Journal Title
XX(X):1–17

Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Learning Attribute Grammars for Movement Primitive Sequencing

Rudolf Lioutikov[1], Guilherme Maeda[2], Filipe Veiga[3], Kristian Kersting[4] and Jan Peters[5,6]

Abstract
Movement Primitives are a well studied and widely applied concept in modern robotics. However, composing primitives out of an existing library 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 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. We exploit the physical nature of robot movement primitives to restrict and efficiently search the grammar space. The grammar is learned by 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. Moreover, we present an approach for the categorization of probabilistic movement primitives and discuss how the connectibility of two primitives can be determined. These characteristics in combination with restrictions to the operators guarantee continuous sequences while reducing the grammar space. Additionally, a set of attributes and conditions is introduced that augments probabilistic context-free grammars in order to solve primitive sequencing tasks with the capability to adapt single primitives within the sequence. The method was validated on tasks that require the generation of complex sequences consisting of simple movement primitives using a 7 degree-of-freedom lightweight robotic arm.

Introduction

Movement primitives (MPs) are a well established concept in robotics. MPs are used to represent atomic, simple movements and are, therefore, appropriate for tasks consisting of a single stroke-based or rhythmic movement (Paraschos et al. 2017). They have been used in a large variety of applications, e.g., table tennis (Muelling et al. 2013), pancake flipping (Kormushev et al. 2010) and hockey (Paraschos et al. 2017). However, for more complex tasks a single MP is often not sufficient. Such tasks 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 (Kulic et al. 2012), Mixture Models (Lioutikov et al. 2017) and other hierarchical approaches (Stulp and Schaal 2011). These approaches can be regarded as mechanisms that produce sequences of MPs, revealing 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.

This paper introduces 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 sequences of primitives. Formal grammars represent a formal description of symbols and rules, encoding the structure of a corresponding language. They have been extensively studied in both natural language processing and compiler construction but have also been applied in a variety of fields such as molecular biology (Chiang et al. 2006), bioinformatics (Rivas and Eddy 2000), computer vision (Zhu and Mumford 2007; Kitani et al. 2005) and robotics (Dantam and Stilman 2013; Lee et al. 2013; Sarabia et al. 2015). The choice of learning a probabilistic representation over a single deterministic plan is based on the insight that probabilistic representations of behavior are generally more robust to changes than deterministic representations, especially in dynamic environments. For instance, in a collaborative task a fixed plan describing the behavior between a robot and a user

[1]Personal Autonomous Robotics Lab, UT Austin
[2]ATR Computational Neuroscience Labs, Kyoto
[3]MIT Computer Science & Artificial Intelligence Lab
[4]CS Dept. and Centre for Cognitive Science, TU Darmstadt
[5]Intelligent Autonomous Systems, TU Darmstadt
[6]Max Planck Institute for Intelligent Systems, Tübingen

Corresponding author:
Rudolf Lioutikov
Department of Computer Science
The University of Texas at Austin
2317 Speedway, Stop D9500
Austin, Texas 78712-1757, U.S.A.
Email: lioutikov@utexas.edu

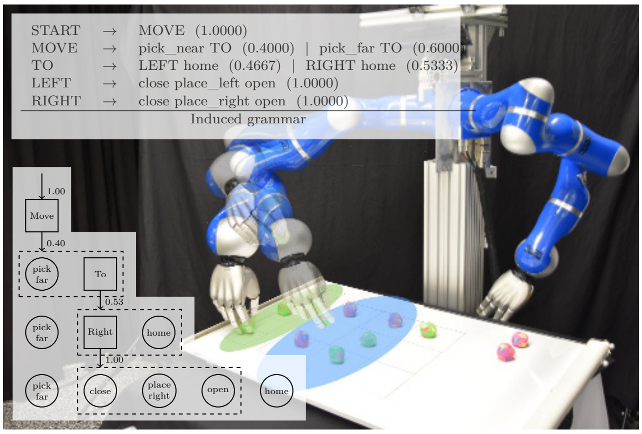


Figure 1. The robot executes a turn in the tic-tac-toe game, represented as a sequence of movement primitives. The sequence was generated by a probabilistic context-free grammar learned from previously labeled observations.

would require the user to always behave according to the set plan. A distribution over plans allows for at least some flexibility as long as the plan is still within the distribution. Furthermore, 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.

The understandability of the grammar itself highly depends on the size and structure of the grammar. 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. (Stolcke 1994) 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 (Talton et al. 2012), where the scores are defined as posteriors over the grammars given the observations.

To ease the learning of grammars the proposed approach exploits the structure inherently presented in the physical motions . We assume each segment of the observed sequences to be a sample from an underlying library of movement primitives (e.g., (Lioutikov et al. 2017; Niekum et al. 2015)). 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 sentence of a natural language. Furthermore, the category of a movement primitive, e.g., hand or arm movement, can be easier deduced than the category of a word, e.g., verb or noun. We discuss how to determine the category of a movement primitive later in this paper.

An important restriction that improves 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 exclusively produce valid grammars, by ensuring the connectibility between two consecutive movement primitives.

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 disadvantages of existing methods based on minimum description length and Dirichlet distributions. 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. This paper extends the work presented in (Lioutikov et al. 2018) with details on the categorization and the assessment of connectibility of movement primitives. Additionally, we enhance the induced grammars with attributes and an evaluation scheme for movement primitive sequencing tasks. The presented method is evaluated on a Tic-Tac-Toe task, where a grammar is induced that sequences primitives in order to pick up a stone and place it on a Tic-Tac-Toe playing field. In addition, we evaluate the method on a collaborative chair assembly task, where the robot induced a grammar describing a sequence of required hand-over primitives.

Related Work

Movement primitives are usually used to solve tasks consisting of single, atomic stroke-based or periodic movements (Paraschos et al. 2017). 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 (Lioutikov et al. 2014) with Dynamical Movement Primitives (DMPs) (Ijspeert et al. 2013). However, in (Lioutikov et al. 2014) the sequences were not learned, but predefined. An approach combining the segmentation of observations and the learning of a sequencing mechanism is presented in (Kulic et al. 2012). 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 (Manschitz et al. 2014). The graph is learned from demonstrations through an agglomerative clustering scheme. 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

$\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$
$\mathcal{A} = \{\mathsf{a}, \mathsf{b}\}\,,\ \ \mathcal{V} = \{\mathsf{A}, \mathsf{B}\}\,,$
$\mathcal{S} = \{\mathsf{A}\}\,,$
$\mathcal{R} = \{$

START	→	A	(1.0)
A	→	aA	(0.3)
	→	aB	(0.7)
B	→	bB	(0.3)
	→	b	(0.7)

$\}$

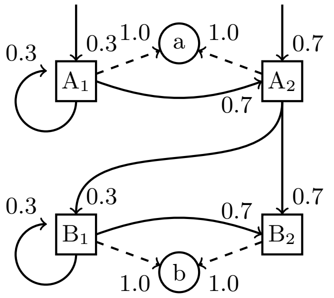


Figure 2. Left: A regular grammar describing the language $\mathsf{a}^+\mathsf{b}^+$. The corresponding parse tree for the sequence `aabb` is shown in Figure 3b. Right: A hidden Markov model that is equivalent to the regular grammar. Squares describe hidden states and circles describe the emissions. Figure 3a shows an instantiated HMM for the sequence `aabb`

representing hierarchies in a principled and intuitive manner.

Motion grammars (Dantam and Stilman 2013) are extensions of context-free grammars modeling the discrete and continuous dynamics of hybrid systems (Dantam and Stilman 2012). Motion grammars aim at fast task verification and have not yet been induced from observations. In (Dantam et al. 2011) and (Sarabia et al. 2015) the benefits of applying context-free grammars in human-robot interaction scenarios such as playing chess or making music collaboratively. In (Lee et al. 2012, 2013) probabilistic context-free grammars are used to sequence discrete actions. Analogously to (Stolcke 1994) 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 (Stolcke 1994) to observed sequences. In contrast to (Stolcke 1994) 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 (Kitani et al. 2008). The meaning of the operators shall become clear later in the paper.

While sharing the motivation of learning intuitive, probabilistic context-free grammars for primitive sequencing, our work differs from (Lee et al. 2012) in several ways. 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 Markov chain Monte Carlo optimization to find the grammar maximizing the posterior, similarly to (Talton et al. 2012), which is more robust to local optima than beam-search.

The grammar induction approach described in (Talton et al. 2012) uses the Metropolis-Hastings algorithm (Andrieu et al. 2003) 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. In addition, the structure of the observed sequences differs significantly from our problem setting. In (Talton et al. 2012), 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 (Stolcke 1994). 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.

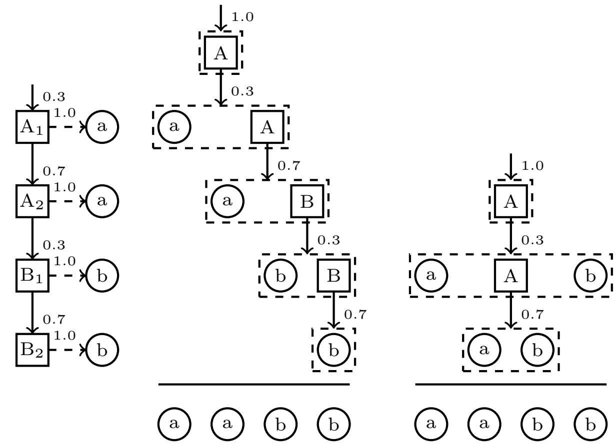
Background

Before presenting the induction of movement primitive grammars, this section briefly introduces the general concepts of formal grammars and movement primitives.

Formal Grammars

A formal grammar is a description of a formal language in terms of symbols and production rules. The symbols, $(\mathcal{A} \cup \mathcal{V})$, are commonly separated into two disjoint sets called terminals $\mathcal{A}$ and nonterminals $\mathcal{V}$, with the convention that terminals represent the atomic elements of the language, while nonterminals can be substituted with sequences of symbols. Each production rule $r_\beta \in \mathrm{R}_\alpha$ substitutes the symbol sequence $\alpha \in (\mathcal{A} \cup \mathcal{V})^+$ with $\beta \in (\mathcal{A} \cup \mathcal{V})^+$. A rule is commonly denoted as $\alpha \to \beta$, where α and β are referred to as the left and right-hand side respectively. With these definitions a grammar can be described as a 4 tuple $\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$, where $\mathcal{R}$ denotes the set of all R_α's and $\mathcal{S}$ is the set of all starting symbols $\mathcal{S} \subseteq (\mathcal{A} \cup \mathcal{V})^+$. A grammar can contain multiple rules for the same left hand side, i.e., $|\mathrm{R}_\alpha| > 1$, resulting in a nondeterministic grammar. Weighting each $\mathrm{R}_\alpha \in \mathcal{R}$ with a corresponding multinomial $\boldsymbol{\rho}_\alpha \in \Delta^{|\mathrm{R}_\alpha|-1}$, leads to a stochastic or probabilistic grammar. Furthermore, grammars can be recursive, i.e., a series of productions starting with a rule in R_α results in a sequence containing α. The nondeterministic and recursive properties allow grammars to represent complex, hierarchical relations between symbols in relatively simply structured production rules.

Formal grammars are commonly classified via the Chomsky hierarchy. The most constrained and, therefore, least expressive grammars in this hierarchy are the so called regular grammars. Languages described by these grammars are known as regular expressions, e.g., the expression $\mathrm{a}^+\mathrm{b}^+$ represents all sentences that consist of one or more as followed by one or more bs. Probabilistic regular grammars are equivalent to hidden Markov models (HMM). Similar to instantiated HMMs formal grammars explain observed sequences in so called parse trees. Figure 2 shows a regular grammar describing the expression $\mathrm{a}^+\mathrm{b}^+$ as well as a corresponding HMM, while Figure 3 shows an (a)



(a) HMM (b) RG (c) CFG

Figure 3. Parse trees for the sequence aabb of (a) and (b) the hidden Markov model and the regular grammar shown in Figure 2. (c) shows the corresponding parse tree for the context-free grammar shown in Table 1.

instantiated HMM and (b) a parse tree for the sequence aabb of the HMM and the regular grammar respectively. The left hand side of each rule is a single nonterminal and the right-hand side can be either a terminal or a terminal followed by a nonterminal. In addition, each production of a single rule is weighted by a probability, e.g., the nonterminal A produces the sequences aA and aB with a probability of 0.3 and 0.7 respectively. While the parse tree and the graph for the time series look fairly similar the description of the higher level policy as a grammar is more intuitive to understand than the graphical model and is, therefore, better suited for non-expert users. However, given that HMMs are an extensively studied tool for the learning and analysis of time series, they are a much more common choice for the sequencing of discrete actions, such as movement primitives, than regular grammars. Despite their simplicity regular grammars can describe complex languages. In fact, every finite language is regular and can, therefore, be described by a regular grammar. Furthermore, every infinite language satisfying the pumping lemma is also regular, as for instance the described language a^+b^+.

Probabilistic Context-Free Grammars are able to describe infinite languages that can not be described by regular grammars, and, therefore, neither by HMMs. An example for such a language is a^nb^n, which contains any sequence with n as that are always followed by the same number of bs. Regular grammars do not contain any mechanism to keep track of the number of produced as. However, such languages can be described by context-free grammars like the one shown in Table 1. Context-free grammars (CFGs) still have only a single nonterminal on the left hand side, but can now contain an arbitrary sequence of terminals and nonterminals of the right-hand side. Despite the more complex language, the context-free grammar is at least as intuitive as the previous regular grammar. Figure 3c shows the corresponding parse tree for

$$
\begin{aligned}
&\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle \\
&\mathcal{A} = \{\mathsf{a}, \mathsf{b}\}, \ \ \mathcal{V} = \{\mathsf{A}\}, \ \ \mathcal{S} = \{\mathsf{A}\} \\
&\mathcal{R} = \{ \\
&\quad \text{START} \ \rightarrow \ \text{A} \qquad (1.0) \\
&\quad \text{A} \ \rightarrow \ \text{ab} \qquad (0.7) \\
&\quad \phantom{\text{A}} \ \rightarrow \ \text{aAb} \qquad (0.3) \\
&\}
\end{aligned}
$$

Table 1. A context free grammar describing the language a^nb^n, i.e., the language of all sentences consisting of a number of as followed by the exact same number of bs.

the sequence aabb. The probabilistic extension of a context-free grammar, as in the given example, is a so called probabilistic or stochastic context-free grammar (PCFG). 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.

Attribute Grammars are an enhancement of context-free grammars, where each terminal and nonterminal can be assigned multiple inherited or synthesized attributes. An inherited attribute belongs to a symbol on the right-hand side of a rule which obtains its value from attributes of the nonterminal of the left-hand side or other symbols on the right-hand side. A synthesized attribute is an attribute of the nonterminal on the left-hand side of a rule whose value is computed using attributes of the right-hand side symbols. The above example for instance can be transformed into an attribute grammar containing the attributes `depth` and `max_depth`. The indices of A_1 and A_2 simply

$$
\begin{aligned}
&\text{START} \ \rightarrow \ \text{A} && (1.00) \\
&A_1 \ \rightarrow \ \text{ab} && (0.70) \\
&\qquad A_1.\texttt{depth} = 1 && \\
&\qquad \rightarrow \ \text{a}A_2\text{b} && (0.30) \\
&\qquad A_1.\texttt{depth} = A_2.\texttt{depth} + 1 && \\
&\qquad A_2.\texttt{max\_depth} = A_1.\texttt{max\_depth} - 1 && \\
&\qquad \texttt{assert} : A_1.\texttt{max\_depth} > 0 &&
\end{aligned}
$$

distinguish between the same nonterminal within a single rule. The synthesized attribute `depth` evaluates to the number of recursions that occurred during the production of a sentence, while the inherited attribute `max_depth` defines how many recursions are at most supposed to occur. The latter is achieved by defining the condition that the second rule is only chosen if A_1.`max_depth` > 0, resulting in sentences with at most `max_depth` number of as and bs. Such conditions extend the expressiveness of attribute grammars beyond the one of context-free grammars. For instance, the language $a^n\ b^n\ c^n$ can not be represented by context-free grammars. This language requires a context-sensitive grammar that encodes pre- and post-conditions within the context symbols. Alternatively, a simple counting attribute that keeps track of how many `c`'s have been produced can be added to a PCFG.

Grammar Inductions refers to the learning of formal grammars from sequences of terminals. Commonly the task is formulated as a search through a grammar space $\mathfrak{G}$, where the connections between grammars are represented as different operators, as illustrated in Figure 4. Such operators manipulate the set of

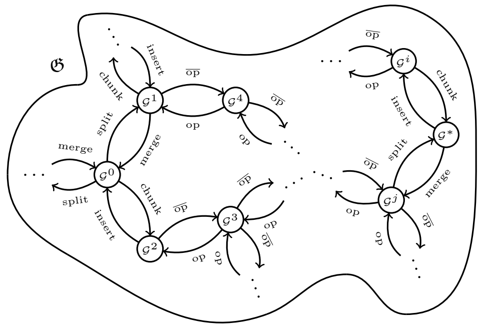


Figure 4. The grammar space $\mathfrak{G}$ contains all valid grammars $\mathcal{G}^0 \dots \mathcal{G}^*$. The space is traversed by applying operators $\text{op} \in \mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$ on the current grammar. For every operator op generating $\mathcal{G}'$ from $\mathcal{G}$, there exists an $\overline{\text{op}}$ that generates $\mathcal{G}$ from $\mathcal{G}'$, e.g., $\overline{\text{merge}} = \text{split}$, $\overline{\text{chunk}} = \text{insert}$.

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 (Stolcke 1994; Lee et al. 2013) and Markov chain Monte Carlo optimization (Talton et al. 2012).

When searching for concise yet general grammars, a common problem is the induction of overly general grammars. For instance, the language a^*b^* is a superset of the language a^nb^n. Hence, a grammar representing a^*b^* can explain every sequence in an observed data set produced by the language a^nb^n. Therefore, the corresponding grammar is a valid entry in the grammar space. This general limitation is known as Gold's Law and states that a correct grammar can not be learned from positive demonstrations alone (Gold 1967). However, the induction method proposed in this work decreases the chances of inducing an overly general grammar by defining a posterior distribution that takes the observed data stronger into account than other related methods.

Movement Primitive Representation

A Movement Primitive (MP) encapsulates a movement or action as a discrete entity. Although simple point attractors are also sometimes referred to as MPs, MPs usually represent the shape of the movement in addition to a start and goal position. Furthermore, MPs are commonly parameterized, allowing for the modification of a MP originally learned from demonstrations. Two examples of parameterized MPs are the well known Dynamical Movement Primitives (DMPs) (Ijspeert et al. 2013) and the more recent Probabilistic Movement Primitives (ProMPs) (Paraschos et al. 2017).

In this paper we choose Probabilistic Movement Primitives (ProMPs) as primitive representation (Paraschos et al. 2017)*. Each observed trajectory $\boldsymbol{\tau}$ is assumed to be sampled from the conditional distribution

$$p\left(\boldsymbol{\tau} \,|\boldsymbol{w}\right) = \prod_t \mathcal{N}\left(\boldsymbol{\tau}_t \,|\boldsymbol{\Phi}_t \boldsymbol{w}, \boldsymbol{\Sigma}_{\text{obs}}\right), \tag{1}$$

with $\boldsymbol{\Sigma}_{\text{obs}}$ being the observation noise. The feature matrix $\boldsymbol{\Phi}_t$ projects the the trajectory $\boldsymbol{\tau}$ onto a lower dimensional weight vector $\boldsymbol{w}$ for every time step t as defined in (Paraschos et al. 2017). The features $\boldsymbol{\Phi}_t$ are usually represented as radial basis functions for stroke like movements and von Mises functions for rhythmic movements. The weight $\boldsymbol{w}$ can be learned from the observed demonstration $\boldsymbol{\tau}$ by applying a maximum-a-posteriori optimization

$$\boldsymbol{w} = \arg\max_{\boldsymbol{w}'} p\left(\boldsymbol{\tau}|\boldsymbol{w}'\right) p\left(\boldsymbol{w}'\right), \tag{2}$$

with $p\left(\boldsymbol{w}'\right)$ denoting a prior over the weights. Depending on the prior choice the optimization yields different types of regressions. For instance choosing a standard normal prior $p\left(\boldsymbol{w}'\right) = \mathcal{N}\left(\boldsymbol{w}'\,|0, \mathbf{I}\right)$ yields a common ridge regression.

The projection of the demonstrated trajectories into lower dimensional spaces is a common property of movement primitives, e.g., in Dynamic Movement Primitives (Ijspeert et al. 2013). However, ProMPs additionally define a Gaussian distribution over the projected trajectories

$$\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{w}\,|\boldsymbol{\theta}\right), \boldsymbol{\theta} = \left(\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right), \tag{3}$$

with mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}$. Every primitive is characterized through its parameters $\boldsymbol{\theta}$ and the corresponding state space distribution is obtained by integrating out the weights

$$p\left(\boldsymbol{\tau}\,|\boldsymbol{\theta}\right) = \int_{\boldsymbol{w}} p\left(\boldsymbol{\tau}\,|\boldsymbol{w}\right) \mathcal{N}\left(\boldsymbol{w}\,|\boldsymbol{\theta}\right) \mathrm{d}\boldsymbol{w}, \tag{4}$$

$$= \prod_t \mathcal{N}\left(\boldsymbol{\tau}\,\middle|\,\boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Phi}_t\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Phi}_t^T + \boldsymbol{\Sigma}_{\text{obs}}\right) \tag{5}$$

For simplicity, the remaining paper will refer to $\boldsymbol{\theta}$ as the primitive itself instead of its parameters.

Problem Statement

Given a set of demonstrations $\mathcal{D} = \left\{\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_{|\mathcal{D}|}\right\}$ a set of primitives $\Theta = \left\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_{|\Theta|}\right\}$, each demonstration represents a labeled sequence of primitives $\boldsymbol{d}_i \in \Theta^+$. The goal of this work is to learn an Attribute grammar $\mathcal{G}^*_{\text{att}}$ that is concise and expressive yet has an easily comprehensible structure. For instance, given a set of demonstrations of turns in a game of tic-tac-toe, the following grammar is a concise representation of the possible sequences.

The learned grammar represents a generalized structure over the observed demonstrations and allows the sampling of new sequences while the attributes allow for the adaptation of individual primitives. For instance, the attribute stone contains the position of the stone which is supposed to be played next and the primitives,

*Note that the presented method is equally applicable when using other popular representations such as Dynamical Movement Primitives (Ijspeert et al. 2013), Gaussian Mixture Models (Calinon et al. 2007), and Gaussian Processes (Schneider and Ertel 2010).

$\mathcal{D} = \{$
(pick_far, close, place_right, open, home),
(pick_near, close, place_right, open, home),
⋮
(pick_far, close, place_left, open, home),
(pick_near, close, place_left, open, home)
$\}$

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

e.g., pick_near and pick_far can now be conditioned on the passed down position. In addition, attributes are introduced that ensure a smooth and continuous trajectory across each sampled primitive sequence despite the adaptation of individual primitives. The desired grammar is learned by inducing a PCFG $\mathcal{G}^*$ from the demonstrations $\mathcal{D}$ and enhancing it afterwards with a general attribute scheme for sequencing movement primitives $\mathcal{G}^* \underset{\text{att}}{\Longrightarrow} \mathcal{G}^*_{\text{att}}$. The set of terminals is defined as the set of primitives $\mathcal{A} = \Theta$ and 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 $\mathfrak{G}$, where 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 $\mathcal{G}'$ while the grammar space itself is explored via a Markov chain Monte Carlo optimization. In order to optimize for grammars with concise and comprehensible structures a novel prior based on Poisson distributions is introduced. The grammar induction and the prior are presented in later in the paper.

Given that the sequences produced by the grammar directly result in the movement of a robot it is important that there are no state jumps at the transition between two consecutive primitives. Throughout this paper this requirement is referred to as primitive connectibility. The connectibility of two primitives depends on the category of each primitive and the transition overlap between the primitives. The category of a primitive classifies which degrees of freedom are effectively controlled by the primitive. Primitives assigned to disjoint categories are not subject to the connectibility requirement. The transition overlap describes how much the end of one primitive and the beginning of the next primitive overlap. If the overlap is too small a smooth transition is unlikely and the connectibility requirement is violated. Next, we present a method for identifying the categories of primitive. Afterwards we introduce an approach to compute the transition overlap between two subsequent ProMPs.

Identifying the Primitive Category

Given a robotic platform with independent kinematic chains, e.g., an arm and a hand, each of these chains represents a category of movements. Such categories allow the relaxation of the connectibility requirement to hold only between primitives of the same category. Hence, the connectibility requirement of two subsequent primitives of the same category is independent of any primitive executed between them that is does not belong to that category. Given that some primitives might contain significant movement across categories, we treat the identification of the primitive category as a simple multi-label classification problem. The classification is based on the degrees of freedom that are active during the movement and assigns each primitive $\boldsymbol{\theta}$ to a category $\mathcal{C}$.

In this work each primitive is represented as a single ProMP. The distribution over the trajectory $\boldsymbol{\tau}$ given the parameters $\boldsymbol{\theta}$ is defined in Equation (4). Assuming equidistant time steps, the distribution

$$p\left(\dot{\boldsymbol{\tau}}\right) = \prod_t \mathcal{N}\left(\dot{\boldsymbol{\tau}} \,\middle|\, \dot{\boldsymbol{\Phi}}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \dot{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \dot{\boldsymbol{\Phi}}_t^T + \boldsymbol{\Sigma}_{\text{obs}}\right), \tag{6}$$

describes the velocity trajectories of the corresponding ProMP, with $\dot{\boldsymbol{\Phi}}_t$ being the first time derivative of the basis functions and $\boldsymbol{\Sigma}_{\text{obs}}$ denoting the observation noise with respect to the velocity trajectory. In order to identify the active degrees of freedom we analyze the mean and standard deviation of the velocity distribution. In particular, we are interested in the maximal absolute velocity over the time steps and define the maximum velocity feature as

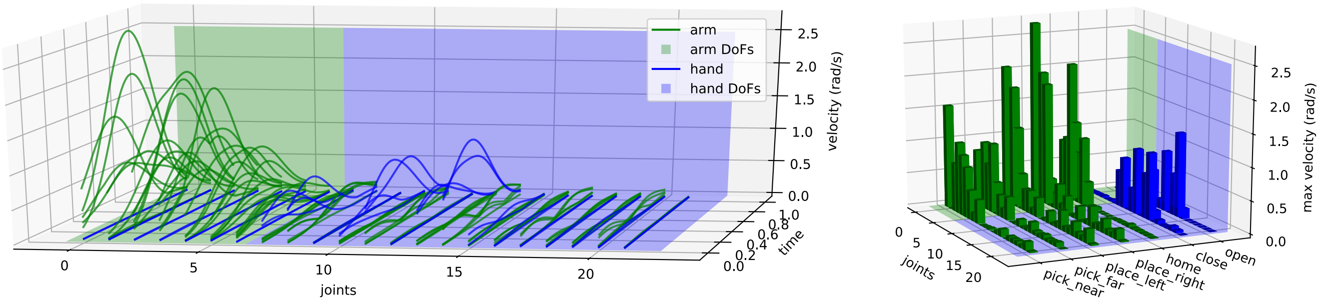
$$\boldsymbol{\psi}^{\text{vel}}\left(\boldsymbol{\theta}\right) = \max_t \left(\left|\dot{\boldsymbol{\Phi}}_t \boldsymbol{\mu}_{\boldsymbol{w}}\right| + 2\sqrt{\operatorname{diag}\left(\dot{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \dot{\boldsymbol{\Phi}}_t^T + \boldsymbol{\Sigma}_{\text{obs}}\right)} \right). \tag{7}$$

A primitive is considered active with respect to the category $\mathcal{C}_i$ if and only if any of the corresponding elements in $\boldsymbol{\psi}^{\text{vel}}\left(\boldsymbol{\theta}\right)$ is above the threshold ϵ_i, i.e.,

$$\boldsymbol{\theta} \in \mathcal{C}_i \iff \bigvee_{d \in \text{dof}(\mathcal{C}_i)} \psi_d^{\text{vel}}\left(\boldsymbol{\theta}\right) > \epsilon_i, \tag{8}$$

with $\text{dof}(\mathcal{C}_i)$ being the set of degrees of freedom associated with category $\mathcal{C}_i$ and $\psi_d^{\text{vel}}\left(\boldsymbol{\theta}\right)$ is the maximal velocity of the d^{th} degree of freedom. The threshold ϵ_i is chosen manually.

In the tic-tac-toe task, two different categories are distinguished, arm movements and hand movements. If two subsequent hand movements are connectible it does not matter how many arm movements are sequenced in between them. The connectibility requirement is still fulfilled. At the same time hand movements can now be executed at arm configurations at which the hand movement has not been observed in the demonstrations. Figure 5 shows the velocity trajectories of the seven



(a) velocity trajectories of the tic-tac-toe ProMPs

(b) maximal velocity of the tic-tac-toe ProMPs

Figure 5. (a) shows the |mean| + 2 std of the velocity distribution defined in Equation (6) for each ProMP, while (b) shows the maximal velocities as computed in Equation (7). The colors indicate which categories were assigned to each ProMP and background colors highlight to which category each joint belongs.

primitives used in the tic-tac-toe task where the colors of the trajectories indicate the identified category and the background color highlights the degrees of freedom associated with each category. Given the demonstrations, every primitive was assigned exactly one category, even though the described approach allows the association of multiple categories to a single primitive. While the given example could also have been solved by a simple annotation of the observed data, examples can be thought of where the presented automated annotation is of great benefit. For instance, a significantly larger set of observed sequences or tasks that require multiple interacting kinematic chains, e.g., a bi-manual task.

Determining Connectibility of Primitives

In this section we discuss how to automatically determine if two ProMPs are connectible, that is if two subsequent ProMPs would result in a jump in the state space or not. Given that ProMPs are a probabilistic trajectory representation it seems fitting to use probabilistic similarity measures, such as the Kullback-Leibler divergence or the Hellinger distance to test if two ProMPs are connectible. However, considering that avoiding jumps in the state space is a spatial requirement probabilistic similarity measures can be misleading. We solve the connectibility problem in the spatial domain by treating each ProMP as a multidimensional tube. At every time step t a ProMP $\boldsymbol{\theta}_i$ can be approximated by a hyper ellipsoid

$$\mathcal{E}_t(\boldsymbol{\theta}_i) = \left(\boldsymbol{c}_t = \boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{w}},\ \boldsymbol{S}_t = \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T + \boldsymbol{\Sigma}_{\text{obs}}\right)$$

with center $\boldsymbol{c}_t$ and shape matrix $\boldsymbol{S}_t$. Two ProMPs, $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$, are now considered connectible if the transition overlap between their corresponding tubes is larger than a predefined threshold

$$\boldsymbol{\theta}_i \underset{\texttt{con}}{\longrightarrow} \boldsymbol{\theta}_j \iff \texttt{overlap}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \geq \epsilon_{\texttt{overlab}}.$$

The transition overlap from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_j$ is defined over the last ellipsoid of the preceding ProMP $\mathcal{E}_T(\boldsymbol{\theta}_i)$ and the first ellipsoid of the succeeding ProMP $\mathcal{E}_1(\boldsymbol{\theta}_{i+1})$

$$\texttt{overlap}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \frac{\texttt{vol}\left(\mathcal{E}_T\left(\boldsymbol{\theta}_i\right) \cap \mathcal{E}_1\left(\boldsymbol{\theta}_j\right)\right)}{\texttt{vol}\left(\mathcal{E}_T\left(\boldsymbol{\theta}_i\right)\right)},$$

with $\texttt{vol}$ denoting the volume and $\mathcal{E}_T(\boldsymbol{\theta}_i)$ and $\mathcal{E}_1(\boldsymbol{\theta}_j)$ being the last ellipsoid of $\boldsymbol{\theta}_i$ and the first primitive of $\boldsymbol{\theta}_j$ respectively. Hence, the transition overlap describes the percentage of the end of $\boldsymbol{\theta}_i$ that is covered at the beginning of $\boldsymbol{\theta}_j$.

Unfortunately computing the volume of the intersection between two hyper ellipsoids is considered #P-complete (Bringmann and Friedrich 2010). We circumvent this problem by computing the overlap for each degree of freedom independently and choosing the minimal value as an approximation of the ellipsoidal overlap. As discussed in the previous section the connectibility between two ProMPs is only considered if both ProMPs share at least one primitive category $\mathcal{C}$

$$\texttt{overlap}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \approx \min_{d \in \texttt{dof}(\mathcal{C})} \frac{|\texttt{intersec}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, d)|}{|c_{i,T,d} \pm n\ s_{i,T,d}|},$$

$$\texttt{intersec}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, d) = c_{i,T,d} \pm n\ s_{i,T,d} \cap c_{j,1,d} \pm n\ s_{j,1,d},$$

where $c_{i,T,d}$ and $s_{i,T,d}$ are the d^{th} element of the ellipsoid center and the d^{th} standard deviation for primitive $\boldsymbol{\theta}_i$ at time step T. The constant n decides how many standard deviations wide the considered interval will be.

The threshold $\epsilon_{\texttt{overlab}}$ can be defined either manually or derived from observations. Given that the learned grammar should at least be capable to reproduce the initially given observations all ProMPs that were pairwise connected in the observations have to be considered connectible. Therefore, the threshold

$$\epsilon_{\texttt{overlab}} = \alpha \min_{(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \in \texttt{pairs}(\mathcal{D})} \texttt{overlap}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j),$$

is defined as a percentage α of the minimal overlap value of all ProMPs that were connected in the observations. $\texttt{pairs}(\mathcal{D})$ is a function that returns all consecutive primitive pairs in the set of demonstrations $\mathcal{D}$. We can now define two sets for each primitive. One set contains all primitives it is connectible to $\overrightarrow{\texttt{Con}}(\boldsymbol{\theta})$ and the other

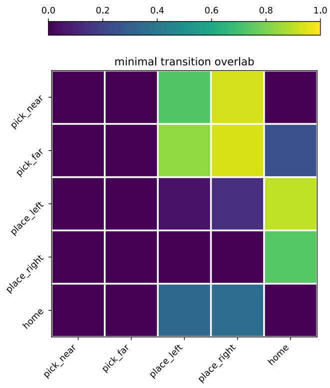


Figure 6. The transition overlap for each arm primitives of the tic-tac-toe task. A value above the threshold $\epsilon_{\texttt{overlab}} = 0.69$ signifies that the primitive at that row is connectible to the primitive of that column. The ellipsoids were $n = 2$ standard deviations wide and the threshold was determined from the observations with $\alpha = 0.95$.

set contains all primitives it is connectible from $\overleftarrow{\mathtt{Con}}(\boldsymbol{\theta})$

$$\overrightarrow{\mathtt{Con}}(\boldsymbol{\theta}_i) = \left\{ \boldsymbol{\theta}_j \middle| \boldsymbol{\theta}_j \in \Theta \wedge \boldsymbol{\theta}_i \xrightarrow[\mathtt{con}]{} \boldsymbol{\theta}_j \right\},$$
$$\overleftarrow{\mathtt{Con}}(\boldsymbol{\theta}_i) = \left\{ \boldsymbol{\theta}_j \middle| \boldsymbol{\theta}_j \in \Theta \wedge \boldsymbol{\theta}_j \xrightarrow[\mathtt{con}]{} \boldsymbol{\theta}_i \right\}.$$

Figure 6 illustrates the transition overlap between each pair of primitives of the tic-tac-toe task arranged in and adjacency like matrix, where each row and column indicate which primitives belong into $\overrightarrow{\mathtt{Con}}(\boldsymbol{\theta})$ and $\overleftarrow{\mathtt{Con}}(\boldsymbol{\theta})$ respectively.

Deciding the connectibility of two primitives using their overlap rather than purely basing it on the observations has the significant advantage of allowing to connect two primitives which might not have been observed connected during the demonstrations but are nevertheless safely connectible.

Inducing PCFGs for Movement Primitives

In this section we introduce an grammar induction approach where the grammar search is defined as a maximum-a-posteriori problem and the grammar space is traversed using a Markov chain Monte Carlo approach. We introduce a novel prior over the grammar structure based on three Poisson distributions allowing to define a desired grammar structure in more detail than common grammar priors. Furthermore, we discuss problems of common grammar priors and the advantages of the presented prior. We present four operators that allow the traversal of the grammar space and define distributions over each given a grammar. The proposal distribution of the MCMC approach is defined as a mixture over the operator distributions.

Learning grammars through posterior optimization

The posterior $p(\mathcal{G}|\mathcal{D})$ 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

$$\mathcal{G}^* = \arg\max_{\mathcal{G}} p(\mathcal{G}|\mathcal{D}) = \arg\max_{\mathcal{G}} p(\mathcal{D}|\,\mathcal{G})\, p(\mathcal{G}), \quad (9)$$

where $p(\mathcal{D}|\,\mathcal{G})$ is the likelihood of the labeled demonstrations $\mathcal{D}$ given the grammar $\mathcal{G}$. The likelihood will be presented in the next section. Afterwards we discuss common choices for the prior $p(\mathcal{G})$ and finally we introduce a novel grammar prior based on Poisson distributions.

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

$$p(\mathcal{D}|\,\mathcal{G}) = \prod_{\boldsymbol{d}\in\mathcal{D}} p(\boldsymbol{d}|\,\mathcal{G}). \quad (10)$$

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

$$p(\boldsymbol{d}\,|\mathcal{G}) = \sum_{\mathtt{t}\in\mathtt{T}(\boldsymbol{d},\mathcal{G})} \prod_{(A,r,\rho)\in\mathtt{t}} \rho,$$

where $\mathtt{t}$ represents a single parse tree and $\mathtt{T}(\boldsymbol{d},\mathcal{G})$ denotes a function producing all feasible parse trees. The 3-tuple (A, r, ρ) represents an edge in the parse tree $\mathtt{t}$ 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 $\mathtt{T}$ creating all possible parse trees for a given demonstration $\boldsymbol{d}$, is implemented by the Earley parser (Earley 1983). 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.

The grammar prior $p(\mathcal{G})$ is commonly modeled as a joint distribution over the grammar probabilities $\boldsymbol{\rho}_{\mathcal{G}} = \{\boldsymbol{\rho}_A | A \in \mathcal{V}\}$ and the grammar structure $\mathcal{G}_{\mathcal{R}} = \{(A, \mathrm{R}_A) | A \in \mathcal{V}\}$ (Stolcke 1994; Kitani et al. 2008; Talton et al. 2012; Lee et al. 2013),

$$p(\mathcal{G}) = p\left(\boldsymbol{\rho}_{\mathcal{G}}\middle|\,\mathcal{G}_{\mathcal{R}}\right) p(\mathcal{G}_{\mathcal{R}}). \quad (11)$$

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 $A \in \mathcal{V}$,

$$p\left(\boldsymbol{\rho}_{\mathcal{G}}\middle|\,\mathcal{G}_{\mathcal{R}}\right) = \prod_{\boldsymbol{\rho}_A\in\boldsymbol{\rho}_{\mathcal{G}}} p(\boldsymbol{\rho}_A). \quad (12)$$

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 (Kitani et al. 2008; Lee et al. 2013). 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 (Talton et al. 2012) 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.

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

$$p\left(\boldsymbol{\rho}_{\mathcal{G}}, \mathcal{G}_{\mathcal{R}}\right) = \frac{p\left(|\mathcal{R}|\,|\eta_{\mathcal{R}}\right)}{|\mathcal{R}|} \sum_{(A,\mathrm{R}_A,\boldsymbol{\rho}_A)\in\mathcal{R}} \gamma\left(A, \mathrm{R}_A, \boldsymbol{\rho}_A\right) \tag{13}$$

$$\gamma\left(A, \mathrm{R}_A, \boldsymbol{\rho}_A\right) = p\left(|\mathrm{R}_A|\,|\eta_{\mathrm{R}}\right) p\left(\mathrm{R}_A|\boldsymbol{\rho}_A, \eta_r\right), \tag{14}$$

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

$$p\left(\mathrm{R}_A|\boldsymbol{\rho}_A, \eta_r\right) = \sum_{r\in\mathrm{R}_A,\rho\in\boldsymbol{\rho}_A} \rho\; p\left(|r||\eta_r\right), \tag{15}$$

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 hyper-parameters $\eta_{\mathrm{R}}, \eta_r$ 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.

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 4. After 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 (Baker 1979).

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 connectibility 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 connectibility requirement any grammar $\mathcal{G}'$ resulting from an application of op on $\mathcal{G}$ also fulfills the requirement. We incorporate the connectibility requirement into the definition of the two common operators merge and split.

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.

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 connectibility requirement. We avoid this problem by restricting the domain Ω_{merge} to only contain compatible nonterminal pairs. Assuming the sets $\texttt{first}(A)$ and $\texttt{last}(A)$ contain all possible primitives that could be at first or last position of any sequence produced starting from A. Two nonterminals A_j and A_k are now considered compatible if

$$\bigcup_{\boldsymbol{\theta}\in\texttt{first}(A_j)} \overleftarrow{\texttt{Con}}(\boldsymbol{\theta}) = \bigcup_{\boldsymbol{\theta}\in\texttt{first}(A_k)} \overleftarrow{\texttt{Con}}(\boldsymbol{\theta})$$
and
$$\bigcup_{\boldsymbol{\theta}\in\texttt{last}(A_j)} \overrightarrow{\texttt{Con}}(\boldsymbol{\theta}) = \bigcup_{\boldsymbol{\theta}\in\texttt{last}(A_k)} \overrightarrow{\texttt{Con}}(\boldsymbol{\theta}).$$

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 (Stolcke 1994) and define the new insert operator that negates the effects of chunk.

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

insert selects a nonterminal $A \in \Omega_{\text{insert}}$ 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 $\mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$. 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 $\mathcal{O} \setminus \{\text{op}\}$ that is able to produce $\mathcal{G}'$ from $\mathcal{G}$.

Finding $\mathcal{G}^*$

Similarly to (Talton et al. 2012) we search for the optimal grammar $\mathcal{G}^* = \arg\max_{\mathcal{G}} p(\mathcal{G}|\mathcal{D})$ using Markov chain Monte Carlo (MCMC) optimization. A main advantage of MCMC over local search methods is that it's stochastic exploration traverses the grammar space better than local search methods. Given the definition of the grammar score the corresponding landscape is highly multimodal. Often several operators that each lead to a lower scoring grammar are required to be executed sequentially in order to arrive at a new maximum. Even with a broad beam width, beam search often fails to surpass such valleys whereas MCMC due to it's stochasticity manages to reach at least better local optima and even offers theoretical guarantees to find the global optimum in the limit.

In (Talton et al. 2012) 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 (Stolcke 1994). The insert operator ensures the irreducibility of the Markov chain. Analogously to (Talton et al. 2012), we apply the Metropolis Hastings algorithm. However, since (Talton et al. 2012) 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 $\mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$,

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

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

$$p\left(\text{op}' \middle| \mathcal{G}, \eta_{\text{op}'}\right) = \frac{\eta_{\text{op}'}\left(1 - \delta_{|\Omega_{\text{op}'}|}\right)}{\sum_{\text{op}\in\mathcal{O}'} \eta_{\text{op}}\left(1 - \delta_{|\Omega_{\text{op}}|}\right)} \qquad (16)$$

where $\eta_{\text{op}} \in \mathbb{R}$ is a weighting for the operator op, $\delta_{|\Omega_{\text{op}}|}$ 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 $q_{\text{op}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right)$ should not contribute any probability mass if $\text{op} \neq \text{op}'$. This restriction is achieved by the Kronecker deltas $\delta_{\text{op}',\text{op}}$ in the following mixture components.

$q_{\text{split}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right)$ 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 $A_i \in \Omega_{\text{split}}$ is $1/|\Omega_{\text{split}}|$. 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. There are exactly $2^{|\mathrm{R}_{A_i}|} - 2$ possibilities of assigning the productions to either R_{A_j} or R_{A_k}. Finally, the N_{A_i} occurrences of A_i across all productions in $\mathcal{R}$ have been replaced by A_j or A_k in $\mathcal{R}'$. The chosen replacements have been one out of a total of $2^{|N_{A_i}|} - 2$ possibilities, considering that A_j and A_k had to be chosen at least once. Combining the possibilities for assigning the productions and for assigning the occurrences results in redundancies, since there are always two combinations that will result in the same $\mathcal{R}'$. The overall probability of $\mathcal{G}'$ being produced from $\mathcal{G}$ by using a split operator is given as

$$q_{\text{split}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right) = \frac{\delta_{\text{op}',\text{split}}}{|\Omega_{\text{split}}|} \frac{2}{(2^{|\mathrm{R}_{A_i}|} - 2)(2^{|N_{A_i}|} - 2)}. \qquad (17)$$

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

$$q_{\text{merge}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right) = \frac{\delta_{\text{op}',\text{merge}}}{|\Omega_{\text{merge}}|}. \qquad (18)$$

$q_{\text{chunk}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right)$ 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 is

$$q_{\text{chunk}}\left(\mathcal{G}' \middle| \mathcal{G}, \text{op}'\right) = \frac{\delta_{\text{op}',\text{chunk}}}{|\Omega_{\text{chunk}}|}. \qquad (19)$$

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

$$q_{\text{insert}}\left(\mathcal{G}'\middle|\,\mathcal{G},\text{op}'\right)=\frac{\delta_{\text{op}',\text{insert}}}{|\Omega_{\text{insert}}|}. \tag{20}$$

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

$$\text{acc}\left(\mathcal{G}',\text{op}'\middle|\,\mathcal{G}\right)=\min\left(1,\frac{p\left(\mathcal{G}'|\mathcal{D}\right)^{1/T}q\left(\mathcal{G},\overline{\text{op}'}\middle|\,\mathcal{G}'\right)}{p\left(\mathcal{G}|\mathcal{D}\right)^{1/T}q\left(\mathcal{G}',\text{op}'\middle|\,\mathcal{G}\right)}\right), \tag{21}$$

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. For instance, Table 3 shows a grammar induced by the presented method given sequences of the previously described tic-tac-toe task. The semantically meaningful names of the nonterminals were chosen manually.

Given that the MCMC optimization finds high scoring grammar after only few iterations, the hyper-parameter optimization is inexpensive. Furthermore, a good rule of thumb for the number of productions per nonterminal and the number of symbols per production are 2 and 3 respectively, leaving the number of nonterminals the only free parameter of the presented prior.

Enhancing PCFGs with Attributes for Movement Primitive Sequencing

So far, the presented approach induces grammars, that do not violate the connectibility requirement. However, connectibility as defined in this work only guarantees that the transition area of two consecutive primitives is large enough to produce a continuous state space trajectory. In order to ensure smooth trajectories the start of the subsequent primitive has to be conditioned to the end of the current primitive. This can be achieved within the grammar formulation by introducing attributes. Furthermore, attributes can be used for defining points of interest that primitives need to reach for a successful execution. We introduce an evaluation scheme for movement primitive sequencing tasks that enhance given probabilistic context-free grammars with attributes and conditions. The scheme generalizes to different movement primitive sequencing tasks and, therefore, needs only little to no adaptation for specific tasks, with the exception of the initialization of the task-specific attribute values.

We define the following three attributes, common to primitive sequencing tasks:

- `transition` This attribute defines where the current primitive ends and the next primitive is supposed to start. It is solely defined for nonterminals, and ensures that the produced primitives result in a continuous state space trajectory.

- `endpoint` The endpoint of a movement primitive. It is solely defined for terminals and after the terminal has been evaluated, the `transition` attribute of the left hand side nonterminal is set to the `endpoint` of the corresponding primitive.

- `viapoints` An ordered list of points that are supposed to be traversed by the sequence of primitives. The points are given in the state-space of the primitives. Once the first point is traversed by a primitive it is removed from the list and the next point is considered.

In addition to the attributes we define two conditions for necessary for the evaluation scheme. If preceded with an assert these conditions have to be satisfied for a successful evaluation.

- reachable Given a primitive and a point in the primitive state-space, this condition is satisfied if the point is reachable by the primitive. In this work we use probabilistic movement primitives over the joint configuration of the robot. A point given in the configuration of the robot is reachable by a particular primitive if it is within two times the standard deviation of the trajectory mean of the movement primitive.

- producible Given a nonterminal this condition is satisfied if at least one of the corresponding right-hand sides is producible. A right-hand side is considered producible if all mandatory conditions are satisfied, given the current set of attributes.

The described attributes enhance context-free grammars for movement primitive sequencing tasks, such that the sequenced primitives can be conditioned to state of the environment, e.g., the pose of an object. The conditions ensure a continuous state space trajectory of the sequenced primitives, even in the case of primitive adaptations.

Evaluation Scheme for the Tic-Tac-Toe Task

We explain the functionality of the attributes in detail using the example of the tic-tac-toe task. We start with the probabilistic context-free grammar shown in Table 3. The grammar was induced from demonstrations as described in the previous section.

The production of the sequence always begins at the START nonterminal. We assign two points to the `viapoints` attribute. One for the position of a stone and one for the field the stone is supposed to be placed on. Furthermore, we set the `transition` attribute to the current position of the robot in the primitive state-space. We use the literals stone_pos, field_pos and cur_pos instead of the actual numerical values, where assert indicates that this condition must be satisfied otherwise the entire right-hand side is removed from consideration as a possible production of the corresponding nonterminal given the current attribute set. If the START nonterminal is not producible,

START.transition = cur_pos
START.viapoints = [stone_pos,field_pos]
assert: producible(START)
assert: empty(START.viapoints)

START → MOVE (1.00)
MOVE.viapoints = START.viapoints
MOVE.transition = START.transition
assert: producible(MOVE)
START.transition = MOVE.transition
START.viapoints = MOVE.viapoints

the task is not solvable under the given attributes. Furthermore, if the `viapoints`-list is not empty after evaluating START not all points were traversed and the task is not considered solved.

An important convention in the attribute notation is that whenever a nonterminal appears as an argument of a condition or on the right-hand side of an assignment it has been evaluated before. For instance, the producibility of START and MOVE can only be asserted once the respective nonterminal has been fully evaluated.

The MOVE nonterminal contains multiple productions, each consisting of multiple symbols. The productions can be evaluated in parallel, i.e., the evaluation of each of the productions begins with the same set of attributes, independent of the changes that have occurred during the evaluation of the other productions. In contrast, the symbols of a single production are evaluated sequentially, i.e., every symbol begins with the attributes set after the evaluation of the previous symbol. As mentioned before, terminals represent single movement primitives. It is important that a sequence of primitives does not contain any jumps in the state-space, since a real robot platform will not be able to make significant changes in its configuration instantaneously. Therefore, we ensure that every selected primitive starts where the previous primitive ended. In the proposed evaluation scheme this is achieved by ensuring that the reachable condition holds for the primitive and the current `transition`-point. If the primitive can start from the `transition`-point, the `transition` attribute is set to the `endpoint` of the primitive afterwards. Furthermore, we define a function to traverse the `viapoint`-list.

- traverse The function expects a terminal and a list of points. If the first point in the list is reachable by the terminal the corresponding primitive will traverse the point, the point will be removed from the list and the function evaluates to true.

Given that the possible adaptation of the primitive to the point could change the endpoint, traverse has to be evaluated before the `transition`-point is adapted.
Only the evaluation for one of the two productions is shown. The evaluation of the other production is defined analogously, but with the terminal pick_far instead of pick_near. Despite that both of the productions next evaluate the TO nonterminal, the actual evaluations might differ due to two different sets of attribute values.

MOVE → pick_near TO (0.40)
assert: reachable(pick_near,MOVE.transition)
traverse(pick_near, MOVE.viapoints)
MOVE.transition = pick_near.endpoint
TO.viapoints = MOVE.viapoints
TO.transition = MOVE.transition
assert: producible(TO)
MOVE.viapoints = TO.viapoints
MOVE.transition = TO.transition

TO → LEFT home (0.47)
LEFT.viapoints = TO.viapoints
LEFT.transition = TO.transition
assert: producible(LEFT)
TO.viapoints = LEFT.viapoints
TO.transition = LEFT.transition
assert: reachable(pick_near, TO.transition)
traverse(pick_near, TO.viapoints)
TO.transition = pick_near.endpoint

Again two different possible productions are evaluated in parallel but only one is shown. The evaluation of the other production is defined equivalently, but with the nonterminal RIGHT instead of the LEFT. In contrast to the evaluation of MOVE the productions of TO require the evaluation of a nonterminal before the evaluation of a terminal.

A General Evaluation Scheme for Sequencing Tasks

A structure for both terminal and nonterminal evaluations is clearly evident. Every terminal a on the production of a rule with nonterminal A on the left-hand side is evaluated using the statements

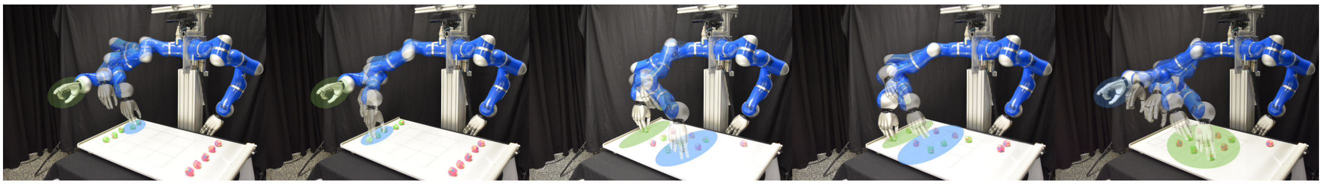
assert: reachable(a, A.transition)
traverse(a, A.viapoints)
A.transition = a.endpoint

and every nonterminal B on the right-hand side of a rule with nonterminal A on the left-hand side is evaluated using

B.viapoints = A.viapoints
B.transition = A.transition
assert: producible(B)
A.viapoints = B.viapoints
A.transition = B.transition

The presented evaluation scheme is very general and can be applied to any movement primitive sequencing task. Using not further specified via-points has the advantage that the evaluation does not restrict which primitive traverses which point. For instance, in the case of an obstacle it might be sufficient that the obstacle is passed at some point, but it does not necessarily matter which primitive avoids it. However, the unspecified list of via-points has a significant disadvantage. A primitive might require a certain via-point, for instance pick_near and pick_far have to know where the stone is positioned in order to pick it up successfully. Nothing in the current scheme associates via-points with a certain primitives. We solve this problem by introducing two additional attributes.

- `keywords` An unordered list of keywords. This attribute is assigned only to terminals before the



pick_near pick_far place_left place_right home

Figure 7. The five arm primitives used in the sequences, representing turns in the tic-tac-toe game. While both pick_near and pick_far approach a stone from the home position they differ in the stone positions they can reach. Similarly the primitives place_left and place_right position the stone in different areas of the playing field.

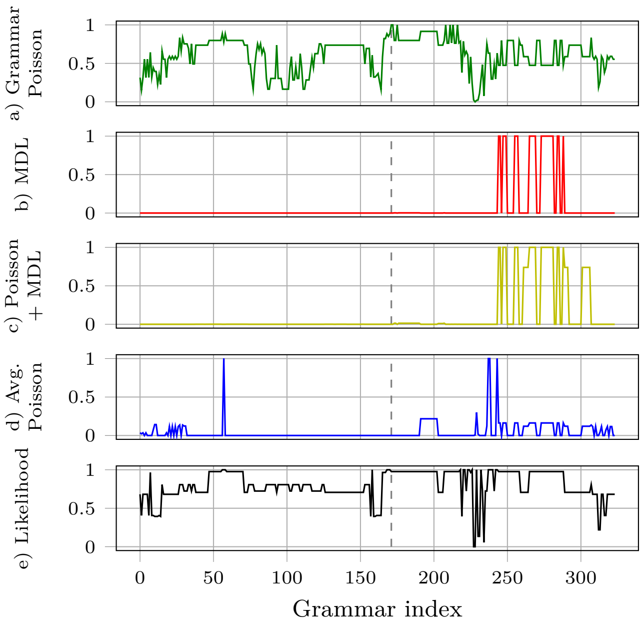


Figure 8. The posteriors and the likelihood for the tic-tac-toe turn grammar. The vertical, dashed line indicates the index of the highest posterior (171), given the presented Poisson prior.

evaluation and contains keywords identifying relevant points in the `targets` attribute.

- `targets` A dictionary that maps keywords to ordered lists of points. The points are defined in the primitive state-space. A primitive containing a matching keyword in its `keywords` attribute extracts the first point in the corresponding list.

The evaluation scheme for terminals is now defined as

```
assert: reachable(a, A.transition)
for: key in a.keywords
    assert: key in A.targets
    assert: traverse(a, A.target[key])
traverse(a, A.viapoints)
A.transition = a.endpoint
```

We introduce the for: notation to indicate an iteration and the in notation to indicate the existence of an element in a list. The `targets` attribute can strongly influence the production of a sequence. The given target could be outside of the distribution of the primitive associated with the terminal. For instance, both terminals pick_near and pick_far have a stone keyword. If the `targets` attribute associates stone with a value outside of the pick_near primitive but within the pick_far primitive, the assert statement would only hold for pick_far, ensuring that every sequence produced with this set of targets will contain a pick_far and never a pick_near. This way the target attributes directly influence the effective structure of the grammar.

The evaluation scheme for nonterminals only changes such that the `targets` attribute is additionally passed down and received afterwards, analogously to the `viapoints` attribute.

Evaluating Parallel Attribute Sets

We already established that the right-hand sides of a single nonterminal are evaluated in parallel. If more than one right-hand side does not violate any asserts, multiple parallel sets of attributes return from that nonterminal evaluation. Given that within one right-hand side the attributes are passed sequentially from symbol to symbol, the question arises which of the multiple attribute sets should to be considered. A naive approach would be to select a random attribute set. However, one attribute set might result in an unproducible right-hand side while another might not. We address this problem by storing every attribute set corresponding to a producible right-hand side in an ordered list. The order is defined randomly, while being weighted with the probabilities of the right-hand sides. Only the first set of attributes is considered, unless the set results in an assert violation, then the attribute set is discarded and the evaluation continues with the next set in the list. If no sets are left, the right-hand side is considered unproducible. It is possible that a given set of `targets` results in an effective grammar structure that is not capable of producing any sequence of primitives. For instance, neither place_left nor place_right are able to place the stone outside of the playing field. Hence, if the corresponding target is set outside the playing field neither of the to productions of the TO terminal will be producible and the non-producibility will be propagated up until the start symbol. In this case the grammar would return an empty sequence. This can easily be used to prompt the user that the current grammar can not produce a sequence satisfying the given set of targets. Therefore, different targets or new demonstrations extending the grammar are required.

The presented attributes and evaluation scheme are independent of the actual task itself and generalize

START	→	DEMO1 (0.33) | DEMO2		(0.20)
		DEMO3 (0.27) | DEMO4		(0.20)
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)

Table 2. Initial grammar. Grammar index 0 in Figure 8

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)

Table 3. Grammar with the highest posterior after 400 iterations of the MCMC optimization. Grammar index 171 in Figure 8

over movement primitive sequencing tasks. The only attribute that has to be accessed and potentially adapted by the user are the `targets`. Hence, the remaining attributes and the evaluation scheme itself can be considered constants and can be hidden from the user, concealing necessary complexity that does not affect the representation of the behavior. We further simplify the presentation of the attribute grammar, by presenting the keywords of the `targets` attribute as grammar attributes themselves. By applying these simplifications we arrive at the attribute grammar as presented initially in the problem statement.

Experiments

We evaluated the proposed approach on several 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 (Paraschos et al. 2017). 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 (Talton et al. 2012). The Poisson + MDL prior weights the description language for every production with the Poisson probability over the length of the production (Kitani et al. 2008). Finally, the Avg. Poisson prior discards the MDL completely and is solely represented by a Poisson distribution over the average length of all productions (Lee et al. 2013). 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.

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 the demonstrations and, hence, the learning of the primitives was done beforehand via Probabilistic Segmentation (Lioutikov et al. 2017). The five resulting arm primitives are shown in Figure 7, where the green and blue highlighted areas mark the start and end of the end-effector. While pick_near and pick_far are semantically similar, they actually differ quite substantially in the encoded joint trajectory of the robot and, hence, the segmentation algorithm separated those movements into two separate primitives. The same explanation holds for place_left and place_right.

The grammar learning was initialized with 15 observations of four unique sequences, each consisting of five terminals. The initial grammar is shown in Table 2. We initialized our approach with a desired number of rules $\eta_{\mathcal{R}} = 5$, 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 8 and the grammar with the highest posterior, grammar index 171 is shown in Table 3. 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 9. The parse tree includes keys and values assigned to the `keywords` and `targets` attributes. The production of the sequence was started with the attribute $\texttt{targets} = \{\mathsf{stone} : \text{stone\_pos}, \mathsf{field} : \text{field\_pos}\}$ consisting of the position of the stone that should be played next, stone_pos and the field position field_pos on which the stone should be placed. For simplicity, the parse tree presented to the user replaces the actual position of stone_pos and field_pos but instead the numbering of the corresponding playing field cell.

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 8b-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 behavior 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,

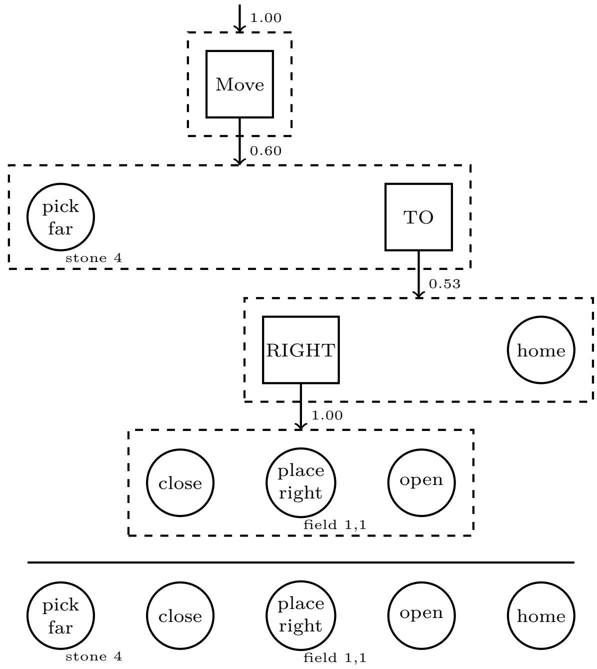


Figure 9. A parse tree of a sequence produced by the learned grammar for tic-tac-toe turns. Nonterminals are presented as squares and terminals as circles. A dashed rectangle represents the production chosen by the parent terminal with the probability next to the connecting arrow. The solid line separates the final sequence from the producing parse tree. The grammar was enhanced with the presented attributes and evaluation scheme, where `stone` and `field` are two keys assigned to the `keywords` attributes of the pick_far and place_right terminals respectively.

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 (10) 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).

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 10, 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 11. 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 Table 4 and three produced sequences are shown in Table 5. 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.

Conclusion

In this work, we have introduced attribute grammars as a mechanism to sequence movement primitives. We have shown how to identify the categories of movement primitives and how to determine if two probabilistic movement primitives are connectible or not. The presented categorization approach is simple yet efficient, however, in future work we want to investigate more sophisticated approaches for the clustering of parameterized time series such as the applied movement primitives. Furthermore, we have presented an approach that induces probabilistic context-free grammars from flat sequences of movement primitive samples, i.e., no hierarchy in the observations, while taking advantage of a stochastic primitive representation. The novel 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 hyper-parameters of the prior have a clear semantic interpretation, namely the number of productions for each nonterminal and the average length of each production. The posterior is learned using a Markov chain Monte Carlo optimization where the proposal distribution is formulated as a mixture

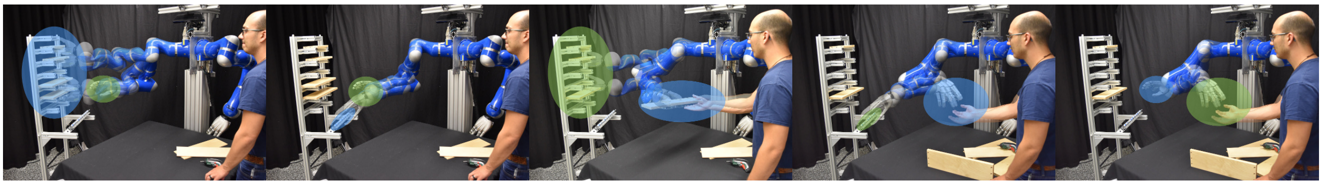


Figure 10. The arm primitives of the box assembly task. The robot applies different primitives for grasping a board, take_board, or picking a screw, take_screw. Similarly the handover for boards and screws is encode in different primitives.

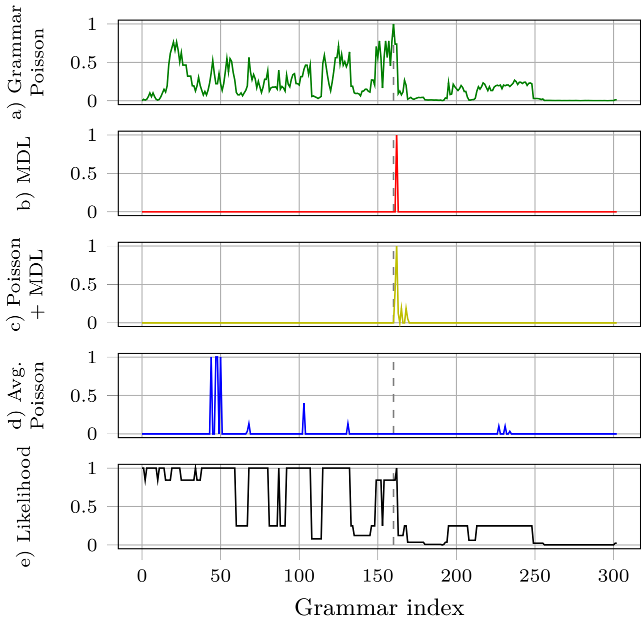


Figure 11. The posteriors and the likelihood for the box assembly task. The vertical, dashed line indicates the index of the highest posterior (160), given the presented Poisson prior.

START	→	ASSEMBLE_SB	(0.5)
	→	ASSEMBLE_BB	(0.5)
BOARD	→	take_board GIVE_B home	(1.0)
SCREW	→	take_screw GIVE_S home	(1.0)
BSB	→	BOARD SCREW BOARD	(1.0)
SBS	→	SCREW BOARD SCREW	(1.0)
GIVE_S	→	close_screw give_screw open_screw	(1.0)
ASSEMBLE_BB	→	BOARD BOARD SBS ... BOARD SCREW SCREW	(1.0)
GIVE_B	→	close_board give_board open_board	(1.0)
ASSEMBLE_SB	→	SBS BOARD SCREW BSB	(0.5)
	→	BOARD SBS BOARD SBS	(0.5)

Table 4. The grammar with the highest posterior for the box assembly task after 400 iterations of the MCMC optimization

model over four operators. We defined attributes and conditions of a general evaluation scheme for sequencing tasks. We enhanced an initially induced probabilistic context-free grammar for making a move in a game of tic-tac-toe with the defined attributes and the evaluation scheme. While the MCMC optimization is less likely to get stuck in local optima than other suggested search strategies, such as beam search, it is not without fault. Depending on the complexity of the task with respect to the length of the observed sequences and the number of terminals, a significant number of samples are required to reach a promising area of the search space. Given that, the actual interest of grammar induction is not the exploration of the posterior, but rather the finding of the optimal grammar inside the search space, future work, will investigate the advantages of Monte Carlo tree search over MCMC for this particular challenge. Another future line of research is the goal to learn more general grammars while avoiding an over generalization, effectively defying Gold's Law. A possible approach is to take advantage of the grammar as a generative model and introduce reinforcement learning techniques to improve the grammar after it has been induced from a given set of demonstrations.

SCREW BOARD SCREW BOARD SCREW BOARD SCREW BOARD
BOARD BOARD SCREW BOARD SCREW BOARD SCREW SCREW
BOARD SCREW BOARD SCREW BOARD SCREW BOARD SCREW

Table 5. Three sample sequences produced by the induced assembly grammar. For the sake of brevity the nonterminals BOARD and SCREW have not been resolved further.

Funding

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). Furthermore, this project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640554 (SKILLS4ROBOTS).

This article is also based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Andrieu C, de Freitas N, Doucet A and Jordan MI (2003) An introduction to MCMC for machine learning. Machine Learning 50(1-2): 5–43.

Baker JK (1979) Trainable grammars for speech recognition. The Journal of the Acoustical Society of America 65(S1): S132–S132.

Bringmann K and Friedrich T (2010) Approximating the volume of unions and intersections of high-dimensional geometric objects. Computational Geometry 43(6): 601 – 610.

Calinon S, Guenter F and Billard A (2007) On learning, representing, and generalizing a task in a humanoid robot. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37(2): 286–298.

Chiang D, Joshi AK and Searls DB (2006) Grammatical representations of macromolecular structure. Journal of Computational Biology 13(5): 1077–1100.

Dantam N, Kolhe P and Stilman M (2011) The motion grammar for physical human-robot games. In: International Conference on Robotics and Automation. IEEE.

Dantam N and Stilman M (2012) The motion grammar calculus for context-free hybrid systems. In: American Control Conference. pp. 5294–5301.

Dantam N and Stilman M (2013) The motion grammar: Analysis of a linguistic method for robot control. IEEE Trans. Robotics 29(3): 704–718.

Earley J (1983) An efficient context-free parsing algorithm (reprint). Commun. ACM 26(1): 57–61.

Gold EM (1967) Language identification in the limit. Information and Control 10(5): 447 – 474.

Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P and Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. Neural computation 25(2): 328–373.

Kitani KM, Sato Y and Sugimoto A (2005) Deleted interpolation using a hierarchical bayesian grammar network for recognizing human activity. In: 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. pp. 239–246.

Kitani KM, Sato Y and Sugimoto A (2008) Recovering the basic structure of human activities from noisy video-based symbol strings. IJPRAI 22(8): 1621–1646.

Kormushev P, Calinon S and Caldwell DG (2010) Robot motor skill coordination with em-based reinforcement learning. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan. IEEE. ISBN 978-1-4244-6674-0, pp. 3232–3237.

Kulic D, Ott C, Lee D, Ishikawa J and Nakamura Y (2012) Incremental learning of full body motion primitives and their sequencing through human motion observation. I. J. Robotics Res. 31(3): 330–345.

Lee K, Kim T and Demiris Y (2012) Learning action symbols for hierarchical grammar induction. In: Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012. IEEE Computer Society. ISBN 978-1-4673-2216-4, pp. 3778–3782.

Lee K, Su Y, Kim T and Demiris Y (2013) A syntactic approach to robot imitation learning using probabilistic activity grammars. Robotics and Autonomous Systems 61(12): 1323–1334.

Lioutikov R, Kroemer O, Maeda G and Peters J (2014) Learning manipulation by sequencing motor primitives with a two-armed robot. In: Intelligent Autonomous Systems 13 - Proceedings of the 13th International Conference IAS-13, Padova, Italy, July 15-18, 2014, Advances in Intelligent Systems and Computing, volume 302. Springer. ISBN 978-3-319-08337-7, pp. 1601–1611.

Lioutikov R, Maeda G, Veiga F, Kersting K and Peters J (2018) Inducing probabilistic context-free grammars for the sequencing of robot movement primitives. In: Proceedings of the International Conference on Robotics and Automation (ICRA).

Lioutikov R, Neumann G, Maeda G and Peters J (2017) Learning movement primitive libraries through probabilistic segmentation .

Manschitz S, Kober J, Gienger M and Peters J (2014) Learning to sequence movement primitives from demonstrations. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014. IEEE, pp. 4414–4421.

Muelling K, Kober J, Kroemer O and Peters J (2013) Learning to select and generalize striking movements in robot table tennis (3): 263–279.

Niekum S, Osentoski S, Konidaris G, Chitta S, Marthi B and Barto AG (2015) Learning grounded finite-state representations from unstructured demonstrations. The International Journal of Robotics Research 34(2): 131–157.

Paraschos A, Daniel C, Peters J and Neumann G (2017) Using probabilistic movement primitives in robotics .

Rivas E and Eddy SR (2000) The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics 16(4): 334–340.

Sarabia M, Lee K and Demiris Y (2015) Towards a synchronised grammars framework for adaptive musical human-robot collaboration. In: 2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). pp. 715–721.

Schneider M and Ertel W (2010) Robot learning by demonstration with local gaussian process regression. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, pp. 255–260.

Stolcke A (1994) Bayesian Learning of Probabilistic Language Models. PhD Thesis, Berkeley, CA, USA.

Stulp F and Schaal S (2011) Hierarchical reinforcement learning with movement primitives. In: 2011 11th IEEE-RAS International Conference on Humanoid Robots. pp. 231–238.

Talton JO, Yang L, Kumar R, Lim M, Goodman ND and Mech R (2012) Learning design patterns with bayesian grammar induction. In: The 25th Annual ACM Symposium on User Interface Software and Technology, UIST ’12, Cambridge, MA, USA, October 7-10, 2012. ACM. ISBN 978-1-4503-1580-7, pp. 63–74.

Zhu SC and Mumford D (2007) A stochastic grammar of images. Foundations and Trends® in Computer Graphics and Vision 2(4): 259–362.