UC San Diego

Learning attribute grammars for movement primitive sequencing

Presenter: Xinyi He

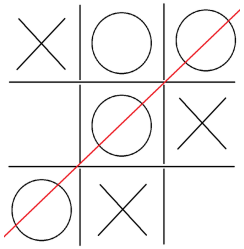
05/19

Outline

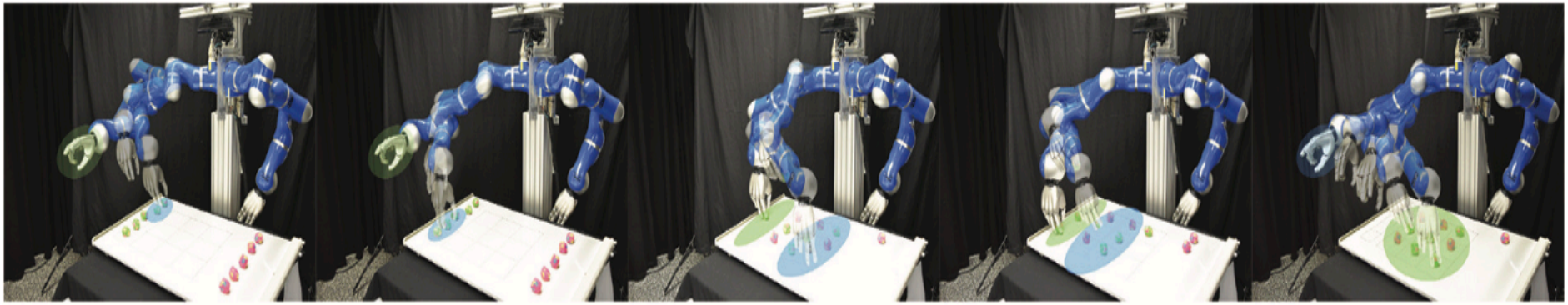
- Introduction
- Related work
- Method
- Experiments
- Conclusion

Introduction

- Movement primitives (MPs)
  - represent atomic, simple movements



tic-tac-toe



MPs in tic-tac-toe game

Introduction

- Sequences of MPs
  - represent more complex tasks

$$
\begin{aligned}
\mathcal{D} = \{ & \\
& (\text{pick\_far, close, place\_right, open, home}), \\
& (\text{pick\_near, close, place\_right, open, home}), \\
& \vdots \\
& (\text{pick\_far, close, place\_left, open, home}), \\
& (\text{pick\_near, close, place\_left, open, home}) \\
\} &
\end{aligned}
$$

sequence of MPs (demonstration)

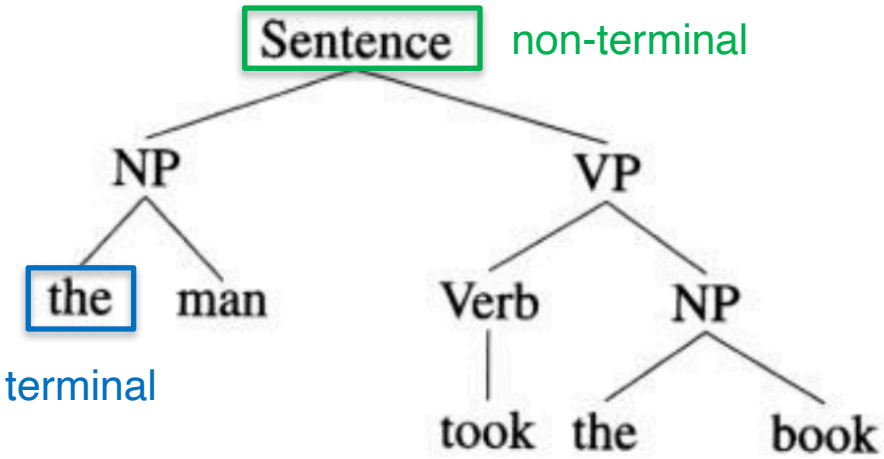
Sequences of MPs in tic-tac-toe game

  - downside: current sequencing mechanisms require a lot of expert knowledge

- Goal: Need more intuitive sequencing mechanisms

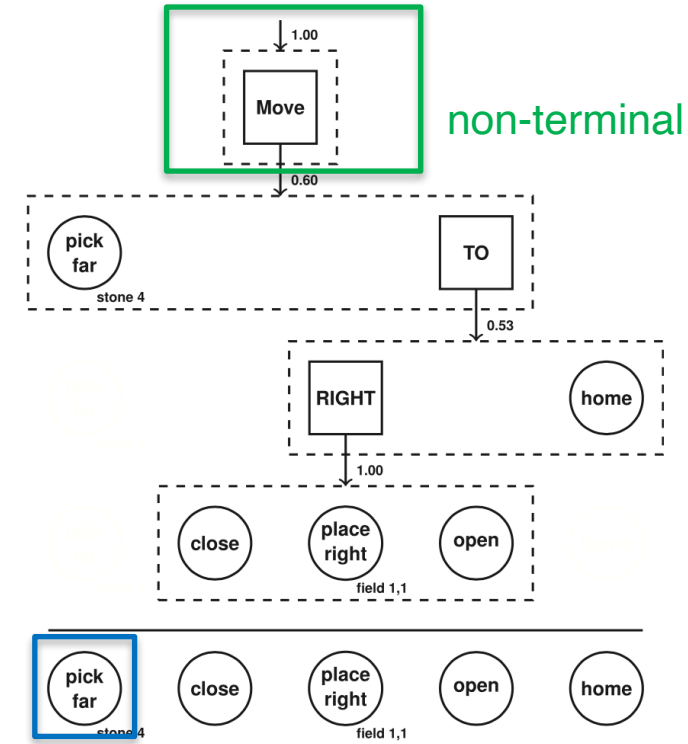
Background

- Formal grammar



Production rule:

Sentence → NP VP
VP → Verb NP
Verb → took
NP → the man
NP → the book



Production rule:

Start	→	MOVE
MOVE	→	pick_near TO
	→	pick_far TO
TO	→	LEFT home
	→	RIGHT home
LEFT	→	close place_left open
RIGHT	→	close place_right open

Background

- Formal grammars

$$\mathcal{G} = \langle \mathcal{A}, \mathcal{V}, \mathcal{R}, \mathcal{S} \rangle$$

  - $\mathcal{A}$ is terminals , $\mathcal{V}$ is non-terminals
  - Production rule $r_\beta \in \mathbf{R}_\alpha$ (α , $\beta \in (\mathcal{A} \cup \mathcal{V})^+$)

$$\alpha \rightarrow \beta \quad = \text{LHS} \rightarrow \text{RHS}$$

  - $\mathcal{R}$ is the set of all production rules $\mathbf{R}_\alpha$
  - $\mathcal{S}$ is the starting symbol ($\mathcal{S} \subseteq (\mathcal{A} \cup \mathcal{V})^+$)
  - Weight each $\mathbf{R}_\alpha \in \mathcal{R}$ with a multinomial $\boldsymbol{\rho}_\alpha$: $\sum \boldsymbol{\rho}_\alpha = 1$

Background

- Formal grammars
  - 1. Probabilistic context-free grammars (PCFGs)
    - describe non-regular expression
    - $\alpha \rightarrow \beta$

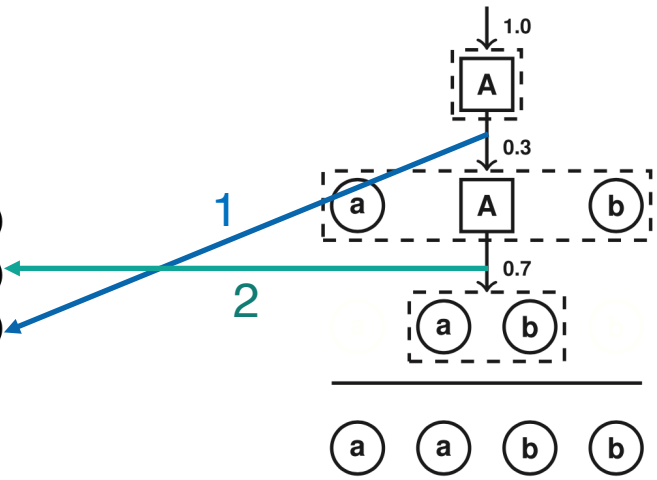
    LHS α: single non-terminal

    RHS β: arbitrary sequence of terminals, non-terminals

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

START	$\rightarrow$	A	(1.0)
A	$\rightarrow$	ab	(0.7)
	$\rightarrow$	aAb	(0.3)

$$\}$$



E.g. $\mathbf{a^n b^n}$

parse tree of **aabb**

Background

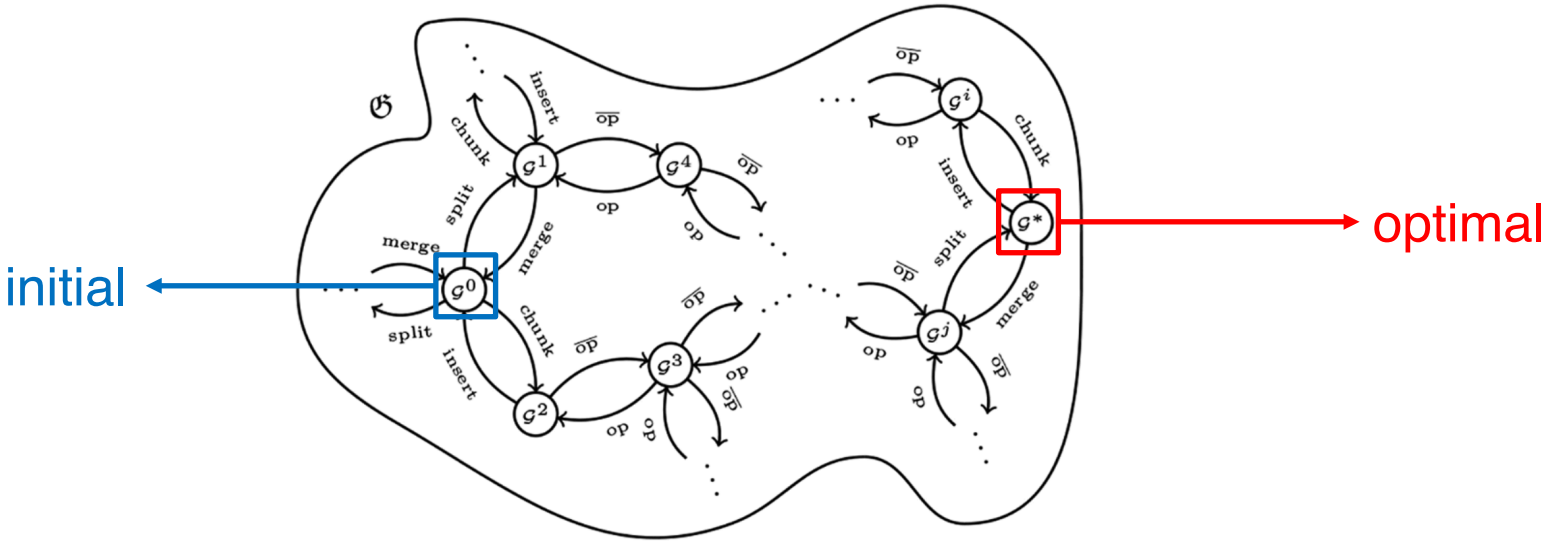
- Formal grammars
  - 2. Attribute grammars
    - extend expressiveness of PCFGs (E.g. $\mathbf{a^n b^n c^n}$)
    - terminal / non-terminal can be assigned attributes

START	$\rightarrow$	A	(1.00)
A_1	$\rightarrow$	ab	(0.70)
		$A_1.\text{depth} = 1$	
	$\rightarrow$	aA_2b	(0.30)
		$A_1.\text{depth} = A_2.\text{depth} + 1$	
		$A_2.\text{max\_depth} = A_1.\text{max\_depth} - 1$	
		assert : $A_1.\text{max\_depth} > 0$	

E.g. $\mathbf{a^n b^n}$

Background

- Formal grammars
  - 3. Grammar induction
    - **learn formal grammars from sequences of terminals**
    - search problem in grammar space $\mathfrak{G}$:
      start from initial grammar $\mathcal{G}^0$
    -> traverse using operators (e.g. split, merge…)
    -> search for optimal grammar $\mathcal{G}^*$



Background

- MP representation
  - probabilistic movement primitives (ProMPs)
    - Each trajectory $\boldsymbol{\tau}$ of demonstration is sampled from:

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

observation noise

feature matrix

weight

    - Learn $\boldsymbol{w}$ from observed demonstration $\boldsymbol{\tau}$:
      Maximum a posteriori optimization

$$\boldsymbol{w} = \arg\max_{\boldsymbol{w}'} p(\boldsymbol{\tau}|\boldsymbol{w}')p(\boldsymbol{w}')$$

prior over weights

    - Gaussian distribution over projected trajectories:

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{\theta}), \boldsymbol{\theta} = (\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$$

primitive

mean, covariance matrix

Background

- MP representation
  - probabilistic movement primitives (ProMPs)
    - Distribution of trajectory $\boldsymbol{\tau}$ given primitive $\boldsymbol{\theta}$:

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

    - Assume equidistant time step t, velocity trajectories :

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

first time derivative

first time derivative

observation noise w.r.t velocity trajectory

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Related Work

- MPs are used for simple tasks
  - single, atomic stroke-based or periodic movements (Paraschos et al. 2018)

- MPs sequences are used for complex tasks
  - predefined MPs sequence (Lioutikov et al., 2014)
  - use PCFGs to sequence discrete actions and learn the grammar (Lee et al. 2012, 2013)
  - grammar induction for hierarchical structures using the Metropolis–Hastings algorithm (Talton et al. 2012)

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Method

- Problem statement
  - Given a set of demonstrations $\mathcal{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{|\mathcal{D}|}\}$, a set of primitives $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|\Theta|}\}$ ($\boldsymbol{d}_i \in \boldsymbol{\Theta}^+$)
  - **Goal: learn a concise, expressive attribute grammar** $\mathcal{G}^*_{\text{att}}$

- Overview
  - Induce PCFG $\mathcal{G}^*$ from the demonstrations $\mathcal{D}$
  - Enhance with a general attribute scheme for sequencing MP $\mathcal{G}^* \underset{\text{att}}{\Rightarrow} \mathcal{G}^*_{\text{att}}$

Method

1. Identify the primitive category
   - multi-label classification, assign primitive $\boldsymbol{\theta}$ to category $\mathcal{C}_i$
   - classify based on active DOF during movement
   - represent primitive $\boldsymbol{\theta}$ using ProMP
     - velocity trajectories

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

     - maximum velocity feature

mean std

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

   - $\boldsymbol{\theta}$ is active w.r.t category $\mathcal{C}_i$

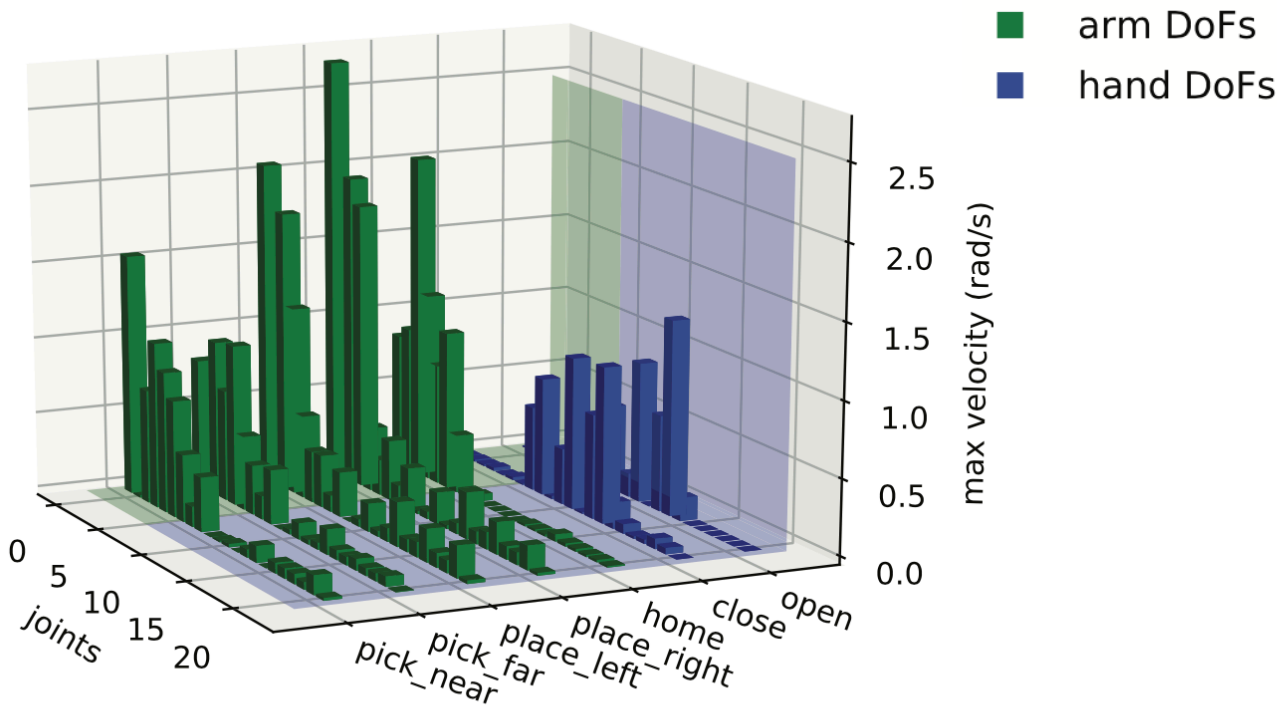
$$\boldsymbol{\theta} \in \mathcal{C}_i \Leftrightarrow \bigvee_{d \in \text{dof}(\mathcal{C}_i)} \psi_d^{\text{vel}}(\boldsymbol{\theta}) > \epsilon_i$$

DOFs associated with $\mathcal{C}_i$

dth DOF

Method

1. Identify the primitive category
   - 2 categories: arm, hand



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

Method

2. Determine connectibility of primitives

- Primitive category allow connectibility requirement hold only between primitives of the same category
- At every time step t, approximate a ProMP $\boldsymbol{\theta}_i$ as a hyper ellipsoid
- For two ProMPs, $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$:

$$\boldsymbol{\theta}_i \underset{\text{con}}{\rightarrow} \boldsymbol{\theta}_j \Leftrightarrow \text{overlap}\left(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j\right) \geqslant \epsilon_{\text{overlab}}$$

- Compute transition overlap for each DOF:

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

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

d_th element of center/std of last ellipsoid of θ_i

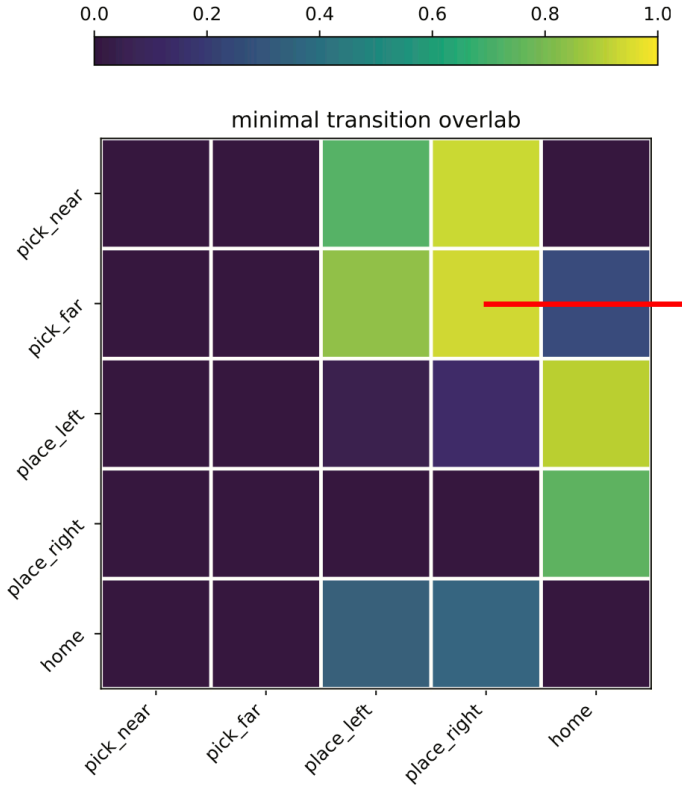
first ellipsoid of θ_j

- Define connectible to $\overrightarrow{\text{Con}}(\boldsymbol{\theta}_i) = \left\{\boldsymbol{\theta}_j | \boldsymbol{\theta}_j \in \boldsymbol{\Theta} \wedge \boldsymbol{\theta}_i \underset{\text{con}}{\rightarrow} \boldsymbol{\theta}_j\right\}$

  connectible from $\overleftarrow{\text{Con}}(\boldsymbol{\theta}_i) = \left\{\boldsymbol{\theta}_j | \boldsymbol{\theta}_j \in \boldsymbol{\Theta} \wedge \boldsymbol{\theta}_j \underset{\text{con}}{\rightarrow} \boldsymbol{\theta}_i\right\}$

Method

2. Determine connectibility of primitives



pick_far connectible to place_right

The transition overlap for each arm primitives of the tic-tac-toe task.

$\epsilon_{\text{overlab}} = 0.69$

Method

3. Induce PCFGs for MPs
   - Learn grammars through posterior optimization
     - Maximize a posterior $p(\mathcal{G}|\mathcal{D})$:

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

likelihood prior

     - Posterior distribution: combination of likelihood and prior

Method

3. Induce PCFGs for MPs
   - Learn grammars through posterior optimization
     - Likelihood: summary of evidence from new observations

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

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

$\mathsf{T}(\boldsymbol{d},\mathcal{G})$: all feasible parse trees

(A,r,ρ) : edge connect non-terminal A and production $r \in R_A$ with probability $\rho \in \boldsymbol{\rho}_A$

Method

3. Induce PCFGs for MPs

- Learn grammars through posterior optimization
  - Prior: belief before taking evidence into account
    - Parameter prior $p(\boldsymbol{\rho}_{\mathcal{G}}|\mathcal{G}_{\mathcal{R}})$, $\boldsymbol{\rho}_{\mathcal{G}} = \{\boldsymbol{\rho}_A | A \in \mathcal{V}\}$
    - Structure prior $p(\mathcal{G}_{\mathcal{R}})$, $\mathcal{G}_{\mathcal{R}} = \{(A, \mathrm{R}_A) | A \in \mathcal{V}\}$
    - Grammar prior $p(\mathcal{G}) = p(\boldsymbol{\rho}_{\mathcal{G}}|\mathcal{G}_{\mathcal{R}})p(\mathcal{G}_{\mathcal{R}})$

    - 3 common structure priors:

    1.MDL: exponential distribution of MDL $(\mathcal{G}_{\mathcal{R}})$

    2.MDL + Poisson: extend MDL with log of Poisson distribution with mean η_r (average production length)

    3.Avg. Poisson: a Poisson distribution over η_r

$$\mathrm{MDL}(\mathcal{G}_{\mathcal{R}}) = \sum_{(A, \mathrm{R}_A) \in \mathcal{G}_{\mathcal{R}}} \sum_{r \in \mathrm{R}_A} \mathrm{MDL}(r)$$

$$\mathrm{MDL}(r) = [1 + |r|] \log_2 |\mathcal{A}| + |\mathcal{V}|]$$

bits to encode symbols

LHS

production length

Method

3. Induce PCFGs for MPs
   - Learn grammars through posterior optimization
     - Prior: before taken evidence into account
       - Novel prior (Grammar Poisson): $p(\mathcal{G}) = p(\boldsymbol{\rho}_{\mathcal{G}}, \mathcal{G}_{\mathcal{R}})$
         model parameter prior and structure prior jointly

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

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

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

Poisson distribution with means $\eta_{\mathcal{R}}$, η_{R} , η_r

avg rules

avg productions per rule

avg production length

Method

3. Induce PCFGs for MPs
   - Traverse grammar space $\mathfrak{G}$
     - Each grammar is a node in $\mathfrak{G}$
     - Operators $\mathrm{op} \in \mathcal{O}$ are directed edges between grammars
     - $\mathfrak{G}$ only contain grammars satisfy connectibility

     - Define $\mathcal{O} = \{\mathrm{merge}, \mathrm{split}, \mathrm{chunk}, \mathrm{insert}\}$

     split: non-terminal $A_i \in \Omega_{\mathrm{split}}$ -> 2 new non-terminals A_j, A_k
     randomly separate $\mathbf{R}_{A_i}$ into 2 disjoint sets $\mathbf{R}_{A_j}$, $\mathbf{R}_{A_k}$

     merge: 2 non-terminals $(A_j, A_k) \in \Omega_{\mathrm{merge}}$ -> new non-terminal A_i
     productions $\mathbf{R}_{A_i} = \mathbf{R}_{A_j} \cup \mathbf{R}_{A_k}$

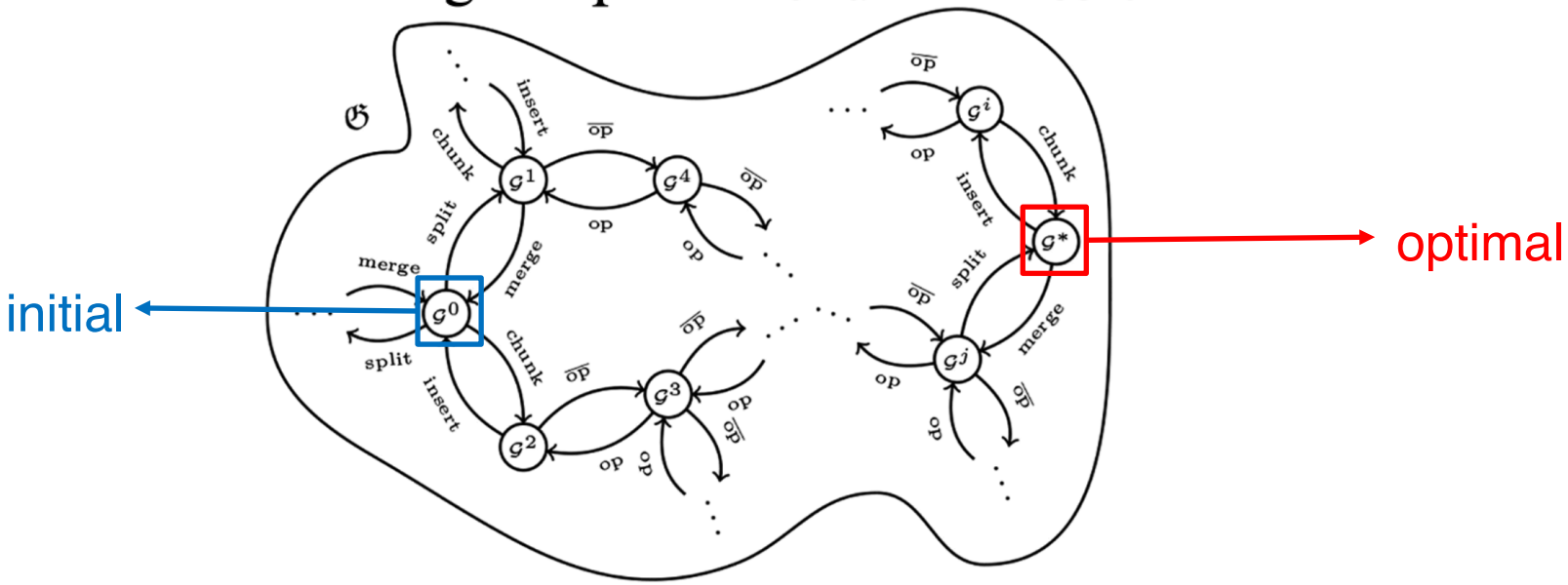
     chunk: create new non-terminal A with production $\mathbf{R}_A = \{r\}$
     every r is replaced by A

     insert: select a non-terminal $A \in \Omega_{\mathrm{insert}}$
     every A is replaced by its production $r \in \mathbf{R}_A$

Method

3. Induce PCFGs for MPs
   - Traverse grammar space $\mathfrak{G}$
     - Define $\mathcal{O} = \{\text{merge}, \text{split}, \text{chunk}, \text{insert}\}$ Not Exchangeable

$$\overline{\text{merge}} = \text{split} \qquad \overline{\text{chunk}} = \text{insert}$$



   - *merge, split* : generalize existing hierarchies in grammars
   - *chunk, insert* : create new hierarchies

Method

3. Induce PCFGs for MPs
   - Find $\mathcal{G}^*$
     - Use Markov chain Monte carlo (MCMC) optimization
     - ***insert*** ensures irreducibility of Markov chain
     - Apply Metropolis-Hasting algorithm
       - At every iteration, sample a random new grammar from the proposal distribution $\mathcal{G}', \mathrm{op}' \sim q(\mathcal{G}', \mathrm{op}' | \mathcal{G})$
       - The new grammar is accepted with a probability of

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

       - After several iteration, return the grammar with the highest posterior

Method

4. Enhancing PCFGs with attributes for MP sequencing
   - Attributes help ensure a continuous state-space trajectory of the sequenced MPs

   - Define 3 attributes:

     transition: define where current primitive ends and next primitive to start (only for non-terminals)
     endpoint: the endpoint of a MP (only for terminals after evaluation)
     viapoints: An ordered list of points that are supposed to be traversed by the sequence of primitives

   - Define 2 conditions:

     reachable: whether a point is reachable by the primitive
     producible: whether at least one of the RHS is producible

Method

4. Enhancing PCFGs with attributes for MP sequencing

- Evaluation scheme for tic-tack-toe

MOVE → pick_near TO (0.40)

evaluate terminal pick_near

```
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
```

evaluate non-terminal TO

traverse(terminal, a list of points): if the first point in the list is reachable by the terminal, the primitive will traverse the point, remove the point from the list, return True

Method

4. Enhancing PCFGs with attributes for MP sequencing
   - General evaluation scheme for sequencing tasks
     - For every production rule is evaluated using:

       LHS: non-terminal A, RHS: terminal a, every non-terminal B

evaluate terminal a

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

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

evaluate every non-terminal B

Method

4. Enhancing PCFGs with attributes for MP sequencing
   - General evaluation scheme for sequencing tasks
     - A primitive may require a certain via point

       e.g. pick_far, pick_near need position of the stone

     - Define 2 additional attributes:

       keywords: unordered list contain keywords identifying relevant points in *targets* attribute (only for terminals before evaluation)

       targets: a dictionary maps keywords to ordered list of points

Method

4. Enhancing PCFGs with attributes for MP sequencing
   - General evaluation scheme for sequencing tasks
     - *target* greatly influence the sequence production
       - e.g. pick_far, pick_near has *keyword* stone
         assert reachable(pick_far, target[stone]) -> success
         assert reachable(pick_near, target[stone]) -> fail
         Then every sequence produced with the target will contain pick_far, never a pick_near.
     - Users can only access and adapt *targets* attribute

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Experiments

- Learn a grammar for Tic-Tac-Toe turns
  - 5 arm MPs
    - pick_near, pick_far, place_left, place_right, home
  - 2 hand MPs
    - close, open

  - Start with the initial grammar (4 unique sequences)

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)

Experiments

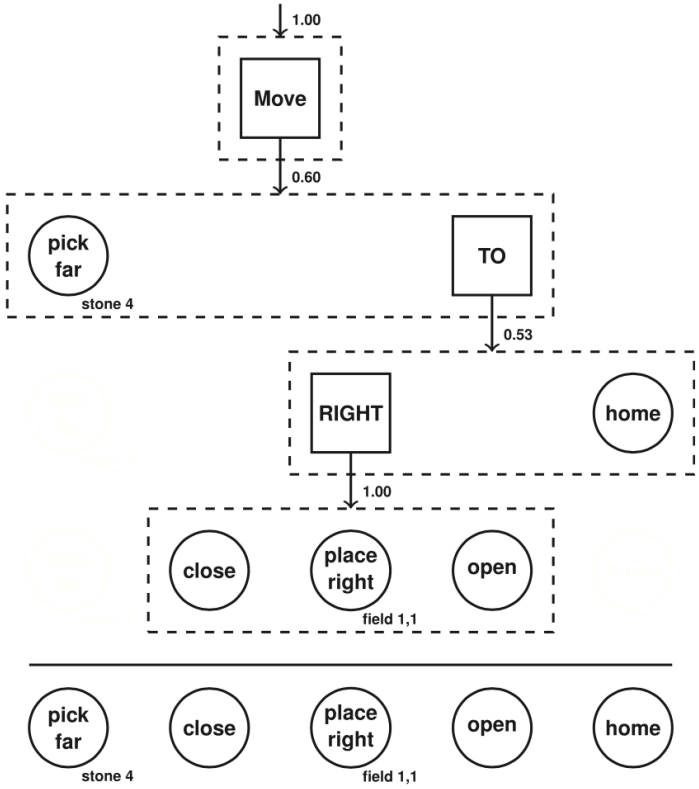
- Learn a grammar for Tic-Tac-Toe turns
  - Initialize the approach
    - desired # rules $\eta_{\mathcal{R}} = 5$
    - desired # avg productions per rule $\eta_{\mathrm{R}} = 2$
    - desired avg length of each production $\eta_r = 3$
    - Weights of each op $\eta_{\mathrm{op}} = 1$
  - Run MCMC optimization for 400 steps
    - Result in 324 accepted grammars

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)

Grammar with the highest posterior

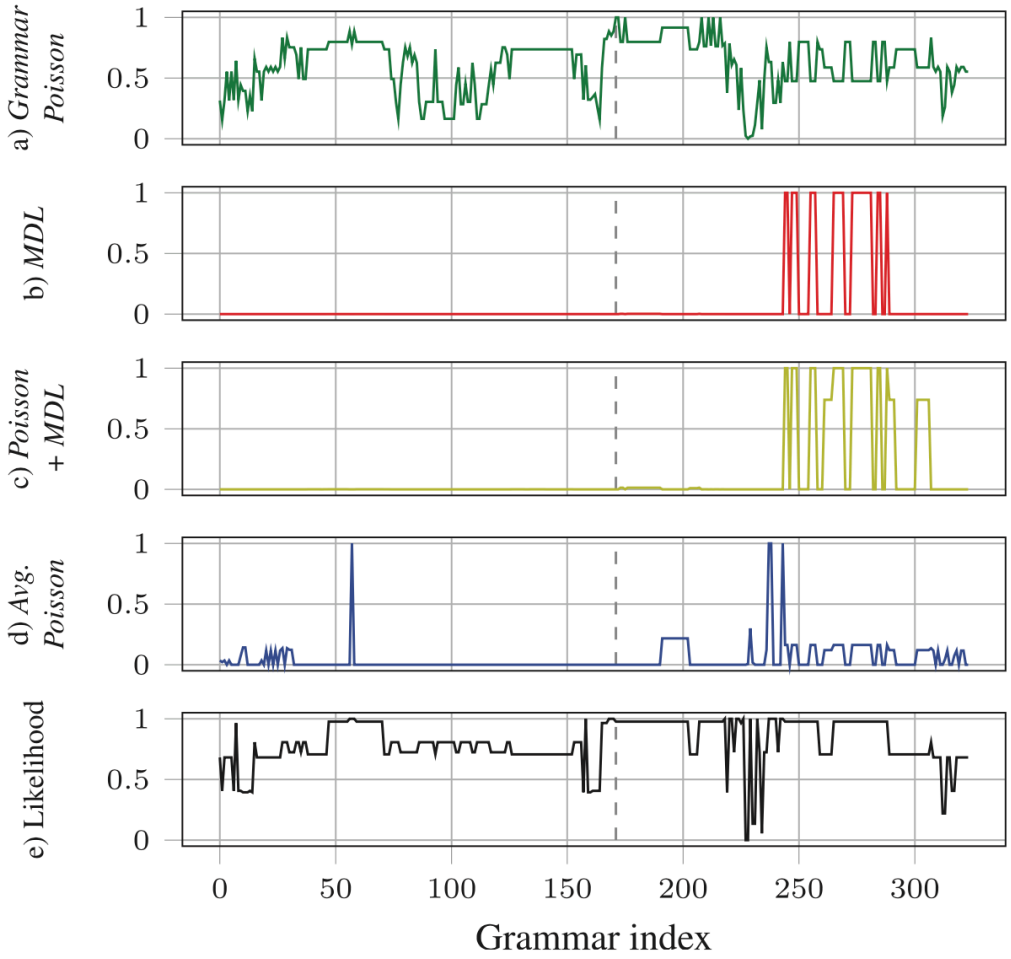
Experiments

- Learn a grammar for Tic-Tac-Toe turns
  - A possible sequence produced by the grammar



Experiments

- Learn a grammar for Tic-Tac-Toe turns



The posteriors and the likelihood for tic-tac-toe turn grammar

Outline

- Introduction
- Related work
- Method
- Experiments
- Conclusion

Conclusion

- Accomplishments
  - Successfully introduced attribute grammars as a mechanism to sequence MPs.
    - More intuitive for non-expert users
    - The novel grammar prior eliminate complications compared to other common priors e.g. MDL
    - MCMC optimization gets a better local optima

Future Work

- Investigate more sophisticated approaches for the clustering of parameterized time series, e.g. applied MPs.
- Investigate the advantage of Monte Carlo tree search over MCMC.
- Learn more general grammars while avoiding over generalization.

Thank you!