Part II
Technical background

Fausto Carcassi

Plan

- Formal grammars
- Semantics for formal grammars
- Bayesian inference

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

Formal grammars

What’s a *grammar*

Start with a *language* (Typically, a natural language)

A notion of **well-formedness** independent of meaning
- E.g., ‘3 is more curious than the past table’
- We call this *grammatical* well-formedness.

We can build an **abstract device** to encode grammaticality
- Two types of such devices are popular: automata and formal grammars.
- There is a correspondence between automata and grammars!
- In the rest of the course we’ll just use grammars.

Formal grammars

Grammars: Infinite use of finite means

Four ingredients:

1. A finite set N of *nonterminal symbols*
2. A finite set Σ of *terminal symbols*
3. A finite set P of *production rules*
4. A symbol S in N: the *start symbol*

Formal grammars

1. *N* (nonterminal symbols)
2. Σ (terminal symbols)
3. *P* (production rules)
4. *S* (start symbol)

1. {S, x, y}
2. {a, b}
3. 1. S → x
   2. x → xy
   3. y → a
   4. x → b

Let's derive a sentence in this grammar!

Formal grammars

1. *N* (nonterminal symbols)
2. Σ (terminal symbols)
3. *P* (production rules)
4. *S* (start symbol)

1. {S, x, y}
2. {a, b}
3. 1. S → x
   2. x → xy
   3. by → ba
   4. x → b

Let's derive a sentence in this grammar!

Formal grammars

1. *N* (nonterminal symbols)
2. Σ (terminal symbols)
3. *P* (production rules)
4. *S* (start symbol)

1. {S, x, y}
2. {a, b}
3. 1. S → x
   2. x → xy
   3. by → ba
   4. x → b
   5. b → a **WRONG!** Why?

Let's derive a sentence in this grammar!

CFG - Context Free Grammars

Context-free grammar (CFGs) are grammars with rules of the form:

$$A \rightarrow \alpha$$

Where:

- A: *single* nonterminal symbol
- α: (possibly empty) string of terminals and/or nonterminals

PCFG - Probabilistic CFG

$G = (N, \Sigma, P, S, \Pi)$

1. *N:* nonterminal symbols
2. Σ: terminal symbols
3. *P:* production rules
4. *S:* start symbol
5. Π: probabilities on production rules

New: a function $\Pi: P \to \mathbb{R}$

- *Conditional* probability of applying rule $\alpha \to \beta$
- *Conditional* on the left side being α.

Every derivation has a probability of being derived

- The product of the probabilities of the applied rules.
- Higher probability to smaller trees

PCFG - Probabilistic CFG

1. N (nonterminal symbols)
2. Σ (terminal symbols)
3. P (production rules)
4. S (start symbol)
5. Π (probabilities on production rules)

1. {S}
2. {a, b} Π
3.
   1. S → aSa 0.3
   2. S → bSb 0.3
   3. S → e 0.2
   4. S → a 0.1
   5. S → b 0.1

Interpreting a grammar

Writing down functions

- 1, 2, 3, …, +, %, ‘every’, ‘some’, … **vs** x, y, z, …
    - Unsaturated vs saturated ‘x + 1’ vs ‘1 + 1’

- Unsaturated to function
    - f(x) = x+1
    - f(x) notation is *inconvenient*: forces us to name

- Solution: *lambda* expressions
    - Start with expression $x + 2$
    - Make λ expression w/ variable $\lambda x. x + 2$
    - *Function* from bound variable to the evaluated expression

Multiple λs and languages

- We can nest lambda expressions!
  - For instance: $\lambda y. \lambda x. x + y$
- Any expression can go inside
  - E.g., English: $\lambda x. x$ is a bird
- We'll mostly use variations of predicate logic.

The notation of λ-calculus

Notation for applying an argument to a function

- $\lambda x. P(x)$
- Apply argument N to function $\lambda x. P(x)$: $\big(\lambda x. P(x)\big)N$
- β-reduction:
  - $\big(\lambda x. P(x)\big)N$
  - Replace N in every occurrence of x and remove the lambda
  - $P(N)$

- We can also rename variables w/ alpha conversion (Let's ignore this)

Example of β reduction

- $\left(\left((\lambda x.\lambda y.x(y))\lambda z.P(z)\right)a\right)b$

Substitute $\lambda z.P(z)$ for x into $\lambda y.x(y)$:

- $\left((\lambda y.\lambda z.P(z)(y))a\right)b$

Substitute a for y in $\lambda z.P(z)(y)$

- $(\lambda z.P(z)(a))b$

Substitute b for z in $P(z)(a)$:

- $P(b)(a)$

Compositional interpretation

Let's build an *interpretation function* for a grammar!

- Associate each sentence with a meaning!

E.g., propositional logic

- $S \rightarrow p \mid q \mid (S \wedge S) \mid (S \vee S) \mid \neg S$

Interpretation of basic symbols:

- $I_{@}(p) = 1$
- $I_{@}(q) = 0$
- $I_{@}(\wedge) = \lambda x. \lambda y. \mathrm{x} = 1 \textit{ and } y = 1$
- $I_{@}(\vee) = \lambda x. \lambda y. x = 1 \textit{ or } y = 1$
- $I_{@}(\neg) = \lambda x. x = 0$

Compositional interpretation

Meaning of complex sentences:

If α has the form $\begin{array}{c} S \\ | \\ \beta \wedge \gamma \end{array}$ then $\mathrm{I}(\alpha) = (\mathrm{I}(\wedge)(\mathrm{I}(\beta))\ \mathrm{I}(\gamma))$

Can you tell what the rules are for the other entries?

Example:

$$\begin{array}{ccccc} & & & \mathrm{S} & \\ & & & | & \\ & \mathrm{S} & & \vee & \mathrm{p} \\ & | & & & \\ \mathrm{S} & \wedge & \mathrm{S} & & \\ | & & | & & \\ \mathrm{p} & & \mathrm{q} & & \end{array}$$

Bayesian inference

Probability & conditional probability

What is probability?

- Feature of system in the world
  - Dice behaviour over many rolls
- Degree of support btw propositions
  - “Given that it’s raining, we’ll *probably* get wet”
- Strength or degree of a belief, or *credence*
  - “I think George is *probably* at the party”

We write

- P(A) for “credence of A”
- P(A | B) for “credence of A given B”

In general, P(A | B) ≠ P(B | A)

- P(X flies | X is a Kakapo) is **high**, P(X is a Kakapo | X flies) is **low**

A motivating example

Question: “Is my new haircut better than the old one?”

- You are completely unsure: 50/50
  - Mother of a friend says “Yes” –> new belief?
  - Your worst enemy says “Yes” –> new belief?
- You are pretty sure it’s worse than it was: 90/10
  - Mother of a friend says “Yes” –> new belief?
  - Your worst enemy says “Yes” –> new belief?

What is at play?
- Prior
- Likelihood

A motivating example

$P(\text{Improvement} \mid \text{`Yes' from mum})$ $\quad P(\text{`Yes' from mum} \mid \text{Improvement})$ $\quad P(\text{Improvement})$

$$P(\text{Improvement} \mid \text{`Yes' from mum}) = \frac{P(\text{`Yes' from mum} \mid \text{Improvement})P(\text{Improvement})}{P(\text{`Yes' from mum})}$$

Why product and not addition?

- What if a component is zero?

Why divide by *P*('Yes' from mum)?

- *P*(improvement | 'Yes' from mum) + *P*(not improvement | 'Yes' from mum)
- Posterior has to sum to 1

The components of Bayes theorem

$$\overbrace{P(H \mid D)}^{\text{Posterior}} = \frac{\overbrace{P(D \mid H)}^{\text{Likelihood}}\,\overbrace{P(H)}^{\text{Prior}}}{\underbrace{P(D)}_{\text{Evidence}}}$$

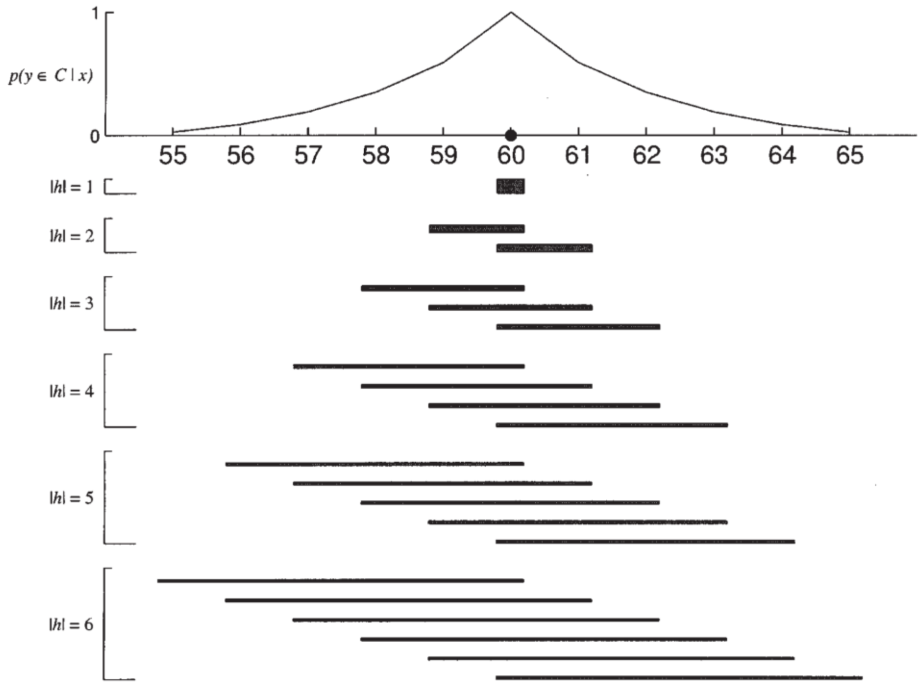
Four ingredients in Bayes theorem:

1. **Posterior** Probability of hypothesis given data
2. **Likelihood** Probability of the data *given* the hypothesis
3. **Prior** Probability of the hypothesis, NOT conditioned on data
4. **Evidence** Probability of the data, NOT conditioned on H

Case study: Simple category learning

Task: learn a category from examples.

- The space is simply the integers from 1 to 50
- The examples are numbers from the category
- The category is a *convex* region
- We get examples from inside the category



Tenenbaum & Griffiths (2001)

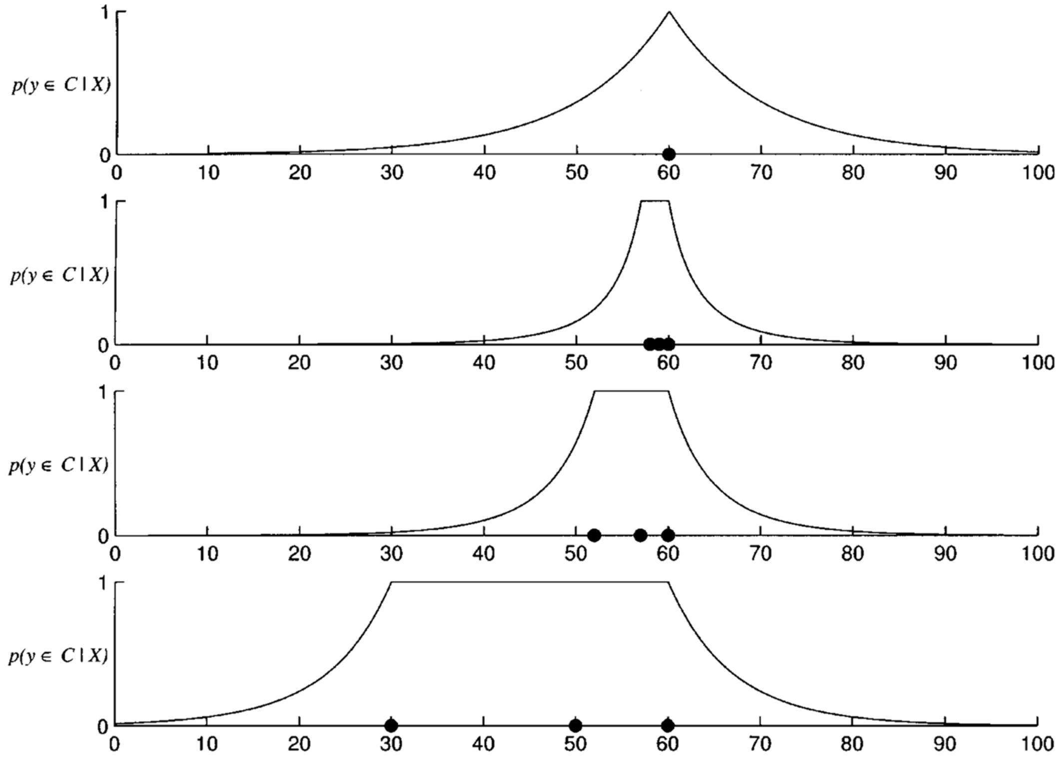
Bayesian category learning

- What's the space of hypotheses?
- What's the posterior, likelihood, and prior?
- What happens if we get more observations?

Case study: Simple category learning

Strong sampling –> *size effect*

- Can you see why intuitively?
- Can you see why formally?



Tenenbaum & Griffiths (2001)

Approximate Bayesian inference

The problem: Bayesian evidence

Bayes theorem again:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)} = \frac{P(D \mid H)P(H)}{\sum_h P(D \mid h)P(h)}$$

E.g., consider:

- P(positive test | sick) = 0.9,
- P(positive test | not sick) = 0.1, P(sick) = 0.1.
- We can calculate P(sick | positive test)

Sum/integrate across all hypotheses is not possible except simplest cases!

But in general, we need an alternative approach.

Note: We care about expectations

Note: all we need is expectations of functions of the posterior.

Suppose we have a bunch of samples $x_1, \ldots, x_N$ from the posterior. Then:

$$\int f(x)P(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

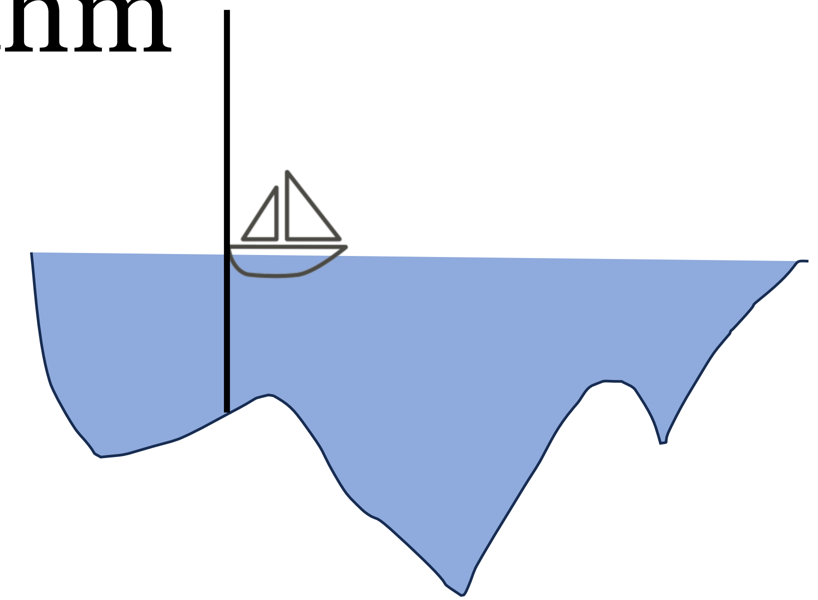
This is *Monte Carlo Integration*

New question: Can we get *posterior samples* knowing only…

- Prior
- Likelihood

Answer: We can! With Monte Carlo Markov Chain algorithms

Metropolis-Hastings algorithm

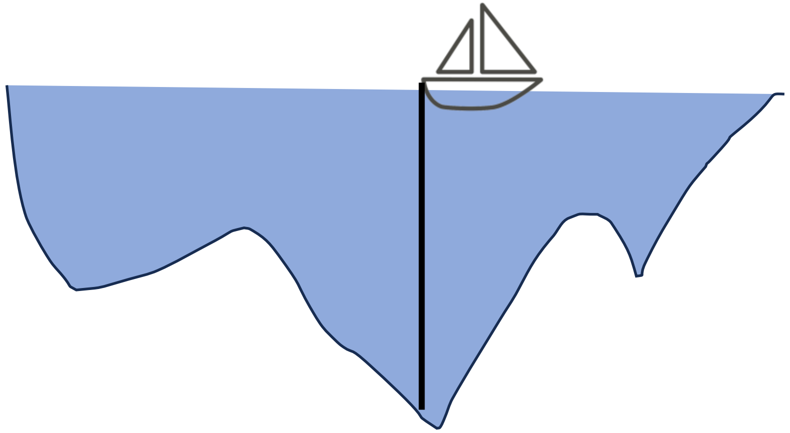


You are on a ship on a lake. You can

- poke the bottom of the lake with a stick
- determine its depth.

Problem:

- Write down a list of points on the lake
- With a frequency proportional to their depth



How would you go about doing this?

Do you see why this is equivalent to the problem we have?

Metropolis-Hastings algorithm

- Start at (any) $P_{current}$
- Then for i=1; i < N; i++:
  - Try a different point $P_{proposed}$ following a proposal distribution
    Proposal can depend on $P_{current}$
  - If depth($P_{proposed}$) > depth($P_{current}$):
    - Move to $P_{proposed}$, i.e., set $P_{current} = P_{proposed}$
  - Else:
    - Move to $P_{proposed}$ with probability depth($P_{proposed}$) / depth($P_{current}$)
    - If they're almost the same, move with higher probability, etc.

Metropolis-Hastings is just this, but instead of depth we have probability!

Asymmetric proposal distribution

Proposal distribution

Candidate state

$$A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$$

Current state

Markov Chain Monte Carlo

If some pretty weak conditions are satisfied, in the limit of infinite samples the distribution of samples converges to the true posterior distribution.

MCMC: a way of getting samples from the posterior

- without knowing the normalization constant
- i.e., the *Bayesian evidence*.

With enough samples, we can accurately approximate the posterior.

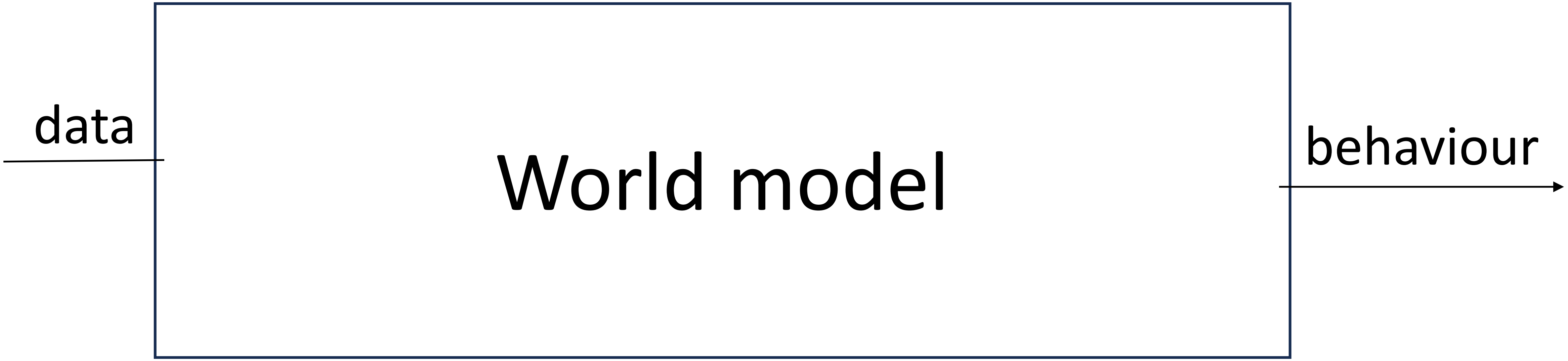
Putting it all together

Our grand plan for the pLoT

Much of cognition consists in manipulating world models

But the models have to be *encoded* somehow

Claim: Our world models are encoded in the LoT



Proposals for learning w/ a grammar

We can use MH to infer a posterior over models given observations.

- Start from some tree
- Select a subtree of the parse tree at random
- Regenerate the proposal w/ same CFG and auxiliary prob σ
- Accept with prob: max of 1 and

Goodman et al (2008)

$$\boxed{\frac{P(o \mid T')P(T' \mid \mathrm{PCFG})}{P(o \mid T)P(T \mid \mathrm{PCFG})}} \cdot \boxed{\frac{|T|}{|T'|} \cdot \frac{P(T \mid \mathrm{CFG}, \sigma)}{P(T' \mid \mathrm{CFG}, \sigma)}}$$

$$A(x', x_t) = \min\left(1, \boxed{\frac{P(x')}{P(x_t)}} \boxed{\frac{g(x_t \mid x')}{g(x' \mid x_t)}}\right)$$

Conclusions

We've learned about

- PCFGs
- their interpretation
- Bayesian category learning

We can use this as a model of the pLoT

Next session:

- Combine into computational 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

If there's time left…

The grand plan of the pLoT

Interpreted PCFG	<- Models the Language of Thought
Defines **hypotheses** *H*	<- Sentences in the LoT
Defines **prior** over *H*	<- Prior probability of LoT sentences
Observations *O*	<- Data for learning
Generated by unknown *H*	<- To be represented in the LoT
Likelihood $P(O \mid H)$	<- From a world model
Bayesian inference	<- Gives us a distribution over LoT sentences
Gets $P(H \mid O)$	Our best guess for the representation content!

Type theory

(This part might be confusing!)

Type theory

- For reasons that will become clear soon, we associate each of our expressions with a *type*.
- Let's define the set of types:
  - e and t are types
  - If σ and τ are types, then $< \sigma, \tau >$ is a type
  - Nothing else is a type
- And how to interpret them:
  - e refers to the set of individuals
  - t refers to the set of truth values
  - $< \sigma, \tau >$ refers to the set of functions from objects of type σ to objects of type τ

Type theory

Let's consider some expressions and what type they are:

- $\lambda x. P(x)$, where P is a predicate and x an individual.
  - $< e, t >$
- $\lambda x. \lambda y. Q(x, y)$, where Q is a predicate with two arguments and x and y individuals
  - $< e, < e, t \gg$
- $\lambda X. X(a)$, where a is an individual and X is a predicate
  - $\ll e, t >, t >$
- $\lambda X. \lambda Y. X(a) \wedge Y(a)$, where a is an individual and X and Y predicates
  - $\ll e, t >, \ll e, t >, t \gg$
- Basically, it can get as complicated as you want!

Type theory

In order to keep things tidy, we can put domain restrictions after a colon.

Therefore, we can write the type of each argument of a lambda function as follows:

- $\lambda x: x \in e. P(x)$, where P is a predicate
- $\lambda x: x \in e. \lambda y: y \in e. Q(x, y)$, where Q is a predicate with two arguments
- $\lambda X: X \in < e, t >. X(a)$, where a is an individual
- $\lambda X: X \in < e, t >. \lambda Y: Y \in < e, t >. X(a) \wedge Y(a)$

Sometimes, you'll also see the type written as a subfix:

- $\lambda x e. P(x)$

Language to grammar

- We've seen grammar –> language
- We can also go language –> grammar
- Let's try to write a grammar that produces all the palindromes in {a, b}*

1. *N* (nonterminal symbols)	1. {S}
2. Σ (terminal symbols)	2. {a, b}
3. *P* (production rules)	3. 1. S → aSa 2. S → bSb 3. S → *e* 4. S → a 5. S → b
4. *S* (start symbol)	

Language to grammar

- All strings with the form: $a^n b^n$

1. *N* (nonterminal symbols)
2. Σ (terminal symbols)
3. *P* (production rules)
4. *S* (start symbol)

1. {S}
2. {a, b}
3. 1. S → aSb
   2. S → *e*