RSS Workshop on Grounding Human-Robot Dialog for Spatial Tasks, July 2011

Make Your Robot Talk Correctly: Deriving Models of Hybrid System

Neil Dantam
ntd@gatech.edu

Magnus Egerstedt
magnus@ece.gatech.edu

Mike Stilman
mstilman@cc.gatech.edu

***Abstract*—Using both formal language and differential equations to model a robotic system, we introduce a calculus of transformation rules for the symbolic derivation of hybrid controllers. With a Context-Free Motion Grammar, we show how to test reachability between different regions of state-space and give several symbolic transformations to modify the set of event strings the system may generate. This approach lets one modify the language of the hybrid system, providing a way to change system behavior so that it satisfies linguistic constraints on correct operation.**

I. Introduction

To make robots *safe and reliable*, we advocate the use of formal verification to *prove* that the robotic system will operate correctly. Because robots are hybrid systems with both discrete event-driven and continuous time-driven dynamics, we must use methods that operate on the combined hybrid dynamics. By accounting for the combined dynamics, we can create and verify controllers to achieve desired goals.

We present a systematic approach for the construction and verification of hybrid controllers based on the Motion Grammar. The Motion Grammar describes the interaction between a robot and its environment as a formal language [1]. Unlike natural language, which can be ambiguous, formal language is precisely specified. Describing this dialog of sensor readings and input commands as a formal language permits model checking to verify correctness and prevent damage or injury [2]. In this paper, we introduce a method to *transform* the grammar to achieve previously unsatisfied correctness constraints. This transformation uses a calculus of symbolic rules that leverage both the discrete and continuous dynamics of the system. By introducing new events at convenient points or showing that regions in state space are unreachable, we can modify the grammar in a way that changes the set of events which may be generated. The *Motion Grammar Calculus* is a formal method for deriving hybrid controllers using a set of symbolic transformations that modify the system language to achieve correct operation.

Our work in this paper is based on the Motion Grammar, an approach to hybrid control using Context-Free Grammars [1, 2]. Hybrid control combines automata theory [3] and state-space control [4]. There are many other methods for hybrid control including the Motion Description Language (MDL) [5], Hybrid Automata [8], Maneuver Automaton [9], and MDLe [10]. Several authors apply Linear Temporal Logic (LTL) to specify and derive controllers for hybrid systems [11, 12, 13]. Our work differs from this primarily in that we introduce methods for expanding the set of discrete events in the system. More detailed background and comparisons with other methods are given in [2, 14].

II. Summary of the Motion Grammar

The Motion Grammar is a Syntax-Directed Definition expressing the language of interaction between agents and real-world uncertain environments. The *syntax* of the grammar describes the structure of system events, the discrete dynamics. The *semantics* of the grammar describes the time-driven response of the system, the continuous dynamics, using state-space differential equations.

Definition 1. *The Motion Grammar, $\mathcal{G}_M$, is a tuple $\mathcal{G}_M = (Z, V, P, S, \mathscr{U}, \mathscr{X}, \mathscr{Z}, \eta, K)$*

Z	*set of tokens representing events*
V	*set of nonterminals*
P	*set of productions*
S	*starting nonterminal, $S \in V$*
$\mathscr{U}$	*space of robot inputs*
$\mathscr{X}$	*continuous robot state space*
$\mathscr{Z}$	*space of robot sensor readings*
η	*tokenizing function, $\eta : \mathscr{Z} \mapsto Z$*
K	*set of semantic rules*

III. Tokenization and Reachability

Tokens are discrete events. One important type of event is entry into some region of interest within the continuous state space. These regions may be areas where the underlying dynamics of the system change, for example a position where contact is made with another object. They may also be areas where we want our input to the system to abruptly change, for example a mobile robot reaching a waypoint and switching to a different trajectory. A new token or event then is generated when the robot enters into that region.

Definition 2. *The token set Z is a set of regions representing a complete partition of the state space $\mathscr{X}$. For $\lfloor \mathbf{x} \in \mathcal{R}_\mathrm{i} \rfloor \in Z$,*

- *$\mathcal{R}_i \cap \mathcal{R}_j = \emptyset,\ i \neq j$, regions are nonoverlapping.*

- $\bigcup_{i=1}^{|Z|} \mathcal{R}_i = \mathscr{X}$, *regions cover the entire state space.*

Definition 3. *A new token is generated when the system crosses the boundary between two regions. In discrete time, when $x_{k-1} \in \mathcal{R}_i \wedge x_k \in \mathcal{R}_j \wedge i \neq j$, token $\lfloor \mathrm{x} \in \mathcal{R}_\mathrm{j} \rfloor$ appended to the input tape.*

This definition of tokens requires that we compute at runtime what side of the region boundary the system is in. One way to do this is to express this codimension-1 manifold $\mathcal{M}$ as the set of all points where some scalar function is zero,

$$\mathcal{M} = \{x \ : \ s(x) = 0\} \tag{1}$$

Then $\operatorname{sign}(s(x))$ will indicate the side of the manifold that the system is in.

A. Region Crossing

Using the definition of tokens as regions, we can now use the continuous dynamics to predict the discrete dynamics. In Sect. IV, we will use this fact to transform the grammar. Note that since tokens are regions, the set of discrete tokens which may be generated is equivalent to the set of reachable regions of continuous state. Consider the autonomous system dynamics given by smooth function $\dot{x} = f(x)$. Let the boundary between $\mathcal{R}_i$ and $\mathcal{R}_j$ be given by the codimension-1 manifold $\mathcal{M}$, $c = s(x)$. The normal vector to $\mathcal{M}$ at point x is $\nabla s(x)$.

To determine if the system will cross $\mathcal{M}$ at some point x, we consider the direction of $\dot{x} = f(x)$ compared to $\nabla s(x)$. If $\nabla s(x) \cdot f(x) < 0$, then the vectors point away from each other the system moves away from the boundary and will not cross it. This product is the Lie Derivative, $L_f s = \nabla s(x) f(x)$.

Theorem 1. *Let $\mathcal{M} = \{x \ : \ s(x) = 0\}$. If $L_f s(x) < 0 \ \forall x \in \mathcal{M}$, the system will never cross $\mathcal{M}$. If $\exists x \in \mathcal{M},\ L_f s(x) > 0$, the system* may *cross $\mathcal{M}$.*

Proof: Consider some $p \in \mathcal{M}$. Then $L_f s(p) = \nabla s(p) \cdot f(p) = \|\nabla s(p)\| \, \|f(p)\| \cos\theta$, where θ is the angle between $\nabla s(p)$ and $f(p)$. When $\cos\theta > 0$, the system moves off $\mathcal{M}$ in the direction of increasing $s(x)$. When $\cos\theta < 0$, the system moves off $\mathcal{M}$ in the direction of decreasing $s(x)$. Since $\operatorname{sign}(\cos\theta) = \operatorname{sign}(L_f s)$ we can use $\operatorname{sign}(L_f s)$ to test which side of $\mathcal{M}$ the system will move to from p. If there is no p for which $L_f s > 0$, then the system cannot move off the manifold to cross it. If there is any p for which $L_f s > 0$, then from that p, the system will move off the manifold thus cross it. ■

Thm. 1 thus shows whether one region is directly reachable from another based on system dynamics $\dot{x} = f(x)$. One way to express $L_f s$ only along $\mathcal{M}$ is to parametrize $\mathcal{M}$ by some $v \in \Re^{n-1}$, where $\mathscr{X} \in \Re^n$.

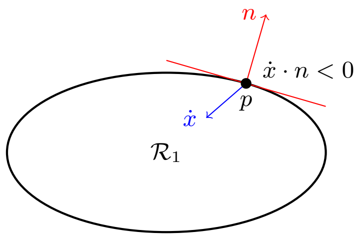


Fig. 1. A region given by $1 = \frac{x^2}{2^2} + y^2$ and tangent planes by $1 + \frac{{x_0}^2}{2^2} + {y_0}^2 = \frac{2x_0}{2^2}x + 2y_0 y$. The system is evolves by $\dot{x} = -x$, indicating that it stays within $\mathcal{R}_1$ at the boundary.

$$\mathcal{M} = \{x \ : \ x = \phi(v)\} \tag{2}$$

For example, if $\mathcal{M}$ is a hyperplane, then $\phi(v) = Mv$, where M is a $n \times (n-1)$ matrix. If $\mathcal{M}$ is a hypersphere, the ϕ can be defined to transform spherical coordinate vector v to Cartesian coordinates x.

B. Example

Consider a region bounded by an ellipse centered on the origin as shown in Fig. 1.

$$M \equiv c = \frac{{x_1}^2}{{a_1}^2} + \frac{{x_2}^2}{{a_2}^2} \tag{3}$$

$$\nabla s(x) = \begin{bmatrix} \frac{x_1}{{a_1}^2} & \frac{x_2}{{a_2}^2} \end{bmatrix} \tag{4}$$

1) System 1: Consider time-driven dynamics $\dot{x} = -x$. This gives us $L_f s = -\frac{{x_1}^2}{{a_1}^2} - \frac{{x_2}^2}{{a_2}^2}$. Since this is always negative, the system will not cross the boundary.

2) System 2: Consider time-driven dynamics $\dot{x} = \begin{bmatrix} y - x & -x - y \end{bmatrix}^T$. This gives us $L_f s = \frac{x_1 x_2}{{a_1}^2} - \frac{x_1 x_2}{{a_2}^2} - \frac{{x_1}^2}{{a_1}^2} - \frac{{x_2}^2}{{a_2}^2}$. We can use a parameterization of the manifold by v, $x = \begin{bmatrix} a_1 \cos v & a_2 \sin v \end{bmatrix}$ to rewrite $L_f s = \frac{a_2}{a_1} c_v s_v - \frac{a_1}{a_2} c_v s_v - 1$. From this, we see that if $\frac{a_1}{a_2} > 1 + \sqrt{2}$ or $\frac{a_2}{a_1} > 1 + \sqrt{2}$, the Lie Derivative will be positive for some values of v meaning that the system will cross the boundary from some, but not all v.

C. Uncontrollable Events

In addition to the *controllable* events resulting from the continuous dynamics, we can model *uncontrollable* events as well. Such events may include fault conditions, uncontrollable continuous dynamics, and even human actions or utterances. All can be included as tokens in the grammar. In this paper however, we focus on the controllable events because it is by appropriately controlling these events that we can transform the grammar to achieve correctness.

IV. The Motion Grammar Calculus

We introduce a calculus of symbolic transformation rules that may be used to construct a hybrid controller for some system. Initially, the system designer must model the hybrid system as a grammar or equivalent automaton. This initial grammar $\mathcal{G}_0$ represents the behavior of the system for any specified input u. Then, these transformations may be applied to rewrite $\mathcal{G}_0$ into another grammar $\mathcal{G}$ that satisfies the desired constraints. These transformations are thus *changing the system language*, $L(\mathcal{G}_0) \neq L(\mathcal{G})$. Through this process, we modify the behavior of the system to make it *correct*.

There are two types of transformations that we consider here. One type operates purely on the discrete grammar, converting it into a more convenient form. The other type of transformation uses continuous domain knowledge to modify the language.

A. Discrete Rules

We can apply any transformation to the grammar which does not modify the system language but instead rewrites the grammar in a more convenient form. Transformations like this are used, for example, to convert grammars to normal forms such as Chomsky Normal Form or Greibach Normal Form [3]. This process of rewriting grammars without changing the language is useful in many domains. We apply it here to dynamical systems in order to rewrite the grammar in a form amenable to our rules which exploit the *continuous* domain semantics.

B. Continuous Rules

These rules use knowledge of the continuous dynamics to modify the language of the grammar based on what regions the system may enter, corresponding to what tokens it may generate. For these rules, we specify some precondition on the production set P, and then specify the resulting token set Z', nonterminal set V' and production set P'.

1) Input Specification: When the continuous dynamics are of the form $\dot{x} = f_0(x, u)$, we are always able to specify an input u.

Transform 1. *Given $p = A \to \alpha f_0(x,u)\beta$, define $f(x) = f_0(x, g(x))$. Then the new production set is $P' = P - p \cup \{A \to \alpha f(x)\beta\}$.*

2) Token Splitting: Some region represented by a token can be split into two regions, creating two new tokens. We then create new productions to indicate entry into these new regions.

Transform 2. *Given some $\zeta_0 = \lfloor \mathrm{x} \in \mathcal{R}_0 \rfloor \in Z$, create tokens $\zeta_1 = \lfloor \mathrm{x} \in \mathcal{R}_1 \rfloor$ and $\zeta_2 = \lfloor \mathrm{x} \in \mathcal{R}_2 \rfloor$ such that $\mathcal{R}_1 \cup \mathcal{R}_2 = \mathcal{R}_0 \wedge \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$ and update token set $Z' = Z - \zeta_0 \cup \{\zeta_1, \zeta_2\}$. The new nonterminal set is $V' = V \cup \{A_0, A_1, A_2, A_3, A_4\}$. The new production set is $P' = P - \{(A \to \alpha_1\zeta_0\kappa\alpha_2) \in P\} \cup \{(A \to \alpha_1 A_0), (A_0 \to A_1|A_2) \,:\, (A \to \alpha_1\zeta_0\kappa\alpha_2) \in P\} \cup \{(A_1 \to \zeta_1\kappa A_3), (A_2 \to \zeta_2\kappa A_4) \,:\, (A \to \alpha_1\zeta_0\kappa\alpha_2) \in P\} \cup \{(A_3 \to A_2|\alpha_2), (A_4 \to A_1|\alpha_2) \,:\, (A \to \alpha_1\zeta_0\kappa\alpha_2) \in P\}$.*

3) Adjacency: If two regions in state space are not adjacent, then the system may not pass directly between them. Thus we can eliminate productions which allow this to happen.

Transform 3. *For $p_1 = A \to r_A\kappa_A B$, $B \to \beta_1| \ldots |\beta_n$, if r_A is not adjacent to $\mathcal{R}_0(\beta_n)$ WLOG, then $P' = P - p_1 \cup \{A \to r_A\kappa_A B'\} \cup \{B' \to \beta_1| \ldots |\beta_{n-1}\}$*

4) Pruning: The continuous dynamics f provide information that may be used to remove productions from the grammar. From Thm. 1, we can show whether the system following some function $\dot{x} = f(x)$ may actually cross into any of the regions specified in the grammar.

Transform 4. *Given productions $p_1 = A \to r_1 f B$ and $p_2 = r_1 f C$, if $\forall r_2 \in \mathsf{first}(C)$ $L_f s(p) < 0$ for all p along the manifold bordering region r_1 and r_2, then $P' = P - p_2$.*

a) Bounce: If the system in moving from region r_1 to region r_2 will immediately reenter r_1, then we can eliminate productions showing that the system will pass through r_2 into some third region.

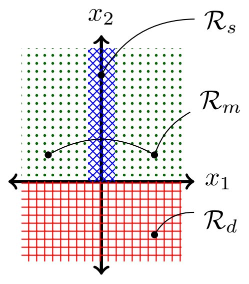
Transform 5. *Given productions $p_1 = A \to r_1\kappa_A B$, $p_2 = B \to r_2\kappa_B C$, and $p_3 = C \to r_1\kappa_B\alpha$, if $L_{\kappa_A} s_{12}(x) > 0 \;\forall x \in \{x : s_{12}(x) = c\} \wedge L_{\kappa_B} s_{21}(x) > 0 \;\forall x \in \{x : s_{21}(x) = c\}$, then $P' = P - p_1 - p_2 \cup \{(A \to r_1\kappa_A B'), (B' \to r_2\kappa_B r_1\kappa_B\alpha)\}$*

C. Using the Calculus to Enforce Correctness

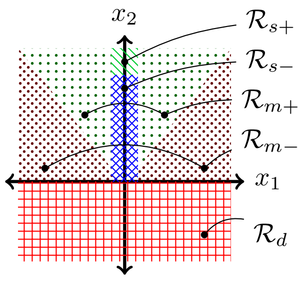
These rules provide important capabilities to work with hybrid models. Tf. 1 allows us to specify the input to the robot to drive toward desired tokens. Tf. 2 allows us to introduce new surfaces where we can switch control inputs. Tf. 3, Tf. 4, and Tf. 5 allow us to *remove* productions from the grammar. We can use this to satisfy a correctness constraint by eliminating certain bad productions causing the constraint violation. In this way, we can systematically produce a grammatical model that specifies the correct operation.

V. Example Derivation

We now demonstrate this approach with a simple example. Consider a mobile robot moving in one dimension, x_1, with a battery, x_2, that discharges as it moves.



(a) Initial Regions



(b) Derived Regions

Fig. 2. Regions/Tokens for 1-dimensional robot with battery.

$$\begin{aligned}
\langle \mathrm{S} \rangle &\rightarrow \lfloor \mathrm{s} \rfloor \left\{ \dot{x} = [u \quad k_s]^T \right\} \langle \mathrm{M} \rangle \\
\langle \mathrm{M} \rangle &\rightarrow \lfloor \mathrm{m} \rfloor \left\{ \dot{x} = [u \quad -|u|]^T \right\} \langle \mathrm{N} \rangle \\
\langle \mathrm{N} \rangle &\rightarrow \langle \mathrm{S} \rangle \mid \langle \mathrm{D} \rangle \\
\langle \mathrm{D} \rangle &\rightarrow \lfloor \mathrm{d} \rfloor \{ \dot{x} = 0 \}
\end{aligned}$$

Fig. 3. Initial grammar for 1-dimensional battery robot.

There is a recharging station at the zero position. When the battery level falls to zero, the robot can no longer operate. The tokens and continuous state space are shown in Fig. 2(a). The initial grammar for this system is given in Fig. 3.

Because we want the robot to keep operating, its battery should never run down. This constraint can be expressed as the LTL formula,

$$\square \left(\neg \lfloor \mathrm{d} \rfloor \right) \tag{5}$$

The initial grammar does not satisfy (5). For example, the grammar generates the string $\lfloor \mathrm{s} \rfloor \lfloor \mathrm{m} \rfloor \lfloor \mathrm{d} \rfloor$, which violates the constraint. Thus, we must apply our transformations to the grammar in order to make it correct.

There are two main ideas to maintaining the battery charge. First, the robot must never stray far enough from the charging station such that its battery level is too low for it to return. Second, the robot must wait in the charging station for some time to recharge. These two ideas indicate how we should split the state space to produce new surfaces for switching controllers, Fig. 2(b), using Tf. 2. Tf. 1 lets us specify controllers to drive the robot appropriately. Finally, we can eliminate bad productions to derive the grammar in Fig. 4.

In this derived grammar, we have removed all productions which may generate $\lfloor \mathrm{d} \rfloor$. Observe that in $\langle \mathrm{M}_2' \rangle$, the robot will return to the charging station when its battery is too low. Since we only momentarily contact $\mathcal{R}_{m-}$ (Tf. 5) and since $\mathcal{R}_{m+}$ is not adjacent to $\mathcal{R}_d$ (Tf. 3), we eliminated the dead battery production. Thus, the grammar now satisfies (5).

VI. Conclusion

In this paper, we have introduced a method for symbolically deriving hybrid controllers using context free

$$\begin{aligned}
\langle \mathrm{S} \rangle &\rightarrow \langle \mathrm{S}_1 \rangle | \langle \mathrm{S}_2 \rangle \\
\langle \mathrm{S}_1 \rangle &\rightarrow \lfloor \mathrm{s+} \rfloor \left\{ \dot{x} = [u \quad k_s]^T \right\} \langle \mathrm{M}_1 \rangle \\
\langle \mathrm{S}_2 \rangle &\rightarrow \lfloor \mathrm{s-} \rfloor \{ \dot{x} = [-x_1 \quad k_s] \} \langle \mathrm{S}_1 \rangle \\
\langle \mathrm{M}_1 \rangle &\rightarrow \lfloor \mathrm{m+} \rfloor \left\{ \dot{x} = [u \quad -|u|]^T \right\} \langle \mathrm{M}_3 \rangle \\
\langle \mathrm{M}_3 \rangle &\rightarrow \langle \mathrm{M}_2' \rangle | \langle \mathrm{S} \rangle \\
\langle \mathrm{M}_2' \rangle &\rightarrow \lfloor \mathrm{m-} \rfloor \{ \dot{x} = [-x \quad -|u|] \} \langle \mathrm{M}_2'' \rangle \\
\langle \mathrm{M}_2'' \rangle &\rightarrow \lfloor \mathrm{m+} \rfloor \{ \dot{x} = [-x \quad -|u|] \} \langle \mathrm{S} \rangle
\end{aligned}$$

Fig. 4. Derived Grammar for 1-dimensional battery robot.

grammars. Using a variety of transformation rules, one can modify an initial grammatical model of the system, changing the system language in the process. We use knowledge of the continuous system dynamics to show which regions are reachable. Following this approach, we can derive a new grammar for the system that will satisfy constraints on correct operation.

References

[1] N. Dantam, P. Kolhe, and M. Stilman, "The motion grammar for physical human-robot games," in *IEEE Intl. Conf. on Robotics and Automation*. IEEE, 2011.
[2] N. Dantam and M. Stilman, "The motion grammar: Linguistic planning and control," in *Robotics: Science and Systems (accepted)*. IEEE, 2011.
[3] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, MA, 1979.
[4] W. Brogan, *Modern Control Theory*. Prentice-Hall, Upper Saddle River, NJ, 1991.
[5] R. Brockett, "Formal languages for motion description and map making," *Robotics*, vol. 41, pp. 181–191, 1990.
[6] M. Egerstedt, "Motion description languages for multi-modal control in robotics," *Control Problems in Robotics*, pp. 74–90, 2002.
[7] M. Egerstedt, T. Murphey, and J. Ludwig, "Motion programs for puppet choreography and control," in *Hybrid Systems: Computation and Control*. Springer, 2007, pp. 190–202.
[8] D. Hristu-Varsakelis and W. Levine, Eds., *Handbook of Networked and Embedded Control Systems*. Birkhauser, 2005.
[9] E. Frazzoli, M. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.
[10] V. Manikonda, P. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures and motion control," *Mathematical Control Theory*, pp. 199–226, 1998.
[11] G. Fainekos, H. Kress-Gazit, and G. Pappas, "Hybrid controllers for path planning: a temporal logic approach," in *IEEE Conf. on Decision and Control*, 2005.
[12] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.
[13] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.
[14] N. Dantam and M. Stilman, "The motion grammar: Linguistic perception, planning, and control," College of Computing, Georgia Institute of Technology, Tech. Rep. GT-GOLEM-2010-001, 2010.
[15] C. Belta and L. Habets, "Controlling a class of non-linear systems on rectangles," *Automatic Control, IEEE Transactions on*, vol. 51, no. 11, pp. 1749–1759, 2006.