Advances in Abstract Categorial Grammars

Language Theory and Linguistic Modeling

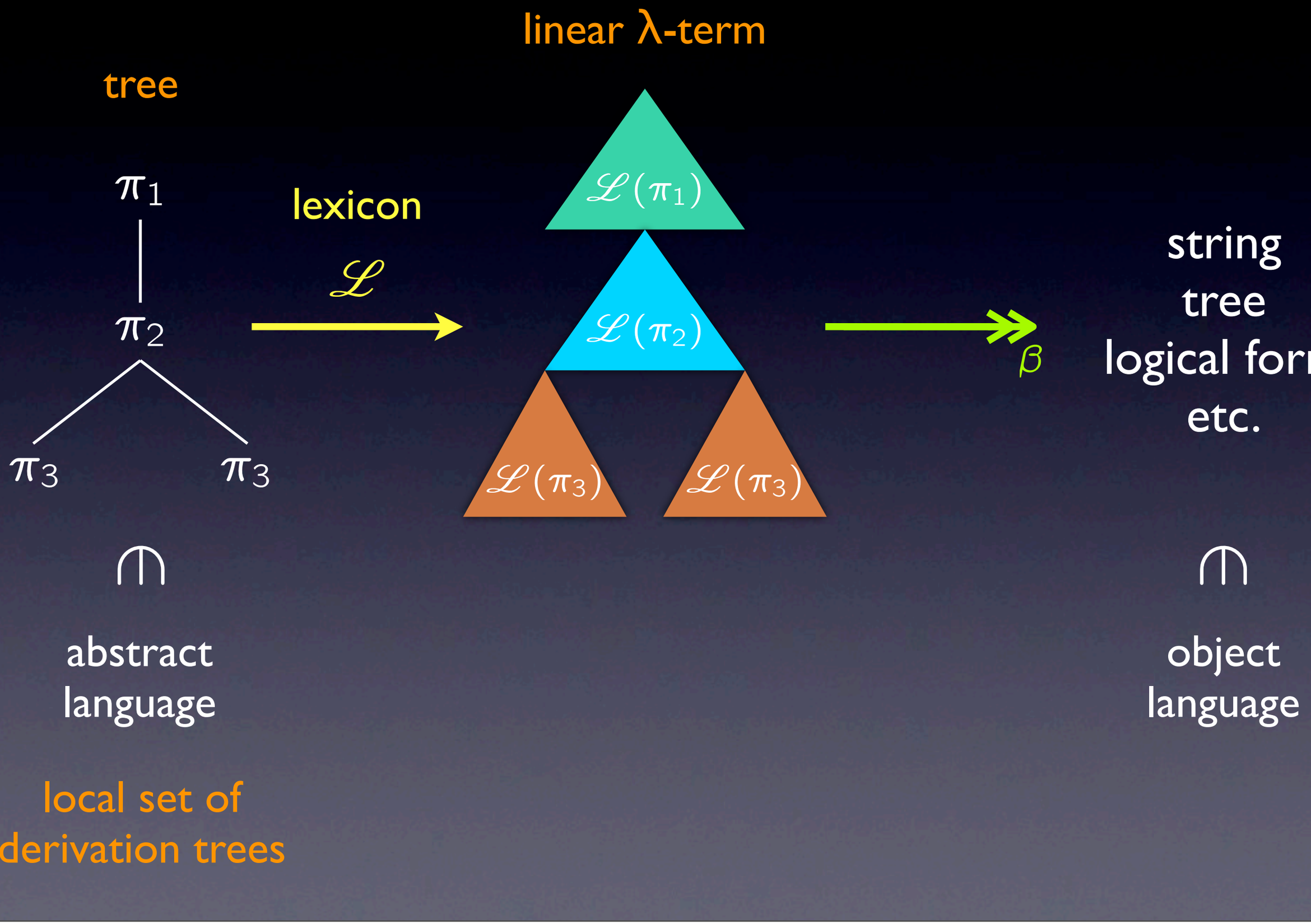
Lecture 1

This course

- Formal properties of second-order ACGs
  - Characterization of generative power
  - Datalog representation
  - Earley recognizer
- Linguistic modeling
  - ACG perspective on syntax-semantics interface

The theory of general ACGs is very difficult, but the special case of second-order ACGs is almost completely understood.

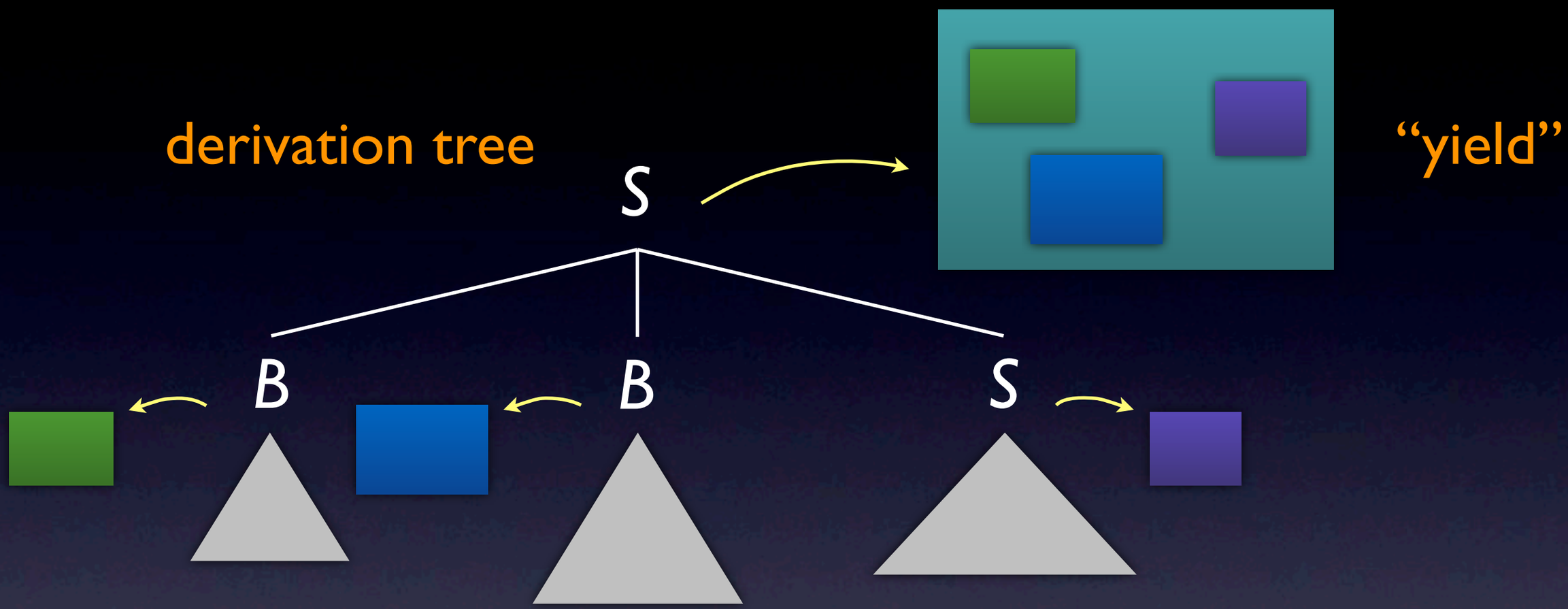
Second-order ACGs



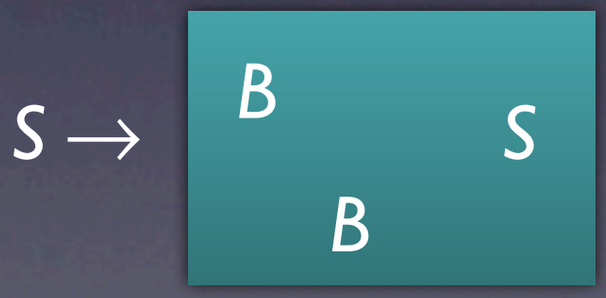
This is what a second-order ACG officially looks like.
Second-order ACGs are “context-free” grammars on linear lambda-terms.
Use notation that makes ACGs closer to other “context-free” formalisms.

“Context-free” grammar formalisms

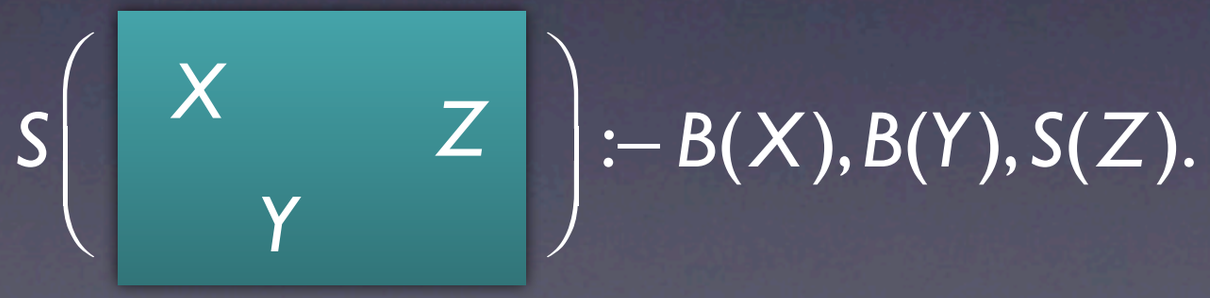
derivation tree



“yield”



top-down view

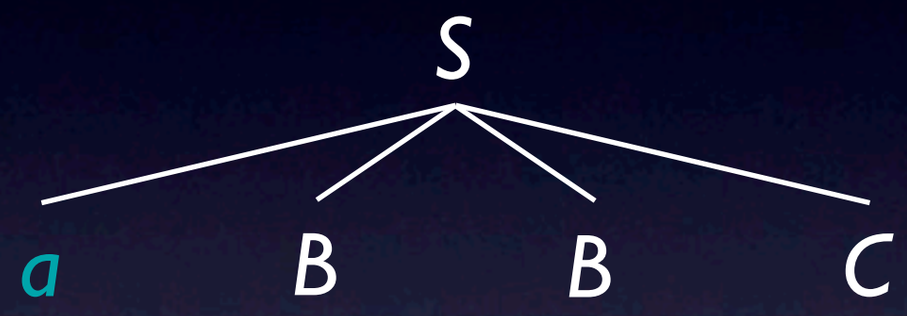


bottom-up view

Context-free grammars

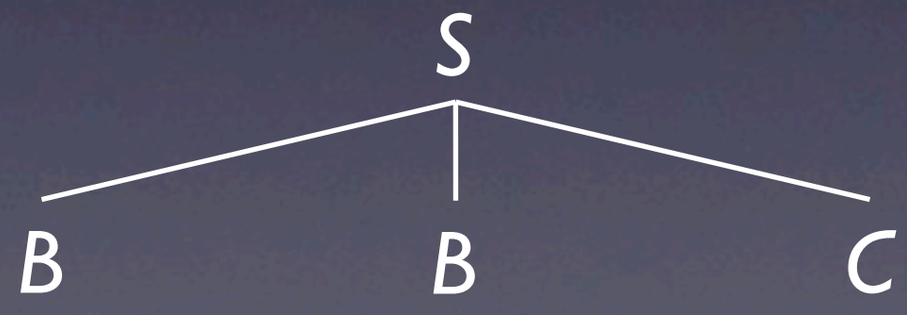
$S \rightarrow aBBS$

top-down view

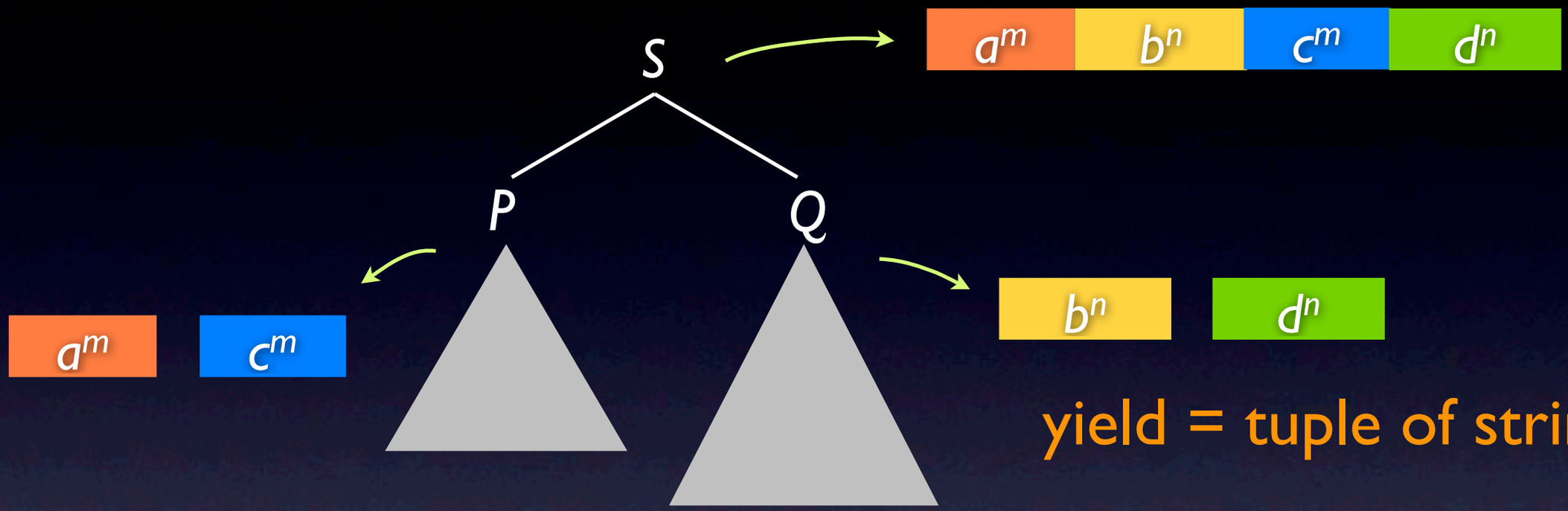


$S(aXYZ) :- B(X), B(Y), S(Z).$

bottom-up view



Multiple context-free grammars



derivation tree

$S(x_1 y_1 x_2 y_2) :- P(x_1, x_2), Q(y_1, y_2).$ bottom-up

$P(\varepsilon, \varepsilon).$ $Q(\varepsilon, \varepsilon).$

$P(ax_1, cx_2) :- P(x_1, x_2).$ $Q(by_1, dy_2) :- Q(y_1, y_2).$

This is an example of a 2-MCFG.
An m-MCFG allows nonterminals to take up to m arguments.

m-multiple context-free grammars

Seki et al. 1991

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

$$\rho(S) = 1 \qquad \rho: N \rightarrow [1,m]$$

$$B(t_1, \ldots, t_r) :- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n}).$$

- $\rho(B) = r, \rho(B_i) = r_i$
- $t_1 \ldots t_r \in (\Sigma \cup X)^*$
- Each $x_{i,j}$ occurs at most once in $t_1 \ldots t_r$

$$L(G) = \{ w \in \Sigma^* \mid P \vdash S(w) \}$$

Top-down vs. bottom-up

Type 0	EFS	Smullyan 1961
Type 1 = CSG	l.b. EFS ≡ CSG	Arikawa et al. 1989
	simple LMG ≡ P	Groenink 1997
	PMCFG	Seki et al. 1991, Groenink 1997
	MCFG	Seki et al. 1991, Groenink 1997
Type 2 = CFG	simple EFS ≡ CFG	Arikawa 1970
Type 3		

rewriting systems

logic programs on strings

Actual world

Thue → Post → Chomsky (1956) → Not much happens for a long time

Smullyan (1961)

Better possible world

Smullyan → Chomsky → Formal grammar flourishes

Second-order ACGs as context-free grammars on λ-terms

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

- $\Sigma = (A, C, \tau)$ is a higher-order signature
  - A is a set of atomic types
  - C is a set of constants
  - $\tau: C \rightarrow \mathcal{T}(A)$

$$\alpha, \beta \in \mathcal{T}(A) \Rightarrow \alpha \rightarrow \beta \in \mathcal{T}(A)$$

- $\sigma: N \rightarrow \mathcal{T}(A)$

complexity = max(ord(σ(*B*)))

Second-order ACGs as context-free grammars on λ-terms

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

- Rules in P are of the form

$$B(M) :- B_1(X_1), \ldots, B_n(X_n).$$

where $X_1 : \sigma(B_1), \ldots, X_n : \sigma(B_n) \vdash_\Sigma M : \sigma(B)$.

- $L(G) = \{ |M|_\beta \mid P \vdash S(M) \}$

β-normal form

Typed linear λ-calculus

$$\frac{}{x : \alpha \vdash_\Sigma x : \alpha} \qquad \frac{}{\vdash_\Sigma c : \tau(c)}$$

$$\frac{\Gamma \vdash_\Sigma M : \alpha \quad \Delta \vdash_\Sigma N : \beta}{\Gamma \cup \Delta \vdash_\Sigma MN : \beta} \text{ if } \Gamma \cap \Delta = \varnothing \qquad \frac{\Gamma, x : \alpha \vdash_\Sigma M : \beta}{\Gamma \vdash_\Sigma \lambda x.M : \alpha \to \beta}$$

MNP means (*MN*)*P*

$\lambda x_1 \ldots x_n.M$ means $\lambda x_1.(\ldots(\lambda x_n.M)\ldots)$

$\alpha_1 \to \cdots \to \alpha_n \to \beta$ means $\alpha_1 \to (\cdots \to (\alpha_n \to \beta)\cdots)$

Second-order ACG

$S \longrightarrow q(\lambda x_1.q(\lambda x_2.fx_1(gx_2)))$

$P \longrightarrow \lambda x_1.q(\lambda x_2.fx_1(gx_2))$

$R \longrightarrow \lambda x_1 x_2.fx_1(gx_2)$

$P \longrightarrow \lambda x.gx$

$\tau(q) = (o \to o) \to o$

$\tau(f) = o \to o \to o$

$\tau(g) = o \to o$

$\sigma(S) = o$

$\sigma(P) = o \to o$

$\sigma(R) = o \to o \to o$

$S(q(\lambda x.Xx)) :\!- P(X).$

$P(\lambda x_1.q(\lambda x_2.Yx_1x_2) :\!- R(Y).$

$P(\lambda x_2.q(\lambda x_1.Yx_1x_2)) :\!- R(Y).$

$P(\lambda x.gx).$

$R(\lambda x_1 x_2.fx_1(Xx_2)) :\!- P(X).$

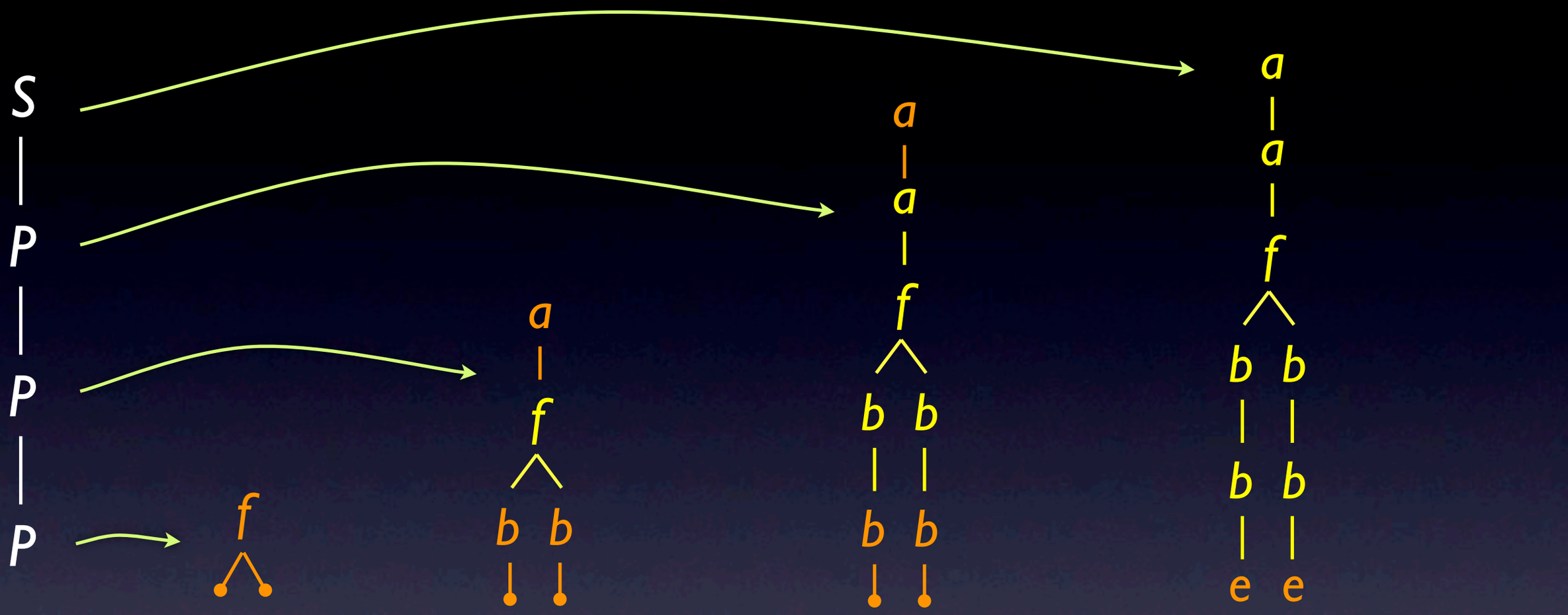
complexity 2

“Context-free” grammars on …

	strings	trees	unary tree contexts $C[x_1]$	n-ary tree contexts $C[x_1,\ldots,x_n]$
single	CFG	RTG	TAG (monadic LCFTG)	LCFTG
multiple	MCFG	MRTG	MCTAG	MLCFTG

- RTG: regular tree grammar
- LCFTG: linear context-free tree grammar
- MCFG: multiple context-free grammar
- MRTG: multiple regular tree grammar
- MCTAG: multi-component tree-adjoining grammar
- MLCFTG: multiple linear context-free tree grammar

Linear Context-Free Tree Grammar



$$S\left(\begin{array}{c} Y \\ \wedge \\ e\ \ e \end{array}\right) :\!-\ P(Y).$$

$$P\left(\begin{array}{c} a \\ | \\ Y \\ \wedge \\ b\ \ b \\ |\ \ | \end{array}\right) :\!-\ P(Y).$$

$$P\left(\begin{array}{c} f \\ \wedge \end{array}\right).$$

Ranked trees

$$\Delta = \bigcup_{r \in \mathbb{N}} \Delta^{(r)}$$

ranked alphabet

$$f \in \Delta^{(n)}, T_1, \ldots, T_n \in \mathbb{T}_\Delta \Rightarrow (f T_1 \ldots T_n) \in \mathbb{T}_\Delta$$

$f(T_1, \ldots, T_n)$

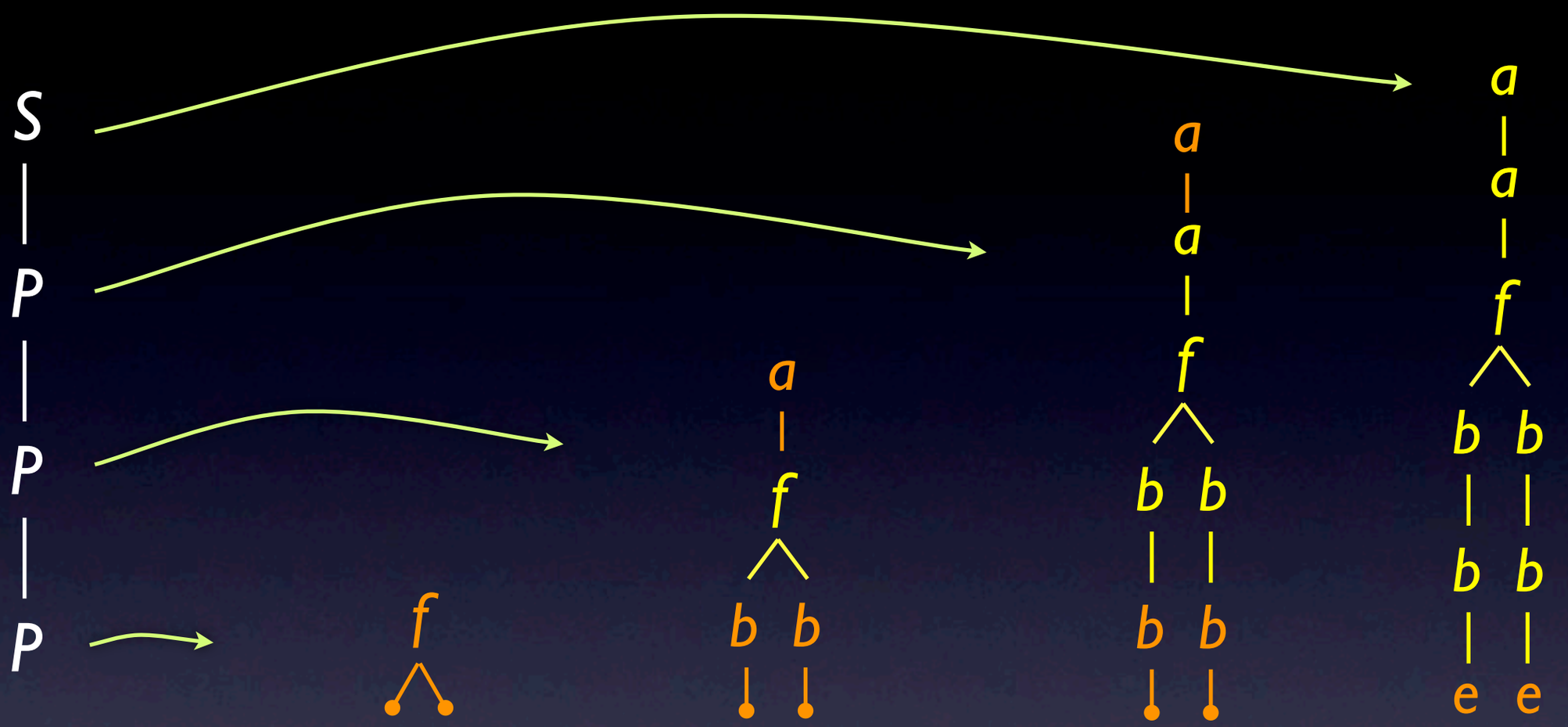
Tree signature

$$\Sigma_{\Delta}^{\text{tree}} = (\{o\}, \Delta, \tau_{\Delta})$$

$$f \in \Delta^{(n)} \Rightarrow \tau_{\Delta}(f) = o^n \rightarrow o$$

- Trees over Δ are β-normal closed λ-terms of type *o*
- *n*-ary tree contexts over Δ are β-normal closed λ-terms of type $o^n \rightarrow o$

LCFTG as second-order ACG

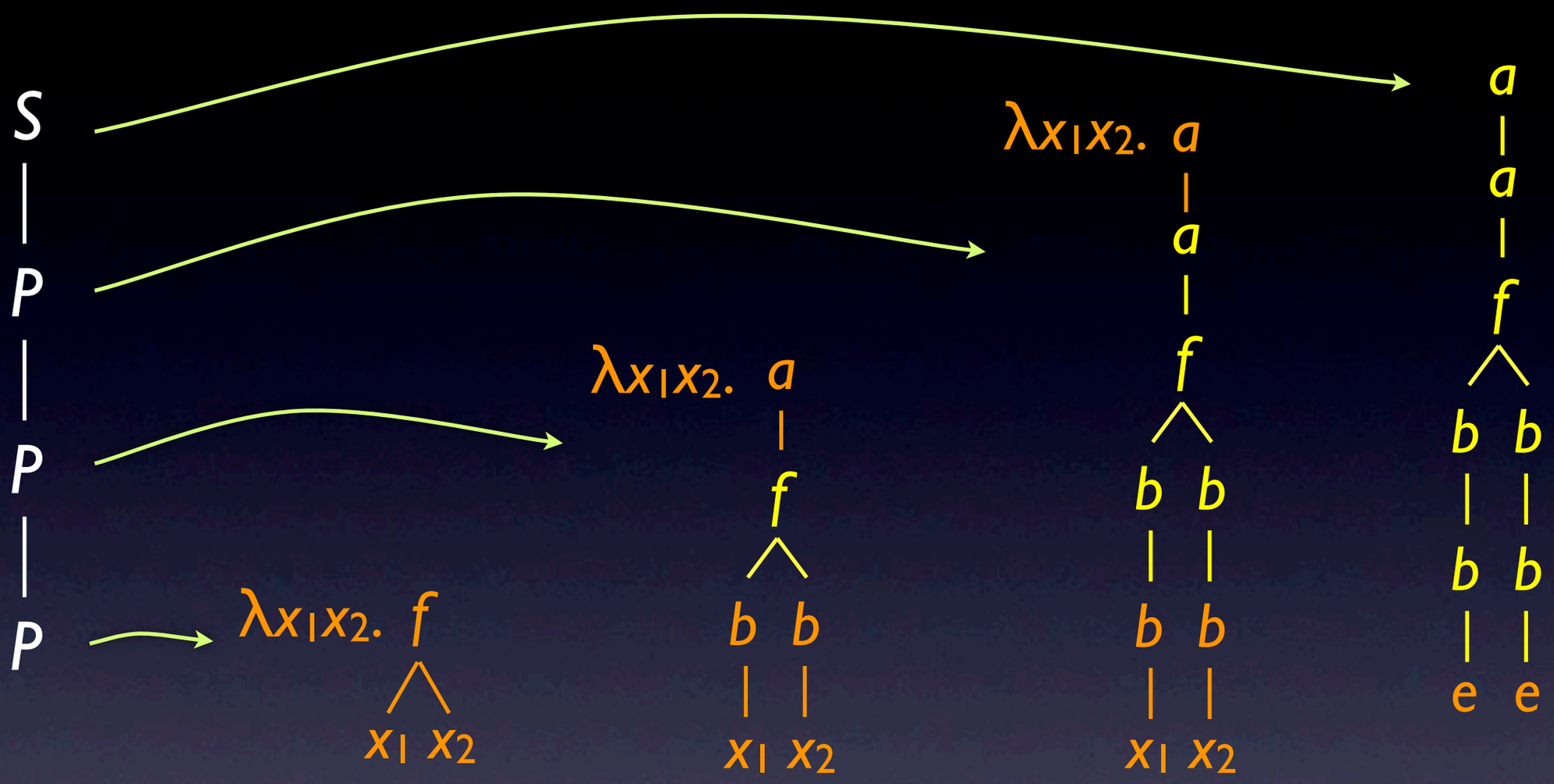


$$S\left(\begin{array}{c} Y \\ \wedge \\ e\ \ e \end{array}\right) :\!-\ P(Y).$$

$$P\left(\begin{array}{c} a \\ | \\ Y \\ \wedge \\ b\ \ b \end{array}\right) :\!-\ P(Y).$$

$$P\left(\begin{array}{c} f \\ \wedge \end{array}\right).$$

LCFTG as second-order ACG



$$S\left(\begin{array}{c} Y \\ \wedge \\ e\ \ e \end{array}\right) :\!- P(Y).$$

$$P\left(\begin{array}{c} \lambda x_1 x_2.\ a \\ | \\ Y \\ \wedge \\ b\ \ b \\ |\ \ | \\ x_1\ x_2 \end{array}\right) :\!- P(Y).$$

$$P\left(\begin{array}{c} \lambda x_1 x_2.\ f \\ \wedge \\ x_1\ x_2 \end{array}\right).$$

$\sigma(P) = o \to o \to o$
complexity 2

Context-free grammars usually do not use lambda.
Cleaner notation with lambda.

second-order
ACGs of
complexity 2

$$TR(\mathrm{ACG}_{(2,2)}) \supseteq \mathrm{LCFTL}$$

tree
generating
power

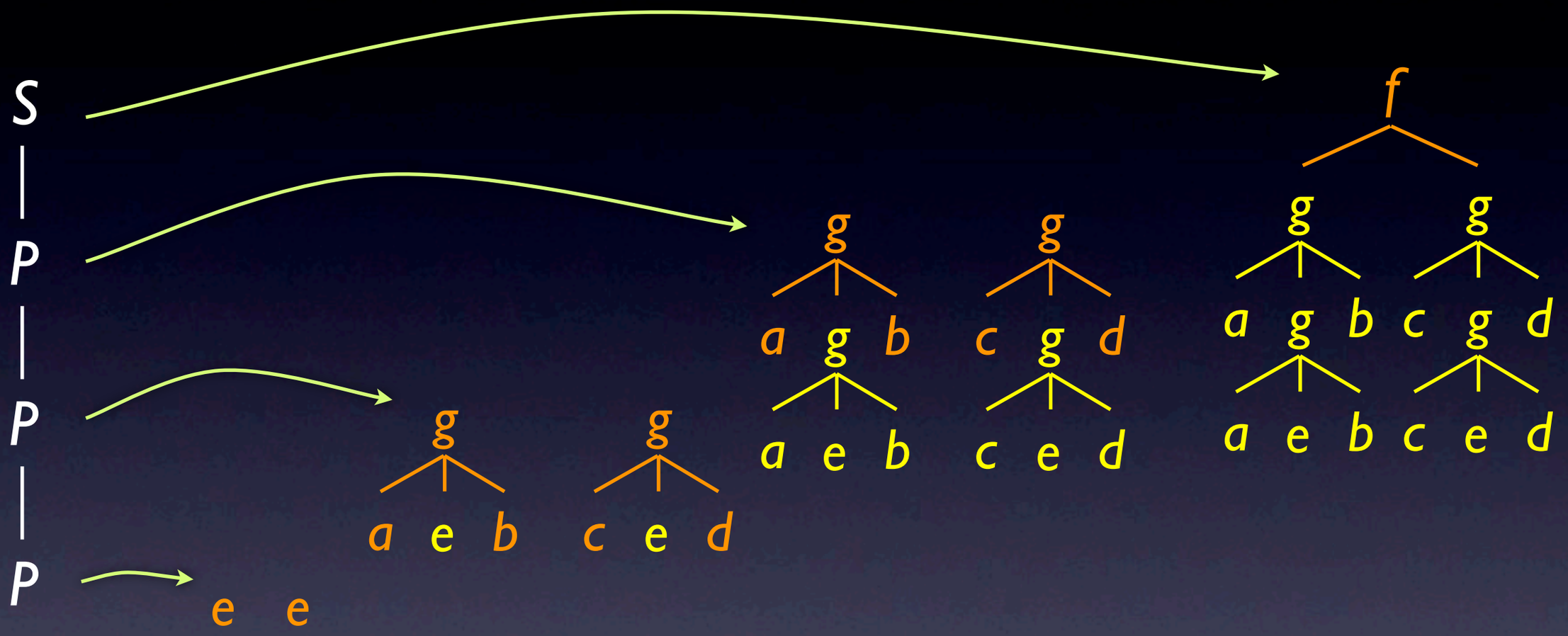
- Converse can be shown as well.

second-order
ACGs of
complexity 2

$$TR(\mathrm{ACG}_{(2,2)}) = \mathrm{LCFTL}$$

tree
generating
power

Multiple regular tree grammar



$$S\left(\begin{array}{c} f \\ X_1\ X_2 \end{array}\right) :\!-\ P(X_1, X_2).$$

$$P\left(\begin{array}{c} g \\ a\ X_1\ b \end{array}, \begin{array}{c} g \\ c\ X_2\ d \end{array}\right) :\!-\ P(X_1, X_2).$$

$$P(e, e).$$

Encoding tuples

M_1, M_2: linear λ-terms of type α_1, α_2

$$\langle M_1, M_2 \rangle \;=\; \lambda w.wM_1M_2$$

linear

type $(\alpha_1 \multimap \alpha_2 \multimap \beta) \multimap \beta$ for any β

Usage: $\langle M_1, M_2 \rangle\, (\lambda x_1 x_2.C[x_1, x_2]) \twoheadrightarrow_\beta C[M_1, M_2]$

$C[\ ,\ ]$ must be of type β

MRTG

$$S\left(\begin{array}{c} f \\ X_1\ X_2 \end{array}\right) \coloneq P(X_1, X_2).$$

$$P\left(\begin{array}{ccc} g & & g \\ a\ X_1\ b & , & c\ X_2\ d \end{array}\right) \coloneq P(X_1, X_2).$$

ACG

$$S\left(X\left(\lambda x_1 x_2 .\ \begin{array}{c} f \\ x_1\ x_2 \end{array}\right)\right) \coloneq P(X).$$

context of type o

$$\times \quad P\left(X\left(\lambda x_1 x_2 .\lambda w.w\left(\begin{array}{c} g \\ a\ x_1\ b \end{array}\right)\left(\begin{array}{c} g \\ c\ x_2\ d \end{array}\right)\right)\right) \coloneq P(X).$$

context of type $(o \to o \to o) \to o$

$P(e, e).$

$P(\lambda w.wee).$

Encoding tuples

M_1, M_2: linear λ-terms of type α_1, α_2

$$\langle M_1, M_2 \rangle \;=\; \lambda w.wM_1M_2$$

linear

type $(\alpha_1 \to \alpha_2 \to o) \to o$

Usage:

$$\lambda y_1 \ldots y_m.\, \langle M_1, M_2 \rangle\, (\lambda x_1 x_2.C[x_1, x_2]y_1 \ldots y_m) \twoheadrightarrow_{\beta\eta} C[M_1, M_2]$$

where $C[\ ,\ ]$ is of type $\beta_1 \to \cdots \to \beta_m \to o$

MRTG

$$S\left(\begin{array}{c} f \\ X_1\ X_2 \end{array}\right) \coloneq P(X_1, X_2).$$

$$P\left(\begin{array}{ccc} g & & g \\ a\ X_1\ b & , & c\ X_2\ d \end{array}\right) \coloneq P(X_1, X_2).$$

ACG

$$S\left(X\left(\lambda x_1 x_2.\ \begin{array}{c} f \\ x_1\ x_2 \end{array}\right)\right) \coloneq P(X).$$

context of type o

context of type $(o \to o \to o) \to o$

✗ $$P\left(X\left(\lambda x_1 x_2.\lambda w.w\left(\begin{array}{c} g \\ a\ x_1\ b \end{array}\right)\left(\begin{array}{c} g \\ c\ x_2\ d \end{array}\right)\right)\right) \coloneq P(X).$$

✓ $$P\left(\lambda w.X\left(\lambda x_1 x_2.w\left(\begin{array}{c} g \\ a\ x_1\ b \end{array}\right)\left(\begin{array}{c} g \\ c\ x_2\ d \end{array}\right)\right)\right) \coloneq P(X).$$

$\sigma(P) = (o \to o \to o) \to o$
complexity 3

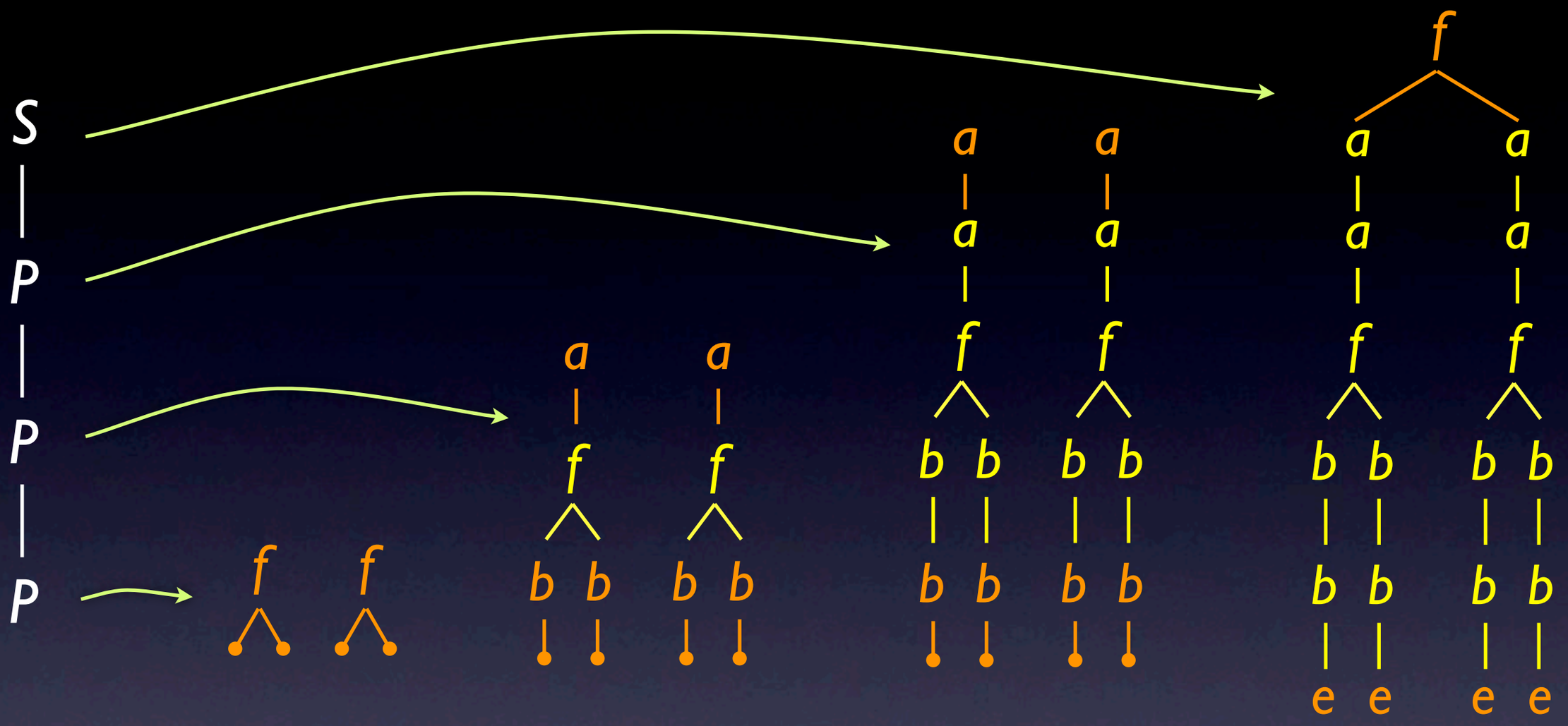
second-order
ACGs of
complexity 3

$$TR(\mathrm{ACG}_{(2,3)}) \supseteq \mathrm{MRTL}$$

tree
generating
power

- Inclusion is proper because LCFTL $\not\subseteq$ MRTL

Multiple Linear Context-Free Tree Grammar



$$S\left(\begin{array}{c} f \\ Y_1 \quad Y_2 \\ e\ e\ e\ e \end{array}\right) :\!-\ P(Y_1, Y_2).$$

$$P\left(\begin{array}{c} a \quad a \\ Y_1 \quad Y_2 \\ b\ b\ b\ b \end{array}\right) :\!-\ P(Y_1, Y_2).$$

$$P\left(\begin{array}{c} f \quad f \end{array}\right).$$

second-order
ACGs of
complexity 4

$$TR(\mathrm{ACG}_{(2,4)}) \supseteq \mathrm{MLCFTL}$$

tree
generating
power

Encoding tree grammars

$$TR(\mathrm{ACG}_{(2,4)}) \supseteq \mathrm{MLCFTL}$$

$$TR(\mathrm{ACG}_{(2,3)}) \supseteq \mathrm{MRTL} \cup \mathrm{LCFTL}$$

$$TR(\mathrm{ACG}_{(2,2)}) = \mathrm{LCFTL}$$

$$TR(\mathrm{ACG}_{(2,1)}) = \mathrm{RTL}$$

Encoding strings

- View strings as unary tree contexts over a monadic alphabet

$$/aaba/ = \lambda z.\ a(a(b(a(z))))$$

String signature

$$\Sigma_{\Upsilon}^{\text{string}} = (\{o\}, \Upsilon, \tau)$$

$$\tau(a) = o \to o$$

- Concatenation is expressed by

$$\mathbf{B} = \lambda xyz.x(yz)$$

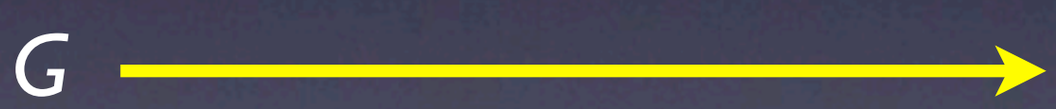
Yield

$$\text{yield} : \Sigma_{\Delta}^{\text{tree}} \to \Sigma_{\Delta^{(0)}}^{\text{string}}$$

$f \mapsto \lambda x_1 \ldots x_n z.x_1(\ldots(x_n z)\ldots)$ if $f \in \Delta^{(n)}$ for $n \geq 1$

$c \mapsto \lambda x.cx$ if $c \in \Delta^{(0)}$

$0 \mapsto 0 \to 0$



tree generating complexity n

string generating complexity $n+1$

$L(G') = yL(G)$

Encoding string grammars

?

$STR(\mathrm{ACG}_{(2,4)}) \supseteq \mathrm{MCFL} = y\mathrm{MRTL}$

$STR(\mathrm{ACG}_{(2,3)}) = y\mathrm{LCFTL}$

$STR(\mathrm{ACG}_{(2,2)}) = \mathrm{CFL} = y\mathrm{RTL}$

Salvati's Theorem

$$STR(\mathrm{ACG}_{(2,n)}) \subseteq \mathrm{MCFL}$$

- This implies (for $n \geq 5$)

$$STR(\mathrm{ACG}_{(2,n)}) = STR(\mathrm{ACG}_{(2,4)}) = \mathrm{MCFL}$$

Extraction of a tuple of strings from a λ-term

$$\vdash_{\Sigma_Y^{\text{string}}} M : \alpha$$

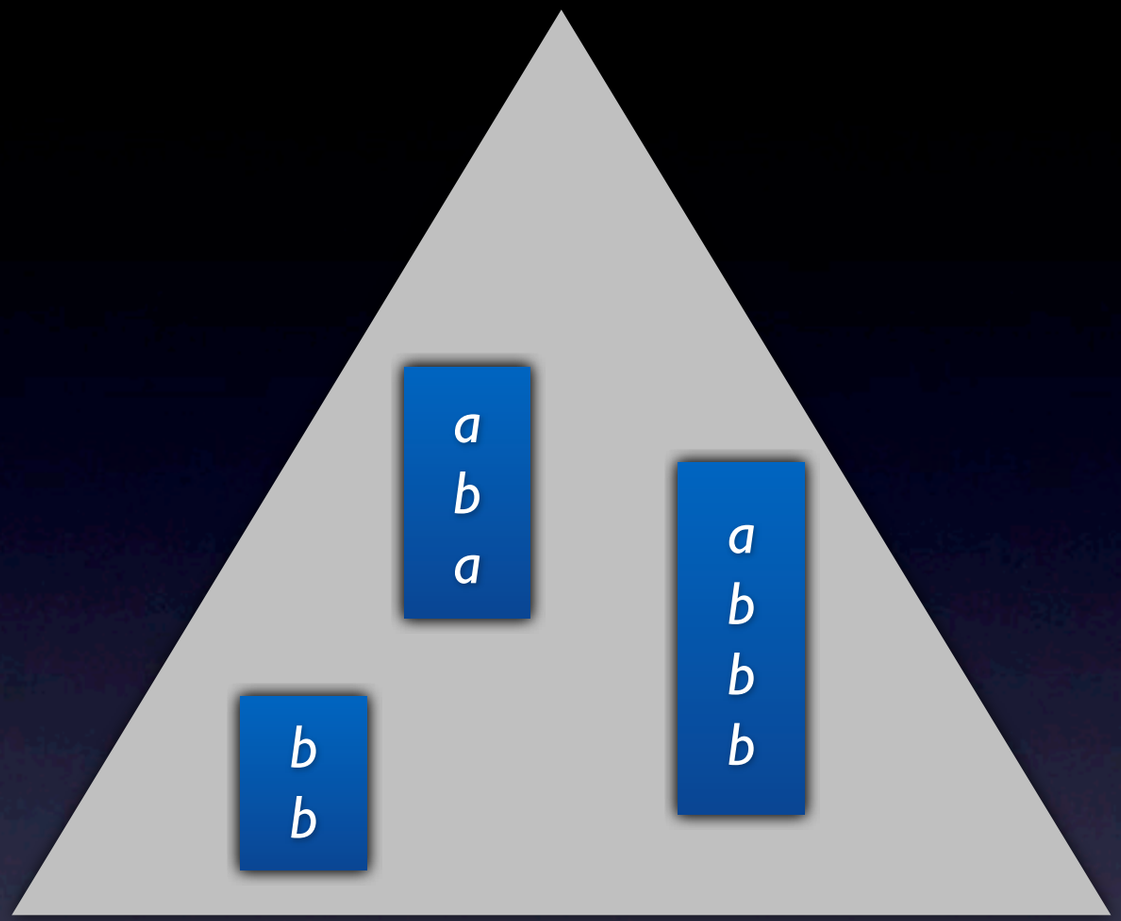
tuple of strings $(w_1, \ldots, w_m)$ $\qquad$ $P[z_1, \ldots, z_m]$ pure λ-term

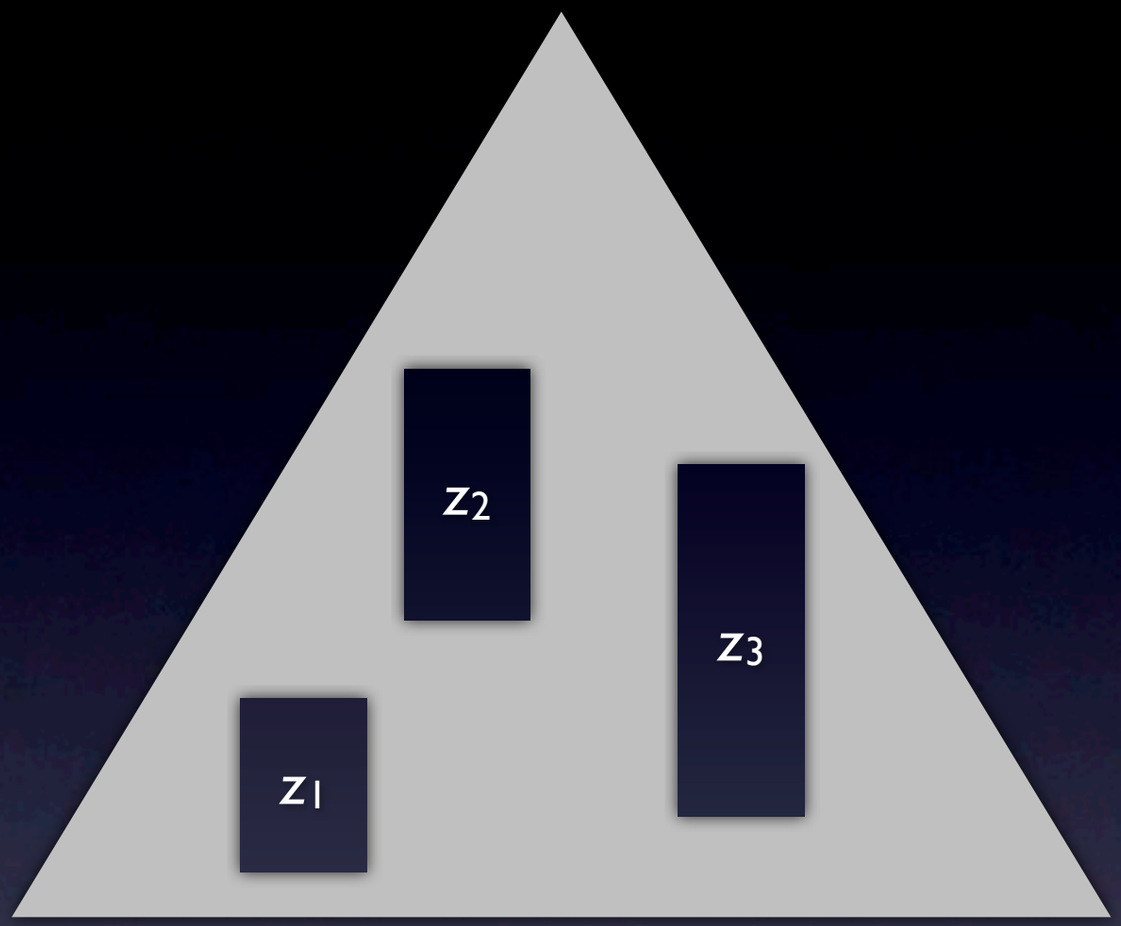
$$z_1 : o \to o, \ldots, z_m : o \to o \vdash P[z_1, \ldots, z_m] : \alpha$$

$$P[/w_1/, \ldots, /w_m/] \twoheadrightarrow_\beta M$$

$$2m \leq |\alpha|$$

Given α, there are only finitely many possibilities for $P[z_1,\ldots z_m]$.
Information carried by M is just $(w_1,\ldots,w_m)$ plus some bounded amount of information.


a
b
a
b
b
a
b
b
b



b
b

a
b
a

a
b
b
b

$$z_1 : o \to o, \ldots, z_m : o \to o \vdash P[z_1, \ldots, z_m] : \alpha$$

only finitely many

$B(M)$

$(B, P[z_1, \ldots, z_m])(w_1, \ldots, w_n)$

ACG

MCFG

- The same technique applies to the tree case.

Create a new nonterminal with B and $P[z_1,\ldots,z_m]$.
You get a tuple of tree contexts in the tree case.

Second-order ACG hierarchy

$STR(\mathrm{ACG}_{(2,n)})$ $(n \geq 5)$ $TR(\mathrm{ACG}_{(2,n)})$ $(n \geq 5)$

$\shortparallel$ $\shortparallel$

$STR(\mathrm{ACG}_{(2,4)}) = \mathrm{MCFL}$ $TR(\mathrm{ACG}_{(2,4)}) = \mathrm{MLCFTL}$

$STR(\mathrm{ACG}_{(2,3)}) = y\mathrm{LCFTL}$ $TR(\mathrm{ACG}_{(2,3)}) \supseteq \mathrm{MRTL} \cup \mathrm{LCFTL}$

$STR(\mathrm{ACG}_{(2,2)}) = \mathrm{CFL}$ $TR(\mathrm{ACG}_{(2,2)}) = \mathrm{LCFTL}$

$TR(\mathrm{ACG}_{(2,1)}) = \mathrm{RTL}$