Part III
Category learning with a pLoT

Fausto Carcassi

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Summary so far

pLoT as a picture of cognition

Some formal tools:

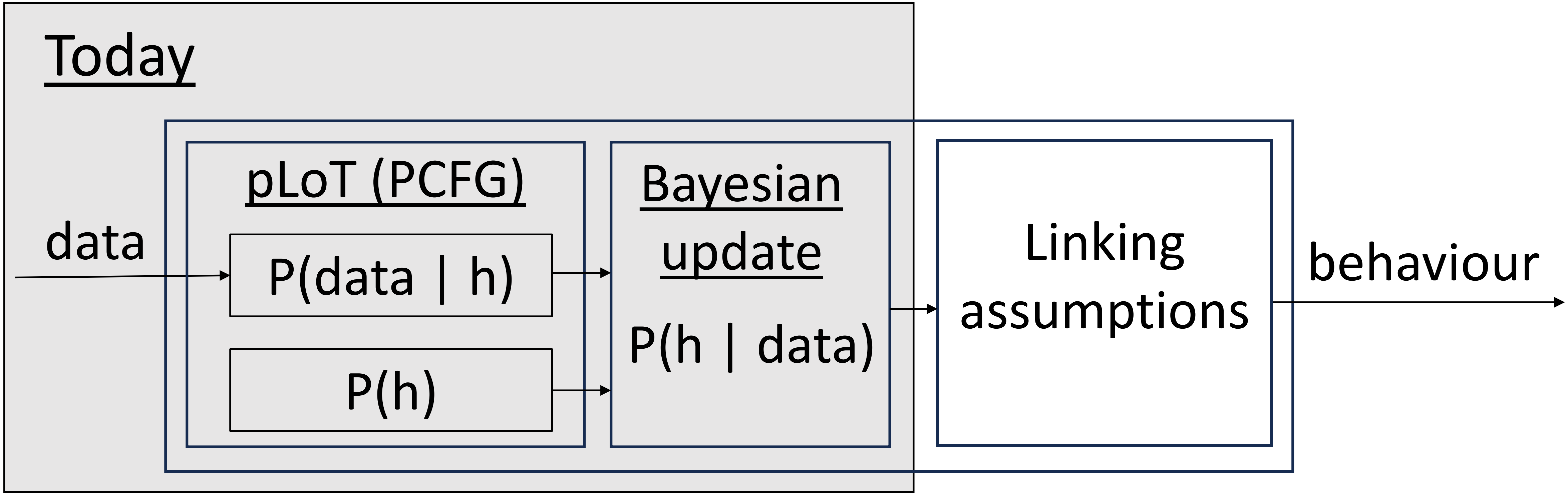
- Probabilistic PCFG
- Compositional interpretation
- Bayesian inference

Plan for session

- We'll consider some examples
- Combine the technical tools
- Build up to a pLoT category learning simulation

Part I	Introduction: On the very idea of an LoT
Part II	Technical background
Part III	**Bayesian program induction (LOTlib3)**
Part IV	Case studies
Part V	Summary & Future prospects

Our grand plan for the pLoT



Example 1: The visual domain

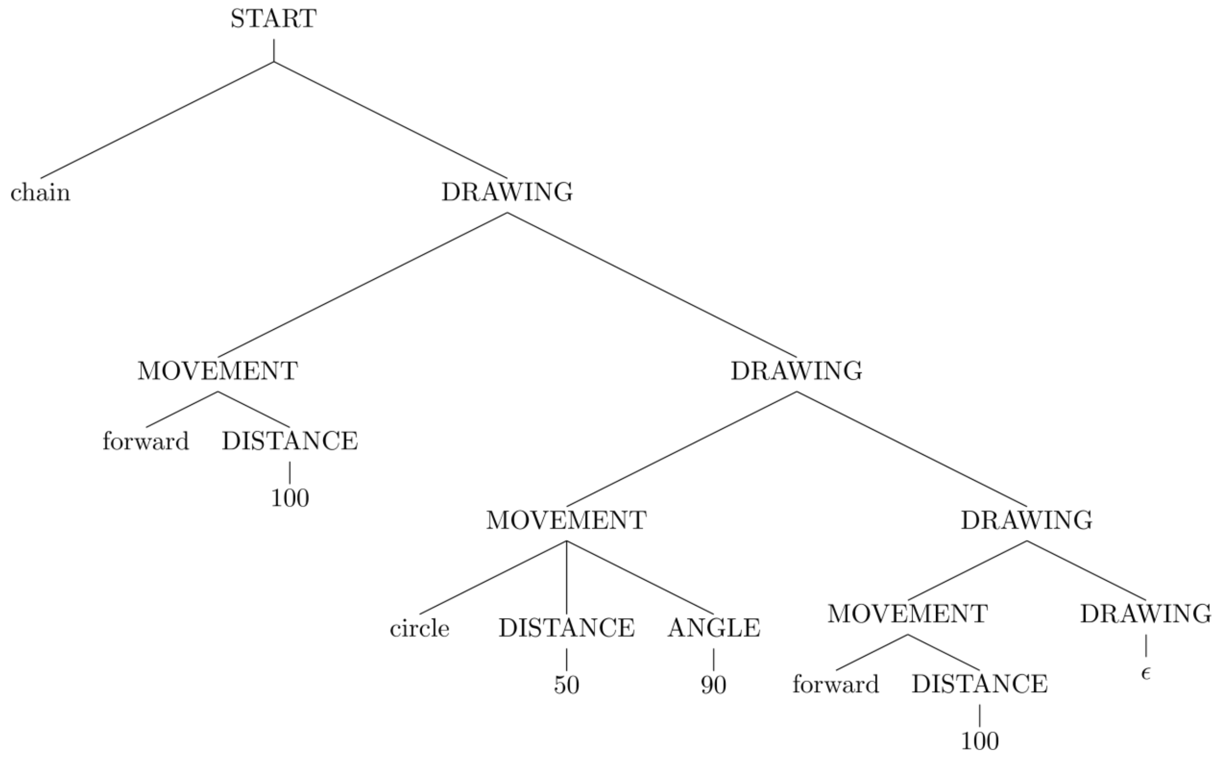
Shape grammar

What could we mean by *shape grammar*?

What would you include in a shape grammar?

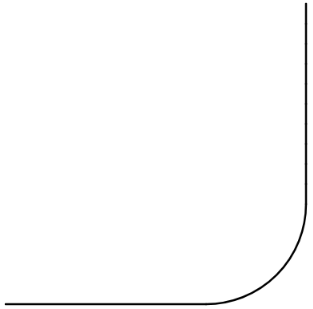
Shape grammar

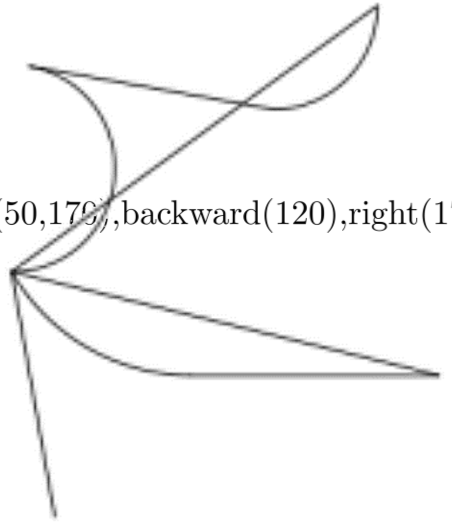
$$
\begin{aligned}
\langle \text{START} \rangle &\to \text{chain}(\langle \text{DRAWING} \rangle) \\
\langle \text{DRAWING} \rangle &\to \langle \text{MOVEMENT} \rangle, \langle \text{DRAWING} \rangle \\
&\to \epsilon \\
\langle \text{MOVEMENT} \rangle &\to \text{forward}(\langle \text{DISTANCE} \rangle) \\
&\to \text{backward}(\langle \text{DISTANCE} \rangle) \\
&\to \text{right}(\langle \text{ANGLE} \rangle) \\
&\to \text{left}(\langle \text{ANGLE} \rangle) \\
&\to \text{circle}(\langle \text{DISTANCE} \rangle, \langle \text{ANGLE} \rangle) \\
&\to \text{goto}(0, 0) \\
\langle \text{DISTANCE} \rangle &\to 50 \mid 100 \mid 120 \\
\langle \text{ANGLE} \rangle &\to 30 \mid 60 \mid 90 \mid 170
\end{aligned}
$$



Shape semantics

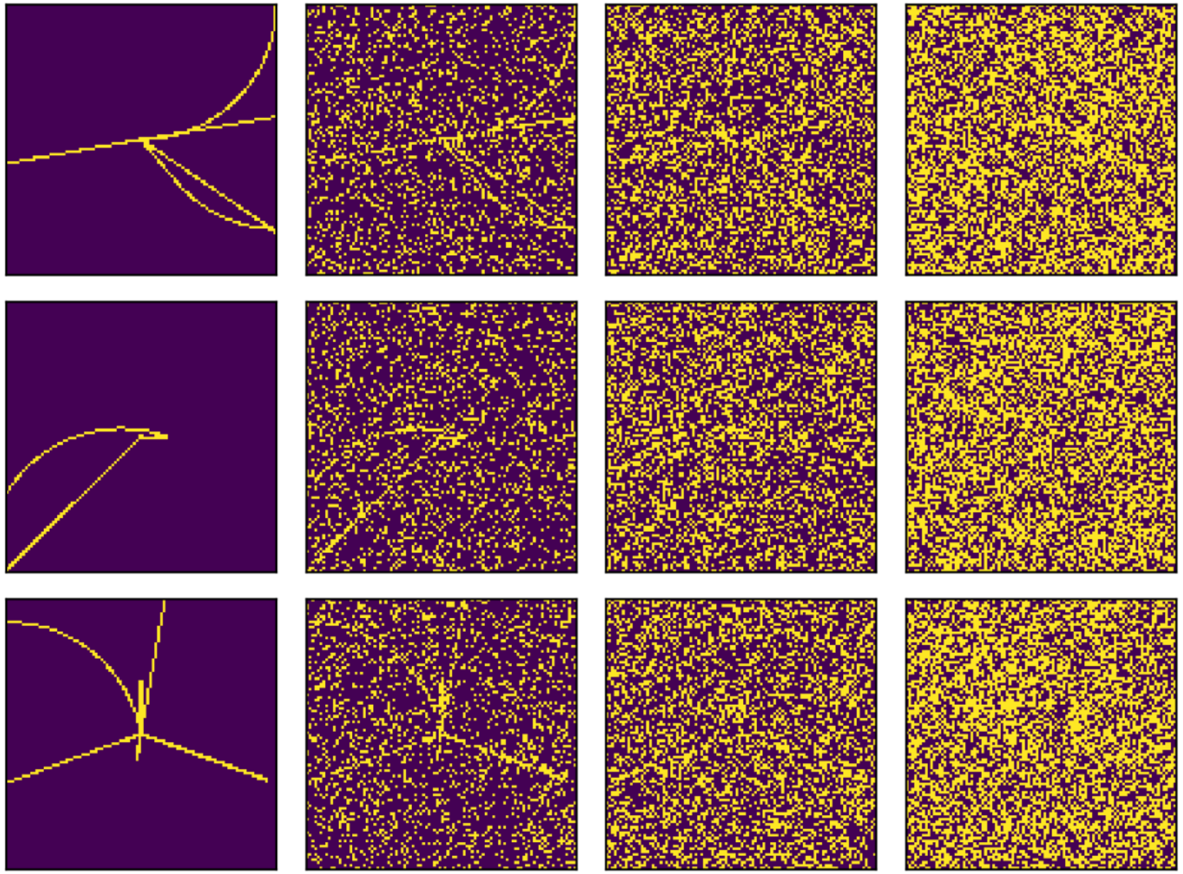
chain(forward(100), circle(50,90), forward(100),)





chain(right(60),circle(100,60),forward(120),goto(0,0),circle(50,170),backward(120),right(170),circle(50,90),goto(0,0),right(170),forward(120),right(90),)

Shape observations

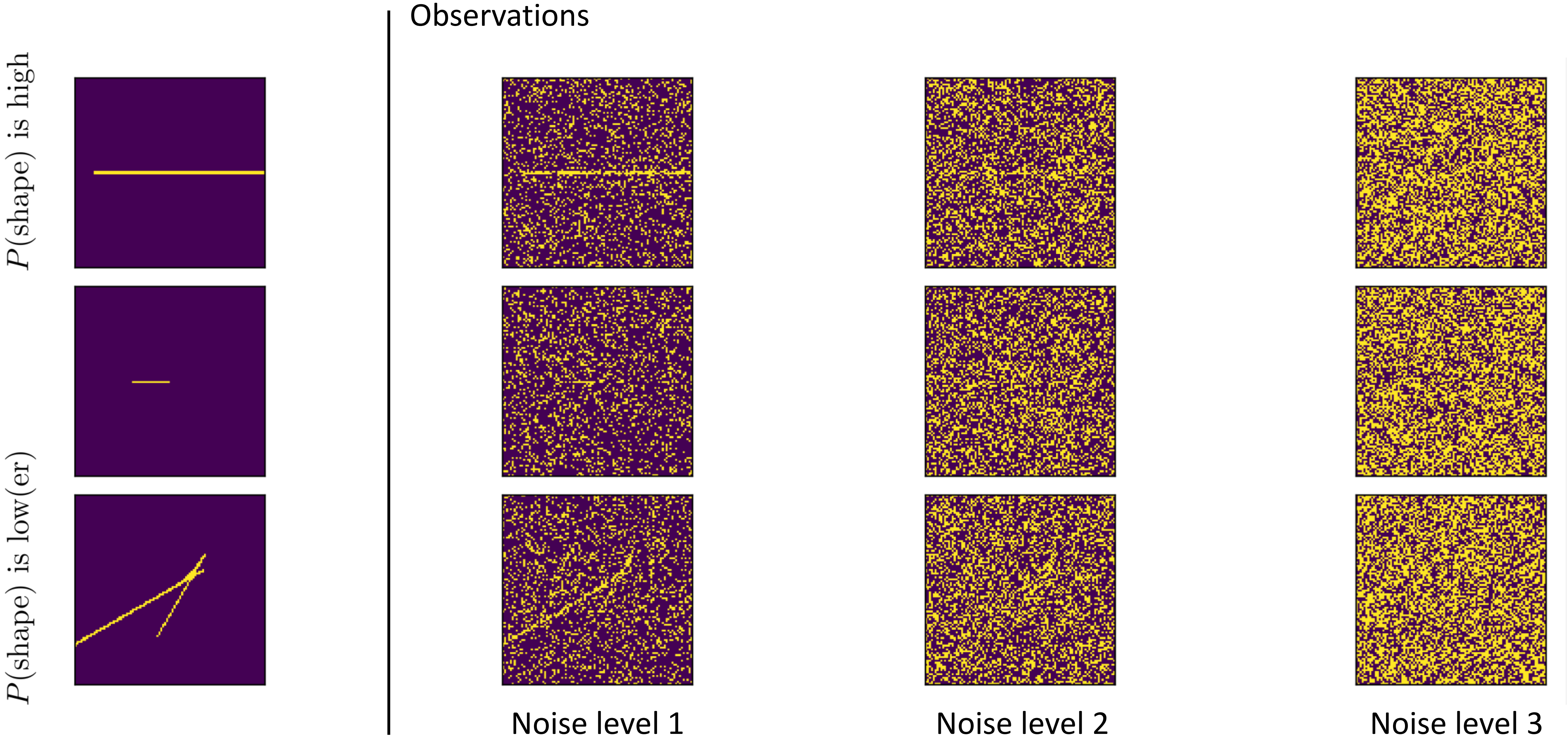


Noise:

- Random pixel flip (salt & pepper)
- W/ probability p

Can we write a likelihood?

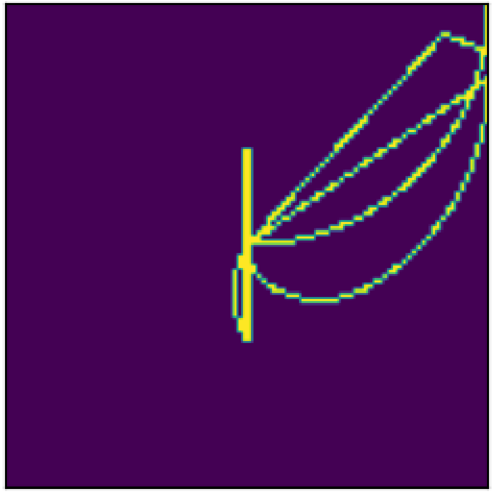
Priors and likelihoods



Bayesian inference in the shape grammar

```
chain(
circle_(50,90),
backward_(10),
right_(90),
right_(170),
right_(30),
forward_(10),
right_(90),
left_(170),
goto_(0,0),
circle_(20,60),
right_(30),
goto_(0,0),
goto_(0,0),
goto_(0,0),
backward_(20),
forward_(20),
goto_(0,0),
circle_(10,60),
circle_(20,60),
circle_(50,60),
left_(170),
goto_(0,0),
right_(60),
left_(30),
)
```

True shape



chain(circle_(50,90),goto_(0,0),forward_(20),goto_(0,0),backward_(20),)

LoT learning models: the big picture

1. Interpreted PCFG
   Defines hypotheses H
   Defines a prior over H

2. Observations O
   Generated by true H

3. Likelihood P(O | H)

4. Bayesian inference algorithm
   Approximate!

1. Shapes grammar
   Each H is a sentence…
   …which encodes a shape

2. Image (with noise)
   Multiple H produce same shape!

3. Likelihood P(image| shape)

4. MCMC w/ e.g., LOTlib3
   Gives us samples from posterior!

Example 2: The auditory domain

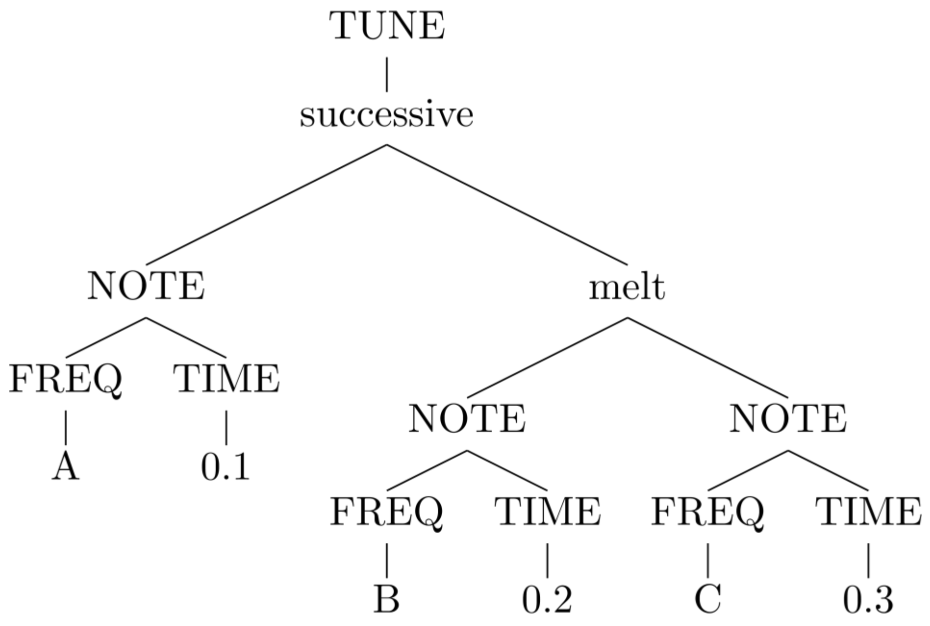
Sound grammar

What could we mean by *sound grammar*?

What would you include in a sound grammar?

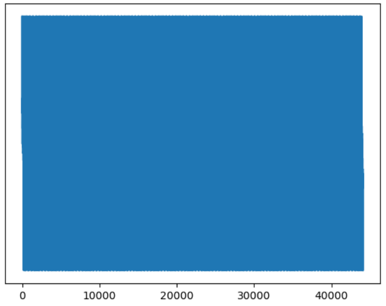
Sound grammar

$$
\begin{aligned}
\langle \mathrm{TUNE} \rangle &\rightarrow \langle \mathrm{NOTE} \rangle \\
&\rightarrow \mathrm{successive}(\langle \mathrm{TUNE} \rangle, \langle \mathrm{TUNE} \rangle) \\
&\rightarrow \mathrm{melt}(\langle \mathrm{TUNE} \rangle, \langle \mathrm{TUNE} \rangle) \\
\langle \mathrm{NOTE} \rangle &\rightarrow \mathrm{defineNote}(\langle \mathrm{FREQ} \rangle, \langle \mathrm{TIME} \rangle) \\
\langle \mathrm{TIME} \rangle &\rightarrow 0.1 \mid 0.2 \mid 0.3 \\
\langle \mathrm{FREQ} \rangle &\rightarrow \mathrm{A} \mid \mathrm{Bb} \mid \mathrm{B} \mid \mathrm{C} \mid \mathrm{Db} \mid \mathrm{D} \mid \mathrm{Eb} \mid \mathrm{E} \mid \mathrm{F} \mid \mathrm{Gb} \mid \mathrm{G} \mid \mathrm{Ab}
\end{aligned}
$$

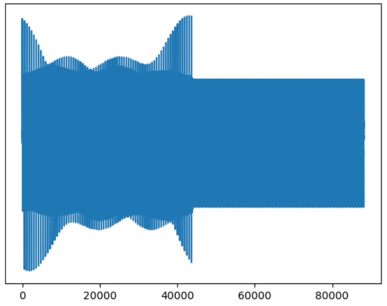


(For the sake of our ears, let’s only keep A, C, D, E, G)

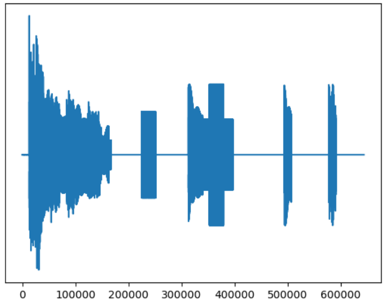
Sound semantics



A

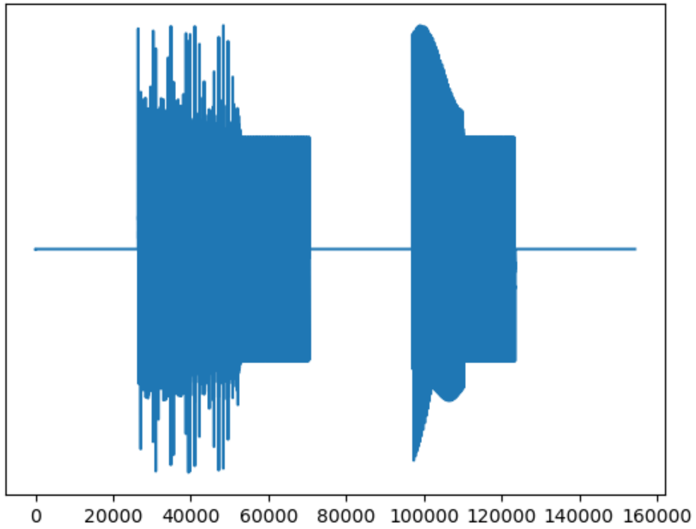


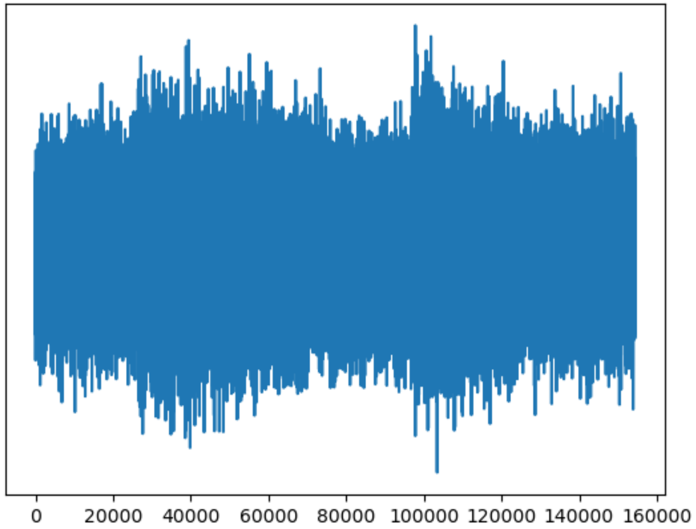
melt(defineNote(E, 1), defineNote(D, 2))



successive(defineNote(’D’, 0.3), melt(defineNote(’A’, 0.3), ” ”melt(melt(melt(melt(defineNote(’E’,
1.), ” ”successive(melt(successive(defineNote(’D’, 0.6), successive(defineNote(’E’,
” ”0.6), defineNote(’E’, 0.6))), melt(successive(melt(melt(defineNote(’C’, ” ”0.3),
successive(melt(melt(successive(successive(defineNote(’G’, 0.3), ” ”melt(defineNote(’E’,
1.), defineNote(’A’, 0.6))), successive(defineNote(’A’, ” ”0.6), successive(defineNote(’E’,
0.6), melt(defineNote(’G’, 0.3), ” ”melt(defineNote(’G’, 1.), successive(defineNote(’A’,
0.6), defineNote(’D’, ” ”0.3))))))), melt(defineNote(’D’, 0.6), successive(defineNote(’C’,
1.), ” ”successive(defineNote(’D’, 1.), defineNote(’G’, 1.)))))), ” ”successive(successive(melt(defineNote(’C’,
1.), ” ”successive(successive(melt(melt(successive(melt(successive(melt(defineNote(’E’,
” ”0.3), defineNote(’E’, 0.6)), defineNote(’E’, 0.6)), defineNote(’C’, 0.3)), ” ”de-
fineNote(’A’, 0.3)), defineNote(’A’, 0.6)), ” ”successive(melt(melt(defineNote(’E’,
0.3), defineNote(’A’, 1.)), ” ”successive(melt(defineNote(’E’, 1.), successive(defineNote(’G’,
0.3), ” ”successive(melt(melt(defineNote(’D’, 0.3), defineNote(’D’, 1.)), ” ”de-
fineNote(’A’, 1.)), defineNote(’A’, 0.3)))), melt(defineNote(’A’, 0.3), ” ”melt(defineNote(’E’,
0.6), melt(successive(defineNote(’A’, 0.3), ” ”defineNote(’C’, 0.6)), melt(defineNote(’C’,
0.3), defineNote(’E’, ” ”0.3)))))))), defineNote(’E’, 1.))), defineNote(’E’, 1.)),
defineNote(’A’, ” ”0.3))), melt(defineNote(’G’, 1.), melt(defineNote(’C’, 1.), ”
”melt(defineNote(’D’, 0.6), defineNote(’D’, 0.6)))))), defineNote(’G’, 1.))), ” ”suc-
cessive(melt(defineNote(’C’, 0.6), successive(defineNote(’G’, 0.6), ” ”defineNote(’C’,
0.3))), successive(melt(defineNote(’C’, 0.6), ” ”defineNote(’C’, 1.)), defineNote(’E’,
0.6))))), defineNote(’E’, 0.3)), ” ”defineNote(’D’, 1.)), defineNote(’D’, 0.3))),
melt(defineNote(’C’, 0.3), ” ”successive(defineNote(’A’, 0.6), successive(melt(successive(defineNote(’E’,
” ”0.3), successive(defineNote(’D’, 1.), defineNote(’E’, 0.3))), ” ”successive(melt(successive(melt(defineNote(’G’,
0.3), defineNote(’D’, 0.3)), ” ”successive(successive(defineNote(’E’, 1.), defineNote(’C’,
0.6)), ” ”successive(melt(defineNote(’D’, 0.3), defineNote(’E’, 0.3)), defineNote(’D’,
” ”0.6)))), defineNote(’D’, 0.6)), defineNote(’E’, 0.3))), defineNote(’G’, ” ”0.3)))))),
defineNote(’C’, 0.3)), successive(melt(defineNote(’D’, 0.3), ” ”melt(melt(defineNote(’A’,
0.6), successive(successive(defineNote(’C’, 0.3), ” ”successive(defineNote(’D’, 1.),
defineNote(’A’, 0.3))), defineNote(’C’, ” ”1.))), defineNote(’G’, 0.3))), defineNote(’C’,
1.))), ” ”successive(defineNote(’D’, 0.3), defineNote(’E’, 0.3)))))

Audio observations

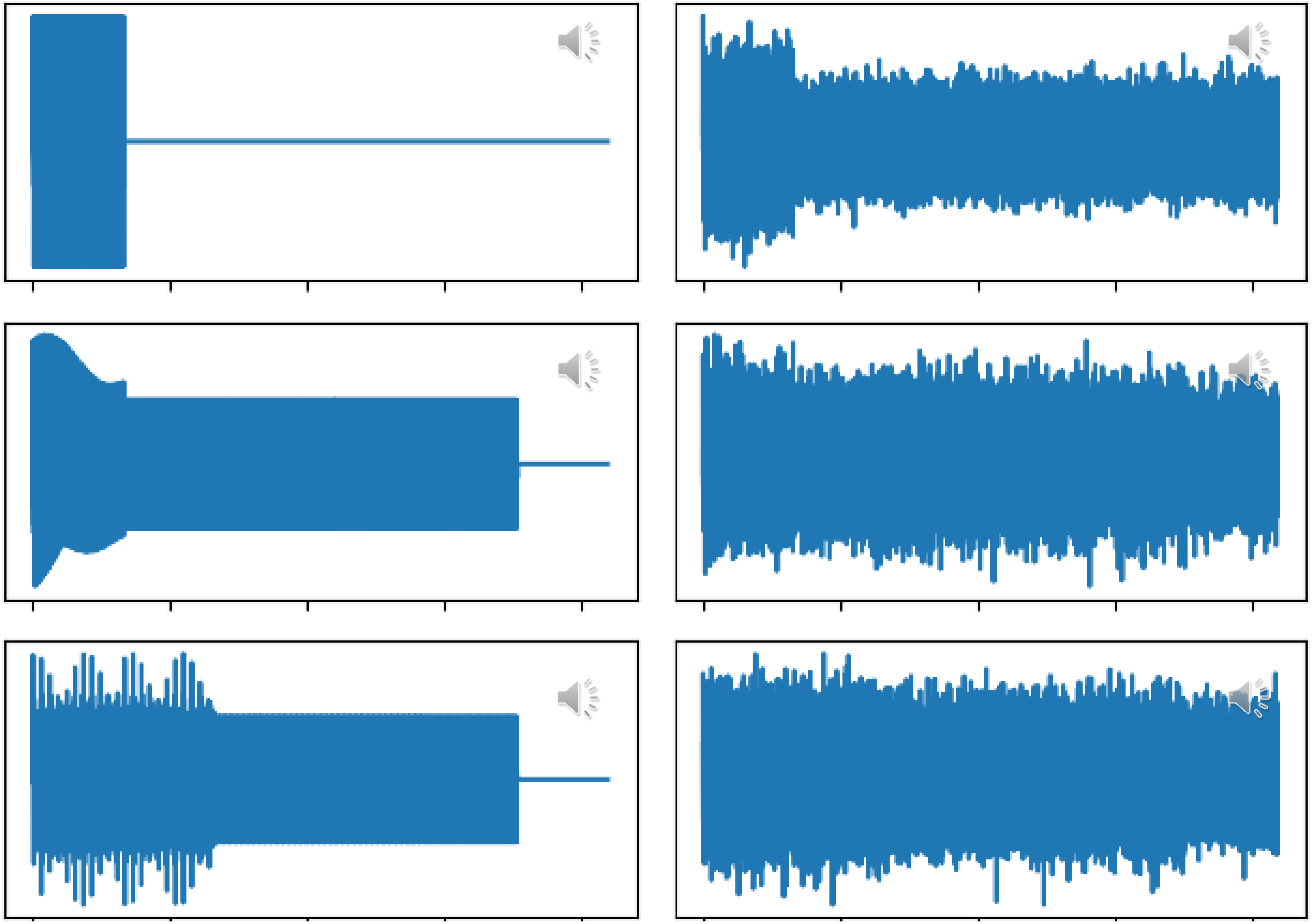




successive(defineNote(’G’, 0.6), successive(defineNote(’D’, 0.6), successive(successive(successive(successive(defineNote(’G’, 0.3), melt(melt(defineNote(’A’, 0.3), successive(defineNote(’C’, 1.), defineNote(’D’, 0.3))), defineNote(’G’, 0.6))), defineNote(’E’, 0.6)), defineNote(’E’, 0.6)), successive(melt(melt(defineNote(’E’, 0.6), successive(defineNote(’C’, 0.3), defineNote(’D’, 1.))), defineNote(’G’, 0.3)), defineNote(’G’, 0.3)))))

Priors and likelihoods

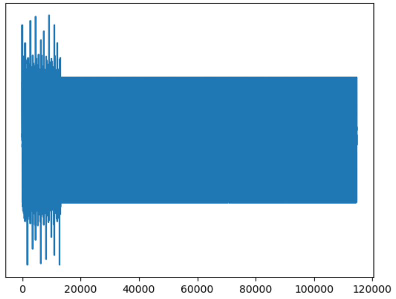
$P(\text{sound})$ is high



$P(\text{sound})$ is low(er)

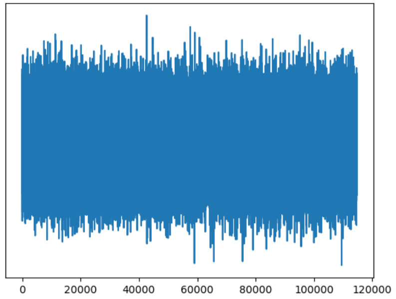
Learning: Bayesian inference in a PCFG

melt(defineNote('G', 0.3), successive(successive(defineNote('A', 0.6), defineNote('C', 1.)), defineNote('E', 1.)))

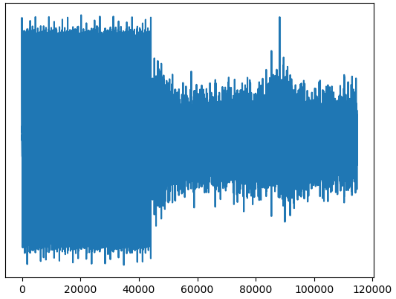


MAP

successive(successive(defineNote('A', 1.), defineNote('A', 1.)), defineNote('D', 0.6))



Posterior mean



LoT learning models: the big picture

1. Interpreted PCFG
   Defines hypotheses H
   Defines a prior over H

2. Observations O
   Generated by true H

3. Likelihood P(O | H)

4. Bayesian inference
   Approximate!

1. Sound grammar
   Each H is a sentence…
   …which encodes a sound

2. Sound (with noise)
   Multiple H produce same sound!

3. Likelihood P(audio | sound)

4. MCMC w/ e.g., LOTlib3
   Gives us samples from posterior!

Summary: Building a pLoT model

1. Pick a domain, e.g., music, geometry, logical concepts
2. Write a (plausible) list of primitive concepts
3. Write a PCFG
4. Define a likelihood function
5. Produce an observation
   1. Sample from the PCFG
   2. Hand-design
   3. Use naturally occurring
6. Run inference algorithm

Interim remarks

Important features

Simpler representations have a higher prior

- Simpler <--> More probable in the pLoT (PCFG)

Three ways probabilities enter the picture

1. Likelihood function
2. Prior given by PCFG
3. Stochastic primitives

Important features

What we have been doing is essentially *program induction*

- Define a simple programming language
- Find (a distribution over) programs that produce some output given some input

In domains above, pLoT sentences denoted single objects

…but each sentence in the pLoT can denote a **function**

- Observations are (inputs, output) tuples

Categories (sets) are a special case with Boolean output

- Observations can be e.g., samples from the category
- Size effect!

Some hypotheses types

Input	**Output**	**Stochastic primitives**	**Meaning**	**Example**
$\emptyset$	Domain object	No	Single domain object	
		Yes	Graded category	
Domain object	Boolean	No	Category	
		Yes	Graded category	
Domain object	Domain object	No	Transformation	
		Yes	Stochastic transf	
…	…	…	…	

Example 3: Binary strings

Let's build it together!

1. Write a PCFG
2. Write an interpretation function
3. Construct an example of data
4. Run the inference

Conclusions

- We saw how to construct a model of category learning w/ a pLoT
- Two example domains:
    - Shapes
    - Sounds
    - Many more to be explored!
- Next step: Literature using this kind of model.

Part I	Introduction: On the very idea of an LoT
Part II	Technical background
Part III	**Bayesian program induction (LOTlib3)**
Part IV	Case studies
Part V	Summary & Future prospects