(Sub)regular Robotic Languages

Chetan Rawal, Herbert G. Tanner and Jeffrey Heinz

Abstract— **This paper brings together concepts from linguistics and formal language theory and applies them to model robot behavior. This is done by defining a class of formal languages that capture the abstract behavior of robots which can be described as hybrid systems with stable continuous dynamics. It is shown that this class of languages falls within the Subregular hierarchy, thereby enabling computationally efficient operations between elements of the class. Specifically, we show that the languages in question are Star-free, but do not belong into two well-known subclasses, members of which have been used in linguistics to construct models of natural language sound pattern acquisition.**

I. Introduction

This paper is a stepping stone toward automated planning and control design synthesis for robotic systems. We present a language-theoretic framework that allows exploitation of existing results in formal language theory to bring in fresh perspective for analysis of systems whose behaviors can be abstractly described as a sequence of controllers. Such sequences bear close resemblance with linguistic and computational analysis studies since controller sequences can be viewed as strings of symbols or words that describe a logical overall behavior, just like strings of words in a language follow grammatical rules to form logical sentences.

The continuous component of hybrid systems can present significant challenges in terms of characterizing sets of states reachable from given initial conditions, or verifying whether certain state-dependent properties hold over families of continuous-state trajectories. To overcome these challenges, discrete abstractions for continuous and hybrid systems have been proposed [2], [3], [20], [25]. This type of abstract system discretization can broadly be classified into two categories: environment-driven discretization and control-driven discretization [2]. Environment-driven discretization typically involves discretizing the workspace of the system using polygons [3]. System control strategies then evolve within each polygon and the system "flows" between adjacent polygons to reach the desired goal. Highly complex environments can be discretized into simpler polygons that allow specifications in terms of temporal logic formulas [6], [17], [26]. These techniques rely on model checking algorithms [8], [12] to ensure that the design conforms to the desired specifications, and simulation or bisimulation relations [9], [27] to ensure that the design is implementable on an concrete system. Whereas environment-driven strategies discretize the systems' workspace, control-driven strategies discretize the continuous dynamics [7], [14], [16], [20], [29]. The overall behavior is expressed as a sequence of motion primitives [7] or motion description atoms [4], [5], [18].

The approaches using temporal logics and motion description languages are mostly top-down, where specifications are designed first, translated into a discrete dynamical system and then intersected with the target system to block whatever behavior of the concrete system does not match the specifications. The output of the control design algorithm is a supervising strategy that dictates directly the evolution of the low-level continuous dynamics. In contrast, the hypothesis in this paper is based on a more bottom-up design, in which the rules by which *existing* control designs can be interlaced, and the control specifications are encoded into formal grammars, and then the question asked is whether the language generated by the former is contained in the language of the latter. The difference is that instead of instructing the practitioner how to re-design their control strategies, the method suggests ways to *combine existing solutions*, and under certain assumptions analysis can be performed using the rules (grammars) only rather than the automata representation of the system.

We hypothesize that the conditions that facilitate working with grammars (as well as automata) in this framework are those that restrict the languages describing the behaviors of the systems, into the regular category of languages. Specifically, we believe that there is "room at the bottom" of the Chomsky hierarchy, and focus on *regular languages* to capitalize on their superior decidability and cheaper set operation properties [13]. There is precedence in existing literature for the use of regular languages in symbolic planning and control [7], but the expressive power of the resulting models when they are subject to constraints on what input symbol can come after which, has not been explored.

Chetan Rawal and Herbert Tanner are with the Department of Mechanical Engineering at the University of Delaware. Jeff Heinz is with the Department of Linguistics and Cognitive Science at the University of Delaware. This work is supported by NSF under grant #1035577.

Regular languages may not be able to express all the type of specifications that ω-regular [6], [17], [26], [30] or context-free [15] languages can, but they can still capture meaningful constraints and objectives. In this way, we essentially plan to trade off expressive power for computational and analytic simplicity.

We start with a very special class of hybrid systems with stable continuous dynamics and no jumps in the continuous variables, and we implicitly use stability to obtain an asymptotic abstraction [20] for each continuous mode. Asymptotic abstraction [20] assigns discrete symbols to each of the component closed-loop stable continuous dynamics, and abstracts away the exact continuous evolution of the system between regions of the continuous space defined similarly to the partitioning seen in predicate abstraction [1], [10]. This discretization of the continuous dynamics can be shown to yield finite transition systems (or semiautomata) which are weakly bisimilar to the concrete hybrid dynamics [28].

The purpose of this paper is not to document the abstraction procedure (see [28] for that) but rather to introduce the type of languages that describe the behavior of the hybrid systems considered, at the very high abstract level and investigate their position within the Subregular hierarchy in order to determine their properties and the computational and analysis tools they might admit. We refer to this class of languages as *regular robotic languages* (RRLs). The regular robotic languages are shown to be an intersection of languages belonging to particular subregular classes. Due to the structure present in these classes, intersection of languages within a particular class is only as expensive as the intersection of finite sets [11], unlike the intersections of context-free or $\omega-$regular languages [31], [32].

II. Preliminaries

Let Σ represent a finite collection of symbols (alphabet) with Σ^* the Kleene-star of Σ, which is the set of all possible finite strings (including the empty string ϵ) over Σ. The term "word" and "sequence" are used interchangeably and is defined as a generic sequence taken from Σ^*. We write $\Sigma^{\leq k}$ to denote the set of all words with length up to k. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, and $|w|_l$ represents the number of times the string l appears in word w. For example, $|baaa|_{ba} = 1$ and $|baaa|_{aa} = 2$. The symbols $\rtimes$ and $\ltimes$ denote the start and end of a string respectively, and are not considered part of the alphabet. Unless specified otherwise, we assume $w, v, u, f, l, x \in \Sigma^*$, while $k, n, m, t, i, j \in \mathbb{N}$.

Definition 2.1 ([19]): A *regular expression* over an alphabet Σ is

- Any letter of the alphabet;
- ϵ (the null word) and $\emptyset$ (the empty set).
- If ψ and χ are regular expressions, then all of the following are regular expressions: *(i)* $\psi \cup \chi$; *(ii)* $\psi \cap \chi$; *(iii)* $\overline{\psi}$ (the complement with respect to Σ^*); *(iv)* $\psi\ \chi$; *(v)* ψ^*.

Definition 2.2 (Regular Languages): A language is *regular* if it can be represented by a regular expression.[1]

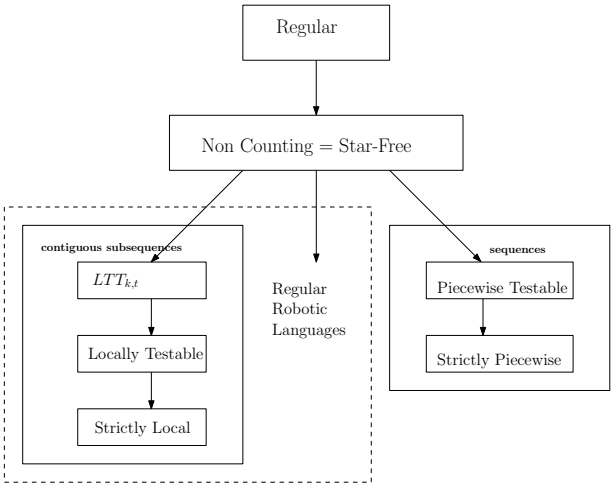

Fig. 1. Regular robotic languages in the Subregular hierarchy.

A. The Subregular Hierarchy

In formal language theory, the Subregular hierarchy [19], [21]–[23] is a subset of the Chomsky hierarchy in which one finds two distinct branches: the *local* branch, and the *piecewise* branch (figure 1).

Definition 2.3 (cf. [21]): The k factors of a word $w \in \Sigma^*$ are the set of k-length contiguous sub-sequences of symbols in w:

$$F_k(w) := \begin{cases} v \in \Sigma^k : w = uvx;\ u, x \in \Sigma^*, & |w| \geq k \\ w & \text{otherwise.} \end{cases}$$

Definition 2.4 (cf. [21]): A (strictly) k-local grammar is a set of k-factors including the start and end of string symbols: $G_{SL_k} \subseteq F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\})$.
Such grammars allow us to specify which words belong in a particular language, by constraining the set of k-factors that they are allowed or forbidden to have.

Definition 2.5 (cf. [21]): A stringset L over Σ is *strictly k-local* iff there is some strictly k-local grammar G over Σ (for some k) such that L is the set of all strings that agree with G:

$$L(G_{SL_k}) := \{w \in \Sigma^* \mid F_k(\rtimes \cdot w \cdot \ltimes) \subseteq G_{SL_k}\}\ .$$

[1]Regular languages are also defined as the class of languages accepted by a finite state automata. However, we will not deal with automata in this paper and therefore only use the definition that establishes equivalence with regular expressions.

The set of all strictly k-local languages forms the language class denoted

$$\mathcal{L}_{SL_k} = \{L(G_{SL_k}) \mid G_{SL_k} \subseteq F_k(\{\rtimes\} \cdot \Sigma^* \cdot \{\ltimes\}\}\,.$$

Definition 2.6 (cf. [21]): A language is said to be *locally threshold k-testable up to t* ($LTT_{k,t}$) iff deciding whether a word belongs to the language rests only with its *multiset* of k-factors with multiplicity up to t. More formally, $\exists\, k, t \in \mathbb{N}$ such that $\forall w, v \in \Sigma^*$ and $\forall l \in F_k(\rtimes \cdot w \cdot \ltimes) \cup F_k(\rtimes \cdot v \cdot \ltimes)$, the following is true:

$$\Big[|w|_l = |v|_l \ \vee \ \min\{|w|_l, |v|_l\} \geq t\Big] \Rightarrow \Big[w \in L \Leftrightarrow v \in L\Big]\,.$$

Definition 2.7 (cf. [22]): A *subsequence* w of a word v is a permutation of the strings formed from the symbols of the word while retaining their relative ordering:

$$w \sqsubseteq v \Leftrightarrow w = \sigma_1\sigma_2\cdots\sigma_n \wedge v \in \Sigma^*\sigma_1\Sigma^*\sigma_2\Sigma^*\cdots\Sigma^*\sigma_n\Sigma^*,$$

where $\sigma_i \in \Sigma$.

The set of all subsequences of length k in a word $w \in \Sigma^*$ is denoted

$$P_k(w) \coloneqq \{v \in \Sigma^k | v \sqsubseteq w\}\,,$$

while the set of all subsequences of length *up to* k is denoted $P_{\leq k}(w) \coloneqq \{v \in \Sigma^{\leq k} | v \sqsubseteq w\}$.

Definition 2.8: A k-piecewise grammar G_{SP_k} is a set of subsequences of length k:

$$G_{SP_k} \subseteq P_k(\Sigma^*)\,.$$

The class of Piecewise Testable languages is introduced in [24]:

Definition 2.9 (cf. [22]): A language L is *Piecewise k-Testable* (PT_k) iff

$$(\forall\, w, v)[P_{\leq k}(w) = P_{\leq k}(v)] \Rightarrow [w \in L \Leftrightarrow v \in L]\,.$$

Definition 2.10 (cf. [19]): A language is *star-free* (SF) if it can be represented by a regular expression that can be written without the Kleene-Star ($*$).

Star-free languages are sometimes called *non-counting*. There are several equivalent definitions of this class, including automata-theoretic and algebraic ones [19].

III. A Hybrid Robot Model

We start with describing the system that we use in our analysis followed by an example of the system that makes the ideas concrete.

A. The Hybrid Robotic System

Definition 3.1 (Hybrid Robotic System): The hybrid robotic agent is a tuple $\mathbf{H} = \{\mathcal{H}, \mathcal{K}, f, \mathcal{P}, \mathcal{AP}, \overleftarrow{\cdot}, \overrightarrow{\cdot}, s, T\}$, where

- $\mathcal{H} = \mathcal{X} \times \mathscr{L}$ is a set of hybrid states and is the cartesian product of a continuous domain $\mathcal{X} \subseteq \mathbb{R}^n$, and a set of boolean variables $\mathscr{L} \subseteq \{\mathbf{0}, \mathbf{1}\}^r$. In this context $\mathbf{1}$ stands for "true" and $\mathbf{0}$ stands for "false;"
- $\mathcal{K}$ is a finite set of discrete locations (modes) with each $\kappa \in \mathcal{K}$ indexing a unique control law for the closed loop continuous dynamics;
- $\mathcal{P} \subseteq \mathbb{R}^m$ is the set of continuous variables parameterizing each control law in $\mathcal{K}$;
- $F : \mathcal{X} \times \mathscr{L} \times \mathcal{P} \times \mathcal{K} \to T\mathcal{X}$ is a finite collection of asymptotically stable vector fields;
- $\mathcal{AP}$ is a set of atomic propositions on $\mathcal{H} \times \mathcal{P}$;
- $\overleftarrow{\cdot}$: $\mathcal{K} \to 2^{\mathcal{AP}}$ is a map that associates a control law indexed by $\kappa \in \mathcal{K}$ with a set of pre-conditions (Pre) which enable the activation and execution of its control action on F, and we write $(h, p) \models \overleftarrow{\kappa}$ when these preconditions are all satisfied;
- $\overrightarrow{\cdot}$: $\mathcal{K} \to 2^{\mathcal{AP}}$ is a map that associates a control law indexed by $\kappa \in \mathcal{K}$ with a set of post-conditions (Post) which are true when the dynamics F driven by controller κ have reached a steady state, and we write $(h, p) \models \overrightarrow{\kappa}$ for the hybrid state $h \in \mathcal{H}$ in the positive limit set of F when the Post of κ is satisfied with controller parameters set to $p \in \mathcal{P}$;
- $s : \mathcal{H} \to 2^{\mathcal{P}}$ is a set valued reset map for the control parameters p that determines the admissible values for control parameterizations as a function of the system's hybrid state;
- T: $\mathcal{H} \times \mathcal{P} \times \mathcal{K} \to \mathcal{H} \times \mathcal{P} \times \mathcal{K}$ is the transition map that regulates the switching between controllers, according to which $(h, p, \kappa) \to (h, p', \kappa')$ iff $(h, p) \models \overrightarrow{\kappa}$, $p' \in s(h)$, and $(h, p') \models \overleftarrow{\kappa'}$.

The discontinuous changes that occur in $\mathbf{H}$ are only because of the parameters p, and these changes are the ones that activate the transitions between different controllers, assuming that the transition is included in the relation T. We say that a hybrid state h *evolves* into another state h' along a vector F and under a controller κ parameterized by p, and we write $h \overset{\kappa[p]}{\hookrightarrow} h'$. This transition occurs continuously along the integral curves of F and is not instantaneous.

Definition 3.2 (Feasible Controller Sequence): A controller sequence s is said to be *feasible* if for each $\kappa^{\{i\}} \in \mathcal{K}$, $\textsc{Post}(\kappa^{\{i\}}) \Rightarrow \textsc{Pre}(\kappa^{\{i+1\}})$, that is, $\forall\, i \in [1, |s| - 1]$, $\forall\, (h, p) \in \mathcal{H} \times \mathcal{P}$, $\exists\, p' \in s(h) : (h, p) \models \overrightarrow{\kappa^{\{i\}}} \wedge (h, p') \models \overleftarrow{\kappa^{\{i+1\}}}$.

The set of all feasible controller strings for $\mathbf{H}$ form its *language*, $L(\mathbf{H})$:

Definition 3.3: The language $L(\mathbf{H})$ of a hybrid system $\mathbf{H}$ is the set of all feasible controller sequences. The finite set of symbols composing the sequences in $L(\mathbf{H})$ form the *control alphabet* $\Sigma_{\mathbf{H}} \subseteq \mathcal{K}$ for $\mathbf{H}$.

B. An example

A mobile manipulator is equipped with feedback controllers that enable it to navigate in a cluttered environment, and to pick and place small objects. The

components of the hybrid system model for this robot can be defined as follows.

The set of continuous states $\mathcal{X}$ contains tuples of the form (q_p, θ, q_m, q_o), where $(q_p, \theta) \in \mathcal{F}_1 \subset \mathsf{SE}(2)$ [2] are the position and orientation of the robot's base within its collision-free planar workspace $\mathcal{F}_1$, $q_m \in \mathcal{W} \subset \mathbb{R}^3$ is (cartesian) position of the manipulator's end-effector relative to its base within its reachable 3D workspace $\mathcal{W}(q_p, \theta)$, and $q_o \in \mathbb{R}^3$ is the position of a object to be manipulated.[3]

The set of logical variables $\mathscr{L}$ contains only one boolean variable denoted $\mathfrak{g}$, which marks whether the end-effector of the robot's arm holds an object ($\mathfrak{g} \Leftrightarrow \mathbf{1}$) or not ($\mathfrak{g} \Leftrightarrow \mathbf{0}$).

The control parameter set $\mathcal{P} \subset \mathbb{R}^6$ consists of tuples (p_p, p_o), where $p_p \in \mathsf{SE}(2)$ is a desired base configuration for the robot, and $p_o \in \mathbb{R}^3$ is a reference position for the manipulated object. The robot is equipped with three controllers, each giving rise to a discrete system mode. There is one controller for the robot base to navigate from $q_i \in \mathcal{F}_1$ to $q_d \in \mathcal{F}_1$, is associated to symbol A. The arm controller for picking an object at p_o and keep it at the end-effector's home position q_m^h, is associated to symbol B, and the arm controller for placing the object held at p_o is associated with symbol C; thus $\mathcal{K} = \{A, B, C\} = \Sigma$. The set of atomic propositions $\mathcal{AP}$ defined for this system consists of:

- $\alpha_1 \Leftrightarrow q_p \in p_p + \mathcal{B}_\epsilon$, which when true it implies that the base is close to its reference position ($\mathcal{B}_\epsilon$ is an appropriately dimensioned ball of radius ϵ);
- $\alpha_2 \Leftrightarrow q_o \in p_o + \mathcal{B}_\epsilon$, which when true implies that the object is in the neighborhood of location p_o;
- $\alpha_3 \Leftrightarrow p_o \in \mathcal{W}(q_p, \theta)$, which when true suggests that with the base at q_p, the reference location p_o lies within the reachable workspace $W(q_p, \theta)$ of the manipulator; and
- $\alpha_4 \Leftrightarrow \mathfrak{g}$, is a binary variable indicating the state of the gripper at the robot's end effector.

	A	B	C
Pre	$\mathbf{1}$	$\{\alpha_2, \alpha_3, \neg\alpha_4\}$	$\{\neg\alpha_2, \alpha_3, \alpha_4\}$
Post	$\{\alpha_1\}$	$\{\neg\alpha_2, \alpha_3, \alpha_4\}$	$\{\alpha_2, \alpha_3, \neg\alpha_4\}$

TABLE I

Pre and Post conditions for the hybrid robotic agent in the example. The sets are interpreted as conjunctions, *i.e.*, $\{\alpha_2, \alpha_3, \neg\alpha_4\} \Leftrightarrow \alpha_2 \wedge \alpha_3 \wedge (\neg\alpha_4)$.

The continuous dynamics of the system can be trivially described by a vector field F in the form:

$$\begin{bmatrix} \dot{q}_p \\ \dot{\theta} \\ \dot{q}_m \end{bmatrix} = \begin{cases} \begin{bmatrix} v\,(\cos\theta, \sin\theta)^\top \\ \omega \\ 0 \end{bmatrix}, & k = A \\ \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix}, & \text{otherwise.} \end{cases}$$

C. Regular Robotic Languages

Constraints between controllers naturally arise due to the conditions imposed by the transition map, expressed in terms of the Pres and Posts of each discrete system location. The nature of the binary relations of the form $\text{Post}(k) \Rightarrow \text{Pre}(k')$ in Definition 3.2, naturally give rise to one of the two following constraints.

a) Adjacency constraints: each 2-factor of the form kk' in a controller sequence s needs to satisfy $\text{Post}(k) \Rightarrow \text{Pre}(k')$, and since the kk' pair is contiguous, we call the relation between the Pre of k' and the Post of k, an *adjacency constraint*. An example of such adjacency constraints is the fact that in the case of the mobile manipulator just discussed, we can never have CC or BB as a 2-factor in any of its feasible controller sequences. Since these constraints on pairs of consecutive symbols can be expressed by a strictly 2-local grammar, adjacency constraints force the language of a hybrid robotic system to exhibit strictly 2-local constraints (SL_2). The class of SL_2 languages is denoted $\mathcal{L}_{SL_2}$.

b) Long distance constraints: Long distance (LD) dependencies between controllers arise in part because of the inability of some controllers to arbitrarily change the hybrid state of the system in order to make (h, p), for some $p \in s(h)$, to satisfy the Pre of another controller. In the example considered, the gripper constraints that prevent CC or BB from appearing are (also) long distance constraints because for example, once $\mathfrak{g}$ is set to $\mathbf{1}$ by B, only C can turn it back to $\mathbf{0}$ to allow another B to be executed, irrespectively of how many As can go in between. This type of long distance dependencies cannot be captured by a language subclass within any of the PT or LTT branches of Figure 1 because the distance between the (discontiguous) occurrences of B (or C) is not upper bounded in general. The class of languages that are specified by this type of *binary* (SL_2-type) long distance dependencies will be denoted $\mathcal{L}_{LD}$.

We introduce a projection operator E_{LD}, which acts on a string w and yields the (sub-)string containing only the symbols that are involved in some long distance constraint according to $\mathcal{T}$. Therefore, E_{LD} "erases" the symbols that are not long distance dependent. Thus if $w = \sigma_1\sigma_2\cdots\sigma_n$, then

$$E_{LD}(w) := \begin{cases} \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_m}, & \sigma_{i_j} \in F_1(\mathcal{T}) \\ \epsilon & \text{otherswise} \end{cases}$$

Thus, whenever a symbol found in any of the subsequences in $\mathcal{T}$ appears, it goes through the E_{LD} filter.

[2] $\mathsf{SE}(2)$ is the special Euclidean group of dimension 3.

[3] Typically, q_o would be part of another system representing the environment. For the purposes of this example, however, this variable is lumped together with the robot's states.

To see why $\mathcal{T}$ cannot be used directly as a 2-piecewise grammar to yield (a 2-piecewise) L_{LD}, it suffices to observe that in order to decide whether a word belongs in a piecewise k-testable language, we need to verify that *all* subsequences in the word can be found in the k-piecewise grammar. In contrast, here we filter the string first through E_{LD}, discarding any "irrelevant" symbols, and then we check for compliance with $\mathcal{T}$.

The grammar that would work as an inclusion checking mechanism for $\mathcal{L}_{LD}$ is an object that needs to defined separately. We will refer to these grammars, defined through the projection operator E_{LD}, as *tier grammars*:

Definition 3.4: A tier grammar G_{LD_k} is a set of permitted factors of length k defined as

$$G_{LD_k} \subseteq F_k(\{\rtimes\} \cdot E_{LD}(\Sigma^*) \cdot \{\ltimes\}).$$

The (tier) language L_{LD} can now be defined in terms of its grammar:

$$L(G_{LD}) = \{w \in \Sigma^* \mid F_2(E_{LD}(w)) \in G_{LD_2}\}.$$

We denote this class of tier languages $\mathcal{L}_{LD}$.

Note that G_{LD_k} contains permitted $k-$factors of words filtered through E_{LD}. The tier language can also be thought of as a language which excludes exactly those words whose filtered 2−factors are forbidden. For a grammar G_{LD_k}, the forbidden 2−factors is the set $\overline{G_{LD_k}} = F_k(\{\rtimes\} \cdot E_{LD}(\Sigma^*) \cdot \{\ltimes\}) - G_{LD_k}$.

Formally, let the *tier based container* of w be

$$C_T(w) = \{u \in \Sigma^* : w \in F_2(\rtimes E_{LD}(u) \ltimes)\}.$$

Thus $C_T(w)$ is all words which contain w as a factor on tier T. It follows that $\tilde{C}_T(w)$ is all words which do not contain w on tier T. Since a language can also be thought of as those words which do not contain any forbidden factors on tier T, it follows (from DeMorgan's laws) that

$$L(G_{LD}) = \cap_{w \in \overline{G_{LD_k}}} \overline{C_T(w)}.$$

Languages with words that satisfy conjunctions of adjacency and long-distance constraints form a class which we call *Regular Robotic Languages* (RRLs).

Definition 3.5: The class of Regular Robotic Languages, $\mathcal{L}_{RRL}$ is the set of finite intersections of languages in $\mathcal{L}_{SL_2}$ and $\mathcal{L}_{LD}$.

In Section IV, we show that $\mathcal{L}_{RRL}$ is a Star-free class, but neither a proper subset of $\mathcal{L}_{LTT}$ nor $\mathcal{L}_{PT}$ (figure 1).

IV. Position of RRLs within the Subregular Hierarchy

Robotic languages being regular, they can generally be equivalently represented in the form of a (class of) finite state (semi)automata. We are interested in finding where exactly they lay within the Subregular hierarchy. We first show that the regular robotic languages are star-free languages. We then show by counter-examples that control sequences feasible in some hybrid robotic system may fall outside both the local, and the piecewise branches of the Subregular hierarchy shown in Figure 1.

Theorem 4.1: Regular robotic languages belong to the star-free class of the Subregular hierarchy.

Proof: Let $L \in \mathcal{L}_{RRL}$. Then, by definition

$$L = L_1 \cap L_2 \cap L_3 \cdots \cap L_n,\ L_i \in \mathcal{L}_{SL_2} \cup \mathcal{L}_{LD},\ i \in \{1, \ldots, n\}.$$

Since the star-free class is closed under intersection [19] and it is known that $\mathcal{L}_{SL_2}$ is a sub-class of star-free class, it only remains to show that $\mathcal{L}_{LD}$ is star-free.

Consider any language L' in $\mathcal{L}_{LD}$. By definition, there is some forbidden tier grammar $\overline{G_{LD}}$ such that $L' = \cap_{w \in \overline{G_{LD_k}}} \overline{C_T(w)}$. Since the star-free languages are closed under complement and finite intersection, it remains to show the $C_T(w)$ is star free for all $w \in \rtimes T * \ltimes$.

First consider any $w = \sigma_1 \ldots \sigma_n$. Since $(\Sigma \setminus \mathcal{T})^* = \overline{\Sigma^* \mathcal{T} \Sigma^*}$ and $\Sigma^* = \bar{\emptyset}$, the set $C_T(w)$ can be written as $\bar{\emptyset}\mathcal{T}\bar{\emptyset}\sigma_1 \cdots \sigma_n \bar{\emptyset}\mathcal{T}\bar{\emptyset}$, which is a regular expression without the Kleene-star. Hence, the languages $\mathcal{L}_{LD}$ are star free. ∎

Theorem 4.2: The robotic patterns do not belong to the local branch of the Subregular hierarchy.

Proof: It is sufficient to prove this theorem with a counter example, using the system of Section III-B. Observe that $\forall k$ and any t, $w = A^k B A^k B A^k C A^k \notin L(\mathbf{H})$ and $v = A^k B A^k C A^k B A^k \in L(\mathbf{H})$. Then, we have $F_k(\rtimes \cdot w \cdot \ltimes) = F_k(\rtimes \cdot v \cdot \ltimes)$ and $\forall l \in F_k(\rtimes \cdot w \cdot \ltimes)$ it is the case that $|w|_l = |v|_l$ Therefore, two words v and w have the same k-factors and the same number of them, but do not both belong to the same language $L(\mathbf{H})$. By definition, $L(\mathbf{H})$ is not LTT, since there is at least one system $\mathbf{H}$ that generates a language outside the class. ∎

Theorem 4.3: Regular robotic languages do not belong to the piecewise branch of the Subregular hierarchy.

Proof: Again, it is proven by a counter-example using the system of Section III-B. Consider and arbitrary k and let $w = A^k(BA^kBA^kCA^kCA^k)^k \notin L(\mathbf{H})$, and $v = A^k(BA^kCA^kBA^kCA^k)^k \in L(\mathbf{H})$. Then observe that $P_{\leq k}(w) = P_{\leq k}(v)$. Hence, even though the two words have exactly the same k-subsequences (for any k), not both the words are in the language. The language generated by $\mathbf{H}$ is not a piecewise testable language. ∎

V. Conclusion and Future Work

In robotic systems which operate by switching between different controllers each ensuring asymptotic stability for the particular task designed for, the possible sequences of controllers activated over time form a language with very special structure. The short and long-distance dependencies between these controllers induce a controller transition logic that can be captured by a new combination of models within the subregular category of formal languages. This paper characterizes

this combination as a finite intersection of strictly 2-local languages, and languages from a new class, defined through a set of rules on different tiers of the language alphabet, as in autosegmental phonology. The languages in this intersection are named *regular robotic languages* in this paper. It is shown that these languages are star free, and they are not properly contained in either the locally testable or the piecewise testable categories within regular languages. Ongoing work targets problems that involve compositions operations on these type of languages, and exploration of the possibility for automatic identification of these languages in the limit from positive data (as in sound pattern learning).

References

[1] Rajeev Alur, Thao Dang, and Franjo Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *ACM Trans. Embed. Comput. Syst.*, 5:152–199, February 2006.

[2] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G.J. Pappas. Symbolic planning and control of robot motion [grand challenges of robotics]. *Robotics Automation Magazine, IEEE*, 14(1):61–70, 2007.

[3] C. Belta, V. Isler, and G.J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *Robotics, IEEE Transactions on*, 21(5):864–874, 2005.

[4] R.W. Brockett. On the computer control of movement. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 534–540 vol.1, April 1988.

[5] Magnus Egerstedt. *Motion Description Languages for Multi-Modal Control in Robotics*, volume 4, pages 75–89. Springer Berlin / Heidelberg, 2003.

[6] G.E. Fainekos, H. Kress-Gazit, and G.J. Pappas. Temporal logic motion planning for mobile robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2020–2025, 2005.

[7] E. Frazzoli, M.A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *Robotics, IEEE Transactions on*, 21(6):1077–1091, 2005.

[8] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44585-4_6.

[9] Antoine Girard and George J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45:566–571, 2009.

[10] Susanne Graf and Hassen Saidi. Construction of abstract state graphs with pvs. In Orna Grumberg, editor, *Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer Berlin / Heidelberg, 1997. 10.1007/3-540-63166-6_10.

[11] Jeffrey Heinz. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[12] Gerard J. Holzmann. The model checker spin. *IEEE Trans. Softw. Eng.*, 23:279–295, May 1997.

[13] John E. Hopcroft and Jeffrey D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.

[14] D. Hristu and S. B. Andersson. Symbolic feedback control for navigation. *IEEE Transactions on Automatic Control*, 51(6):926–937, 2006.

[15] D. Hristu-Varsakelis, M. Egerstedt, and P.S. Krishnaprasad. On the structural complexity of the motion description language mdle. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 4, pages 3360–3365 vol.4, 2003.

[16] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 51(6):949–962, 2006.

[17] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from ltl specifications. In João Hespanha and Ashish Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 333–347. Springer Berlin / Heidelberg, 2006. 10.1007/11730637_26.

[18] V. Manikonda, P.S. Krishnaprasad, and J. Hendler. A motion description language and a hybrid architecture for motion planning with nonholonomic robots. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 2, pages 2021–2028 vol.2, May 1995.

[19] Robert McNaughton and Seymour A. Papert. *Counter-Free Automata (M.I.T. research monograph no. 65)*. The MIT Press, 1971.

[20] J.L. Piovesan, H.G. Tanner, and C.T. Abdallah. Discrete asymptotic abstractions of hybrid systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 917–922, 2006.

[21] James Rogers. Aural pattern recognition experiments and the subregular hierarchy. In *10th Mathematics of Language Conference*, pages 1–7, 2007.

[22] James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Computer Science*, pages 255–265. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-14322-9_19.

[23] Imre Simon. Piecewise testable events. In H. Brakhage, editor, *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer Berlin / Heidelberg, 1975. 10.1007/3-540-07407-4_23.

[24] Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222. 1975.

[25] P. Tabuada. Symbolic sub-systems and symbolic control of linear systems. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 18–23, 2005.

[26] P. Tabuada and G.J. Pappas. Linear time logic control of discrete-time linear systems. *Automatic Control, IEEE Transactions on*, 51(12):1862–1877, 2006.

[27] Paulo Tabuada. Approximate simulation relations and finite abstractions of quantized control systems. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 529–542. Springer-Verlag, 2007.

[28] Herbert G. Tanner, Chetan Rawal, Jie Fu, Jorge L. Piovesan, and Chaouki T. Abdallah. Finite asymptotic abstractions for hybrid systems with stable continuous dynamics. Discrete Event Dynamic Systems (submitted), 2010.

[29] A. Tiwari and G. Khanna. Nonlinear systems: Approximating reach sets. In R. Alur and G. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 600–614. Springer, March 2004.

[30] Moshe Vardi. An automata-theoretic approach to linear temporal logic. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer Berlin / Heidelberg, 1996. 10.1007/3-540-60915-6_6.

[31] Moshe Y. Vardi. *An automata theoretic approach to automatic program verification*. Yorktown Heights, N.Y. : International Business Machines Inc., Thomas J. Watson Research Center, 1986.

[32] Sheng Yu, Qingyu Zhuang, and Kai Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.