \varphi_w) \xrightarrow{a} (\mathbf{0110}, \text{stay})$.

The set-valued maps $\boldsymbol{\lambda}$ appearing in the write transitions of $\mathbf{R}(\mathbf{H})$ are defined through (1):

$$\boldsymbol{\lambda}\big(\tau;\mathbf{1000},\mathbf{0000},a\big) = \big\{p' \in \mathcal{P} \,|\, p_a' = p_a, p_p' \in \{\mathbb{R}^2 \times \mathbb{S}^1 \setminus W^{-1}(p_a)\} \setminus \{p_p\}\big\}$$
$$\boldsymbol{\lambda}\big(\tau;\mathbf{1000},\mathbf{1110},b\big) = \big\{p' \in \mathcal{P} \,|\, p_p' = p_p \in W^{-1}(X_o), p_a' = X_o\big\}$$
$$\boldsymbol{\lambda}\big(\tau;\mathbf{1011},\mathbf{0001},a\big) = \big\{p' \in \mathcal{P} \,|\, p_p' \in \mathbb{R}^2 \times \mathbb{S}^1 \setminus \{W^{-1}(p_a)\}, p_a' = p_a\big\}$$
$$\boldsymbol{\lambda}\big(\tau;\mathbf{1011},\mathbf{1011},c\big) = \big\{p' \in \mathcal{P} \,|\, p_p' = p_p, p_a' \in W(p_p) \setminus \{p_a\}\big\}$$
$$\boldsymbol{\lambda}\big(\tau;\mathbf{1001},\mathbf{1011},c\big) = \big\{p' \in \mathcal{P} \,|\, p_p' = p_p, p_a' \in W(p_p)\big\}$$
$$\boldsymbol{\lambda}\big(\tau;\mathbf{1110},\mathbf{0110},a\big) = \big\{p' \in \mathcal{P} \,|\, p_p' \in W^{-1}(p_a) \setminus \{p_p\}, p_a' = p_a\big\} \quad .$$

For a fixed tuple $(q, q', \sigma) \in \mathcal{Q} \times \mathcal{Q} \times \Sigma$, the set-valued map $\boldsymbol{\lambda}(\cdot\,; q, q', \sigma)$ maps $\tau \in \mathcal{P}$ to a subset of $\mathcal{P}$, and can be inverted on appropriate subsets of $\mathcal{P}$:

$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1000},\mathbf{0000},a\big)=\{p\in\mathcal{P}|p_p\in\mathbb{R}^2\times\mathbb{S}^1\setminus\{\{p'_p\}\cup W^{-1}(p_a)\},p_a=p'_a\}$$
$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1000},\mathbf{1110},b\big)=\{p\in\mathcal{P}|p_p=p'_p\in W^{-1}(p'_a),p_a=\mathbb{R}^3\setminus W(p'_p)\}$$
$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1011},\mathbf{0001},a\big)=\{p\in\mathcal{P}|p_p\in W^{-1}(p'_a),p_a=p'_a\}$$
$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1011},\mathbf{1011},c\big)=\{p\in\mathcal{P}|p_p=p'_p,p_a\in W(p'_p)\setminus\{p'_a\}\}$$
$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1001},\mathbf{1011},c\big)=\{p\in\mathcal{P}|p_p=p'_p,p_a\in\mathbb{R}^3\setminus W(p'_p)\}$$
$$\boldsymbol{\lambda}^{-1}\big(p';\mathbf{1110},\mathbf{0110},a\big)=\{p\in\mathcal{P}|p_p\in W^{-1}(p'_a)\setminus\{p'_p\},p_a=p'_a\}\ .$$

The transformation semiautomaton $\mathbf{TR}(\mathbf{H}) = \langle \bar{\mathcal{Q}}, \overline{\Sigma}, \bar{\Delta} \rangle$ is described graphically in Fig. 1. The assignment of labels $\{\lambda_i\}$ to transitions in $\mathbf{R}(\mathbf{H})$ is done by the function $\boldsymbol{\Lambda} : \Lambda \to \mathcal{Q}\times\mathcal{Q}$. Explicitly, $\boldsymbol{\Lambda}(\lambda_1) = (\mathbf{1000}, \mathbf{0000})$, $\boldsymbol{\Lambda}(\lambda_2) = (\mathbf{1000}, \mathbf{1110})$, $\boldsymbol{\Lambda}(\lambda_3) = (\mathbf{1011}, \mathbf{0001})$, $\boldsymbol{\Lambda}(\lambda_4) = (\mathbf{1001}, \mathbf{1011})$, $\boldsymbol{\Lambda}(\lambda_5) = (\mathbf{1011}, \mathbf{1011})$, $\boldsymbol{\Lambda}(\lambda_6) = (\mathbf{1110}, \mathbf{0110})$.

D. Task specification

Given some initial configuration for the robot, $X_p(0) = (0, 1, \frac{\pi}{4})^\intercal$, $\Psi(0) = \Psi^h = (0, \pi)^\intercal$, $\mathfrak{g} = \mathbf{0}$, and the manipulated object $X_o(0) = (-1, 2, 0.3)^\intercal$, we seek a time-optimal plan for the robot to pick the object and deliver it to a user located at $X_u = (2, 3, 0.4)^\intercal$. To avoid trivial solutions, we assume that $X_o(0) \notin W(X_p(0))$, and $W^{-1}(X_o(0)) \cap W^{-1}(X_u) = \varnothing$, which means the object is not within the vicinity of initial base location, and that the arm cannot deliver the object to the user without the robot base having to reposition itself.

Assume the register initialized with $p_0 = (X_p(0)^\intercal, X_u^\intercal)^\intercal$, which sets the register semiautomaton to state $\mathbf{1000}$. When the user receives the object at time t_f, for $t > t_f$ the system holds. At time t_f, we have $X_u \in W(X_p(t_f)), X_p(t_f) \in \pi_1(p_f)+\mathcal{B}_\varepsilon(0)$, and $X_o = X_u$, $\mathfrak{g} = 0$; thus $\alpha_1, \alpha_2, \alpha_3$ evaluate true. The semiautomaton would then be at state $\mathbf{1110}$, while $\pi_2(p_f) = X_u$. Thus, when (z, p) satisfies SPEC, this means that $V_M(z, p) = \mathbf{1110}$, and $\pi_2(p) = \pi_2(p_f) = X_u$. The objective is thus to find the shortest walk $\mathfrak{w}$ from $(\mathbf{1000}, \mathfrak{p})$ to $(\mathbf{1110}, \mathfrak{p})$ in $\mathbf{TR}(\mathbf{H})$, which ensures (5) is satisfied for some data word in the family $\bar{w}$ given by the translation procedure.

E. Solving the planning problem

For the DFA obtained from $\mathbf{TR}(\mathbf{H})$ with initial $(\mathbf{1000}, \mathfrak{p})$ and final $(\mathbf{1110}, \mathfrak{p})$ states, the equivalent $\mathbf{RE}(\mathbf{H})$ is:

$$\mathbf{RE}(\mathbf{H}) = (\lambda_1\, a)^*\Big(\,\lambda_2\, b\,\big(\,\lambda_3\, a\lambda_4\, c + (\,\lambda_5 + \theta\,)\, c\,\big) \\ \big(\,\theta\, b\,\big(\,\lambda_3\, a\lambda_4\, c + (\lambda_5 + \theta)\, c\,\big) + \lambda_6\, a\,\big)^*\Big)\ . \quad (11)$$

By replacing the Kleene-star in (11) with natural numbers, we obtain strings that correspond to walks of certain length in the graph of $\mathbf{TR}(\mathbf{H})$. Let us denote $\mathfrak{W}(m)$ the set of walks of length m. Substitution in (11) verifies that the set of walks has to be of even length with $m > 3$. For $m = 4$, we find $\mathfrak{W}(4) = \{\lambda_2\, b\, \lambda_5\, c,\ \ \lambda_2\, b\, \theta\, c\}$. Set $\mathfrak{w} = \lambda_2\, b\, \lambda_5\, c$ translates to $\bar{w} = (b, p_1)(c, p_2)$ and $\{M_j(\cdot)\}_{j=1}^2 = \{M_1 = \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(\lambda_2), b), M_2 = \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(\lambda_5), c)\}$, in which $\boldsymbol{\Lambda}(\lambda_2) = (\mathbf{1000}, \mathbf{1110})$ and $\boldsymbol{\Lambda}(\lambda_5) = (\mathbf{1011}, \mathbf{1011})$. The resulted map $M(p_0) = M_2 \circ M_1\big((X_p(0)^\intercal, X_u^\intercal)^\intercal\big) = \varnothing$ since $M_1\big((X_p(0)^\intercal, X_u^\intercal)^\intercal\big) = \{(X_p(0)^\intercal, {p'_a}^\intercal)^\intercal \mid p'_a \in \{X_o(0)\} \cap W(X_p(0))\} = \varnothing$ as the initial position of the object is not within the workspace of the mobile platform. The same procedure applies to the other walk and it turns out none of them satisfies (5). For $m = 6$ (11) generates $\mathfrak{W}(6) = \{\lambda_1\, a\, \lambda_2\, b\, \lambda_5\, c,\ \lambda_1\, a\, \lambda_2\, b\, \theta\, c,\ \lambda_2\, b\, \lambda_3\, a\lambda_4\, c,\ \lambda_2\, b\, \lambda_5\, c\, \lambda_6\, a,$ $\lambda_2\, b\, \theta\, c\, \lambda_6\, a\}$, all of which are rejected. For example, walk $\mathfrak{w} = \lambda_1\, a\, \lambda_2\, b\, \lambda_5\, c$ translates to $w = (a, p_1)(b, p_2)(c, p_3)$ and $M(\cdot) = \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(\lambda_5), c) \circ \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(\lambda_2), b) \circ \boldsymbol{\lambda}(\cdot\,; \boldsymbol{\Lambda}(\lambda_1), a)$, so $M(p_0) = \{(p'^\intercal_p, p'^\intercal_a)^\intercal \mid p'_p \in W^{-1}(X_o(0)), p'_a \in W^{-1}(p'_p) \setminus \{X_o(0)\}\}$; but $\forall\, p'_p \in W^{-1}(X_o(0))$, one has $X_u \notin W^{-1}(p'_p) \setminus \{X_o(0)\}$, unless the user can get the object without the robot's base moving—which is trivial.

Finally, for $m = 8$, we find a walk $\mathfrak{w} = \lambda_1\, a\, \lambda_2\, b\, \lambda_3\, a\, \lambda_4\, c$, which translates to $\bar{w} = (a, p_1)(b, p_2)(a, p_3)(c, p_4)$, and $M_1(\cdot) = \boldsymbol{\lambda}\big(\cdot\,; \boldsymbol{\Lambda}(\lambda_1), a\big)$, $M_2(\cdot) = \boldsymbol{\lambda}\big(\cdot\,; \boldsymbol{\Lambda}(\lambda_2), b\big)$, $M_3(\cdot) = \boldsymbol{\lambda}\big(\cdot\,; \boldsymbol{\Lambda}(\lambda_3), a\big)$, $M_4(\cdot) = \boldsymbol{\lambda}\big(\cdot\,; \boldsymbol{\Lambda}(\lambda_4), c\big)$. Since the composition of maps $M(p_0) = \big\{(p'^\intercal_p, p'^\intercal_a)^\intercal \mid p'_p \in \mathbb{R}^2 \times \mathbb{S}^1, p'_a \in W(p'_p)\big\}$ allows $p'_a = X_u$, this walk is a candidate, and the search is terminated.

Now we resort to DP to obtain the optimal sequence of parameter vectors $p_i = (p_{pi}^\intercal, p_{ai}^\intercal)^\intercal$ for $i = 1, \ldots, 4 = N$. We have $p_4 = p_f$, and must satisfy $\pi_2(p_4) = X_u$. The range of possible parameter values at each stage is:

$$\begin{aligned}
\mathcal{P}_1 &= M_1\big([X_p(0)\, X_u]\big) \cap \big(M_4 \circ M_3 \circ M_2\big)^{-1}\big(W^{-1}(X_u) \times \{X_u\}\big) \\
&= \mathbb{R}^2 \times \mathbb{S}^1 \setminus \{\{X_p(0)\} \cup W^{-1}(X_u)\} \times \{X_u\} \\
&\quad \cap\, W^{-1}(X_o(0)) \times \mathbb{R}^3 = W^{-1}(X_o(0)) \times \{X_u\} \\
\mathcal{P}_2 &= \big(M_4 \circ M_3\big)^{-1}\big(W^{-1}(X_u) \times \{X_u\}\big) \cap M_2 \circ M_1\big([X_p(0)\, X_u]\big) \\
&= \big(\mathbb{R}^2 \times \mathbb{S}^1 \setminus W^{-1}(X_u)\big) \times \big(\mathbb{R}^3 \setminus \{X_u\}\big) \\
&\quad \cap\, W^{-1}(X_o(0)) \times \{X_o(0)\} = W^{-1}(X_o(0)) \times \{X_o(0)\} \\
\mathcal{P}_3 &= M_3 \circ M_2 \circ M_1\big([X_p(0)\, X_u]\big) \cap M_4^{-1}\big(W^{-1}(X_u) \times \{X_u\}\big) \\
&= \big(\mathbb{R}^2 \times \mathbb{S}^1 \setminus W^{-1}(X_o(0))\big) \times \{X_o(0)\} \\
&\quad \cap\, W^{-1}(X_u) \times \mathbb{R}^3 \setminus \{X_u\} = W^{-1}(X_u) \times \{X_o(0)\} \\
\mathcal{P}_4 &= W^{-1}(X_u) \times \{X_u\}\ .
\end{aligned}$$

We discretize the domain of parameter p_p using a polar coordinate system, in which the radial increment between successive parameter settings is 0.06 m, and the angular increment is 10°. Figure 2 shows two sets of possible parameter settings for the position component of p_p, clustered around the object's position at $X_o = (-1, 2, 0.3)^\intercal$, and the user's location at $X_u = (2, 3, 0.4)^\intercal$. The geometric parameters of the robot are $l_1 = l_2 = 0.2$ m, and $h_p = 0.15$ m. Then, after setting $r_{\mathsf{min}}(z) = \sqrt{\max\{0, (l_1 - l_2)^2 - (z - h_p)^2\}}$, and $r_{\mathsf{max}}(z) = \sqrt{(l_1 + l_2)^2 - (z - h_p)^2}$, the domains P_1, P_2, P_3 and P_4 are covered by sets of points $\{p_1[k]\}_{k=1}^{N_1}$, $\{p_2[k]\}_{k=1}^{N_2}$, $\{p_3[k]\}_{k=1}^{N_3}$, and $\{p4[k]\}_{k=1}^{N_4}$, respectively, where $N_1 = N_2 = 36\big\lfloor \frac{r_{\mathsf{max}}(0.3) - r_{\mathsf{min}}(0.3)}{0.06} \big\rfloor$, $N_3 = N_4 = 36\big\lfloor \frac{r_{\mathsf{max}}(0.4) - r_{\mathsf{min}}(0.4)}{0.06} \big\rfloor$.

The DP algorithm described in Section IV runs as follows.

$N = 4$: For every $p_3[k]$, compute (6)

$$P_4{}^*(p_3[k]) = \mathrm{argmin}_{p_4[j] \in \mathcal{P}_4}\ \bar{g}_c(p_3[k], p_4[j])\ .$$

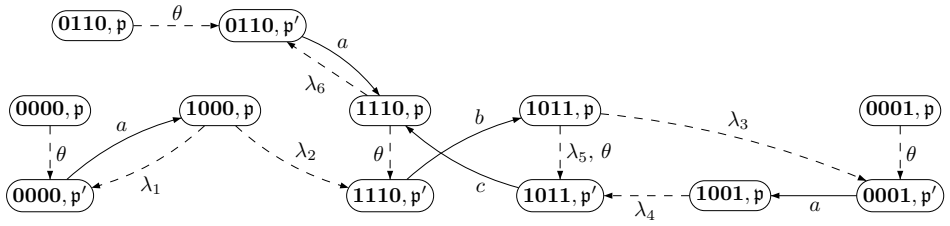

Fig. 1. The transformation semiautomaton $\mathbf{TR}(\mathbf{H})$ of hybrid agent $\mathbf{H}$, for the task specification considered.

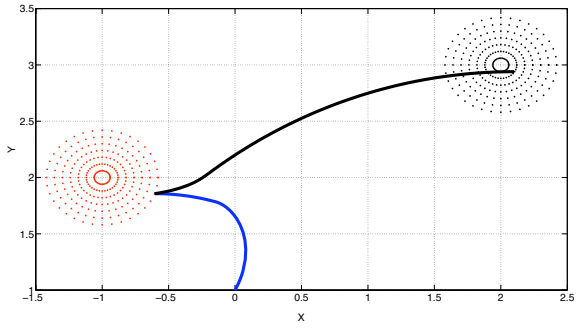

Fig. 2. Discretized workspace for the mobile manipulator and optimal path. The two concentric collection of points mark parameter class representatives around the object and user positions.

$N = 3, 2$: For every $p_2[k]$ and $p_1[k]$, compute (7)

$$P_3{}^*(p_2[k]) = \text{argmin}_{p_3[j]\in\mathcal{P}_3}\ \{\bar{g}_a(p_2[k], p_3[j]) + \bar{J}_4^*(p_3[j])\}$$
$$P_2{}^*(p_1[k]) = \text{argmin}_{p_2[j]\in\mathcal{P}_2}\ \{\bar{g}_b(p_1[k], p_2[j]) + \bar{J}_3^*(p_2[j])\}$$

$N = 1$: Finish by evaluating (8) for $z_0 = [X_p(0)\ X_o(0)]$

$$P_1{}^*(p_0) = \text{argmin}_{p_1[j]\in\mathcal{P}_1}\ \{g_a(z_0, p_1[j]) + \bar{J}_2^*(p_1[j])\}\ .$$

We find $p_1^* = (-0.60, 1.85, 2.79, 2, 3, 0.4)$, $p_2^* = (-0.60, 1.85, 2.79, -1, 2, 0.3)$, $p_3^* = (2.10, 2.91, 2.62, -1, 2, 0.3)$ and $p_4^* = (2.10, 2.91, 2.62, 2, 3, 0.4)$. The accumulated cost is $\bar{J}_w(z_0, p_0) = \bar{g}_c + \bar{g}_a + \bar{g}_b + g_a = 5.49 + 31.72 + 6.08 + 35.80$ seconds. The resulting path on the horizontal plane of the mobile manipulator is shown in Fig. 2.

VI. Conclusions

Certain type of hybrid systems where the continuous dynamics are convergent, afford a partitioning of the continuous state space based on the asymptotic properties of the vector fields and the capacity of the system to re-parametrize its continuous controllers. This partitioning gives rise to purely discrete abstractions—no dynamics on the continuous values—which are weakly simulated by the underlying concrete hybrid dynamics. The abstractions permit a solution to a planning problem at the discrete level. Solutions obtained through this process are in general suboptimal, unless under some special conditions which are identified.

Appendix

A. Example of a register automaton

Consider a language over $\Sigma \times \mathcal{D}$ of the following property: the data value in an atom that immediately follows an atom containing symbol a, has to be the same as the data value in the atom with the symbol a. This language is recognized by a one-way, 2-register automaton $\mathbf{R}_2 = \langle \mathcal{Q}, q_0, F, \Sigma, 2, \tau, \Delta\rangle = \langle \{q_0, q_1, q_2\}, q_0, \{q_0, q_1\}, \{a, b, c\}, 2, \tau : 2 \to \mathbb{R}^2 \cup \{\varnothing\}, \Delta\rangle$ where $\tau_0(1) = \tau_0(2) = \varnothing$, $\varphi_r \Leftrightarrow d = \tau(i)$, $\varphi_w \Leftrightarrow d \neq \tau(i)$, and Δ: $(q_0, \varphi_w) \xrightarrow{b,c} (q_0, 1, \text{right})$, $(q_0, \varphi_w) \xrightarrow{a} (q_1, 2, \text{right})$, $(q_1, \varphi_w) \xrightarrow{a,b,c} (q_2, 2, \text{right})$, $(2, q_1, \varphi_r) \xrightarrow{b,c} (q_0, \text{right})$, $(2, q_1, \varphi_r) \xrightarrow{a} (q_1, \text{right})$, $(2, q_2, \varphi_r) \xrightarrow{a,b,c} (q_2, \text{right})$, $(1, q_2, \varphi_r) \xrightarrow{a,b,c} (q_2, \text{right})$, $(q_2, \varphi_w) \xrightarrow{a,b,c} (q_2, 2, \text{right})$.

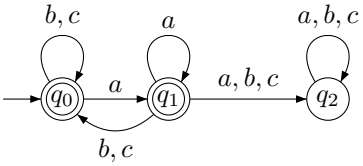

Fig. 3. An example of a 2-register automaton. On receiving initially an atom with symbol a, the machine stores the data value in $\tau(2)$ and enters q_1 which indicates "I have just seen an a." If the data value of the next atom doesn't equal $\tau(2)$, then machine will enter q_2 and stay there forever, otherwise, depending on the symbol in the atom and the machine returns to q_0 (if b or c) or stays in q_1 (if a). The values associated with b or c are stored $\tau(1)$.

B. Asymptotic (T, d) equivalence classes

Denote $\text{dist}\big(x, A\big)$ the distance between point x and set A and let $\text{dist}\big(x, A\big) \stackrel{\text{def}}{=} \inf_{y\in A} \|x - y\|$.

Theorem 3 (Zubov [37]): The set Ω is the region of attraction of a periodic orbit $x = \varphi(t)$ with period T, if and only if there exist two functions $V(x)$ and $W(x)$ defined on Ω satisfying: 1) $V(x)$ is continuous on Ω and the domain of $W(x)$ can be extended to entire $\mathcal{X}$, 2) $V(x) \in (0, 1)$ $\forall x \in \Omega \setminus \varphi$, and $V(x) = 0$ for $\text{dist}\big(x, \varphi\big) = 0$, 3) $W(x) > 0$ for $\text{dist}\big(x, \varphi\big) > 0$ and $W(x) = 0$ for $\text{dist}\big(x, \varphi\big) = 0$, 4)

$$\nabla V^T f(x) = -W(x)\sqrt{1 + \|f\|^2}(1 - V)\ , \qquad (12)$$

5) $\lim_{x\to\partial\Omega} V(x) = 1$.

Proposition 3: Consider a system $\dot{x} = f(x)$ and assume that its trajectories remain inside a compact set Ω and the (attractive) limit set L^+ of the trajectories contain a single, isolated component. Denote $\phi(t; x(0))$ the trajectory of f starting at $x(0)$, and let $V(x)$ be a solution to (12) that satisfies the requirements of Theorem 3. Then the trajectories of f starting in Ω will enter an ε-neighborhood of L^+, in finite time at most $T = \ln\left(\frac{1-c}{1-C}\right)$, where[10] $c \stackrel{\text{def}}{=} \min_{x\in\text{cl}\{L^+\oplus\mathcal{B}_\varepsilon(0)\}} V(x)$ and $C \stackrel{\text{def}}{=} \max_{x\in\Omega} V(x)$.

Proof: Pick $W(x) = \text{dist}\big(x, \varphi\big)$. This choice trivially satisfies the requirements of Theorem 3. Now let $V(x)$ be a solution of (12) that conforms with the conditions of Theorem

[10]cl is used to denote set closure.

3. Then from (12), for $x \in \Omega \setminus \{L^+ \oplus \mathcal{B}_\varepsilon(0)\}$ it follows that $\dot{V} \leq -d(1-V)$ and applying the Comparison Lemma with $C = \max_{x\in\Omega} V(x)$ one obtains $V(x(t)) \leq 1-(1-C)\mathrm{e}^{dt}$. Let $V(x) = c$ be the largest level set of V included in the closure of $L^+ \oplus \mathcal{B}_\varepsilon(0)$. Then, the following upper bound for the time required for the flows starting within the level set $V(x) = C$ to reach $L^+ \oplus \mathcal{B}_\varepsilon(0)$ can be obtained: $t \leq \frac{1}{d} \ln\left(\frac{1-c}{1-C}\right)$. Setting $T \triangleq \ln\left(\frac{1-c}{1-C}\right)$, the proof is completed. ■

REFERENCES

[1] S. K. Banala, S. K. Agrawal, S. H. Kim, and J. P. Scholz, "Novel gait adaptation and neuromotor training results using an active leg exoskeleton," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 2, pp. 216–225, 2010.

[2] J. Schiff, P. Li, and G. Goldstein, "Robotic microsurgical vasovastomy and vasoepididymostomy: a prospective random study in a rat model," *Journal of Urology*, vol. 171, pp. 1720–1725, 2004.

[3] A. Glattfelder and W. Schaufelberger, *Control Systems with Input and Output Constraints*. Springer, 2003.

[4] U. Kiencke and L. Nielsen, *Automotive Control Systems*. Springer-Verlag, 2005.

[5] R. Milner, *Communication and Concurrency*. Prentice Hall, 1989.

[6] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, July 2000.

[7] T. A. Henzinger and P. W. Kopke, "Discrete-time control for rectangular hybrid automata," *Theoretical Computer Science*, vol. 221, pp. 369–392, June 1999.

[8] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[9] M. E. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, F. W. Vaandrager and J. H. van Schuppen, Eds. Springer-Verlag, 1999, vol. 1569, pp. 61–75.

[10] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, pp. 566–571, 2009.

[11] P. Tabuada, "Approximate simulation relations and finite abstractions of quantized control systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Springer-Verlag, 2007, vol. 4416, pp. 529–542.

[12] R. Alur, T. D. Verimag, and F. Ivancic, "Predicate abstractions for reachability analysis of hybrid systems," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 1, pp. 152–199, 2006.

[13] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon, "Supervisory control of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1026–1049, 2000.

[14] J. Lunze, B.Nixdorf, and J. Schroder, "Deterministic discrete-event representations of linear continuous-variable systems," *Automatica*, vol. 35, no. 3, pp. 395–406, 1999.

[15] J. Raisch and S. O'Young, "Discrete approximations and supervisory control of continuous systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 569–573, 1998.

[16] S. Kowalewski, S. Engell, J. Preußig, and O. Stursberg, "Verification of logic controllers for continuous plants using condition/event-system models," *Automatica*, vol. 35, pp. 505–518, 1999.

[17] A. Tiwari and G. Khanna, "Nonlinear systems: Approximating reach sets," in *Hybrid Systems: Computation and Control*, ser. LNCS, R. Alur and G. Pappas, Eds. Springer, Mar. 2004, vol. 2993, pp. 600–614.

[18] Y. Tazaki and J. Imura, "Finite abstractions of discrete-time linear systems and its application to optimal control," in *Proceedings of the 17th IFAC World Congress*, 2008, pp. 4656–4661.

[19] J. L. Piovesan, H. G. Tanner, and C. T. Abdallah, "Discrete asymptotic abstractions of hybrid systems," in *Proceedings of 45th IEEE Conference on Decision & Control*, 2006, pp. 917–922.

[20] S. Graf and H. Saidi, "Construction of abstract state graphs with pvs," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, O. Grumberg, Ed. Springer Berlin Heidelberg, 1997, vol. 1254, pp. 72–83.

[21] M. Kaminski and N. Francez, "Finite-memory automata," *Theoretical Computer Science*, vol. 134, no. 2, pp. 329–363, 1994.

[22] F. Neven, T. Schwentick, and V. Vianu, "Finite state machines for strings over infinite alphabets," *ACM Transactions on Computational Logic*, vol. 5, no. 3, pp. 403–435, 2004.

[23] H. Axelsson, Y. Wardi, M. Egerstedt, and E. I. Verriest, "Gradient descent approach to optimal mode scheduling in hybrid dynamical systems," *Journal of Optimization Theory and Applications*, vol. 136, pp. 167–186, 2008.

[24] M. Mazo Jr. and P. Tabuada, "Symbolic approximate time-optimal control," *Systems & Control Letters*, vol. 60, no. 4, pp. 256 – 263, 2011.

[25] J. Fu and H. Tanner, "Optimal planning on register automata," in *American Control Conference (ACC), 2012*, june 2012, pp. 4540 –4545.

[26] D. Figueira, P. Hofman, and S. Lasota, "Relating timed and register automata," in *EXPRESS'10*, 2010, pp. 61–75.

[27] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[28] H. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[29] J. Lygeros, K. Johansson, S. Simić, and S. Sastry, "Dynamical properties of hybrid automata," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 2–17, 2003.

[30] G. F. Simmons, *Introduction to Topology and Modern Analysis*. Krieger Publishing Company, 2003.

[31] F. Moller, "Logics for concurrency: structure versus automata," *ACM Computing Surveys*, p. 50, 1996.

[32] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic, 2001.

[33] J. A. Brzozowski, "Derivatives of regular expressions," *J. ACM*, vol. 11, pp. 481–494, October 1964.

[34] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, Nov. 2000, vol. I.

[35] L. Valbuena and H. G. Tanner, "Hybrid potential field based control of differential drive mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 68, no. 3-4, pp. 307–322, 2012.

[36] V. S. Ravi N. Banavar, *Switched finite time control of a class of underactuated systems*, ser. Lecture notes in control and information sciences, M. M.Thoma, Ed. Springer, 2006, vol. 333.

[37] V. I. Zubov, *Mathematical Methods for the Study of Automatic Control Systems*. Pergamon Press/Macmillan, 1963.

Jie Fu received the B.S. degree and the M.S. degree in control science and engineering from Beijing Institute of Technology, Beijing, China, in 2007 and 2009 respectively.

She is currently pursuing the Ph.D. degree in mechanical engineering at the University of Delaware. Her research interests include hybrid systems, symbolic control and machine learning.

Herbert G. Tanner received his Eng. Diploma and his Ph.D. in mechanical engineering from the National Technical University of Athens, Greece, in 1996 and 2001, respectively.

From 2001 to 2003 he was a post doctoral fellow with the Department of Electrical and Systems Engineering at the University of Pennsylvania. From 2003 to 2005 he was an assistant professor with the Department of Mechanical Engineering at the University of New Mexico, and he held a secondary appointment with the Electrical and Computer Engineering Department at UNM. He received the NSF Career award in 2005. In 2008 he joined the Department of Mechanical Engineering at the University of Delaware, where he is currently an associate professor. He is currently serving as a director of the graduate certificate program in cognitive science at the University of Delaware. His research focuses on planning and control for multi-agent robotic systems, deployment and inference in mobile sensor networks, and abstraction and planning in hybrid systems. He is a member of the ASME, and IEEE's Control Systems and Robotics and Automation Societies.