Intl. Symposium on Experimental Robotics, June 2012

Linguistic Composition of Semantic Maps and Hybrid Controllers *

Neil Dantam, Carlos Nieto-Granda, Henrik Christensen, and Mike Stilman

Center for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. ntd@gatech.edu, carlos.nieto@gatech.edu, hic@cc.gatech.edu, mstilman@cc.gatech.edu

Abstract. This work combines semantic maps with hybrid control models, generating a direct link between action and environment models to produce a control policy for mobile manipulation in unstructured environments. First, we generate a semantic map for our environment and design a base model of robot action. Then, we combine this map and action model using the Motion Grammar Calculus to produce a combined robot-environment model. Using this combined model, we apply supervisory control to produce a policy for the manipulation task. We demonstrate this approach on a Segway RMP-200 mobile platform.

1 Introduction

This paper provides an approach to generate robot policies by automatically combining Semantic Mapping and Hybrid Control. Semantic mapping and hybrid control are both effective approach within robotics. Semantic mapping produces detailed models of *unstructured environments* [19, 26, 24, 20, 27]; however, this approach provides no direct link to robot action. Hybrid models combine continuous and discrete robot dynamics to efficiently and verifiably represent *robot action* [8, 6, 7, 2, 11, 4]; however, there is no automatic method to produce control models for large, complicated *systems*. While superficially, it appears that semantic mapping and hybrid control are fundamentally different approaches, they are actually closely related. The topological graph of a semantic map and the discrete event system of a hybrid control model are both instances of *formal language*. Thus, we propose to combine the linguistic representations of semantic maps and robot action models to produce an efficient and verifiable control policy for mobile manipulation in unstructured environments.

This work focuses on the application domain of service robots in human environments. Previously, we developed new techniques for mapping using Semantic SLAM [19, 26] and for hybrid systems using our Motion Grammar [8, 6, 7]. Here, we integrate these approaches to produce a combined robot-environment action model. Then, we apply established methods in supervisory control [5] to derive a robot control policy for a mobile manipulation task. This control design approach formally guarantees that the resultant policy satisfies the task specification. Finally, we demonstrate of this approach on a Segway RMP-200 mobile robot.

* This work supported by NSF grants CNS1146352 and CNS1059362 and by grants from the ARL by the MAST CTA and Boeing Corporation.

2 Related Work

Simultaneous Localization and Mapping (SLAM) is the concurrent pose estimation of both the robot and objects in its environment. This is a well studied area with many useful results. Smith and Cheeseman [23] proposed one of the first solutions to the SLAM problem using the Extended Kalman Filter (EKF) to jointly represent the landmark positions along with the robot pose. Folkesson and Christensen developed GraphSLAM [9], an efficient solution to the SLAM problem which preserves landmark independence and is able to find loop closures through nonlinear optimization. Semantic SLAM augments a map with semantically relevant object labels. In this work, we utilize the Semantic SLAM method of Trevor and Nieto [26, 27, 19] to compose a map and hybrid controller.

Hybrid Control is an ongoing research area describing systems with both discrete, event-driven, dynamics and continuous, time-driven, dynamics. Ramadge and Wonham [21] first applied Language and Automata Theory [13] to Discrete Event Systems (DES). Hybrid Automata generally associate differential equations with each state of a Finite Automaton (FA). This method is well studied in control [5, 14, 18, 11, 2]. In this paper, we model hybrid systems using the Motion Grammar which represents continuous dynamics with differential equations and discrete dynamics using a Context-Free Grammar (CFG) [8], and we extend the hybrid approach by automating system modeling using semantic maps. Supervisory control restricts the operation of a DES based on a linguistic specification. This is applied to mobile robot motion planning with FA and Linear Temporal Logic (LTL) by [4, 17]. In our approach, we apply supervisory control to CFGs. Composition of multiple robot behaviors as FA is described in [16]. In this work, we compose a robot model with an automatically generated map while preserving a system representation as a CFG to maintain verifiability and efficiency of execution.

Model checking is the practice of verifying system behavior by modeling the system and verifying that it satisfies a desired specification [3]. This approach is applied to computer software [12] and motion planning for mobile robots [17]. [6] summarizes the classes of models and specifications for which model checking of robotic systems is decidable.

There are several related techniques and alternative approaches for the service robotics domain. Topp and Christensen, [25, 24], provide a separation of regions relating to a user's view on the environment and detection of transitions between them. O'Callaghan [20] developed a new statistical modeling technique for building occupancy maps by providing both a continuous representation of the robot's surroundings and an associated predictive variance employing a Gaussian process and Bayesian learning. In this work we focus on integrating robot mapping with hybrid control methods. The notion of affordances originated in Psychology [10] to describe interaction between agents and environments and has previously provided inspiration for robotics research [22]. We rather focus our approach on direct symbol manipulation techniques with clear algorithmic implementation.

3 Background

The method of this paper produces a robot control policy for unstructured environments by combining Simultaneous Localization and Mapping (SLAM) with

Hybrid Control. We combine these two approaches through Formal Language. First, we produce a basic grammar for the robot's actions and generate the map of the environment via SLAM. Then we compose the action grammar and environment map using the Motion Grammar Calculus. Finally, we apply a supervisory controller to generate the policy for the robot.
We now explain some background on formal language, define our hybrid systems model, the Motion Grammar, and summarize the SLAM technique.

3.1 Formal Language

Formal language is the underlying representation we use to combine mapping and hybrid control. Language and automata theory provide a rigorous method for reasoning about the discrete dynamics of a robotic system. A *formal language* is a set of strings. Strings are sequences of atomic symbols which we can use to describe discrete events, predicates, locations, or actions within our system. A *grammar* defines a formal language. We first briefly review some relevant points of language theory. For a thorough coverage of formal language and its applicability to robotic systems, please see [13, 5, 6].

Definition 1 (Context-Free Grammar, CFG). $G = (Z, V, P, S)$ *where* Z *is a finite alphabet of symbols called* tokens*,* V *is a finite set of symbols called* nonterminals*,* P *is a finite set of mappings* $V \mapsto (Z \cup V)^*$ *called* productions*, and* $S \in V$ *is the* start symbol*.*

The productions of a CFG are conventionally written in Backus-Naur form. This follows the form $A \to X_1 X_2 \ldots X_n$, where A is some nonterminal and $X_1 \ldots X_n$ is a sequence of tokens and nonterminals. This indicates that A may *expand* to all strings represented by the right-hand side of the productions. The symbol ϵ is used to denote an empty string. For additional clarity, nonterminals may be represented between angle brackets $\langle\rangle$ and tokens between square brackets [].
Grammars have equivalent representations as *automata* which *recognize* the language of the grammar. This automata form provides a more convenient representation for some tasks, such as defining languages for maps in Sect. 4.1. The equivalence of grammars and automata means that we can freely choose whichever representation is most convenient. In the case of a Regular Grammar – where all productions are of the form $\langle \mathrm{A} \rangle \to [\mathrm{a}]\, \langle \mathrm{B} \rangle$, $\langle \mathrm{A} \rangle \to [\mathrm{a}]$, or $\langle \mathrm{A} \rangle \to \varepsilon$ – the equivalent automaton is a Finite Automaton (FA), similar to a Transition System with finite state. A CFG is equivalent to a Pushdown Automaton, which is an FA augmented with a stack; the addition of a stack provides the automaton with memory and can be intuitively understood as allowing it to count.

Definition 2 (Finite Automata, FA). $M = (Q, Z, \delta, q_0, F)$*, where* Q *is a finite set of* states*,* Z *is a finite alphabet of* tokens*,* $\delta : Q \times Z \mapsto Q$ *is the* transition function*,* $q_0 \in Q$ *is the* start state*,* $F \in Q$ *is the set of* accept states*.*

Definition 3 (Acceptance and Recognition). *An automaton* M accepts *some string* σ *if* M *is in an accept state after reading the final element of* σ*. The set of all strings that* M *accepts is the* language *of* M*,* $\mathfrak{L}_M$*, and* M *is said to* recognize $\mathfrak{L}_M$*.*

Regular Expressions [13] and Linear Temporal Logic (LTL) [3] are two alternative notations for finite state languages. These representations are convenient forms for defining supervisory controllers as in Sect. 4.3. The basic Regular Expression operators are concatenation $\alpha\beta$, union $\alpha|\beta$, and Kleene-closure α^*. Some additional common Regular Expression notation includes $\neg\alpha$ which is the complement of α, the dot (.) which matches any token, and $\alpha?$ which is equivalent to $\alpha|\epsilon$. Regular Expressions are equivalent to Finite Automata and Regular Grammars. LTL extends propositional logic with the binary operator *until* $\cup$ and unary prefix operators *eventually* $\Diamond$ and *always* $\Box$. LTL formula are equivalent to Büchi automata, which represent *infinite* length strings, termed ω-Regular languages. We can also write ω-Regular Expressions by extending classical Regular expressions with infinite repetition for some α given as α^ω. These additional notations are convenient representations for finite state supervisors.

3.2 The Motion Grammar

Next, we model robot action using the Motion Grammar (MG), giving an initial set of hybrid control actions the robot can perform. MG represents the operation of a robotic system as a Context-Free language, augmenting a Context-Free Grammar with additional variables to handle the continuous dynamics. We use this combined representation to describe the operation of the full robotic *system* [8, 6].

Definition 4. *The Motion Grammar is a tuple*
$\mathcal{G}_M = (Z, V, P, S, \mathcal{X}, \mathcal{Z}, \mathcal{U}, \eta, K)$ *where,*
Z *set of events, or tokens*
V *set of nonterminals*
$P \subset V \times (Z \cup V \cup K)^*$ *set of productions*
$S \in V$ *start symbol*
$\mathcal{X} \subseteq \Re^m$ *continuous state space*
$\mathcal{Z} \subseteq \Re^n$ *continuous observation space*
$\mathcal{U} \subseteq \Re^p$ *continuous input space*
$\eta : \mathcal{Z} \times P \times \mathbb{N} \times \mapsto Z$ *tokenizing function*
$K \subset \mathcal{X} \times \mathcal{U} \times \mathcal{Z} \mapsto \mathcal{X} \times \mathcal{U} \times \mathcal{Z}$ *semantic rules*

The Motion Grammar describes the *language* of the robotic system. The terminal symbols of this language are robot events and predicates, representing a discrete abstraction of the system path.
We use two properties to ensure the validity of a system modeled as a Motion Grammar: *completness* and *correctness*. Completeness ensures that our model $\mathcal{G}$ is a faithful representation of the physical system F. We define this property using the *simulation* relation, that all paths in F are also paths in $\mathcal{G}$. Correctness ensures that our model $\mathcal{G}$ satisfies some desired property S. We define correctness using the subset relation.

Definition 5. *Given $\mathcal{G}_M$ and system F then* $\mathsf{complete}\,\{\mathcal{G}\} \equiv F \preceq \mathcal{G}_M$

Definition 6. *A Motion Grammar $\mathcal{G}$ is correct with respect to some specification S when all strings in the language of $\mathcal{G}$ are also in S:* $\mathsf{correct}\,\{\mathcal{G}, \mathsf{S}\} \equiv \mathfrak{L}(\mathcal{G}) \subseteq \mathfrak{L}(S)$

3.3 Semantic Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is the concurrent execution of both *Localization* and *Mapping* on a robot. *Localization* means determining the current position of the robot based on observations. *Mapping* means determining the positions of objects in the environment based on observations. Typical SLAM implementations combine odometry and other geometric measurements such as point clouds or camera features to simultaneously produce an estimate of the position of both the robot and objects. Using this technique, the robot models unstructured environments.

Our mapping system identifies surfaces and connected free spaces in the world [26, 27]. We use the surfaces, such as walls and tables, to localize the robot based on its relative position to these object. We represent free spaces as Gaussian regions in $\Re^3$ with mean at the center of the free space and standard deviation indicating the dimensions of the free space [19]. Topological connections between these Gaussian regions indicate connected free spaces in the environment. For example, a door or hallway between two rooms would connect the Gaussian regions for those rooms.

We then extend the metric and topological information of the map surfaces and connected Gaussians with additional semantic information by labeling each of the Gaussian regions. These *Semantic Maps* provide useful information for navigation and localization of the robot. In addition, the semantic content of the map permits higher-level reasoning about the spatial regions of the environment. We exploit this semantic information in our composition of the map with a grammar for robot action.

4 Composing Maps and Grammars

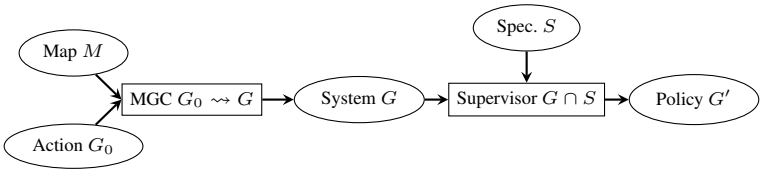
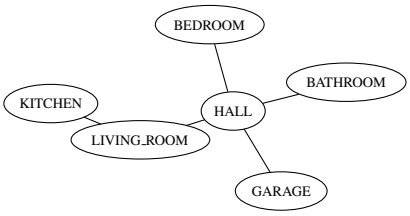


Fig. 1. Sequence of operations to generate policy.

We produce the control policy for the robot by composing a semantic map and a base action grammar, following Fig. 1. We will explain this approach using the example map for the Georgia Tech Aware home, Fig. 2(a), and the base grammar for mobile manipulation, Fig. 2(b). First, we convert the map graph into a grammar for the map language. Then, we compose the map grammar and the action grammar using the Motion Grammar Calculus (MGC) to model the robotic system operating within the mapped environment. Finally, we produce a task policy by applying a supervisory controller to this system model.



(a) Semantic Map M

$$\begin{aligned}\langle \mathrm{S} \rangle &\rightarrow [\mathrm{room}]\,\langle \mathrm{S} \rangle \\ &\mid [\mathrm{object}]\,[\mathrm{pick}]\,\langle \mathrm{S}' \rangle \\ \langle \mathrm{S}' \rangle &\rightarrow [\mathrm{room}]\,\langle \mathrm{S}' \rangle \\ &\mid [\mathrm{place}]\,\langle \mathrm{S} \rangle\end{aligned}$$

(b) Base Grammar G_0

Fig. 2. Example of Semantic Map M and base manipulation grammar G_0. This map represents the Georgia Tech Aware Home.

4.1 Map Languages

To better analyze the semantic map, we first represent this map using formal language. The Gaussian free space regions of the map are arranged in a graph, indicating connectivity between these regions. The graph for the Aware Home is Fig. 2(a). This graph is equivalent to a Regular Language representing the set of all traces through the map.

Definition 7. *Let Map $M = (N, V)$, where N is a finite set of location symbols, and $V \subset N \times N$ is the set of adjacent symbols $n_i \rightarrow n_j$.*

We can transform any Map M into a regular grammar. We note that when analyzing Finite Automata, the language symbols are typically given along transitions [13, 1] wheres in a map, location symbols mark a state. For regular languages, these two conventions – terminal language symbols on states and terminal language symbols on edges – are equivalent. Algorithm 1 transforms the state terminal map to an edge terminal automaton. Then, we can directly convert this automaton to a Regular Grammar.
We demonstrate the conversion for the map in Fig. 2(a). First, we apply Algorithm 1 to produce a FSM from the map graph. Since the output of this algorithm is a Nondeterminisic Finite Automaton with more than the minimum necessary number of states, we convert the NFA to a DFA [1, p152] and minimize the number of DFA states with Hopcroft's Algorithm [1, p180]. This result is Fig. 3(a). Note that in this example, we save two states over the original map in Fig. 2(a). Finally, we convert the FSM to the Regular grammar in Fig. 3(b).

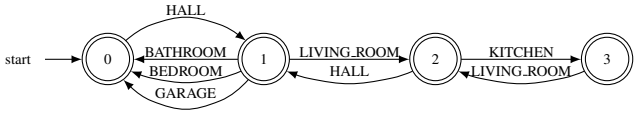
4.2 Composition using the Motion Grammar Calculus

In order to semantically merge the robot and environment models, we apply our Motion Grammar Calculus (MGC). MGC is a set of rewrite rules for hybrid systems modeled in the Motion Grammar [7]. According to these rules, we extend our action grammar with each map symbol while maintaining only those transitions allowed by the map. While supervisory control can only operate to restrict system G using existing symbols, the MGC crucially describes how to introduce *new symbols* into G. There are two relevant rewrite rules from the MGC that we use here.

Algorithm 1: State to Edge Symbols

Input: Q ; // Initial States
Input: $E : Q \times Q$; // Initial Edges
Output: Q' ; // Final States
Output: Z' ; // Edge Symbols
Output: $E' : Q' \times Z' \times Q'$; // Final Edges

$Z' = Q$;
$Q' = E$;
$E' = \emptyset$;
forall $q \in Q$ **do**
  forall $(e_i = Q \rightarrow q) \in E$ **do**
    forall $(e_j = q \rightarrow Q) \in E$ **do**
      $E' = E' \cup e_i \xrightarrow{q} e_j$



(a) Map Automaton

$\langle 0 \rangle \rightarrow$ [hall] $\langle 1 \rangle$
$\langle 1 \rangle \rightarrow$ [bathroom] $\langle 0 \rangle$
| [bedroom] $\langle 0 \rangle$
| [garage] $\langle 0 \rangle$
| [livingroom] $\langle 2 \rangle$
$\langle 2 \rangle \rightarrow$ [hall] $\langle 1 \rangle$
| [kitchen] $\langle 3 \rangle$
$\langle 3 \rangle \rightarrow$ [livingroom] $\langle 2 \rangle$

(b) Map Grammar

Fig. 3. Representing maps with formal language.

Transform 1 (Symbol Splitting) *Given some* $\zeta_0 = [\mathrm{x} \in \mathcal{R}_0] \in Z$*, create tokens* $\zeta_1 = [\mathrm{x} \in \mathcal{R}_1]$ *and* $\zeta_2 = [\mathrm{x} \in \mathcal{R}_2]$ *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* $P' = P - \{(A \to \alpha_1 \zeta_0 \kappa \alpha_2) \in P\} \cup \{(A \to \alpha_1 A_0), (A_0 \to A_1 | A_2) : (A \to \alpha_1 \zeta_0 \kappa \alpha_2) \in P\} \cup \{(A_1 \to \zeta_1 \kappa A_3), (A_2 \to \zeta_2 \kappa A_4) : (A \to \alpha_1 \zeta_0 \kappa \alpha_2) \in P\} \cup \{(A_3 \to A_2 | \alpha_2), (A_4 \to A_1 | \alpha_2) : (A \to \alpha_1 \zeta_0 \kappa \alpha_2) \in P\}$.

Transform 2 (Adjacency Pruning) *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* $P' = P - p_1 \cup \{A \to r_A \kappa_A B'\} \cup \{B' \to \beta_1 | \ldots | \beta_{n-1}\}$

By applying these transforms, we can introduce the map symbols into the action grammar while preserving the validity of the model. Each derivation step maintains the *completeness* of the model according to the path of the hybrid system. By assuming that the initial model is complete, this ensures that all derived models are also complete. For the remainder of the MGC and proofs of its correctness, please see [7].
In addition to these two transforms, we also use the $\mathsf{first}()$ and $\mathsf{follow}()$ sets [1] to define initial and adjacent symbols. The $\mathsf{first}()$ set defines all terminals which may begin some derivation of a grammar symbol. The $\mathsf{follow}()$ set defines all terminals which may appear immediately to the right of some symbol in a grammatical derivation [1][p221].

Definition 8 (First Set). *Define* $\mathsf{first}(X)$ *for some grammar symbol* X *to be the set of terminals which may begin strings derived from* X*.*

Definition 9 (Follow Set). *Define* $\mathsf{follow}(X)$ *for grammar symbol* X *to be the set of terminals* a *that can appear immediately to the right of* X *in some sentential form.*

Note that for map grammars such as Fig. 3(b), the follow set for each terminal symbol is equivalent to the adjacent nodes in the map graph Fig. 2(a).

Proposition 1. *Given a grammar* G *representing some map* M*,* $\mathsf{follow}(z)$ *of some terminal symbol* z *of* G *represents the set of all map locations adjacent to* z*.*

Algorithm 2 describes how we apply these transforms to compose the Map and Action grammars. First, we introduce all map symbols into the action grammar by repeatedly splitting the initial terminal symbol of the action grammar by direct application of Transform 1. Next, we prune out productions indicating transitions between non-adjacent map locations. To prune these productions, we apply Transform 2 by intersecting the grammar with sets of allowable transitions. The disallowed transitions are indicated by the regular expression $L = (.^* z_1 {Z_A}^* z_2 .^*)$ in line 8 of Algorithm 2. The complement of this regular expression defines all paths which do not move directly from z_1 to z_2. Since z_1 and z_2 are non-adjacent, intersecting with $\overline{L}$ will preserve only paths which do not contain the disallowed transition. The result is a grammar which contains the original action model and all permissible transitions from the semantic map.

Algorithm 2: Composing Map and Action Grammars

Input: (Z_M, V_M, P_M, S_M) ; // Map Grammar
Input: (Z_A, V_A, P_A, S_A) ; // Action Grammar
Output: (Z, V, P, S) ; // Combined Grammar

$(Z, V, P, S) \leftarrow (Z_A, V_A, P_A, S_A)$;
/* Add map symbols by splitting first($\mathtt{S_A}$) */
$z_0 = \mathsf{first}(S_A)$;
forall $z \in Z_M$ **do**
  $(Z, V, P, S) \leftarrow$ Transform 1 to split z_0 into z and z_0
/* Prune non-adjacent map symbols */
forall $z_1 \in Z_M$ **do**
  forall $z_2 \in Z_M$ **do**
    if $z_2 \notin \mathsf{follow}(z_1)$ **then**
      $(Z, V, P, S) \leftarrow (Z, V, P, S) \cap \overline{\mathfrak{L}\{.^* z_1 Z_A{}^* z_2 .^*\}}$;

We apply Algorithm 2 to combine the map grammar, Fig. 3(b), with the base grammar for mobile manipulation, Fig. 2(b). In this process, the initial nonterminal of the base grammar, $[\mathrm{room}]$, is repeatedly split into all the symbols of the semantic map. Then all transitions between non-adjacent map symbols are pruned away. This produces the combined grammar of Fig. 4(a).

4.3 Supervisory Control

Finally, we use supervisory control to produce the policy G' from our system model G and task specification S, [5, p133]. This application of supervisory control will permit only those transitions of the model G which are also contained in specification S. We represent this as the intersection,

$$G' = G \cap S \tag{1}$$

Given that G is Context-Free and S is Regular, we use the algorithm defined in [13, p135] to produce Context-Free G', ensuring that we can efficiently execute the policy given by G'. This algorithm operates on a Context-Free language model for system G and a Regular language specification for correct operation S with the assumption that we can *block* any undesirable transitions in G. The corrected system language, then, is $G' = G \cap S$, where G' is also Context-Free. We note in addition that to prune non-adjacent regions permitted by Transform 2 in Algorithm 2, we apply this same language intersection operation.

We use supervisory control of the grammar in Fig. 4(a) to perform the desired mobile manipulation task. To instruct the robot to bring an object from the kitchen to the human in the bedroom, we construct our supervisor according to the regular expressions in Fig. 4(b). Thus, our controlled system is,

$$G' = G \cap \bigcap_{i=0}^{4} S_i = [\mathrm{h}]\,[\mathrm{l}]\,[\mathrm{k}]\,[\mathrm{object}]\,[\mathrm{pick}]\,[\mathrm{l}]\,[\mathrm{h}]\,[\mathrm{b}]\,[\mathrm{place}]\,[\mathrm{h}] \tag{2}$$

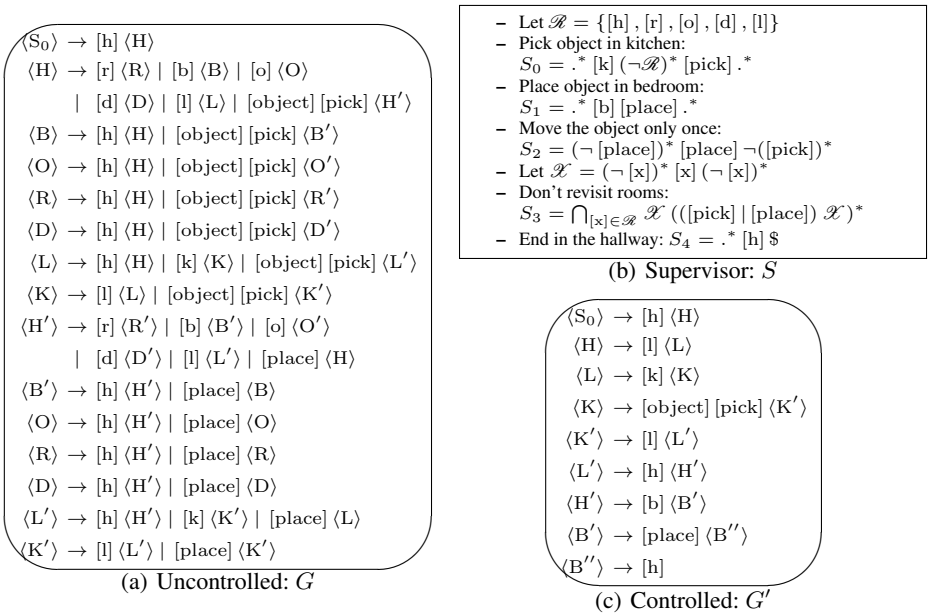


Fig. 4. Grammars for the Uncontrolled and Controlled mobile manipulator in the Aware Home.

5 Experiments

We implemented this approach on a Segway RMP-200 mobile platform as shown in Fig. 5. This platform is equipped with an ASUS Xtion PRO LIVE camera, providing RGBD information for plane and surface extraction and with a UTM-30LX Hokuyo laser used to label the spatial regions as Gaussian models. It includes a Schunk parallel jaw gripper to manipulate objects. We conducted the experiments in the Georgia Tech Aware Home [15] and RIM center.

For both of the home and office environments, we first drove the robot through each area collecting 3D point clouds, laser, and odometry. Our mapper extracts planes and surfaces in the environment, building the map and localizing the robot. During the navigation, the robot partitions the environment into Gaussian regions. This produces the Gaussian map in Fig. 6. Then, we annotate the Gaussian regions of the map with semantic labels. The result is a graph, shown previously for the Aware Home in Fig. 2(a) and also for the RIM center in Fig. 7. This resulting map is suitable for both human interpretation and automatic symbol manipulation.

Next, we apply the method described in Sect. 4 to generate the symbolic model for the robot in each of the environments. For the Aware home, this model is given in Fig. 4(a), and for the RIM center in Fig. 7. For the Aware Home, we asked the robot to peform the following task, *Collect a soda from the kitchen and bring it to the bedroom*, expressed as the specification in Fig. 4(b). For the RIM Center, we apply a similar supervisor in Fig. 8(b) to collect a soda from kitchen and bring it to library.

The policy for the task in the RIM environment, Fig. 8(c), is more complicated than for the Aware Home, Fig. 4(c). This is because the RIM map contains mul-

(a) Aware Home (b) RIM Center (c) Picking

Fig. 5. Segway RMP-200 mobile platform in the Georgia Tech Aware, the RIM Center, and picking a soda can.

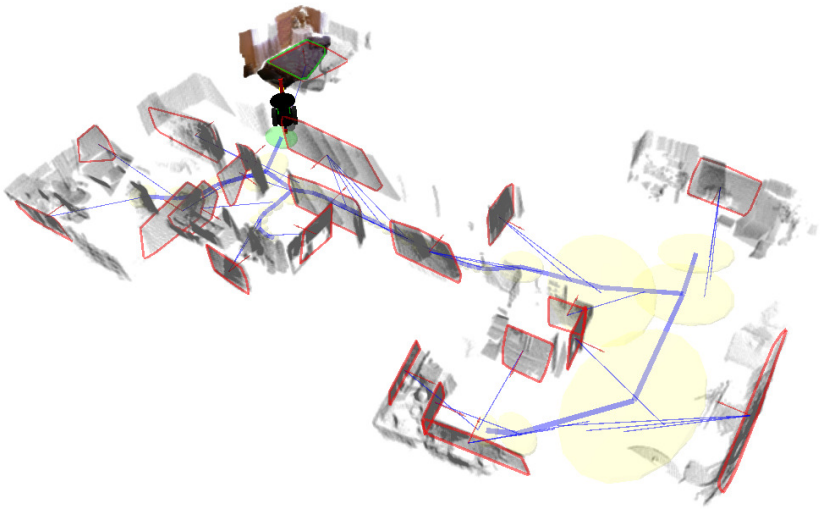
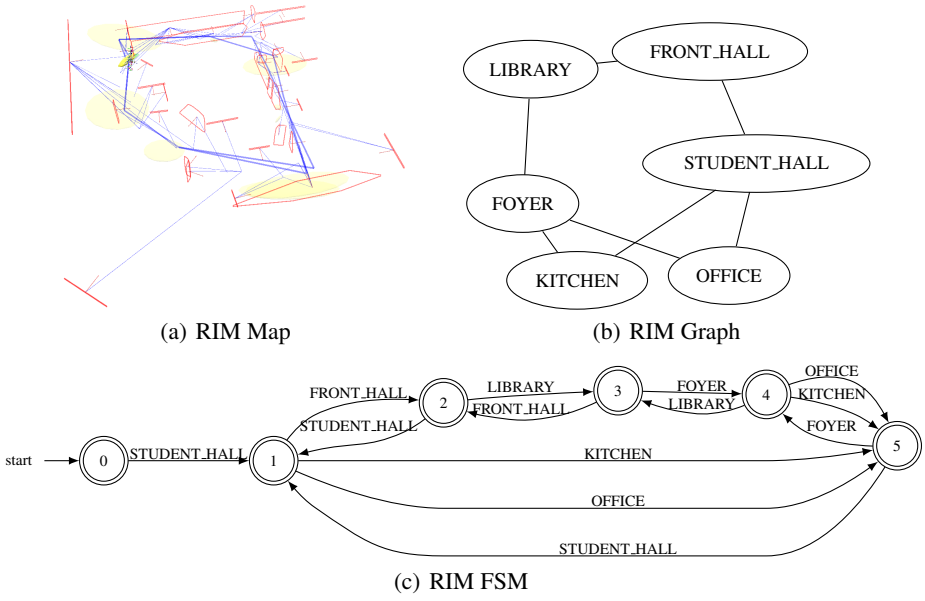


Fig. 6. Generated Semantic Maps for the Aware Home. In the map, black shows 3D robot model, gray shows point clouds, yellow shows connected Gaussian regions (blue edges), and red shows the surfaces.



(a) RIM Map (b) RIM Graph

(c) RIM FSM

Fig. 7. Generated Semantic map of Georgia Tech RIM Center and the equivalent graph and Finite Automata forms.

tiple paths between all rooms. Thus, all these possible paths are captured in the control policy grammar. The result is the nine strings represented by the following regular expression,

$$G' = (k|rlfk|olfk)\,[\text{pick}]\,(fl|fosrl|srl)\,[\text{place}] \tag{3}$$

These generated policies direct the robot along the path to complete the specified task. For the Aware Home, the robot fetches the object from the kitchen and delivers it to the bedroom, illustrated in Fig. 9. This figure shows the path of the robot, both as a trajectory though the map and as the sequence of language symbols.

6 Discussion

In this approach, we combine a Semantic Map and a Motion Grammar using the Motion Grammar Calculus (MGC). This ensures the validity of our final system model because each transform of the MGC preserves *completeness* of the model. Then, applying a supervisory controller guarantees that the final policy is *correct* with regard to the specification. Thus, the overall approach is *correct-by-construction* in the sense that the final system model is guaranteed by the MGC to simulate our initial system, and the resultant policy satisfies the supervisory control specification.

The defining characteristic of this method is the uniform representation of the set of all robot paths as a language with an explicit grammar. This representation

$\langle S_0 \rangle \rightarrow$ [s] $\langle S \rangle$
$\langle S \rangle \rightarrow$ [r] $\langle R \rangle$ | [o] $\langle O \rangle$ | [k] $\langle K \rangle$ | [pick] $\langle S' \rangle$
$\langle O \rangle \rightarrow$ [s] $\langle S \rangle$ | [f] $\langle F \rangle$ | [pick] $\langle O' \rangle$
$\langle K \rangle \rightarrow$ [s] $\langle S \rangle$ | [f] $\langle F \rangle$ | [pick] $\langle K' \rangle$
$\langle F \rangle \rightarrow$ [o] $\langle O \rangle$ | [k] $\langle K \rangle$ | [l] $\langle L \rangle$ | [pick] $\langle F' \rangle$
$\langle L \rangle \rightarrow$ [f] $\langle F \rangle$ | [r] $\langle R \rangle$ | [pick] $\langle L' \rangle$
$\langle R \rangle \rightarrow$ [l] $\langle L \rangle$ | [s] $\langle S \rangle$ | [pick] $\langle R' \rangle$
$\langle S' \rangle \rightarrow$ [r] $\langle R' \rangle$ | [o] $\langle O' \rangle$ | [k] $\langle K' \rangle$ | [place] $\langle S \rangle$
$\langle O' \rangle \rightarrow$ [s] $\langle S' \rangle$ | [f] $\langle F' \rangle$ | [place] $\langle O \rangle$
$\langle K' \rangle \rightarrow$ [s] $\langle S' \rangle$ | [f] $\langle F' \rangle$ | [place] $\langle K \rangle$
$\langle F' \rangle \rightarrow$ [o] $\langle O' \rangle$ | [k] $\langle K' \rangle$ | [l] $\langle L' \rangle$ | [place] $\langle F \rangle$
$\langle L' \rangle \rightarrow$ [f] $\langle F' \rangle$ | [r] $\langle R' \rangle$ | [place] $\langle L \rangle$
$\langle R' \rangle \rightarrow$ [l] $\langle L' \rangle$ | [s] $\langle S' \rangle$ | [place] $\langle R \rangle$

(a) Uncontrolled: G

- Let $\mathscr{R} = \{[s], [k], [o], [f], [l], [r]\}$
- Pick object in kitchen:
  $S_0 = .^* [k] (\neg \mathscr{R})^* [pick] .^*$
- Place object in library:
  $S_1 = .^* [l] [place] \$$
- Move the object only once:
  $S_2 = (\neg [place])^* [place] \neg([pick])^*$
- Let $\mathscr{X} = (\neg [x])^* [x] (\neg [x])^*$
- Don't revisit rooms:
  $S_3 = \bigcap_{[x] \in \mathscr{R}} \mathscr{X} (([pick] \mid [place]) \mathscr{X})^*$

(b) Supervisor: S

$\langle S_0 \rangle \rightarrow$ [s] $\langle S \rangle$
$\langle S \rangle \rightarrow$ [k] $\langle K \rangle$ | [r] $\langle R \rangle$ | [o] $\langle OL \rangle$
$\langle K \rangle \rightarrow$ [pick] $\langle K' \rangle$
$\langle R \rangle \rightarrow$ [l] $\langle OL \rangle$
$\langle OL \rangle \rightarrow$ [f] $\langle F \rangle$
$\langle F \rangle \rightarrow$ [k] $\langle K \rangle$
$\langle K' \rangle \rightarrow$ [f] $\langle F' \rangle$ | [s] $\langle S' \rangle$
$\langle F' \rangle \rightarrow$ [l] $\langle L' \rangle$ | [o] $[O']$
$\langle S' \rangle \rightarrow$ [r] $\langle R' \rangle$
$\langle O' \rangle \rightarrow$ [s] $\langle S' \rangle$
$\langle R' \rangle \rightarrow$ [l] $\langle L' \rangle$
$\langle L' \rangle \rightarrow$ [place]

(c) Controlled: G'

Fig. 8. Grammars for the Uncontrolled and Controlled mobile manipulator in the RIM Center. Notice how the policy captures all possible paths through the environment that satisfy the specification.

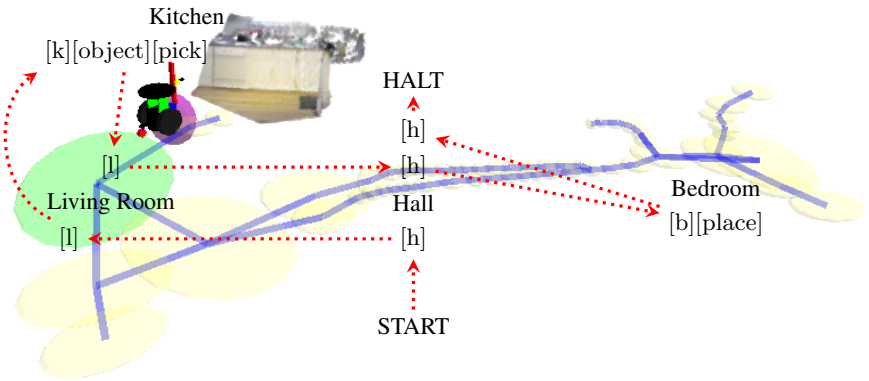


Fig. 9. Path of the robot following controller in Fig. 4(c) and (2), shown as robot enters the living (green oval). Solid blue lines show the map connections between rooms, and dotted red lines show the robot path.

allows iterative development of the grammatical control policy by the progressive application of MGC transformations and supervisory control specifications. At each step of this derivation, the mechanical application of the MGC transforms and supervisory control ensures that we maintain a valid model of the system. Furthermore, because the policy for each task is itself a grammar, we can compose multiple individual task policies to produce a system to perform each of those tasks, all within the same grammatical framework. We expect these capabilities for incremental design and policy composition to be useful as we extend our work to multiple tasks and more complicated systems with larger grammars.

While search-based motion planning could perform some of the tasks in this paper, there are certain advantages given by our linguistic formulation and use of supervisory control for policy generation. Random-sampling planners such as RRTs and PRMs assume a continuous search space, while our application domain includes discrete features for detecting and manipulating objects. General search based planning assumes an explicit goal state and produces a *plan* to reach that state. In contrast, the linguistic approach considers the set of acceptable *paths* and produces a *policy* to stay within that set of paths.

7 Conclusions and Future Work

In this work, we address two significant challenges faced by robot mapping and hybrid controls. Robot mapping produces precise models of the environment, but gives no direct link to robot action. Formal hybrid control models are precise, verifiable, and efficient representations of robot action, but developing these models for large and complicated systems is a tedious task. The linguistic composition demonstrated in this paper eases the challenges posed by each of these separate approaches. Through the automatic, symbolic composition of a map and base hybrid model, we produce a verifiable and executable model of the whole *robotic system*.

We will continue this work with in several ways. First, we will extend our implementation of this method to a variety of mobile manipulation tasks. Next, to provide a natural human interface for the mobile manipulation, we will compose multiple task policies with a grammar for simple human utterances. Finally, to increase the flexibility of this approach, we will extend the offline composition of maps and grammars to online composition as the semantic map is acquired.

Bibliography

[1] A. Aho, M. Lam, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques, & Tools*. Pearson, 2nd edition, 2007.

[2] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid systems*, pages 209–229, 1993.

[3] C. Baier, J.P. Katoen, et al. *Principles of Model Checking*. MIT Press, Cambridge, MA, 2008.

[4] C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5):865–875, 2004.

[5] C.G. Cassandras and Stéphane Lafortune. *Introduction to Discrete-Event Systems*. Springer, 2nd edition, 2008.

[6] N. Dantam and M. Stilman. The motion grammar: Linguistic planning and control. In *Robotics: Science and Systems*, 2011.

[7] N. Dantam and M. Stilman. Deriving models of context-free hybrid systems. In *American Controls Conference*. IEEE, 2012.

[8] 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.

[9] J. Folkesson and H. Christensen. Graphical SLAM - a self-correcting map. *Intl. Conf. on Robotics and Automation*, 2004.

[10] J. J. Gibson. *The senses considered as perceptual systems*. Boston: Houghton Mifflin, 1966.

[11] T.A. Henzinger. The theory of hybrid automata. In *Logic in Computer Science*, pages 278–292. IEEE, 1996.

[12] G.J. Holzmann. *The Spin Model Checker*. Addison Wesley, Boston, MA, 2004.

[13] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, MA, 1979.

[14] D. Hristu-Varsakelis and W.S. Levine, editors. *Handbook of Networked and Embedded Control Systems*. Birkhauser, 2005. ISBN 0817632395.

[15] Julie A. Kientz, Shwetak N. Patel, Brian Jones, Ed Price, Elizabeth D. Mynatt, and Gregory D. Abowd. The Georgia Tech aware home. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, New York, NY, USA, 2008. ACM.

[16] J. Košecká, H.I. Christensen, and R. Bajcsy. Experiments in behavior composition. *Robotics and Autonomous systems*, 19(3):287–298, 1997.

[17] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6): 1370–1381, 2009.

[18] J. Lygeros, K.H. Johansson, S.N. Simic, J. Zhang, and S.S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

[19] C. Nieto-Granda, J. G. Rogers III, A. J. B. Trevor, and H. I. Christensen. Semantic map partitioning in indoor environments using regional analysis. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 1451–1456, Taiwan, Oct 2010. IEEE.

[20] Simon O'Callaghan, Fabio T. Ramos, and Hugh F. Durrant-Whyte. Contextual occupancy maps using gaussian processes. In *IEEE Intl. Conf. on Robotics and Automation*, pages 1054–1060. IEEE, 2009.

[21] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *Analysis and Optimization of Systems*, 25(1):206–230, January 1987.

[22] E. Şahin, M. Çakmak, M.R. Doğar, E. Uğur, and G. Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, 2007.

[23] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Intl. Journal of Robotics Research*, 5(4):56–68, Winter 1987.

[24] E. A. Topp and H. I. Christensen. Detecting region transitions for human-augmented mapping. *IEEE Transactions on Robotics*, pages 1–5, 2010. ISSN 1552-3098.

[25] Elin A. Topp and Henrik I. Christensen. Topological modelling for human augmented mapping. In *IEEE/RSJ Intl.er Conf. on Intelligent Robots and Systems*, pages 2257–2263, Oct. 2006.

[26] A. J. B. Trevor, J. G. Rogers III, C. Nieto-Granda, and H.I. Christensen. Tables, counters, and shelves: Semantic mapping of surfaces in 3d. In *IROS Workshop on Semantic Mapping and Autonomous Knowledge Acquisition*, 2010.

[27] A. J. B. Trevor, J. G. Rogers III, and H.I. Christensen. Planar surface slam with 3d and 2d sensors. In *IEEE Intl. Conf. on Robotics and Automation*, 2012.