Inducing Probabilistic Context-Free Grammars for the Sequencing of Movement Primitives

Rudolf Lioutikov[1], Guilherme Maeda[2], Filipe Veiga[1], Kristian Kersting[3] and Jan Peters[1,4]

***Abstract*— Movement Primitives are a well studied and widely applied concept in modern robotics. Composing primitives out of an existing library, however, has shown to be a challenging problem. We propose the use of probabilistic context-free grammars to sequence a series of primitives to generate complex robot policies from a given library of primitives. The rule-based nature of formal grammars allows an intuitive encoding of hierarchically and recursively structured tasks. This hierarchical concept strongly connects with the way robot policies can be learned, organized, and re-used. However, the induction of context-free grammars has proven to be a complicated and yet unsolved challenge. In this work, we exploit the physical nature of robot movement primitives to restrict and efficiently search the grammar space. The grammar is learned applying a Markov Chain Monte Carlo optimization over the posteriors of the grammars given the observations. The proposal distribution is defined as a mixture over the probabilities of the operators connecting the search space. Restrictions to these operators guarantee continuous sequences while reducing the grammar space. We validate our method on a redundant 7 degree-of-freedom lightweight robotic arm on tasks that require the generation of complex sequences consisting of simple movement primitives.**

I. Introduction

Movement primitives (MPs) are a well established concept in robotics. MPs are used to represent atomic, elementary movements and are, therefore, appropriate for tasks consisting of a single stroke-based or rhythmic movement [1]. They have been used in a large variety of applications, e.g., table tennis [2], pancake flipping [3] and hockey [1]. However, for more complex tasks a single MP is often not sufficient. Such task require sequences of MPs for feasible solutions.

Considering a set or library of MPs, such sequences can be generated in a variety of ways, including Hidden Markov Models [4], Mixture Models [5] and other hierarchical approaches [6]. These approaches can be regarded as mechanisms that produce sequences of MPs. This perspective reveals a common, important downside: understanding these mechanisms requires a significant amount of expert knowledge. However, a declared goal of robotics is the deployment of robots into scenarios where direct or indirect interactions with non-expert users are required. Therefore, more intuitive sequencing mechanisms for non-experts are necessary.

[1]Intelligent Autonomous Systems, TU Darmstadt, 64289 Darmstadt, Germany, {lioutikov, veiga, peters}@ias.tu-darmstadt.de

[2]ATR Computational Neuroscience Labs, 2-2-2 Hikaridai Seika-sho, Soraku-gun, Kyoto 619-0288, Japan, g.maeda@atr.jp

[3]CS Dept. and Centre for Cognitive Science, TU Darmstadt, 64289 Darmstadt, Germany, kersting@cs.tu-darmstadt.de

[4]Max Planck Institute for Intelligent Systems, 72070 Tübingen, Germany

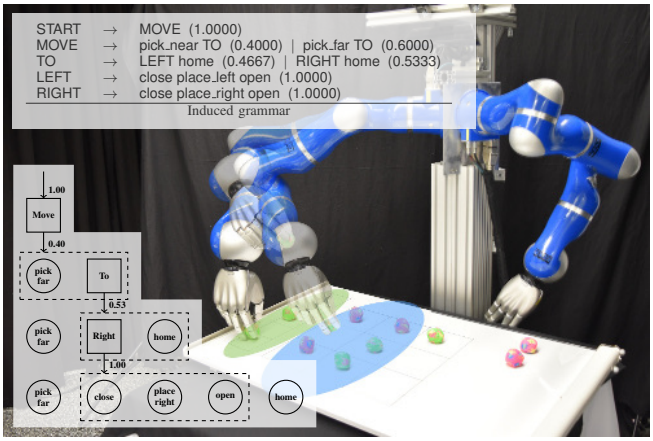


Fig. 1: The robot executes a turn in the tic-tac-toe game, represented as a sequence of movement primitives. The sequence was generated by a probabilistic context-free grammar learned from previously labeled observations.

This work proposes the use of formal grammars for the sequencing of MPs. In particular, we focus on probabilistic context-free grammars (PCFGs) and propose a method to induce PCFGs from observed sampled sequences of primitives. Formal grammars represent a formal description of symbols and rules, representing the structure of a corresponding language. They have been intensively studied in both natural language processing and compiler construction but have also been applied in a variety of fields, e.g., molecular biology [7], bioinformatics [8], computer vision [9] and robotics [10], [11]. PCFGs allow the implicit embedding of hierarchies within the rules of the grammar associating every produced sequence with at least one corresponding parse tree. Such a parse tree represents the derivation of the produced sequence in an intuitive way. Figure 1 shows a learned grammar for placing a stone in a game of tic-tac-toe, including the parse tree for a produced primitive sequence.

However, the understandability of the grammar itself depends on the size of both the grammar, i.e., the number of possible productions, as well as the length of each possible production. The induction of concise but expressive grammars is considered non-trivial and in the context of natural language even an ill-posed problem. A common approach to grammar induction is to formulate the problem as a search problem where each possible grammar is a node in the search space and a set of operators generate the edges between those nodes. This search space can then be traversed through different search methods where a scoring function determines the quality of each grammar. Stolcke et al. [12] suggested

formulating the problem as a Maximum-a-posteriori estimation where the scoring is defined as the posterior given the observations. In order to reduce the possibility of getting stuck in bad local optima, the search space was traversed via beam search. In this work we formulate the search as a Markov Chain Monte Carlo (MCMC) optimization similarly to [13], where the scores are defined as posteriors over the grammars given the observations.

Our proposed approach exploits the structure inherently presented in the physical motions to ease the learning of grammars. We assume each segment of the observed sequences to be a sample from an underlying library of movement primitives (e.g. [5], [14]). Due to the considerably smaller size of a primitive library compared to the corpus of a natural language, the observed sequences of even complex tasks show a simpler structure than a natural language sentence. Furthermore, the type of a movement primitive, e.g., hand or arm movement, can be easier deduced than the type of a word, e.g., verb, noun. In general, the different types of primitives can be determined automatically by identifying their principal components.

An important restriction, improving the induction of grammars for movements is that any produced sequence has to result in a continuous trajectory inside the state space. Therefore, any grammar that would produce a jump in the state space is invalid and has to be removed from consideration. In this work we avoid such grammars directly by restricting the operators to only produce valid grammars.

The contributions of this work are the induction of probabilistic context-free grammars for the sequencing of movement primitives. The posteriors are computed using a novel prior distribution that avoids many pitfalls of existing methods based on minimum description length and Dirichlet distributions (refer to Section II). The search is formulated as a Markov Chain Monte Carlo optimization where the proposed distributions are defined through restrictions put upon the operators connecting the grammar search space. These restrictions include physical constraints presented in the domain of movements. Differently from methods based on greedy beam search (e.g. [12]), MCMC is a global optimizer.

II. Related Work

Movement primitives are usually used to solve tasks consisting of single, atomic stroke-based or periodic movements [1]. For more complex tasks, however, a sequence of primitives has to be applied. An example of such a task is the grasping, positioning, and cutting of a vegetable [15] with Dynamical Movement Primitives (DMPs) [16]. However, in that work the sequences were not learned, but predefined. An approach combining the segmentation of observations and the learning of a sequencing mechanism is presented in [4]. The primitives are encoded using Hidden Markov Models and a graph structure is learned during the segmentation. This graph can be used subsequently to sequence the primitives. Another approach featuring a sequence graph was presented in [17]. The graph is learned from demonstrations through an agglomerative clustering scheme. In [18] a hierarchical version of the Relative Entropy Policy Search algorithm [19] is introduced, capable of learning gating-policies to sequence DMPs. In this work, we propose probabilistic context-free grammars as a means of sequencing movement primitives. Grammars bring the advantage of being a general method capable of representing hierarchies in a principled and intuitive manner.

Motion grammars [10] are extensions of context-free grammars modeling the discrete and continuous dynamics of hybrid systems. The grammars introduced in [10] are predefined and aim at fast task verification. In [11] a probabilistic context-free grammar is used to sequence discrete actions. Analogously to [12] the grammar is learned by applying a beam search for the maximal posterior inside the grammar space. The grammar space is traversed by applying the merge and chunk operators [12] of observed sequences (these operators will be defined in detail in Section III). In contrast to [12] a n-gram like frequency table is used to determine reoccurring patterns in the observations, hence, identifying candidate productions for the chunk operator. To avoid unintuitive, compact grammars the prior definition, originally defined solely by the minimal description length, was extended by a log-Poisson term similar to [20].

While sharing the motivation of learning intuitive, probabilistic context-free grammars for primitive sequencing, our work differs in several ways from [11]. We use a stochastic movement primitive representation and actively take advantage of its properties to induce the grammar. We deviate from the common structure prior definition as an exponential distribution over the minimal description length and define the entire prior as a combination of several Poisson distributions. Furthermore, we use a Markov chain Monte Carlo optimization to find the grammar maximizing the posterior, similarly to [13], which is more robust to local optima than a beam-search.

The grammar induction approach described in [13] uses the Metropolis-Hastings algorithm [21] to learn grammars describing designs in various domains, such as websites and geometric models. The prior is defined using the description length and the grammar learning is not used in any robotics context. Our proposed approach differs significantly in the structure of the observed sequences. In [13], the observations and, hence, the starting points of the grammar induction are already hierarchical structures. Therefore, it is sufficient to traverse the grammar space using solely the merge and split operators. These operators allow the generalization and specialization of grammars, but are not able to introduce new hierarchies like the chunk operator [12]. In this work, we apply all three operators. To achieve the required irreducibility of the Markov chain we additionally introduce the insert operator, negating the effects of the chunk operator. The meaning of the operators shall become clear in the next section.

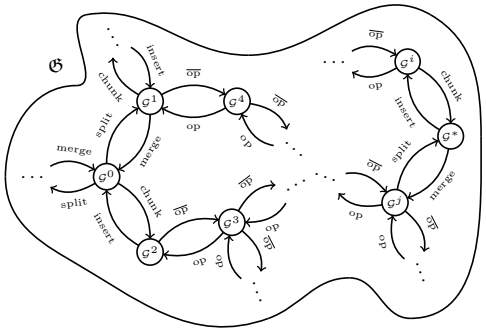


Fig. 2: The grammar space $\mathfrak{G}$ contains all valid grammars $\mathcal{G}^0 \dots \mathcal{G}^*$. The space is traversed by applying operators $\text{op} \in \mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$ on the current grammar. For every operator op generating $\mathcal{G}'$ from $\mathcal{G}$, there exists an $\overline{\text{op}}$ that generates $\mathcal{G}$ from $\mathcal{G}'$, e.g., $\overline{\text{merge}} = \text{split}$, $\overline{\text{chunk}} = \text{insert}$.

III. Background

Before showing how to induce the probabilistic language of movements, this section briefly presents the general concept of probabilistic context-free grammars (PCFGs). A PCFG is a 4-tuple $\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$, consisting of a set of terminals $\mathcal{A} = \left\{a_1, a_2, a_3, \dots, a_{|\mathcal{A}|}\right\}$, a set of nonterminals $\mathcal{V} = \left\{A_1, A_2, A_3, \dots, A_{|\mathcal{V}|}\right\}$, a set of starting symbols $\mathcal{S} \subseteq \mathcal{V}$, and a set of production rules $\mathcal{R} = \{(A, \mathrm{R}_A, \boldsymbol{\rho}_A) | A \in \mathcal{V}\}$. The ordered set $\mathrm{R}_A \in ((\mathcal{A} \cup \mathcal{V})^+)^{|\mathrm{R}_A|}$ is referred to as the productions of the nonterminal $A \in \mathcal{V}$ and the elements $\rho \in \boldsymbol{\rho}_A$ of the multinomial $\boldsymbol{\rho}_A \in \Delta^{|\mathrm{R}_A|-1}$ are the probabilities or parameters of A. A common example grammar is the one describing the language $a^n b^n$

$$\begin{aligned} &\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle, \\ &\mathcal{A} = \{a, b\}, \;\; \mathcal{V} = \{A\}, \;\; \mathcal{S} = \{A\}, \\ &\mathcal{R} = \left\{ \left(A, (ab, aAb), [0.7, 0.3]^T \right) \right\}. \end{aligned}$$

This grammar describes the language of all sequences that consist of any number of as followed by the same number of bs. A more common but less formal notation for the rule set is

$$\begin{aligned} \mathcal{R} = \{ A \to ab \quad & [0.7], \\ A \to aAb \quad & [0.3] \}, \end{aligned}$$

which illustrates, the function of a rule. The nonterminal on the left hand side, A, can produce the sequences ab and aAb on the right hand side with a probability of 0.7 and 0.3 respectively.

Learning formal grammars from sequences of terminals is referred to as grammar induction. Commonly the task is formulated as a search through a grammar space $\mathfrak{G}$, where the connections between grammars are represented as different operators. Such operators manipulate the set of production rules $\mathcal{R}$ and the set of nonterminals $\mathcal{V}$ accordingly. Starting from an initial grammar $\mathcal{G}^0$ these operators are used to traverse the grammar space, searching for the optimal grammar $\mathcal{G}^*$. Various search strategies have been suggested, e.g. beam search [12], [11] and Markov chain Monte Carlo optimization [13].

IV. Inducing a PCFG for Movement Primitives

Given a set of primitives $\Theta = \left\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \dots, \boldsymbol{\theta}_{|\Theta|}\right\}$ and a set of labeled demonstrations $\mathcal{D} = \left\{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \dots, \boldsymbol{d}_{|\mathcal{D}|} \middle| \boldsymbol{d}_i \in \Theta^+\right\}$, the goal is to learn the PCFG $\mathcal{G}^*$ that maximizes the posterior

$$\mathcal{G}^* = \arg\max_{\mathcal{G}} p\left(\mathcal{G}|\mathcal{D}\right). \tag{1}$$

In this work, we define the set of terminals as the set of primitives $\mathcal{A} = \Theta$. During the learning of the grammar the terminals, and, hence the primitives are considered immutable, implying that the search space consists of grammars that only differ in $\mathcal{S}$, $\mathcal{V}$ or $\mathcal{R}$. Each grammar represents a node in the grammar space $\mathfrak{G}$, while the directed edges between nodes are defined by operators. Operators manipulate the rule set $\mathcal{R}$ of a grammar $\mathcal{G}$ and consequently create a new grammar G'. The grammar space $\mathfrak{G}$ is illustrated in Figure 2. In this work we apply four different operators spanning $\mathfrak{G}$, that are described in more detail below. Furthermore, a Markov chain Monte Carlo optimization is used to find the grammar $\mathcal{G}$ that maximizes the posterior $p\left(\mathcal{G}|\mathcal{D}\right)$.

A. Learning grammars through posterior optimization

The posterior $p\left(\mathcal{G}|\mathcal{D}\right)$ describes how probable a given grammar $\mathcal{G}$ is given the observed sequences $\mathcal{D}$. By applying Bayes theorem we can reformulate the posterior, and, hence the maximization as

$$\begin{aligned} \mathcal{G}^* &= \arg\max_{\mathcal{G}} p\left(\mathcal{G}|\mathcal{D}\right), &(2)\\ &= \arg\max_{\mathcal{G}} p\left(\mathcal{D}|\,\mathcal{G}\right) p\left(\mathcal{G}\right), &(3) \end{aligned}$$

where $p\left(\mathcal{D}|\,\mathcal{G}\right)$ is the likelihood of the labeled demonstrations $\mathcal{D}$ given the grammar $\mathcal{G}$, as presented in Section IV-A.1. In Section IV-A.2 we discuss common choices for the prior $p\left(\mathcal{G}\right)$. We finally introduce a novel grammar prior based on Poisson distributions in Section IV-A.3.

1) The likelihood $p\left(\mathcal{D}|\,\mathcal{G}\right)$: is computed for each demonstration independently, yielding

$$p\left(\mathcal{D}|\,\mathcal{G}\right) = \prod_{\boldsymbol{d} \in \mathcal{D}} p\left(\boldsymbol{d}|\,\mathcal{G}\right). \tag{4}$$

Depending on the grammar $\mathcal{G}$ the sequence $\boldsymbol{d}$ could have been produced in multiple ways. Considering every possible derivation results in the sum-product formulation

$$p\left(\boldsymbol{d}\,|\mathcal{G}\right) = \frac{1}{|\text{parse}\left(\boldsymbol{d}, \mathcal{G}\right)|} \sum_{\tau \in \text{parse}(\boldsymbol{d}, \mathcal{G})} \prod_{(A, r, \rho) \in \tau} \rho,$$

where τ represents a single parse tree and $\text{parse}\left(\boldsymbol{d}, \mathcal{G}\right)$ denotes a function producing all feasible parse trees. The 3-tuple (A, r, ρ) represents an edge in the parse tree τ connecting the nonterminal A and its production $r \in R_A$ with a probability of $\rho \in \boldsymbol{\rho}_A$. In this work the function parse creating all possible parse trees for a given demonstration $\boldsymbol{d}$ is implemented by the Earley parser [22]. While the Earley parser suffers from a higher complexity compared to other

parsers, it has the advantage that the parsed grammars do not have to be in any particular form.

2) Commonly the grammar prior $p(\mathcal{G})$*:* is modeled as a joint distribution over the grammar probabilities $\boldsymbol{\rho}_\mathcal{G} = \{\boldsymbol{\rho}_A | A \in \mathcal{V}\}$ and the grammar structure $\mathcal{G}_\mathcal{R} = \{(A, \mathrm{R}_A) | A \in \mathcal{V}\}$ [12], [20], [13], [11],

$$p(\mathcal{G}) = p\left(\boldsymbol{\rho}_\mathcal{G} \middle| \mathcal{G}_\mathcal{R}\right) p(\mathcal{G}_\mathcal{R}). \tag{5}$$

The conditional $p\left(\boldsymbol{\rho}_\mathcal{G} \middle| \mathcal{G}_\mathcal{R}\right)$ itself can be modeled as an independent joint distribution over the parameters of each nonterminal $A \in \mathcal{V}$,

$$p\left(\boldsymbol{\rho}_\mathcal{G} \middle| \mathcal{G}_\mathcal{R}\right) = \prod_{\boldsymbol{\rho}_A \in \boldsymbol{\rho}_\mathcal{G}} p(\boldsymbol{\rho}_A). \tag{6}$$

The dependency on the grammar structure is implicit, since the probabilities $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_\mathcal{G}$ depend on both the set of nonterminals $\mathcal{V}$ and the productions for each nonterminal R_A. The parameters for each nonterminal $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_\mathcal{G}$ form a multinomial distribution, i.e., $\sum_{\rho \in \boldsymbol{\rho}_A} \rho = 1$. Therefore, a Dirichlet distribution would be an obvious choice for the probability distribution over the parameters $p(\boldsymbol{\rho}_A)$ for a single nonterminal $A \in \mathcal{V}$. A significant drawback of using a Dirichlet distribution is its factorial growth in the dimensionality of the multinomial. In fact, using an uninformative Dirichlet distribution, i.e. setting the concentration parameters to 1.0, will result in a probability density of $p(\boldsymbol{\rho}_A) = (\dim(\boldsymbol{\rho}_A) - 1)!$ for any $\boldsymbol{\rho}_A \in \boldsymbol{\rho}_\mathcal{G}$.

To compensate for this growth, the structure prior $p(\mathcal{G}_\mathcal{R})$ is usually modeled as an exponential distribution over the minimal description length (MDL) of the grammar structure $\mathcal{G}_\mathcal{R}$. Every symbol in the production rules, terminal and nonterminal, contributes to the MDL with $\log_2(|\mathcal{A}| + |\mathcal{V}|)$ bits, yielding the over all description length

$$\mathrm{MDL}(\mathcal{G}_\mathcal{R}) = \sum_{(A, \mathrm{R}_A) \in \mathcal{G}_\mathcal{R}} \sum_{r \in \mathrm{R}_A} \mathrm{MDL}(r),$$
$$\mathrm{MDL}(r) = (1 + |r|) \log_2(|\mathcal{A}| + |\mathcal{V}|).$$

A prior $p(\mathcal{G}_\mathcal{R})$ defined as an exponential distribution over the $\mathrm{MDL}(\mathcal{G})$ will prefer small and concise grammars. However, such a prior can lead to grammars that are too compact to be intuitive for non-experts. In order to prefer grammars with a desired production length, η_r, the MDL has been extended with the log of a Poisson distribution with mean η_r[20], [11]. Because of the factorial growth of the parameter prior $p\left(\boldsymbol{\rho}_\mathcal{G} \middle| \mathcal{G}_\mathcal{R}\right)$ the structure prior is often additionally amplified with an exponential weighting term [13] to remain of significance for the overall grammar prior $p(\mathcal{G})$ and, hence, the posterior $p(\mathcal{G}|\mathcal{D})$.

The likelihood $p(\mathcal{D}|\mathcal{G})$ is defined as a product over the average of probabilities, which always results in $p(\mathcal{D}|\mathcal{G}) \leq 1.0$. However, the described grammar prior $p(\mathcal{G})$ is the product of two probability densities, which will very quickly result in $p(\mathcal{G}) \gg 1.0$ and therefore dominate the posterior.

3) The novel prior: presented in this paper aims at inducing PCFGs which are easily understandable for non experts. The key to achieving this goal is the grammar structure, rather than the grammar parameters. Therefore, we suggest a grammar prior, that does not explicitly model a Dirichlet distribution over the parameters but instead implicitly considers the parameters in the overall grammar prior $p(\mathcal{G})$. We model the parameter prior and the structure prior jointly $p(\mathcal{G}) = p\left(\boldsymbol{\rho}_\mathcal{G}, \mathcal{G}_\mathcal{R}\right)$ as

$$p\left(\boldsymbol{\rho}_\mathcal{G}, \mathcal{G}_\mathcal{R}\right) = \frac{p(|\mathcal{R}||\eta_\mathcal{R})}{|\mathcal{R}|} \sum_{(A, \mathrm{R}_A, \boldsymbol{\rho}_A) \in \mathcal{R}} \gamma(A, \mathrm{R}_A, \boldsymbol{\rho}_A) \tag{7}$$

$$\gamma(A, \mathrm{R}_A, \boldsymbol{\rho}_A) = p(|\mathrm{R}_A||\eta_\mathrm{R})\, p(\mathrm{R}_A|\boldsymbol{\rho}_A, \eta_r), \tag{8}$$

where the probabilities over the number of rules $p(|\mathcal{R}||\eta_\mathcal{R})$ and the size of each rule $p(|\mathrm{R}||\eta_\mathrm{R})$ are modeled as Poisson distributions with means $\eta_\mathcal{R}$ and η_R. The probability of each rule is modeled as a weighted average

$$p(\mathrm{R}_A|\boldsymbol{\rho}_A, \eta_r) = \sum_{r \in \mathrm{R}_A, \rho \in \boldsymbol{\rho}_A} \rho\; p(|r||\eta_r), \tag{9}$$

over the probabilities of the corresponding productions. The weighting is given by the grammar parameters $\rho \in \boldsymbol{\rho}_A$ and the probability of each production corresponds to the Poisson distribution over its length $p(|r||\eta_r)$, given a desired production length η_r. Since all components are defined as discrete probabilities, the prior is always $p(\mathcal{G}) \leq 1$, eliminating the need for hard to tune weighting terms to cope with difficult scaling properties. Furthermore, the prior $p(\mathcal{G})$ will now prefer grammars with η_R productions per nonterminal with an average length of η_r symbols per production. The hyperparameters η_R, η_r can be set to achieve a desired simplicity of the grammar. By weighting each production $r \in \mathrm{R}_A$ with the corresponding grammar parameter $\rho \in \boldsymbol{\rho}_A$ the prior gives more significance to production which are more likely to occur.

B. Traversing the grammar space $\mathfrak{G}$

To find the optimal grammar $\mathcal{G}^*$, it is necessary to define mechanisms that generate new grammars. A common choice is to define operators $\mathrm{op} \in \mathcal{O}$, where $\mathcal{O}$ denotes the set of all operators. Each operator op manipulates the rule set $\mathcal{R}$ and consequentially the nonterminal set $\mathcal{V}$ of a given grammar $\mathcal{G}$, therefore, creating a new grammar $\mathcal{G}'$. For each operator op we define a domain Ω_op that op can act upon. The elements in Ω_op depend on the operator itself and can be for instance nonterminals, pairs of nonterminals or productions.

Each grammar represents a node in a grammar space $\mathfrak{G}$. The operators $\mathrm{op} \in \mathcal{O}$ represent directed edges in $\mathfrak{G}$ between two grammars. The grammar space $\mathfrak{G}$ is illustrated in Figure 2. After a grammar $\mathcal{G}'$ was created by applying an operator op on grammar $\mathcal{G}$, the grammar parameters usually have to be recomputed. In this work, the parameters are re-estimated for every new grammar $\mathcal{G}'$ via the Inside-Outside algorithm [23].

Not every possible grammar $\mathcal{G}$ is suitable for sequencing movement primitives. Every sequence produced by $\mathcal{G}$ has to guarantee a smooth, continuous trajectory within the state space of the MPs. In general, this means that a possible next

primitive has to begin close the to the end of the preceding primitive.

We restrict the grammar space $\mathfrak{G}$ to only contain grammars that fulfill this continuity requirement. The restriction is achieved by limiting the domain Ω_{op} of each operator $\mathrm{op} \in \mathcal{O}$, such that if grammar $\mathcal{G}$ fulfills the continuity requirement any grammar $\mathcal{G}'$ resulting from an application of op on $\mathcal{G}$ also fulfills the requirement. We incorporate the continuity requirement into the definition of the two common operators merge and split.

1) split*:* divides the nonterminal $A_i \in \Omega_{\mathrm{split}}$ into two new nonterminals A_j, A_k. The productions R_{A_i} are separated randomly into two corresponding, disjoint sets R_{A_j} and R_{A_k}, where none of the two resulting sets is empty. Each occurrence of A_i is randomly replaced by either A_j or A_k, where both A_j and A_k have to be selected at least once. The domain Ω_{split} contains all nonterminals with at least two productions. Furthermore, every nonterminal in Ω_{split} has to occur at least twice across all productions, including its own.

2) merge*:* combines two nonterminals $(A_j, A_k) \in \Omega_{\mathrm{merge}}$ into a new nonterminal A_i. Correspondingly, the productions of A_i are defined as the union $\mathrm{R}_{A_i} = \mathrm{R}_{A_j} \cup \mathrm{R}_{A_k}$. Every occurrence of A_j and A_k is replaced by A_i. If A_j and A_k contain productions that begin or end in very different MP state spaces a merging would endanger the continuity requirement. We avoid this problem by restricting the domain Ω_{merge}. The domain Ω_{merge} contains only pairs of nonterminals (A_j, A_k), where all productions $\mathrm{R}_{A_j} \cup \mathrm{R}_{A_k}$ begin and end around a similar MP state space.

The split and merge operators negate each other and are capable of generalizing exiting hierarchies in grammars, however they lack the important ability to create new hierarchies. Therefore, we additionally utilize the chunk operator [12] and define the new insert operator that negates the effects of chunk.

3) chunk*:* creates a new nonterminal A with productions $\mathrm{R}_A = \{r\}, r \in (\mathcal{A} \cup \mathcal{V})^+ \wedge r \in \Omega_{\mathrm{chunk}}$. Every occurrence of the sequence r in and production in $\mathcal{R}$ is replaced by A. The domain Ω_{chunk} contains all possible subsequences of all productions in $\mathcal{R}$.

4) insert*:* selects a nonterminal $A \in \Omega_{\mathrm{insert}}$ and replaces each occurrence of A with its production $r \in \mathrm{R}_A$. The domain Ω_{insert} contains all nonterminals with exactly one production.

Given these four operators, we define the set of all possible operators as $\mathcal{O} = \{\mathrm{merge}, \mathrm{split}, \mathrm{chunk}, \mathrm{insert}\}$. Furthermore, the operators in $\mathcal{O}$ are not exchangeable i.e., if a grammar $\mathcal{G}'$ was created by applying the operator op on grammar $\mathcal{G}$, there exists no operator in $\mathcal{O} \setminus \{\mathrm{op}\}$ that is able to produce $\mathcal{G}'$ from $\mathcal{G}$.

C. Finding $\mathcal{G}^*$

Similarly to [13] we search for the optimal grammar $\mathcal{G}^* = \arg\max_{\mathcal{G}} p(\mathcal{G}|\mathcal{D})$ using an Markov chain Monte Carlo optimization. However, the inputs are expected to already be hierarchical, restricting the grammar search to a reorganization of already existing productions by applying solely the merge and split operators. Given that our inputs are flat sequences, that is, pure sequences without hierarchy, of observed primitive samples, we additionally apply the chunk operator, that is capable of creating hierarchies [12]. The insert operator ensures the irreducibility of the Markov chain. Analogously to [13], we apply the Metropolis Hastings algorithm. However, since [13] solely uses the split and merge operator, the paper directly defines the proposal distributions $q\left(\mathcal{G}' \middle| \mathcal{G}\right)$ as the probability of a split or a merge. In this work we define the proposal distribution as a mixture over the four operators $\mathcal{O} = \{\mathrm{merge}, \mathrm{split}, \mathrm{chunk}, \mathrm{insert}\}$,

$$q\left(\mathcal{G}', \mathrm{op}' \middle| \mathcal{G}\right) = \sum_{\mathrm{op} \in \mathcal{O}} p\left(\mathrm{op}' \middle| \mathcal{G}, \eta_{\mathrm{op}'}\right) q_{\mathrm{op}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right),$$

with mixture components $q_{\mathrm{op}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right)$. The mixture probability is defined as

$$p\left(\mathrm{op}' \middle| \mathcal{G}, \eta_{\mathrm{op}'}\right) = \frac{\eta_{\mathrm{op}'}\left(1 - \delta_{|\Omega_{\mathrm{op}'}|}\right)}{\sum_{\mathrm{op} \in \mathcal{O}, \eta_{\mathrm{op}}}\left(1 - \delta_{|\Omega_{\mathrm{op}}|}\right)} \tag{10}$$

where $\eta_{\mathrm{op}} \in \mathbb{R}$ is a weighting for the operator op, $\delta_{|\Omega_{\mathrm{op}}|}$ denotes the Kronecker delta over the size of the domain Ω_{op} for operator op. Given that the operators in $\mathcal{O}$ are not exchangeable, a mixture component $q_{\mathrm{op}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right)$ should not contribute any probability mass if $\mathrm{op} \neq \mathrm{op}'$. This restriction is achieved by the Kronecker deltas $\delta_{\mathrm{op}',\mathrm{op}}$ in the following mixture components.

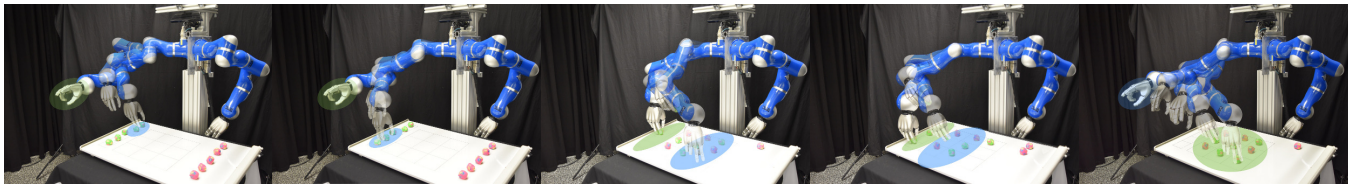
1) $q_{\mathrm{split}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right)$*:* Given that the split operator was applied to produce $\mathcal{G}'$ from $\mathcal{G}$, there exist $A_i \in \mathcal{V}$ and $A_j, A_k \in \mathcal{V}'$. The chance of randomly selecting $A_i \in \Omega_{\mathrm{split}}$ is $1/\left|\Omega_{\mathrm{split}}\right|$. Additionally, every production $r \in \mathrm{R}_{A_i}$ was randomly assigned to either R_{A_j} or R_{A_k}, while each of those two sets had to be selected at least once, resulting in the hypergeometric probability $\mathcal{H}_{\left|\mathrm{R}_{A_j}\right|} = \mathcal{H}\left(\left|\mathrm{R}_{A_j}\right| \middle| 2\left(|\mathrm{R}_{A_i}| - 1\right), |\mathrm{R}_{A_i}| - 1, \left|\mathrm{R}_{A_j}\right| + |\mathrm{R}_{A_k}|\right)$. Finally, the N_{A_i} occurrences of A_i across all productions in $\mathcal{R}$ have been replaced by N_{A_j} and N_{A_k} occurrences of A_j and A_k in $\mathcal{R}'$, contributing yet another hypergeometric probability $\mathcal{H}_{N_{A_j}} = \mathcal{H}\left(N_{A_j} \middle| 2\left(N_{A_i} - 1\right), N_{A_i} - 1, N_{A_j} + N_{A_k}\right)$. The overall probability of $\mathcal{G}'$ being produced from $\mathcal{G}$ by using a split operator is given as:

$$q_{\mathrm{split}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right) = \frac{\delta_{\mathrm{op}',\mathrm{split}}}{\left|\Omega_{\mathrm{split}}\right|} \mathcal{H}_{\left|\mathrm{R}_{A_j}\right|} \mathcal{H}_{N_{A_j}}. \tag{11}$$

2) $q_{\mathrm{merge}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right)$*:* The only stochastic part in the merge operator is the decision which pair $(A_i, A_j) \in \Omega_{\mathrm{op}}$ is selected, therefore the probability for merge is given as

$$q_{\mathrm{merge}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right) = \frac{\delta_{\mathrm{op}',\mathrm{merge}}}{\left|\Omega_{\mathrm{merge}}\right|}. \tag{12}$$

3) $q_{\mathrm{chunk}}\left(\mathcal{G}' \middle| \mathcal{G}, \mathrm{op}'\right)$*:* Given that the domain Ω_{chunk} already contains all possible subsequences of all productions in $\mathcal{R}$, the probability for choosing one sequence at random



(a) pick_near (b) pick_far (c) place_left (d) place_right (e) home

Fig. 3: The five arm primitives used in the sequences, representing turns in the tic-tac-toe game.

START	→	DEMO1 (0.25) | DEMO2 (0.25) | DEMO3 (0.25) | DEMO4 (0.25)
DEMO1	→	pick_far close place_right open home (1.00)
DEMO2	→	pick_near close place_right open home (1.00)
DEMO3	→	pick_far close place_left open home (1.00)
DEMO4	→	pick_near close place_left open home (1.00)

(a) Initial grammar. Grammar index 0 in Figure 5

START	→	MOVE (1.00)
MOVE	→	pick_near TO (0.40) | pick_far TO (0.60)
TO	→	LEFT home (0.47) | RIGHT home (0.53)
LEFT	→	close place_left open (1.00)
RIGHT	→	close place_right open (1.00)

(b) Induced grammar. Grammar index 171 in Figure 5

Fig. 4: (a) The initial grammar for learning a tic-tac-toe turn. (b) The grammar with the best posterior after 400 iterations of the MCMC optimization.

is

$$q_{\text{chunk}}\left(\mathcal{G}'\middle|\,\mathcal{G},\text{op}'\right)=\frac{\delta_{\text{op}',\text{chunk}}}{|\Omega_{\text{chunk}}|}. \tag{13}$$

4) $q_{\text{insert}}\left(\mathcal{G}'\middle|\,\mathcal{G},\text{op}'\right)$*:* The domain Ω_{insert} is already restricted to nonterminals with a single production, therefore the probability of insert is simply

$$q_{\text{insert}}\left(\mathcal{G}'\middle|\,\mathcal{G},\text{op}'\right)=\frac{\delta_{\text{op}',\text{insert}}}{|\Omega_{\text{insert}}|}. \tag{14}$$

At every iteration of the Metropolis-Hasting algorithm a random new grammar is sampled from the proposal distribution $\mathcal{G}',\text{op}'\sim q\left(\mathcal{G}',\text{op}'\middle|\,\mathcal{G}\right)$. This new grammar is then accepted with a probability of

$$\text{acc}\left(\mathcal{G}',\text{op}'\middle|\,\mathcal{G}\right)=\min\left(1,\frac{p\left(\mathcal{G}'|\mathcal{D}\right)^{1/T}q\left(\mathcal{G},\overline{\text{op}'}\middle|\,\mathcal{G}'\right)}{p\left(\mathcal{G}|\mathcal{D}\right)^{1/T}q\left(\mathcal{G}',\text{op}'\middle|\,\mathcal{G}\right)}\right), \tag{15}$$

where T denotes a decaying temperature and $\overline{\text{op}'}$ denotes the complementary operator to op′, i.e. $\overline{\text{split}}=\text{merge}$, $\overline{\text{chunk}}=\text{insert}$. If the new grammar was accepted it is set to the current grammar $G\leftarrow G'$ and the next iteration begins. After a defined number of iterations, the grammar with the highest posterior is returned.

V. Experiments

We evaluated the proposed approach on two real robot tasks. First, we induced a grammar producing turns of the tic-tac-toe game. Second, we learned a grammar that assists a human with the assembly of a simple toolbox. In both tasks the necessary primitives were encoded as Probabilistic Movement Primitives [1]. For each of the tasks we compare the posterior resulting from our proposed prior, *Grammar Poisson*, with the one resulting from three common structure prior choices, *MDL*, *Poisson + MDL*, *Avg. Poisson*. The *MDL* prior is simply defined as an exponential distribution with the MDL as its energy [13]. The *Poisson + MDL* prior weights the description language for every production with the Poisson probability over the length of the production [20]. Finally, the *Avg. Poisson* prior discards the MDL completely and is solely represented by a Poisson distribution over the average length of all productions [11]. A major difference of the *Grammar Poisson* prior to the other discussed priors is that we do not model the distribution over the grammar parameters as a Dirichlet distribution but rather use them as a weighting for the average production length.

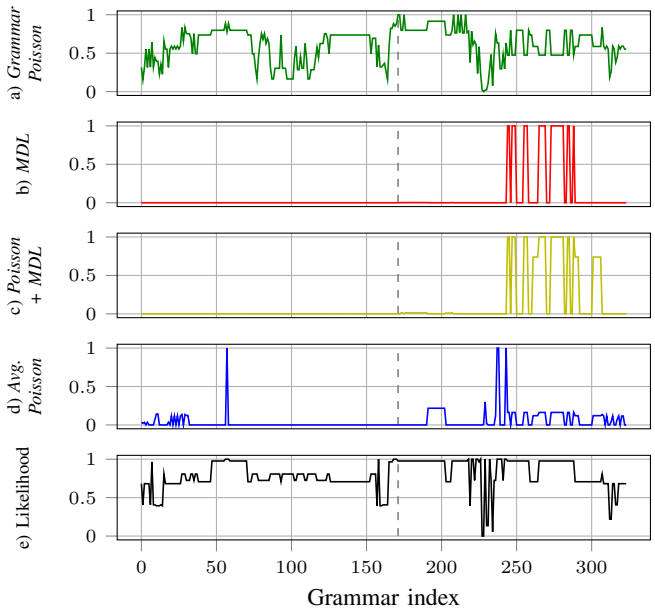
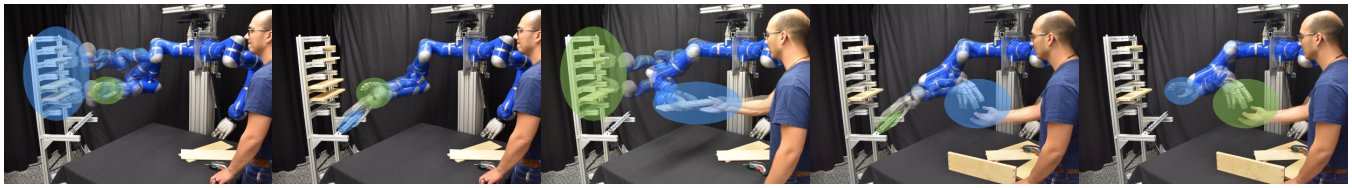


Fig. 5: The posteriors and the likelihood for the tic-tac-toe turn grammar. The vertical, dashed line indicates the index of the highest posterior (171), given the presented Poisson prior.

A. Learning a Grammar for Tic-Tac-Toe Turns

In this task we learned a grammar that allows the robot to play tic-tac-toe against a human. Each produced sequence corresponds to one turn of the game, i.e. picking a stone, closing the hand, placing the stone on the field , opening the hand and returning to the home position. The goal is not to learn the logic behind the game but rather the induction of an intuitive grammar producing valid turns. The segmentation of



(a) take_board (b) take_screw (c) give_board (d) give_screw (e) home

Fig. 6: The arm primitives of the box assembly task.

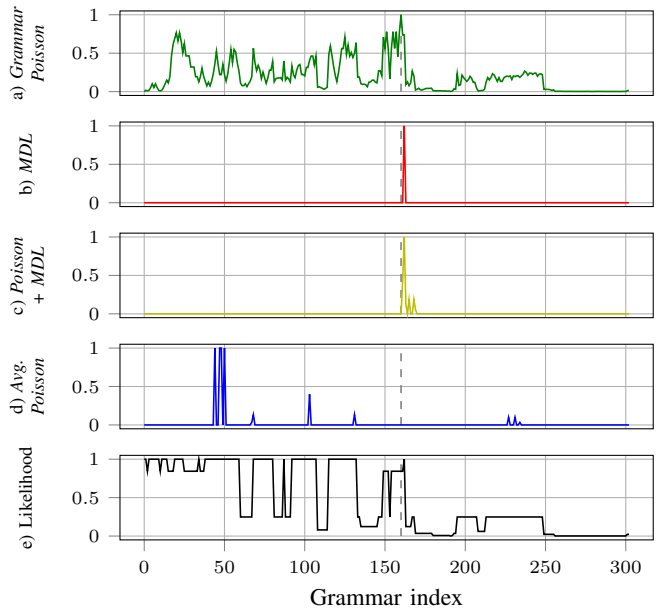


Fig. 7: The posteriors and the likelihood for the box assembly task. The vertical, dashed line indicates the index of the highest posterior (160), given the presented Poisson prior.

START	→	ASSEMBLE_SB (0.50) | ASSEMBLE_BB (0.50)
BOARD	→	take_board GIVE_B home (1.00)
SCREW	→	take_screw GIVE_S home (1.00)
BSB	→	BOARD SCREW BOARD (1.00)
SBS	→	SCREW BOARD SCREW (1.00)
GIVE_S	→	close_screw give_screw open_screw (1.00)
ASSEMBLE_BB	→	BOARD BOARD SBS BOARD SCREW SCREW (1.00)
GIVE_B	→	close_board give_board open_board (1.00)
ASSEMBLE_SB	→	SBS BOARD SCREW BSB (0.50) | BOARD SBS BOARD SBS (0.50)

Induced grammar. Grammar index 160 in Figure 7

Fig. 8: The grammar with the best posterior for the box assembly task after 400 iterations of the MCMC optimization

the demonstrations and, hence, the learning of the primitives was done beforehand via Probabilistic Segmentation [5]. The five resulting arm primitives are shown in Figure 3, where the green and blue highlighted areas mark the start and end of the end-effector.

The grammar learning was initialized with 16 observations of four unique sequences, each consisting of five terminals. The initial grammar is shown in Figure 4a. We initialized our approach with a desired number of rule $\eta_{\mathcal{R}} = 5$, the desired number of average productions per rule $\eta_{\mathrm{R}} = 2$ and the desired average length of each production $\eta_r = 3$. The weights for each operator were set uniformly to $\eta_{\mathrm{op}} = 1, \mathrm{op} \in \mathcal{O}$. The MCMC optimization was run for 400 steps and resulted in 324 accepted grammars. The corresponding normalized posteriors are shown in Figure 5a and the grammar with the highest posterior, grammar index 171 is shown in Figure 4b. The induced grammar intuitively represents, that each produced sequence will move a near or a far stone to either the left or the right side of the playing field. Furthermore, after every closing of the hand there will be a later opening of the hand. A possible sequence produced by the grammar, including the corresponding parse tree is seen in Figure 1. For simplicity the naming of the nonterminals was chosen manually after the grammar learning. An automated naming of the nonterminals corresponding to the semantics of the productions is outside of the scope of this paper and remains part of future work. Figure 5b-d shows the normalized posteriors corresponding to the three common priors. The x-axis corresponds to the different grammars traversed during the MCMC optimization, i.e. the grammar $\mathcal{G}^i, \mathrm{op}^i \sim q\left(\mathcal{G}^i, \mathrm{op}^i \middle| \mathcal{G}^{i-1}\right)$ was sampled from the proposal distribution around $\mathcal{G}^{i-1}$ by applying op^i. The spiky behaviour of the posteriors (b-d) is due to the uninformative Dirichlet prior for the grammar parameters and the exponential distribution over the MDL. Both of these factors can change significantly with a small change in the grammar, e.g. a merge creating a rule with many productions or a chunk reducing the length of a long production.

Furthermore, it is noticeable that the likelihood of the grammar $p\left(\mathcal{G}^i \middle| \mathcal{D}\right)$ does not play significantly into the posteriors of (b-d), whereas our posterior (a) shows a much stronger dependency on the likelihood. This behaviour, is explained by the fact that the likelihood as introduced in Equation (4) is a probability mass function, but the three priors (*MDL*, *Poisson + MDL*, *Avg. Poisson*) are products of probability density functions. In contrast, our prior (*Grammar Poisson*) is defined as a probability mass function, averaging over multiple Poisson distributions. This definition prohibits the prior from completely dominating the likelihood. As a consequence, the proposed prior (*Grammar Poisson*) results in a posterior (a) that takes the given observations much stronger into account than the posteriors in (b-d).

B. Learning a Grammar for a Simple Toolbox Assembly

This task shows the abstraction capabilities of our approach. The demonstrations were again segmented beforehand and resulted in the five arm primitives, shown in Figure 6, and four hand primitives, closing and opening the hand for both a board and a screw grasp. The set of demonstrations contained three different sequences, consisting of

40 terminals each. Every observation showed the grasping and handing over of four boards and four screws, either alternating between the board and the screw or starting with two boards and alternating subsequently. The approach was initialized with $\eta_{\mathcal{R}} = 9$, $\eta_{\mathrm{R}} = 2$, $\eta_r = 2$. The weights for the split and merge operators were set to 1 and the remaining two were set to 2. The MCMC optimization ran for 400 iterations and 303 grammars were accepted. The posteriors for the accepted grammars are shown alongside the likelihood in Figure 7. The posteriors show similar behavior as in the previous task. Both the *MDL* and the *Poisson + MDL* have a maximum at 162, indicating that the corresponding grammar has the minimal description length of all accepted grammars. The *Avg. Poisson* prior has its maximum at 44 due to an average production length close to η_r. However, the corresponding grammar contains 14 rules with one production each. The grammar with the maximum posterior according to the *Grammar Poisson* prior is given at index 160 and presented in Figure 8. The grammar abstracts a full turn from taking a board or screw until going back to the home position. This subsequence was not marked in any way and was detected as a consequence of the grammar learning. The sequence occured multiple times during each observation. Abstracting it into a nonterminal will therefore simplify the grammar significantly. Furthermore, the grammar encodes that a grasping of a board or a screw through the closing of the hand has to be eventually followed by the corresponding opening of the hand. The alternation between handing over a board and a screw is represented in the two rules for SBS and BSB and the rules for ASSEMBLE_SB. The option of starting with two boards is encoded in ASSEMBLE_BB.

VI. Conclusion

In this work, we introduced probabilistic context-free grammars as a mechanism to sequence movement primitives. The paper presents an approach to induce such grammar from flat sequences of movement primitive samples, i.e., no hierarchy in the observations, while taking advantage of a stochastic primitive representation. The new introduced grammar prior is defined over several coupled Poisson distributions, and eliminates the many complications that arise from both Dirichlet parameter priors and minimal description length based structure priors. In our method, the hyperparameters of the prior have an clear semantic interpretation, namely the number of productions for each nonterminal and the average length of each production. The optimal posterior is learned using a Markov chain Monte Carlo optimization where the proposal distribution is formulated as a mixture model over four operators while avoiding local solutions. The learned grammars are simple and intuitive as evaluated in several experiments on a real robot platform.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand) and from the European Union's Horizon 2020 research and innovation programme under grant agreement 640554 (SKILLS4ROBOTS).

References

[1] Paraschos, Daniel, Peters, and Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots (AURO)*, accepted.

[2] Muelling, Kober, Kroemer, and Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research (IJRR)*, pp. 263–279, 2013.

[3] Kormushev, Calinon, and Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*. IEEE, 2010, pp. 3232–3237.

[4] Kulic, Ott, Lee, Ishikawa, and Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *I. J. Robotics Res.*, vol. 31, pp. 330–345, 2012.

[5] Lioutikov, Neumann, Maeda, and Peters, "Learning movement primitive libraries through probabilistic segmentation," *International Journal of Robotics Research (IJRR)*, 2017.

[6] Daniel, Neumann, Kroemer, and Peters, "Hierarchical relative entropy policy search," pp. 1–50, 2016.

[7] Chiang, Joshi, and Searls, "Grammatical representations of macromolecular structure," *Journal of Computational Biology*, vol. 13, pp. 1077–1100, 2006.

[8] Rivas and Eddy, "The language of RNA: a formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, pp. 334–340, 2000.

[9] Zhu, Mumford, *et al.*, "A stochastic grammar of images," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, pp. 259–362, 2007.

[10] Dantam and Stilman, "The motion grammar: Analysis of a linguistic method for robot control," *IEEE Trans. Robotics*, vol. 29, pp. 704–718, 2013.

[11] Lee, Su, Kim, and Demiris, "A syntactic approach to robot imitation learning using probabilistic activity grammars," *Robotics and Autonomous Systems*, vol. 61, pp. 1323–1334, 2013.

[12] Stolcke, "Bayesian learning of probabilistic language models," Ph.D. dissertation, Berkeley, CA, USA, 1994, uMI Order No. GAX95-29515.

[13] Talton, Yang, Kumar, Lim, Goodman, and Mech, "Learning design patterns with bayesian grammar induction," in *25th Annual Symposium on User Interface Software and Technology, October 7-10, 2012, Cambridge, MA, USA,*, Miller, Benko, and Latulipe, Eds. ACM, 2012, pp. 63–74.

[14] Niekum, Chitta, Marthi, Osentoski, and Barto, "Incremental semantically grounded learning from demonstration," in *Robotics: Science and Systems IX, June 24 - June 28, 2013, Technische Universität Berlin, Berlin, Germany*, vol. 9, 2013.

[15] Lioutikov, Kroemer, Maeda, and Peters, "Learning manipulation by sequencing motor primitives with a two-armed robot," in *13th International Conference on Intelligent Autonomous Systems, July 15-18, 2014, Padova, Italy*, ser. Advances in Intelligent Systems and Computing, Menegatti, Michael, Berns, and Yamaguchi, Eds., vol. 302. Springer, 2014, pp. 1601–1611.

[16] Ijspeert, Nakanishi, Hoffmann, Pastor, and Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, pp. 328–373, 2013.

[17] Manschitz, Kober, Gienger, and Peters, "Learning to sequence movement primitives from demonstrations," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 14-18, 2014, Chicago, IL, USA*. IEEE, 2014, pp. 4414–4421.

[18] Daniel, Neumann, Kroemer, and Peters, "Learning sequential motor tasks," in *International Conference on Robotics and Automation, May 6-10, 2013, Karlsruhe, Germany*. IEEE, 2013, pp. 2626–2632.

[19] Peters, Muelling, and Altun, "Relative entropy policy search," in *Twenty-Fourth National Conference on Articial Intelligence, July 1115, 2010, Atlanta, Georgia, USA*, 2010.

[20] Kitani, Sato, and Sugimoto, "Recovering the basic structure of human activities from noisy video-based symbol strings," *IJPRAI*, vol. 22, pp. 1621–1646, 2008.

[21] Andrieu, Freitas, Doucet, and Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, pp. 5–43, 2003.

[22] Earley, "An efficient context-free parsing algorithm (reprint)," *Commun. ACM*, vol. 26, pp. 57–61, 1983.

[23] Baker, "Trainable grammars for speech recognition," *The Journal of the Acoustical Society of America*, vol. 65, pp. S132–S132, 1979.