Automatic Generation of Balletic Motions

Amy LaViers, Yushan Chen, Calin Belta, and Magnus Egerstedt

***Abstract*— As cyber-physical systems become more prevalent, specifications for these systems must be formulated in a more nuanced manner. This paper presents a particular instantiation of such specification by proposing a framework that endows robotic motions with a sense of aesthetic style. Drawing inspiration from classical ballet, poses are cast as discrete states and movements as the transitions between these states. Thus, a given movement style is encoded in the availability of transitions at each state, and the dynamics of a complex physical trajectory are abstracted as a system which moves between these states. Using Linear Temporal Logic (LTL), we are able to further constrain the set of possible sequences through the transition system and thus prevent it from evolving through a sequence of states that is physically impossible or aesthetically undesirable. Our overarching objective is to facilitate subtle degrees of control over systems as such subtleties are required, more and more, to interact in a social and aesthetically driven world.**

I. Introduction

As robots' capabilities and their interactions with humans increase, we desire to endow robotic systems with more sophisticated (typically human-like) behaviors. Such behaviors do not follow predictable, continuous trajectories as humans constantly make discrete decisions and may abruptly change behavioral modes. Thus, this problem requires discrete, non-physical control methods which can interface with the continuous trajectories of robotic systems and at the same time provide discrete decision-making power. Designing such cyber-physical systems requires exploring the temporal structure of human movement by producing specifications which restrict movement patterns such that they match those found in specific human behaviors. In this paper, we use temporal logic statements to begin to enumerate fundamental rules and secondary, aesthetic principles which govern human behavior and can provide a starting point for reasoning about more subtle control design objectives, such as "style," when designing controllers for cyber-physical systems.

Recently, there has been an increasing interest in developing computational frameworks allowing for rich specification languages in robotics. In particular, temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) have been suggested as motion specification languages [20], [16], [6], [10], [3]. The use of such logics allows for a large spectrum of specifications which include: choice of a goal ("go to either A or B"); convergence to a region ("reach A eventually and stay there for all future times"); visiting targets sequentially ("reach A, and then B, and then C"); surveillance ("reach A and then B infinitely often"); the satisfaction of more complicated temporal and logic conditions about the reachability of regions of interest ("Never go to A. Don't go to B unless C or D were visited"). Such robot motion planning and control objectives are achieved based on algorithms inspired from model checking [2] and temporal logic games [15].

Here, we use LTL to express robotic tasks that include an aesthetic component, e.g., "go to goal with grace" or, more specifically, "move in the quick, staccato style of allégro ballet." These tasks have an objective component, such as "take ten steps," and also incorporate subjective qualities like intention and aesthetics, such as "make ten movements that give the impression of being happy." Thus, in this paper, through a particular approach to modeling and formal synthesis, we begin to quantify subjective qualities, which are a significant missing link in human-like robot interaction, by scripting them in the established language of LTL.

This work is a continuation of [12], where classical ballet was first introduced as a potential generator for behavior that is strictly human. Their extracted movement rules have little to do with concrete robotic tasks, to which LTL has been applied previously as described above, and are instead derived from principles of symmetry, aesthetics, and emotive content: concepts which have previously had no equivalent principles in robotic behavior. By scripting these concepts in LTL, we provide a convention that is easily interpreted by appropriate technical communities. In addition, we further develop their set of tools for dancers and dance scholars that quantify aspects of choreography, offering unbiased comparison between movement genres and thus facilitating new debate in the artistic community. In this sense, our contribution here is in the same vein as proposed dance notation systems and other initial attempts at quantification of aesthetic human behavior - most notably those of Rudolf Laban in the early 20^{th} century [8], [13].

Subsequently, our modeling choices also stem from the standard training routine employed by ballet dancers. This routine, known as the barre, warms and trains muscle groups key in executing the full gamut of ballet movements. Thus, the patterns contained in these simple movements (which typically focus on only one half of the body) encode what is essentially a rule set for ballet movement. We model this set of rules for one leg as a finite transition system and use a standard synchronous product to produce a superset of all possible two-leg motions. In our proposed framework,

Amy LaViers and Magnus Egerstedt are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, E-mail: alaviers@gatech.edu, magnus@gatech.edu

Yushan Chen is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA, E-mail: yushanc@bu.edu

Calin Belta is with the Department of Mechanical Engineering, Division of Systems Engineering, and the Bioinformatics Graduate Program, Boston University, MA 02215, USA, E-mail: cbelta@bu.edu

a specification is given as an LTL formula over a set of state observations. These fall into two categories: "hard" specifications, which incorporate all the physical constraints and aesthetic requirements that the robotic system must satisfy and "soft" specifications, which capture certain additional aesthetic requirements that the robotic system is encouraged to achieve. To enforce the overall specification, we use an adaptation of the LTL receding horizon controller from [3]. The remainder of the paper is organized as follows. In Sec. II, we give some definitions that are necessary throughout the paper. We formulate the problem and outline the approach in Sec. III and give the detailed solution in Sec. IV. In Sec. V, we present a case study with concluding final remarks in Sec. VI.

II. Preliminaries

In this section, we briefly review some aspects of Linear Temporal Logic (LTL). Informally, LTL formulas consist of Boolean and temporal combinations of *atomic propositions* α from a set Π. In our context, the propositions capture properties such as "the leg of the robot is off the ground" or "the robot is squatting." Given a system model, LTL allows us to express the time evolution of the state of the system in terms of these atomic propositions. For example, we can know for each time step whether "the robot is squatting."

To model our system, we consider a finite model called weighted transition system:

Definition 1. (Weighted Transition System) *A weighted finite transition system is a tuple $T = (Q, q_0, \rightarrow, \omega, \Pi, h)$, where (i) Q is the finite set of states; (ii) $q_0 \in Q$ is the initial state; (iii) $\rightarrow \subseteq Q \times Q$ is the transition relation; (iv) ω is a weight function that assigns positive values to all the transitions (i.e., to all pairs (q_1, q_2) where $(q_1, q_2) \in \rightarrow$); (v) Π is a finite set of atomic propositions; and (vi) $h : Q \mapsto 2^{\Pi}$ is an output map.*

We assume that the transition system is non-blocking, implying that there is a transition from each state. The labeling function defines for each state $q \in Q$, the set $h(q)$ of all atomic propositions valid in q. For example, the proposition "the left leg of the robot is off the ground" will be valid for all states $q \in Q$ for which the corresponding system configuration is one where left leg is off the ground.

For our transition system we define a run r_T to be an infinite sequence of states $q_0 q_1 q_2 \ldots$ such that $q_0 \in Q_0$, $q_i \in Q$, for all $i \geq 0$, and $(q_i, q_{i+1}) \in \rightarrow$, for all $i \geq 0$. A run r_T defines a word $h(q_0)h(q_1)h(q_2)\ldots$ consisting of sets of atomic propositions valid at each state. This model captures how the behavior of our robot is changing over time in terms of propositions of behavioral relevance (that we will define) rather than in terms of absolute (and behaviorally irrelevant) spatial parameters or even configuration variables.

Definition 2. (Formula of LTL) *An LTL formula ϕ over the atomic propositions Π is defined inductively as follows:*

$$\phi ::= \top \mid \alpha \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{X}\phi \mid \phi \,\mathcal{U}\, \phi$$

*where $\top$ is a predicate true in each state of a system, $\alpha \in \Pi$ is an atomic proposition, $\neg$ (negation) and $\wedge$ (disjunction) are standard Boolean connectives, and **X** and $\mathcal{U}$ are temporal operators.*

LTL formulas are interpreted over infinite words, such as those generated by the transition system T from Def. 1. A run of T satisfies an LTL formula ϕ over Π if ϕ is true at the first state of the run; $\mathbf{X}\phi$ states that at the next state, an LTL formula ϕ is true (i.e., $\mathbf{X}\alpha$ means $\alpha \in h(q_1)$); $\alpha_1 \,\mathcal{U}\alpha_2$ states that there is a future moment when proposition α_2 is true, and proposition α_1 is true at least until α_2 is true. Using the Boolean connectives $\wedge$ and $\neg$, the full power of propositional logic is obtained. Other Boolean connectives such as disjunction $\vee$ and implication $\rightarrow$ can be derived as $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\phi_1 \rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$. From the above temporal operators we can construct two other useful operators Eventually (i.e., future), **F**, defined as $\mathbf{F}\phi := \top \,\mathcal{U}\phi$, and Always (i.e., globally), **G**, defined as $\mathbf{G}\phi := \neg\mathbf{F}\neg\phi$. The formula $\mathbf{F}\alpha$ states that α holds at some future time instance, and $\mathbf{G}\alpha$ states that proposition α holds at all states of the run.

An LTL formula can be represented in an automata-theoretic setting as a *Büchi Automaton*, defined as follows:

Definition 3. (Büchi Automaton) *A Büchi Automaton is a tuple $B = (S, S_0, \Sigma, \delta, F)$, where (i) S is a finite set of states; (ii) $S_0 \subseteq S$ is a set of initial states; (iii) Σ is the input alphabet; (iv) $\delta : S \times \Sigma \rightarrow 2^S$ is a transition relation; (v) $F \subseteq S$ is a set of accepting (final) states.*

The semantics of a Büchi Automaton are defined over infinite input words. Let $\omega = \omega_0\omega_1\omega_2\ldots$ be an infinite input word of automaton B, where $\omega_i \in \Sigma$ for each $i \geq 0$. If we define the input alphabet to be the power set of all atomic propositions ($\Sigma = 2^{\Pi}$), then the semantics are defined over the words that can be produced by a run of the transition system. Thus, the input $\omega = h(q_0)h(q_1)h(q_2)\ldots$ could be a word produced by a run $q_0q_1q_2\ldots$ of the transition system T.

A *run* of the Büchi Automaton over an input word $\omega = \omega_0\omega_1\omega_2\ldots$ is a sequence $r_B = s_0s_1s_2\ldots$, such that $s_0 \in S_0$, and $(s_i, \omega_i, s_{i+1}) \in \delta$, for all $i \geq 0$.

Definition 4. (Büchi Automaton Acceptance) *A word ω is accepted by the Büchi Automaton B if and only if there exists a run r_B over ω so that $\inf(r_B) \cap F \neq \emptyset$, where $\inf(r_B)$ denotes the set of states appearing infinitely often in run r_B.*

The Büchi Automaton allows us to determine whether or not the word produced by a run of the transition system T satisfies an LTL formula. More precisely, for any LTL formula ϕ over a set of atomic propositions Π, there exists a B_ϕ with input alphabet 2^{Π} accepting all and only the infinite words satisfying formula ϕ ([18]). We refer readers to [5], [17] and references therein for efficient algorithms and freely downloadable implementations to translate an LTL formula over Π to a corresponding Büchi Automaton B.

Definition 5. (Weighted Product Automaton) *A weighted product automaton $\mathcal{A} = T \times B$ between a transition system T and a Büchi automaton B is a tuple $(S_\mathcal{A}, S_{\mathcal{A}0}, \delta_\mathcal{A}, \omega_\mathcal{A}, F_\mathcal{A})$, where (i) $S_\mathcal{A} = Q \times S$ is the set of states; (ii) $S_{\mathcal{A}0} = \{q_0\} \times S_0$ is the set of initial states; (iii) $\delta_\mathcal{A} : S_\mathcal{A} \mapsto 2^{S_\mathcal{A}}$ is the transition function defined as $(q_j, s_l) \in \delta_\mathcal{A}((q_i, s_k))$ if and only if $(q_i, q_j) \in \rightarrow$ and $s_l \in \delta(s_k, h(q_i))$; (iv) $\omega_\mathcal{A} : S_\mathcal{A} \times S_\mathcal{A} \mapsto R^+$ is a positive-valued weight function inherited from T, defined as $\omega_\mathcal{A}((q_i, s_k), (q_j, s_l)) = \omega(q_i, q_j)$, where $(q_j, s_l) \in \delta_\mathcal{A}((q_i, s_k))$; and (v) $F_\mathcal{A} = Q \times F$ is the set of accepting states.*

Similar to the acceptance condition of the Büchi Automaton, a run $r_\mathcal{A}$ of $\mathcal{A}$ is accepted if and only if $r_\mathcal{A}$ intersects $F_\mathcal{A}$ infinitely many times. We define the projection of a run $r_\mathcal{A}$ onto the transition system T as: $\gamma_T(r_\mathcal{A}) = r = q_0 q_1 ...$, if $r_\mathcal{A} = (q_0, s_0)(q_1, s_1)....$ Note that if ϕ is an LTL formula and B is its corresponding Büchi Automaton, then the projection of an accepted run $r_\mathcal{A}$ on T is a run of T that generates a word satisfying ϕ ([4]).

Given a transition system T and an LTL formula ϕ over Π, checking whether the words of T[1] satisfy ϕ is called LTL model checking. An off-the-shelf model checker, such as `NuSMV` [1] and `SPIN` [7], proceeds by negating the formula, computing the corresponding Büchi automaton, and checking the emptiness of the language accepted by the product automaton. This procedure was the starting point for previous work [3], [11], where control strategies for a deterministic transition system with inputs from a specification given as an LTL formula were developed. A game theoretic approach was necessary to solve the same problem for a nondeterministic transition system [9].

III. Problem Formulation and Approach

In this paper, we assume that the motion of our system, a two-legged biped that moves in the style of classical ballet, is modeled as a transition system (as defined in Def. 1). Specific modeling choices for the motion of one leg have been defined in previous work [12]. The choices derive from fundamental principles of ballet and the canonical warm-up routine (the barre) which focuses on training the legs of a ballerina. The bulk of this model is described in Fig. 1 and the following table:

Movement	Transition Label
plié	`plie`
relevé	`rele`
battement tendu	`tend`
degajé	`dega`
coupé	`coup`
frappé	`frap`
grand battement	`gran`
possé	`poss`
battement	`batt`
dévelopé	`deve`

[1]In LTL model checking, the transition system (also called Kripke structure) is assumed to be non-blocking and it is not weighted.

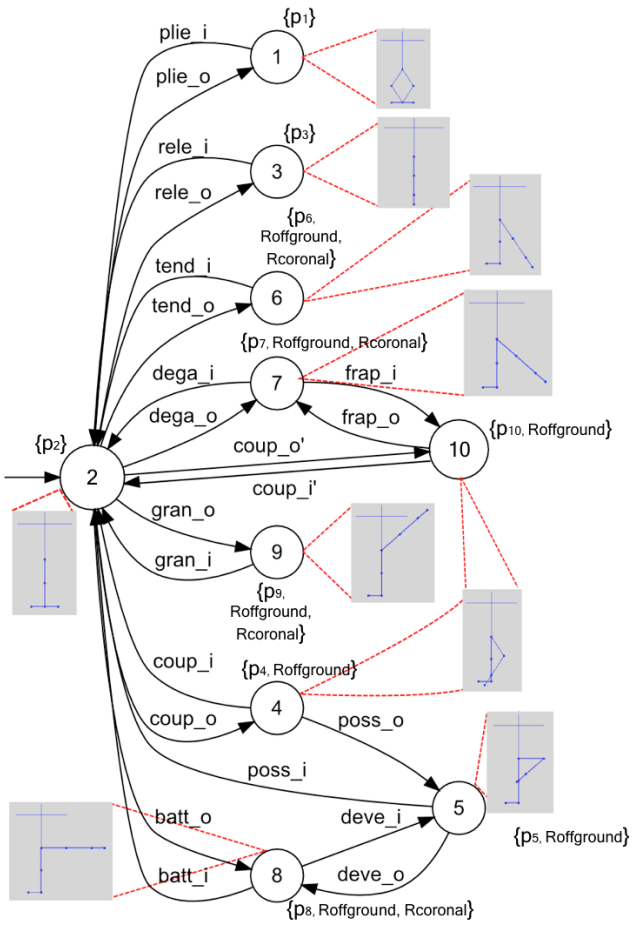

Fig. 1. A transition system which models the working leg (right leg) of a dancer during a ballet barre exercise. The atomic propositions $\{p_1, \ldots, p_{10}\}$ correspond to poses defined by three joint angles: hip, knee, and ankle. Images of the poses and corresponding states satisfying the bracketed propositions are shown. Note that, for clarity, we neglected to draw the self-loops that are needed at each state. Here, we illustrate that in our model state 10 is physically the same as state 4. We differentiate these two related states based on whether the motion of the leg is to remain low (below the hips) or high (at or above hip level) before returning to a neutral state and beginning the next movement. In ballet this pose, formally called *sur le cou-de-pied*, is distinguished with a "wrapped" or "fully-pointed" foot [19], a difference which is indistinguishable in our framework.

The set of states is a selected set of two-dimensional poses (illustrated in Fig. 1) whose shapes typify ballet movements. The set of transition labels corresponds to specific ballet steps from which the model was derived. A run in the transition system starting at the initial state defines a corresponding series of ballet poses; interpolation between these poses represents ballet movement. The weight function from Def. 1 can model the maximum time needed to change between two adjacent poses or the corresponding control effort.

Formally, the transition system in Fig. 1 that models the motion of the right leg of a ballerina during a simple ballet barre is defined as

$$T_R = (Q_R, q_{0_R}, \rightarrow_R, \omega_R, \Pi_R, h_R), \tag{1}$$

where

(i) $Q_R = \{q_1^R, \ldots, q_{10}^R\}$ is the finite set of states;

(ii) $q_{0_R} = q_2^R$ is the initial state representing the initial pose;
(iii) $\rightarrow_R \subseteq Q_R \times Q_R$ is the reflexive transition relation;
(iv) ω_R is a trivial weight function that assigns 1 to all transitions;
(v) $\Pi_R = \{p_1, \ldots, p_{10}\} \cup \{\text{Roffground}, \text{Rcoronal}\}$ is a finite set of atomic propositions;
(vi) $h_R : Q_R \mapsto 2^{\Pi_R}$ is an output map, where state q_i^R satisfies p_i, for all $1 \leq i \leq 10$, states q_j^R satisfies Roffground, for all $4 \leq j \leq 10$ and state q_k^R satisfies Rcoronal, for all $6 \leq k \leq 9$.

The atomic propositions $\{p_1, \ldots, p_{10}\}$ represent different poses corresponding to three joint angles: hip, knee, and ankle. Roffground represents "the right leg of the robot is off ground" and Rcoronal represents "coronal extensions away from the body." In order to allow for both synchronous and asynchronous transitions after we take the Cartesian composition of two such systems, the self-loops are needed at all states; thus we use the reflexive transition relation in Item (iii). In this paper, we do not model the time or effort necessary to transit among poses; therefore, we assign weight 1 to each transition.

Correspondingly, we define the transition system that models the motions of the left leg transition system to be:

$$T_L = (Q_L, q_{0_L}, \rightarrow_L, \omega_L, \Pi_L, h_L), \tag{2}$$

where each component is defined as in Eq. 1; that is Items (i) - (vi) are identical for the left leg's transition system with "R" replaced with "L" as necessary.

The configurations that can be obtained when we consider a robot with two legs are captured by a slightly modified Cartesian composition (product) of the left and right transition systems. We define this product as follows:

Definition 6. (Product of Transition Systems) *The product of two transition systems T_L and T_R, denoted as $T_L \otimes T_R$, is defined as $T_P = (Q_P, q_{0_P}, \rightarrow_P, \omega_P, \Pi_P, h_P)$, where (i) $Q_P \subseteq Q_L \times Q_R$; (ii) $q_{0_P} = (q_{0_L}, q_{0_R})$; (iii) $\rightarrow_P \subseteq Q_P \times Q_P$ is defined by $(q, q') \in \rightarrow_P$ if and only if $q \neq q'$, $(q_L, q'_L) \in \rightarrow_L$ and $(q_R, q'_R) \in \rightarrow_R$, where $q = (q_L, q_R)$ and $q' = (q'_L, q'_R)$; (iv) ω is a weight function such that $\omega(q, q') = \omega(q_L, q'_L) + \omega(q_R, q'_R)$ if and only if $(q, q') \in \rightarrow_P$, where $q = (q_L, q_R)$ and $q' = (q'_L, q'_R)$; (v) $\Pi_P = \Pi_L \cup \Pi_R \cup \{\text{Spose}\}$; and (vi) $h_P : Q_P \mapsto 2^{\Pi_P}$ is defined by* (1) Spose $\in h_P(q_L, q_R)$ *if and only if $q_L = q_R$,* (2) *for all $\pi \in \Pi_L \cup \Pi_R$, $\pi \in h_P(q_L, q_R)$ if and only if $\pi \in h_L(q_L)$ or $\pi \in h_R(q_R)$.*

In other words, T_p captures all possible transitions that can appear in T_L and T_R. The self transitions (when the system holds the pose) are excluded in the configurations. The proposition "Spose" is satisfied by all states at which both legs have the same configuration or pose.

Remark 1. *Rather that defining formal inputs, we assume that one can simply choose available transitions at a state. This corresponds to a deterministic transition system with inputs in which the choice of an available input at a state uniquely determines the transition to the next state. The particular control strategy that we develop in this paper is based on this property.*

Since the one leg transition system does not contain all the information about the physical capabilities of the robot (T_R does not tell us anything about forces required for jumping but instead accepts the correct sequence of leg positions during the jump), it is entirely possible that the product T_P accepts runs that are not physically possible to execute and that are not within the range of our goal aesthetic. Hence, we formulate LTL specifications that enable our system to make discrete decisions about viable trajectories (that is, accept or reject various runs through the transition system) - where viable is defined in terms of both physical and aesthetic constraints.

Specifically, we want to prevent the system from executing any physically infeasible runs. In addition, we are interested in applying aesthetic conditions to the accepted runs of T_P in order to make them adhere to our chosen dance style. For example, we may disallow a list of two-legged body poses which are perhaps considered ugly as judged by the metric of ballet; often, these are asymmetrical poses or poses which cannot be seen from the audience's distant perspective. Finally, even given a system which only produces movements in the style of ballet, we may further restrict our output so as to only produce a specific type of movement phrase within the genre that, for example, is typified by more frequent use of certain movements.

Thus, to define our problem, we assume that our robot is required to satisfy the physical constraints of a bipedal geometry and the aesthetic requirements in ballet. We consider two types of specifications to express the restrictions: (1) *hard* specifications and (2) *soft* specifications. A hard specification incorporates all the physical constraints and aesthetic requirements that the robotic system must satisfy while a soft specification captures certain additional aesthetic requirements that the robotic system is encouraged to achieve.

We use LTL formulas to represent the hard specification. The physical restrictions of the robot and the aesthetic requirements of ballet can be easily translated to LTL formulas as in the following two examples:

1) "visit the pose where both legs are in pose p_2 infinitely many times" can be converted to

$$\mathbf{G}\ \mathbf{F}\ (\text{Spose} \wedge p_2)$$

where Spose implies both begs are in the same pose and Spose $\wedge\ p_2$ implies both legs are in pose p_2.

2) "always squat and then stand up before having two legs off ground" can be converted to

$$\mathbf{G}\ (\neg((p_1 \wedge \text{Spose}) \wedge \mathbf{X}\ (p_2 \wedge \text{Spose})) \quad \rightarrow \\ \mathbf{X}\ \mathbf{X}\ (\neg(\text{Loffground} \wedge \text{Roffground})))$$

where p_1 is the squat pose, p_2 is the standing up pose, **G** implies "always."

The soft specification is introduced to specify aesthetic requirements that the robot is encouraged (instead of forced) to achieve. We define the soft specification, denoted by $\mathcal{S}$, as a collection of elements from the power set of atomic propositions:

$$\mathcal{S} = \{prop_i\}, \text{ where } prop_i \in 2^{\Pi_P}. \tag{3}$$

The collection $\mathcal{S}$ models our desire for the robot to prefer satisfying certain ballet-inspired propositions more than others. With this preference, we will produce a specific type of movement phrase known as *allégro*, which is typified by movements that produce quick and upbeat dynamic quality. In allégro, the ballerina robot is encouraged to perform entrechat (a little jumpy move where the dancer beats his or her feet back and forth) more frequently than other movements. In our system, entrechat would correspond to a word of T_P $\binom{p_1}{\text{Spose}}\binom{p_2}{\text{Spose}}\binom{p_3}{\text{Spose}}\binom{p_2}{\text{Spose}}\binom{p_1}{\text{Spose}}$, which is generated by a run of T_P $(q_1,q_1)(q_2,q_2)(q_3,q_3)(q_2,q_2)(q_1,q_1)$. Consequently, we want to encourage the robot to perform pose $\binom{p_3}{\text{Spose}}$ (i.e., state (p_3,p_3)) more often for a higher appearance frequency of entrechat. In this example, the soft specification is $\mathcal{S} = \{\binom{p_3}{\text{Spose}}\}$.

Formally, we have:

Problem 1. *Given a transition system T_P* (Def. 6)*, a hard specification given as an* LTL *formula ϕ* (Def. 2) *and a soft specification $\mathcal{S}$* (Eqn. 3)*, design a dance or movement phrase for the robot (an infinite run of T_P) such that the produced dance* (1) *satisfies ϕ and* (2) *has a higher frequency of the propositions in $\mathcal{S}$.*

We make the natural assumption that, at each time instant, the ballerina robot can "foresee" only a few steps ahead (imagine the system is performing a free solo without any choreography). To solve Prob. 1, we introduce rewards (positive real numbers) to the states included in $\mathcal{S}$. A state with a reward is called a *target*. A collection of all the potential targets is defined as

$$\mathbb{T} \quad = \quad \begin{array}{ll} \{\mathrm{q}_P \mid h_P(q_P) \in \mathcal{S}\} & \text{if } \mathcal{S} \neq \emptyset, \\ \mathrm{Q}_P & \text{if } \mathcal{S} = \emptyset. \end{array} \tag{4}$$

In other words, when $\mathcal{S} \neq \emptyset$, only states satisfying the propositions in $\mathcal{S}$ can become potential targets. We assume that the rewards are associated to these selected states in a time varying fashion and the appearance and disappearance of rewards and their values are randomized. We denote the non-negative awards associated with the targets by $R(q_P),\ q_P \in Q_P$. Upon visiting a target, the robot collects the corresponding reward and subsequently the state is no longer a target.

We aim to maximize the collected rewards; hence, the robot is encouraged to visit the targets more frequently than the other states without rewards. Specifically, we reduce Prob. 1 to the following problem where we satisfy the LTL formula and maximize the collected reward:

Problem 2. *Given a transition system T_P* (Def. 6)*, an* LTL *formula ϕ* (Def. 2)*, and the potential target set $\mathbb{T}$* (Eqn. 4)*, generate a run of T_P such that the produced run* 1) *satisfies ϕ and* 2) *maximizes the collected rewards from the potential target set $\mathbb{T}$.*

Remark 2. *It is important to note that, since we are interested in infinite robot trajectories, it does not make sense to look for a run that maximizes the total collected rewards, since this can be infinite. Rather, we aim to design a (local, real-time) receding-horizon type controller and find a run that maximizes rewards collected based on local information obtained at current state q.*

To achieve this, we use an approach similar to that in [3], where a receding-horizon type controller was designed. More specifically, we use a measure of progress towards satisfying the formula. If the controller is designed to always increase this progress as defined by our measure, then we can show that the LTL formula is satisfied. The proposed approach relies on 1) the construction of the product automaton (Def. 5) between the transition system and the Büchi Automaton corresponding to ϕ and 2) an assignment of a suitable cost to each state of the resulting product automaton. This cost assignment is computed off-line once and then is used on-line with the real-time controller. The cost assignment is designed so that when used in conjunction with our proposed control strategy, an accepting state on the product automaton will be reached in a finite number of transitions. If this is repeated infinitely many times, the acceptance condition of the product automaton is enforced.

IV. Problem Solution

In this section, we present a state feedback controller providing a solution to Prob. 2. The design of the controller can be divided into two main parts. The first part (Sec. IV-A) is to define a measure of progress towards satisfying formula ϕ. To achieve this, we first construct a product automaton $\mathcal{A}$ (Def. 5) between the transition system T_P (Def. 6) and the Büchi automaton (Def. 3) corresponding to ϕ. Then we assign a cost to each state of $\mathcal{A}$.

The second part (Sec. IV-B) includes two receding horizon controllers: (1) $P_{\mathcal{A}}^{init}(S_{\mathcal{A}0})$, which takes as input the set of initial states of $\mathcal{A}$, and (2) $P_{\mathcal{A}}(p)$ which takes as input a state $p = (q,s)$ of $\mathcal{A}$. These two controllers use the cost assignment computed as described in the first part, maximize the rewards collected from the potential target sets $\mathbb{T}$, and produce finite runs on the product automaton $\mathcal{A}$. Our state-feedback receding horizon controllers generate the desired run of T_P by projecting finite runs produced by these two controllers from $\mathcal{A}$ to T_P (see the text following Def. 5 for the definition of projection of runs of the product automaton $\mathcal{A}$ on the transition system).

By combining the two parts, we present the control algorithm solving Prob. 2 in Alg. 1. As shown in [3], by concatenating the runs produced by two receding horizon controllers, we obtain an accepted run on $\mathcal{A}$, which can be further used to generate a run of T_P satisfying ϕ if at least one such run exists.

A. Cost Assignment

Given the transition system T_P and a Büchi automaton B created from an LTL formula ϕ, we construct the product automaton $\mathcal{A} = (S_\mathcal{A}, S_{\mathcal{A}0}, \delta_\mathcal{A}, \omega_\mathcal{A}, F_\mathcal{A})$ as defined in Def. 5. Our cost assignment scheme applies a cost as a measure of progress to each state of $\mathcal{A}$.

We denote by $d(p_i, p_j)$ the shortest distance (based on the weights of the transitions) from a state $p_i \in S_\mathcal{A}$ to $p_j \in S_\mathcal{A}$. Given p_i, $d(p_i, p_j)$ can be efficiently computed for all $p_j \in S_\mathcal{A}$ by many shortest path algorithms [14] such as Dijkstra's algorithm. We say that a set $A \subseteq S_\mathcal{A}$ is *self-reachable* if and only if all states in A can reach a state in A. We define $F_{\mathcal{A}^\star}$ to be the largest self-reachable subset of $F_\mathcal{A}$. The cost function for all states $p_i \in S_\mathcal{A}$ is defined as:

$$J(p_i) = \begin{cases} \min_{p_j \in F_{\mathcal{A}^\star}} d(p_i, p_j), & \text{if } p_i \notin F_{\mathcal{A}^\star} \\ 0, & \text{if } p_i \in F_{\mathcal{A}^\star} \end{cases} \quad (5)$$

Clearly, $J(p_i) = 0$ if and only if $p_i \in F_{\mathcal{A}^\star}$. This cost encodes the minimum distance from states in $\mathcal{A}$ to the set $F_{\mathcal{A}^\star}$.

We obtain the set $F_{\mathcal{A}^\star}$ by starting with the whole set $F_\mathcal{A}$ and pruning out every state that can not reach another state in the set until all states in the set satisfy the definition of a self-reachable set. Then, we compute the cost measure $J(p_i)$ for all states $p_i \in S_\mathcal{A}$. These are assigned off-line; that is, the costs J associated with each state in $\mathcal{A}$ are saved in a look-up table to be used later in conjunction with the real time controllers (see Sec. IV-B).

B. Receding Horizon Controllers

Now we introduce the two receding horizon controllers, denoted as $P^{init}_\mathcal{A}(S_{\mathcal{A}0})$ and $P_\mathcal{A}(p)$, that produce finite runs $r_{f\mathcal{A}}$ on $\mathcal{A}$, assuming that the cost J defined in (Eq. 5) is accessible from a look-up table. Our desired run is obtained by projecting $r_{f\mathcal{A}}$ onto T_P.

Receding horizon controllers $P^{init}_\mathcal{A}(S_{\mathcal{A}0})$ and $P_\mathcal{A}(p)$ maximize the rewards collected over finite runs with path length no more than a pre-defined "planning horizon length" L (i.e., we make the assumption from Sec. III that the robot can only "foresee" a few steps ahead).

First, we describe the construction of $P_\mathcal{A}(p)$, which is a combination of a "pre controller" $P^{pre}_\mathcal{A}(p)$ and a "post controller" $P^{post}_\mathcal{A}(p)$. Controller $P^{pre}_\mathcal{A}(p)$ drives the system from any state $p \notin F_{\mathcal{A}^\star}$ to a state p' where $J(p') < J(p)$ while maximizing the rewards collected locally (see Remark. 3). The repeated executions of $P^{pre}_\mathcal{A}(p)$ will drive the system from a state $p \notin F_{\mathcal{A}^\star}$ to a state in $F_{\mathcal{A}^\star}$ in finite number of transitions. Controller $P^{post}_\mathcal{A}(p)$ drives the system from a state in $F_{\mathcal{A}^\star}$ to a state $p_i \notin F_{\mathcal{A}^\star}$. $P_\mathcal{A}(p)$ switches between $P^{pre}_\mathcal{A}(p)$ and $P^{post}_\mathcal{A}(p)$ depending on the current state (i.e., choose $P^{pre}_\mathcal{A}(p)$ if $p \notin F_{\mathcal{A}^\star}$ and $P^{post}_\mathcal{A}(p)$ otherwise). This process repeats infinitely many times so that the set $F_{\mathcal{A}^\star}$ is visited infinitely many times, and thus the acceptance condition (Def. 4) of the product automaton is enforced.

Remark 3. *Maximizing the rewards collected locally involves exploring finite runs in a sub-graph of the graph corresponding to $\mathcal{A}$ and searching for the run that maximize the rewards. The size of the sub-graph is decided by the "planning horizon length" L. The states of this sub-graph are $\{p_i \in S_\mathcal{A} \mid d(p, p_i) \leq L\}$ since we do not consider finite runs with the last state further than L away from the current state p. This fact demonstrates the local decision making process of our ballerina robot, as the control decision $r_{f\mathcal{A}}$ is always made on this sub-graph (depending on L and p). As a result, to compute $P^{pre}_\mathcal{A}(p)$, only a local portion of the product automaton $\mathcal{A}$ around p needs to be constructed and explored. This same remark can be made later for $P^{post}_\mathcal{A}(p)$ and $P^{init}_\mathcal{A}(S_{\mathcal{A}0})$.*

To construct the controller $P^{init}_\mathcal{A}(S_{\mathcal{A}0})$, we take the input of the set $S_{\mathcal{A}0}$ because the Büchi automaton B can have a set of initial states S_0. The controller $P^{init}_\mathcal{A}(S_{\mathcal{A}0})$ is similar to $P^{post}_\mathcal{A}(p)$ but it considers a set of (initial) states instead of a single state.

Algorithm 1 Control algorithm for T_P, given an LTL formula ϕ, a potential target set $\mathbb{T}$

INPUT: ϕ, T_P, $\mathbb{T}$ and terminating time t_{end} (instead of generating infinite run of T_P, we set a terminating time to end the algorithm)

OUTPUT: A run r_{dance} of T_P

Executed Off-line:

Construct a Büchi automaton $B = (S, S_0, \Sigma, \delta, F)$ corresponding to ϕ
Construct the product automaton $\mathcal{A} = T_P \times B = (S_\mathcal{A}, S_{\mathcal{A}0}, \delta_\mathcal{A}, \omega_\mathcal{A}, F_\mathcal{A})$
Compute $d(p_i, p_j)$ for all $p_i \in S_\mathcal{A}$ and $p_j \in F_\mathcal{A}$
Set $F_{\mathcal{A}^\star} = F_\mathcal{A}$
WHILE there exist $q \in F_{\mathcal{A}^\star}$ such that

$$\min_{p_j \in F_{\mathcal{A}^\star}, p_i \in \delta_\mathcal{A}(q)} d(p_i, p_j) = \infty$$

do Remove q from $F_{\mathcal{A}^\star}$.
Obtain $J(p_i)$ using Def. 5 for all $p_i \in S_\mathcal{A}$

Executed On-line:

if $\exists\ p_0 \in S_{\mathcal{A}0}$ such that $J(p_0) \neq \infty$ **then**
  Set current time $t_{cur} = 0$
  Update rewards at states in $\mathbb{T}$
  Obtain $P^{init}_\mathcal{A}(S_{\mathcal{A}0}) = r_{f\mathcal{A}} = p_1 ... p_n$; Obtain $\gamma_T(P^{init}_\mathcal{A}(S_{\mathcal{A}0})) = r_f = q_1 ... q_n$; Add r_f to the end of r_{dance}; Set $t_{cur} = t_{cur} + d(q, q_n)$, $q = q_n$ and $p = p_n$
  repeat
    Update rewards at states in $\mathbb{T}$
    Let $P_\mathcal{A}(p) = P^{pre}_\mathcal{A}(p)$ if $J(p) > 0$ and $P_\mathcal{A}(p) = P^{post}_\mathcal{A}(p)$ if $J(p) = 0$; Obtain $P_\mathcal{A}(p) = r_{f\mathcal{A}} = p_1 ... p_n$; Set $P(q) = \gamma_T(P_\mathcal{A}(p)) = r_f = q_1 ... q_n$; Add r_f to the end of r_{dance}; Set $t_{cur} = t_{cur} + d(q, q_n)$, $q = q_n$ and $p = p_n$
  until $t_{cur} > t_{end}$
  return r_{dance}
else
  There is no run originating from q_0 that satisfies ϕ.
end if

The overall algorithm is summarized in Alg. 1. The following theorem shows the correctness of the approach.

Theorem 1. (adapted from [3]) *Assume that there exists a satisfying run for T_P and a LTL formula ϕ. Then, the application of* Alg. 1 *results in a run r_{dance} of T_P satisfying ϕ.*

Remark 4. *(Complexity): Based on* [3]*, the computational complexity of the off-line portion of Alg. 1 is linear in the size of the automaton $\mathcal{A}$, which can have at most $|Q_P| \times |\Pi_P| \times 2^{|\Pi_P|}$ states, where $|\cdot|$ denotes the cardinality of a set. In each loop of the on-line portion of Alg. 1, the computational time is limited by the chosen planning horizon length L. Specifically, it is equivalent to the time of solving a Traveling Salesman Problem on a graph, whose size depends on L.*

V. Generating a dance

In this section, we show that our solution generates motions, or series of poses, within the genre of classical ballet. Such motions satisfy the physical restrictions of the robot and the aesthetic requirements of ballet. Furthermore, we can modify our output motions to satisfy the additional constraints of *allégro* style.

The hard specification is given according to a collection of physical and aesthetic rules which translate to LTL formulas as following:

1) Due to physical restrictions of the robot and the aesthetic requirements, a set of poses are disallowed:
(i) *"always avoid both legs off ground except poses in which two legs are in the same position, and poses satisfying $p_7 \wedge p_{10}$ or $p_5 \wedge p_8$"*

$$\mathbf{G}\ \neg(\text{Roffground} \wedge \text{Loffground} \wedge (\neg\text{Spose}) \wedge \neg(p_7 \wedge p_{10}) \wedge \neg(p_5 \wedge p_8))$$

(ii) *"always avoid both legs in poses p_4, p_6, and p_{10}"*

$$\mathbf{G}\ \neg(\text{Spose} \wedge (p_4 \vee p_6 \vee p_{10}))$$

(iii) *"always avoid both legs in poses satisfying $p_1 \wedge p_2$, $p_1 \wedge p_3$, or $p_2 \wedge p_3$"*

$$\mathbf{G}\ \neg((p_1 \wedge p_2) \vee (p_1 \wedge p_3) \vee (p_2 \wedge p_3))$$

2) Due to physical restrictions of the robot, a collection of sequences of transitions are disallowed:
(i) *"when one leg is in pose p_1 and the other leg is already off ground, always avoid lifting both legs off ground"*

$$\mathbf{G}\ (\neg((p_1 \wedge (\text{Roffground} \vee \text{Loffground})) \rightarrow \mathbf{X}\ (\text{Roffground} \wedge \text{Loffground})))$$

(ii) *"when the right leg is in pose p_2 and the left leg is currently off ground, always avoid performing coronal extensions using the right leg without putting down the left leg first"*

$$\mathbf{G}\ ((\text{p}_2 \wedge \text{Roffground}) \rightarrow \mathbf{X}\ \neg(\text{Roffground} \wedge \text{Lcoronal}))$$

(iii) *"when the left leg is in pose p_2 and the right leg is currently off ground, always avoid performing coronal extensions using the left leg without putting down the right leg first"*

$$\mathbf{G}\ ((\text{p}_2 \wedge \text{Loffground}) \rightarrow \mathbf{X}\ \neg(\text{Loffground} \wedge \text{Rcoronal}))$$

3) Due to physical restrictions of the robot, the robot is required to visit certain poses (e.g., the robot needs to have both standing on the ground once in a while) infinitely many times:
"visiting the pose where both legs are in pose p_2 infinitely many times"

$$\mathbf{G}\ \mathbf{F}\ (\text{Spose} \wedge p_2)$$

The type of dance that we intend to reproduce is known as allégro. Allégro is typified by quick and upbeat dynamic quality; while this type of quality is not directly a part of our discrete model, we can produce runs which contain the movements (series of poses) that ballet dancers use to produce such a movement style. Namely, there are three motions we want to see in the output: *entrechat*, *assemblé*, and *jeté*.

Entrechat is a little jumpy move where the dancer beats his or her feet back and forth. Note that it is a symmetrical move of the body. In our system, entrechat would correspond to a word of T_P (i.e., a sequence of propositions)

$$\tbinom{p_1}{\text{Spose}}\tbinom{p_2}{\text{Spose}}\tbinom{p_3}{\text{Spose}}\tbinom{p_2}{\text{Spose}}\tbinom{p_1}{\text{Spose}},$$

which is generated by a run of T_P (i.e., a sequence of states) $(q_1, q_1)(q_2, q_2)(q_3, q_3)(q_2, q_2)(q_1, q_1)$. Assemblé is another jump but is asymmetrical about the body. In our system, this would also correspond to two words of T_P:

$$\tbinom{p_2}{\text{Spose}}\tbinom{p_1\ ,\ \text{Roffground}}{p_7\ ,\ \text{Rcoronal}}\tbinom{p_2}{\text{Spose}}\tbinom{p_3}{\text{Spose}}\tbinom{p_2}{\text{Spose}}\tbinom{p_1}{\text{Spose}}\tbinom{p_2}{\text{Spose}}$$

or

$$\tbinom{p_2}{\text{Spose}}\tbinom{p_1\ ,\ \text{Loffground}}{p_7\ ,\ \text{Lcoronal}}\tbinom{p_2}{\text{Spose}}\tbinom{p_3}{\text{Spose}}\tbinom{p_2}{\text{Spose}}\tbinom{p_1}{\text{Spose}}\tbinom{p_2}{\text{Spose}}.$$

Hence, the ballerina robot is encouraged to perform entrechat and assemblé more frequently than other movements. Consequently, we encourage the system to enter states satisfying $\binom{p_3}{\text{Spose}}$ (i.e., state (q_3, q_3)) more often for the higher appearance frequency of entrechat, and to enter states satisfying $\binom{p_1\ ,\ \text{Roffground}}{p_7\ ,\ \text{Rcoronal}}$ or $\binom{p_1\ ,\ \text{Loffground}}{p_7\ ,\ \text{Lcoronal}}$ (i.e., states (q_1, q_7) and (q_7, q_1)) more often for a higher appearance frequency of assemblé.

There are two variants of allégro, *grant allégro* and *petit allégro*. We intend to reproduce *grant allégro*, which is distinguished from petit allégro by its large sized movements. We use jeté (a split-leg leap, represented by the proposition $\binom{p_8\ ,\ \text{Roffground}\ ,\ \text{Rcoronal}}{\text{Spose},\ \text{Loffground}\ ,\ \text{Lcoronal}}$ in our system) to characterize our outputs for grant allégro; namely, we enforce recurrence of a pose for *grand jeté* in grand allégro.

Hence, the soft specification can be summarized as

$$\mathcal{S} = \left\{ \begin{matrix} \binom{p_3}{\text{Spose}} & \binom{p_1\ ,\ \text{Loffground}}{p_7\ ,\ \text{Lcoronal}} \\ \binom{p_1\ ,\ \text{Roffground}}{p_7\ ,\ \text{Rcoronal}} & \binom{p_8\ ,\ \text{Roffground}\ ,\ \text{Rcoronal}}{\text{Spose},\ \text{Loffground}\ ,\ \text{Lcoronal}} \end{matrix} \right\}. \quad (6)$$

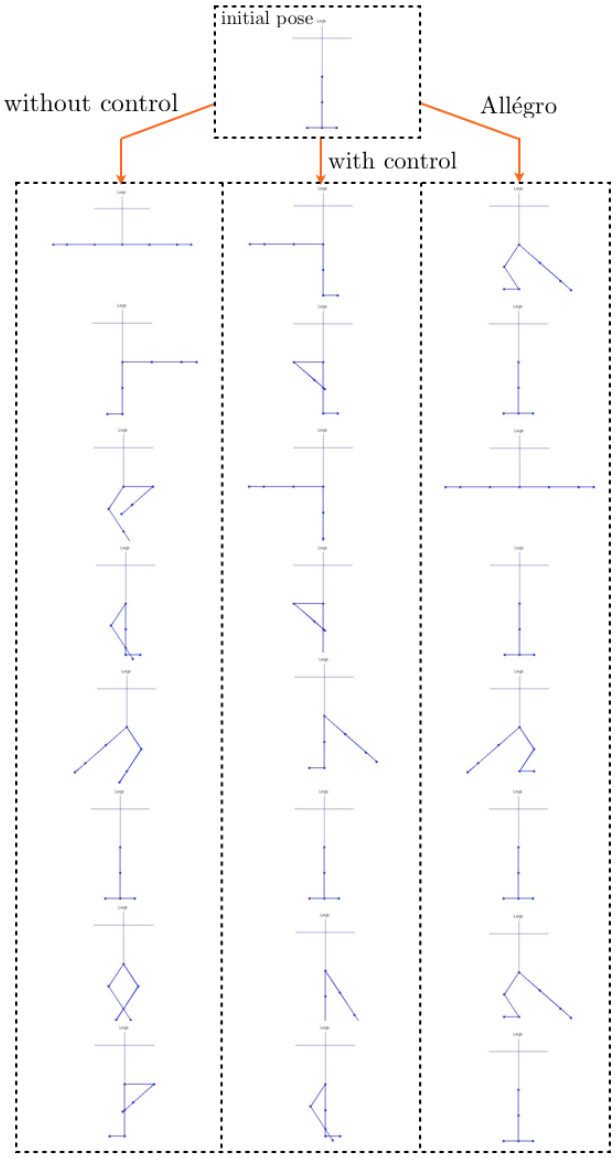

Fig. 2. Three sample sequences demonstrate the results of our control method. The left sequence is an example of a nonphysical (and thus also unaesthetic in some sense) series of poses. The middle sequence is an example of a series of poses endowed with only the hard specifications. Hence, the physical restrictions and aesthetic requirements are satisfied; however, the propositions included in Eqn. 6 do not appear frequently. The right-hand sequence is an example of a series of poses endowed with both the hard and soft specifications (thus is it an allégro phrase). We can see the high occurrence of states (p_1, p_7), (p_7, p_1), and (p_8, p_8). Note that state (p_3, p_3) also occurs frequently in the produced sequence (52 steps in total), but not in the first 9 steps that are shown here. We plot the occurrence rates of these states in Fig. 3.

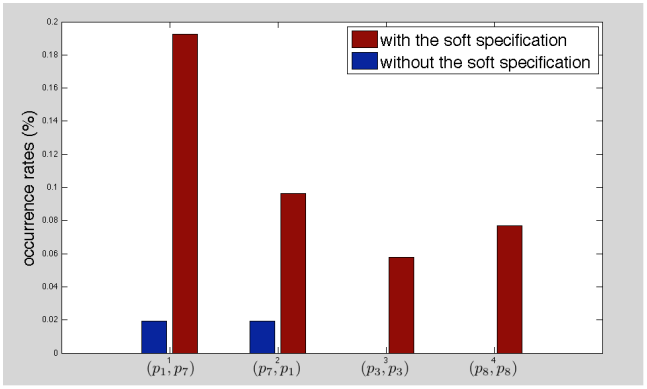

Fig. 3. Occurrence rates of the states satisfying the soft specification for different sample paths show how this specification can encourage certain behaviors of the system. Each sample path contains 52 steps. The red and blue bars show the case with and without the soft specification, respectively. In the case without the encouragement, states (p_3, p_3) and (p_8, p_8) are not visited at all and (p_1, p_7) and (p_7, p_1) are visited only once. In comparison, these states are visited more frequently in case with encouragement from target states. Hence, the soft specifications achieve a differentiated behavior which we call the allégro case as these specifications derive from this style of ballet.

The dance produced by Alg. 1 is animated using a MATLAB script; snapshots of three illustrative sample cases are provided in Fig. 2. These animations have been evaluated by a trained eye and found to be a reasonable initial model of ballet technique. Clearly, significant changes take place between the distinct cases of systems that we animated. In Fig. 3, we show the occurrence rates of the states satisfying the soft specifications in each of the different cases.

VI. Conclusions

This paper has extended results in robot specification to consider more subjective constraints and rewards. The resulting system for robot planning may provide more sophisticated control strategies and, thus, where relevant, a more successful experience of human-like robot behavior. The inherent challenge when attempting to endow an engineered system with such traits is somehow simulating an environment where the human trait is encoded as the system's state. A system with such an augmented view of the world is then, like humans, no longer making decisions based purely on its physical needs but, like many other cyber-physical systems, on a more subjective set of needs (i.e., entering a certain set of states more frequently in order to exude a certain style of ballet).

Furthermore, the careful planning that goes into robotic task specification has applications outside of physical robots. The movement analysis required to produce a system capable of automatically generating movement phrases in the style of classical ballet results in a quantitative phrasing of the rules that govern this somewhat curious example of human behavior. This model may also provide insight into the creativity behind movement choreography as resulting viable trajectories through the transition system for different hard and soft specifications can be compared as part of a quantitative movement study.

Acknowledgment

The work by LaViers and Egerstedt was supported by the US National Science Foundation through grant number 0757317. The work by Chen and Belta was supported by supported by ONR MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020 and NSF CNS-0834260.

REFERENCES

[1] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. *Lecture Notes in Computer Science*, pages 359–364, 2002.

[2] E. M. M. Clarke, D. Peled, and O. Grumberg. *Model checking*. MIT Press, 1999.

[3] Xu Chu Ding, Calin Belta, and Christos G. Cassandras. Receding horizon surveillance with temporal logic specifications. In *the 49th IEEE Conference on Decision and Control*, 2010 (to appear).

[4] G.E. Fainekos, H. Kress-Gazit, and G.J. Pappas. Hybrid controllers for path planning: A temporal logic approach. In *IEEE Conference on Decision and Control*, volume 44, page 4885. Citeseer, 2005.

[5] P. Gastin and D. Oddoux. Fast LTL to Buchi automata translation. *Lecture Notes in Computer Science*, pages 53–65, 2001.

[6] H. Kress Gazit, G. Fainekos, and G. J. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *IEEE Conference on Robotics and Automation*, Rome, Italy, 2007.

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

[8] A. Hutchinson and D. Anderson. *Labanotation.* New Directions., New York, 1954.

[9] M. Kloetzer and C. Belta. Dealing with nondeterminism in symbolic control. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control: 11th International Workshop*, Lecture Notes in Computer Science, pages 287–300. Springer Berlin / Heidelberg, 2008.

[10] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[11] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[12] Amy LaViers and Magnus Egerstedt. The ballet automaton: A formal model for human motion. *(under review) American Control Conference, Proceedings of the 2011*, 2011.

[13] V. Maletic. *Body, space, expression*. Walter de Gruyter & Co., Berlin, 1987.

[14] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Pubns, 1998.

[15] N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. In *VMCAI*, pages 364–380, Charleston, SC, 2006.

[16] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi. Multi-robot motion planning: A timed automata approach. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[17] F. Somenzi and R. Bloem. Efficient büchi automata from ltl formulae. *Twelfth Conference on Computer Aided Verification (CAV'00)*, pages 248–263, 2000.

[18] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

[19] G.W. Warren and S. Cook. *Classical ballet technique*. University of South Florida Press, 1989.

[20] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In *Proc. International Conference on Hybrid Systems: Computation and Control*, 2010.