Distributional Learning of Extensions of Context-Free Grammars

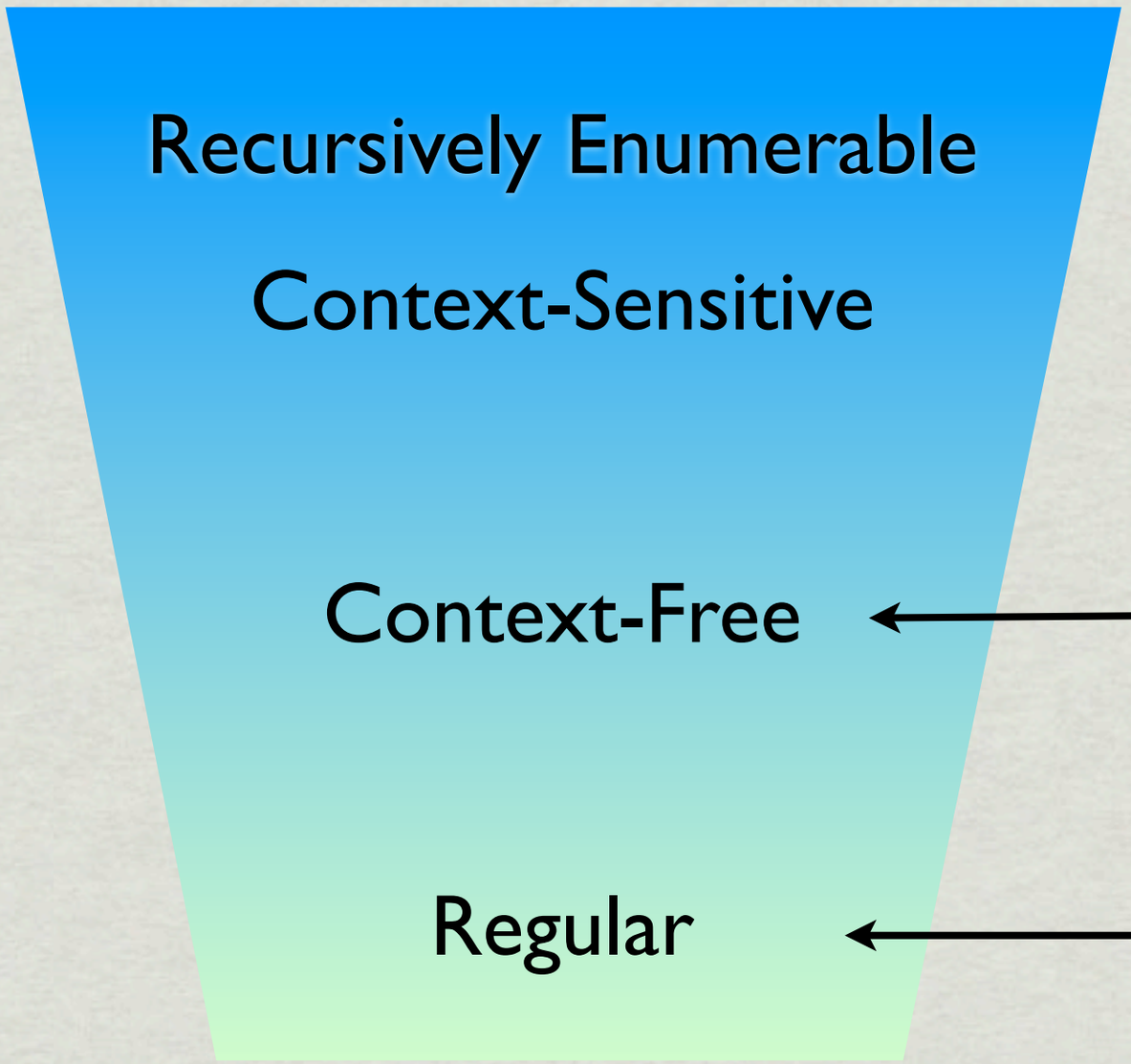
Ryo Yoshinaka
(ERATO Minato Discrete Structure Manipulation System Project,
Japan Science and Technology Agency)

Introduction

Grammatical Inference

- Algorithmic Learning of Formal Languages
  - Mathematical model of natural language acquisition
  - Grammar extraction from tagged/untagged corpora
  - Biological sequences
- More theoretical rather than heuristic
  - *Target*: (subclasses of) regular, linear, context-free, ...
  - *Resource*: positive/negative examples, queries, ...
  - *Criteria*: exact, probabilistic, ...
  - *Efficiency*: computation time, data size, ...

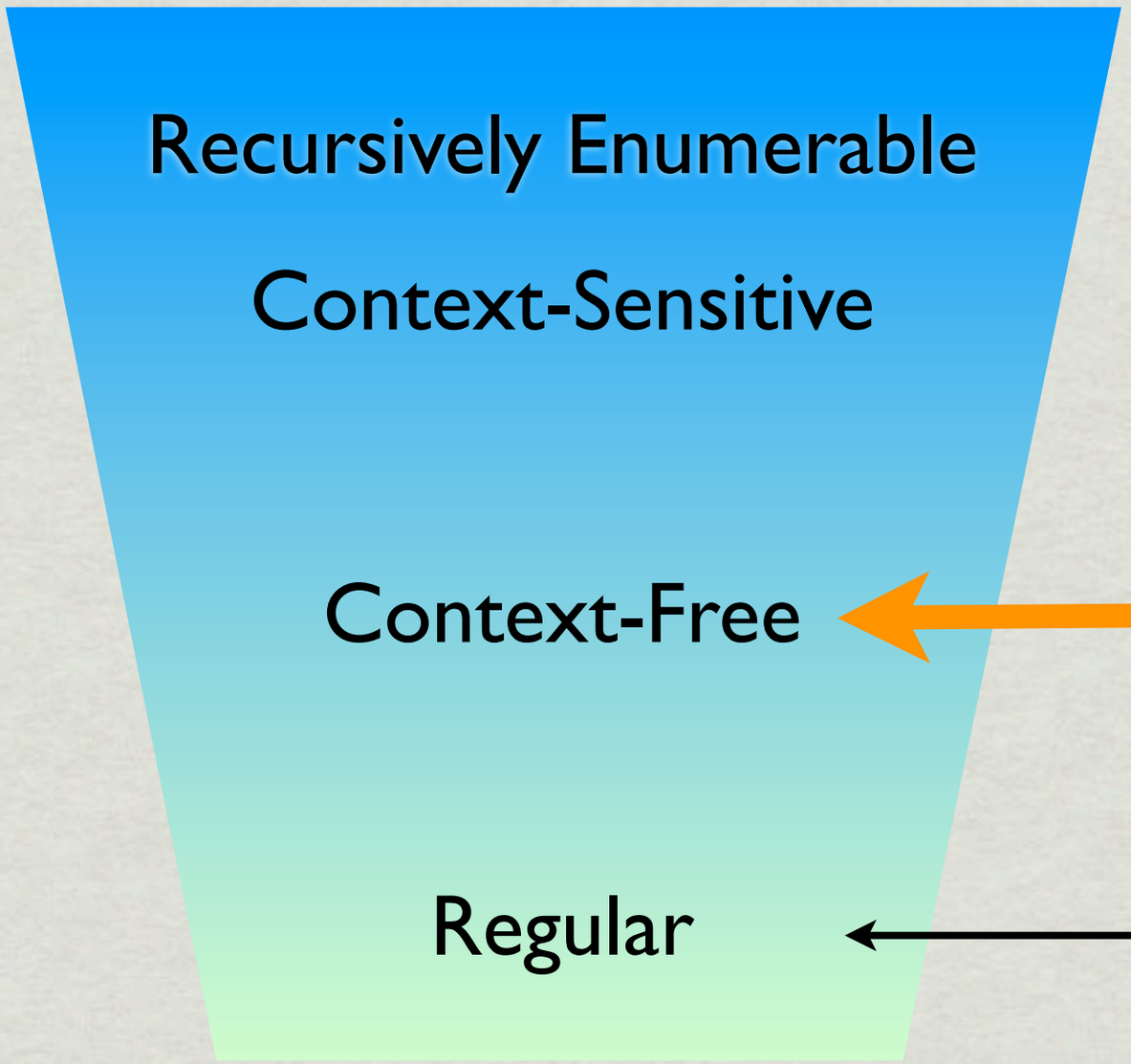
Chomsky Hierarchy & Learning



Few positive results

Many positive results

Chomsky Hierarchy & Learning



Distributional Learning

Many positive results

Distributional Learning

- Models and exploits the distribution of strings in contexts
- Syntactic category of a phrase = Contexts where it occurs

		contexts			
		John ☐ Mary	☐ loves kids	Adam hits ☐	Everyone ☐
strings	John		◎	◎	
	Mary		◎	◎	
	she		◎		
	him			◎	
	loves	◎			
	loves it				◎
	runs				◎

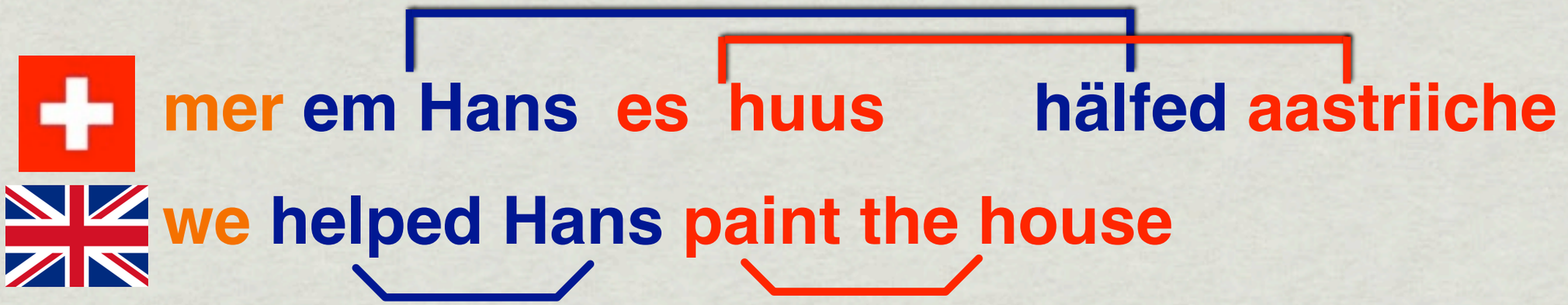
John ≡ Mary, him ≦ Mary, loves it ≡ runs, ...

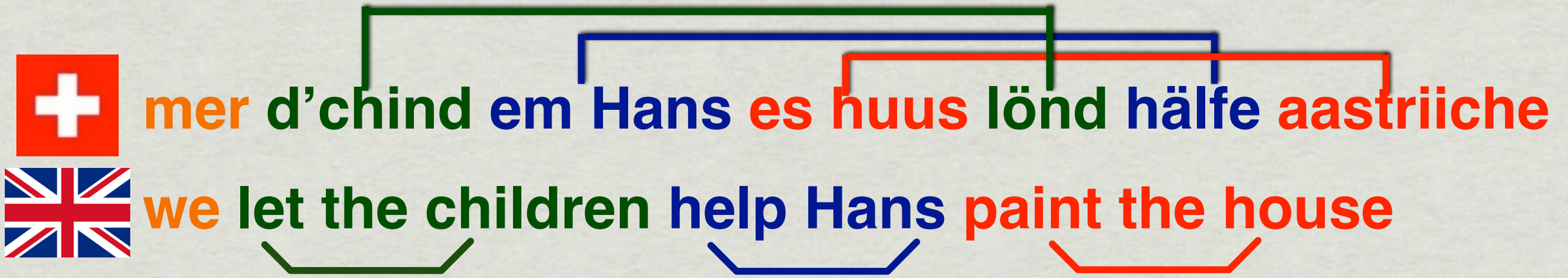
Distributional Learning

- Context-deterministic CFGs by queries (Shirakawa & Yokomori '93)
- Substitutable CFLs by positive data (Clark & Eyraud '05)
- *k*,*l*-Substitutable CFLs by positive data (Yoshinaka'08)
- Probabilistic learning of Unambiguous (*k*,*l*-)NTS Languages (Clark'06, Luque'10)
- New formalisms, learning with queries (Clark et al.'08, Clark'09)
- etc.

Non-context-free phenomena I

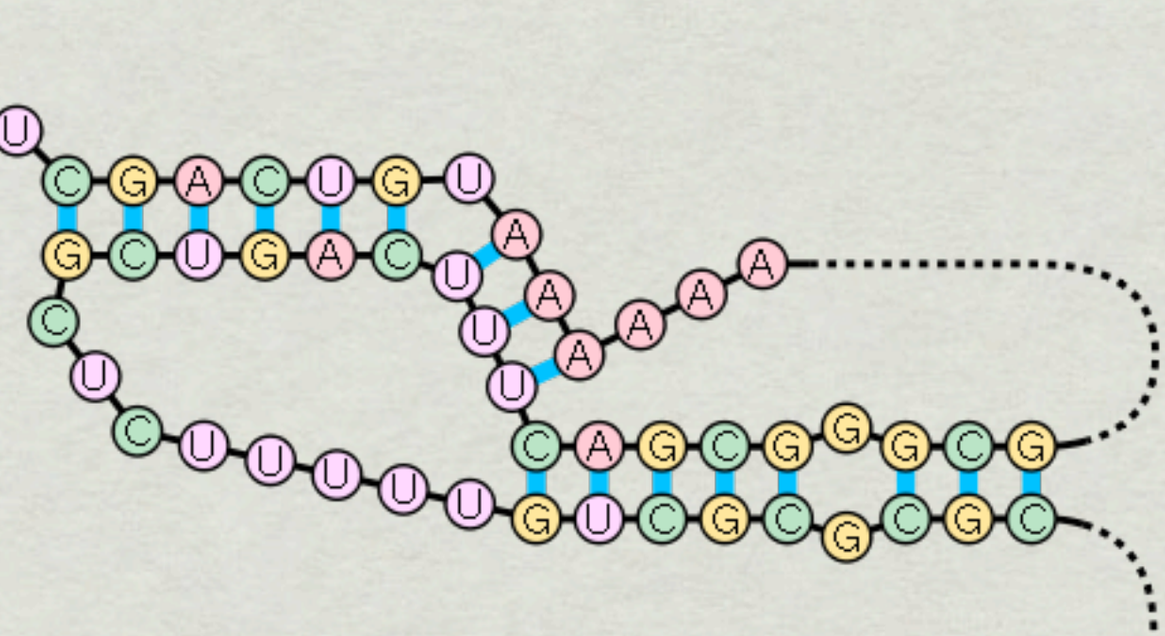
- Swiss-German





Non-context-free phenomena II

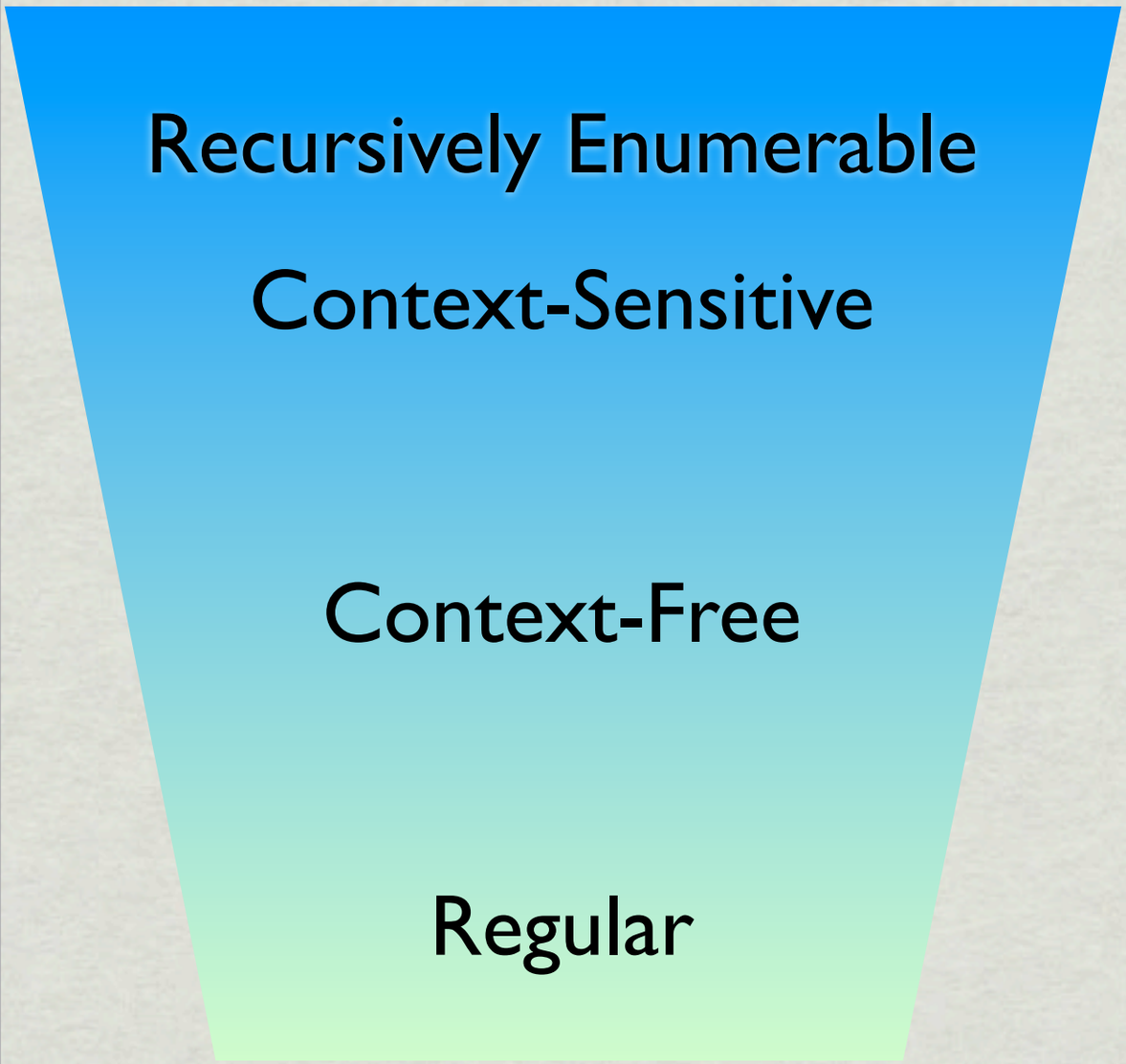
- Pseudoknots in biological sequences



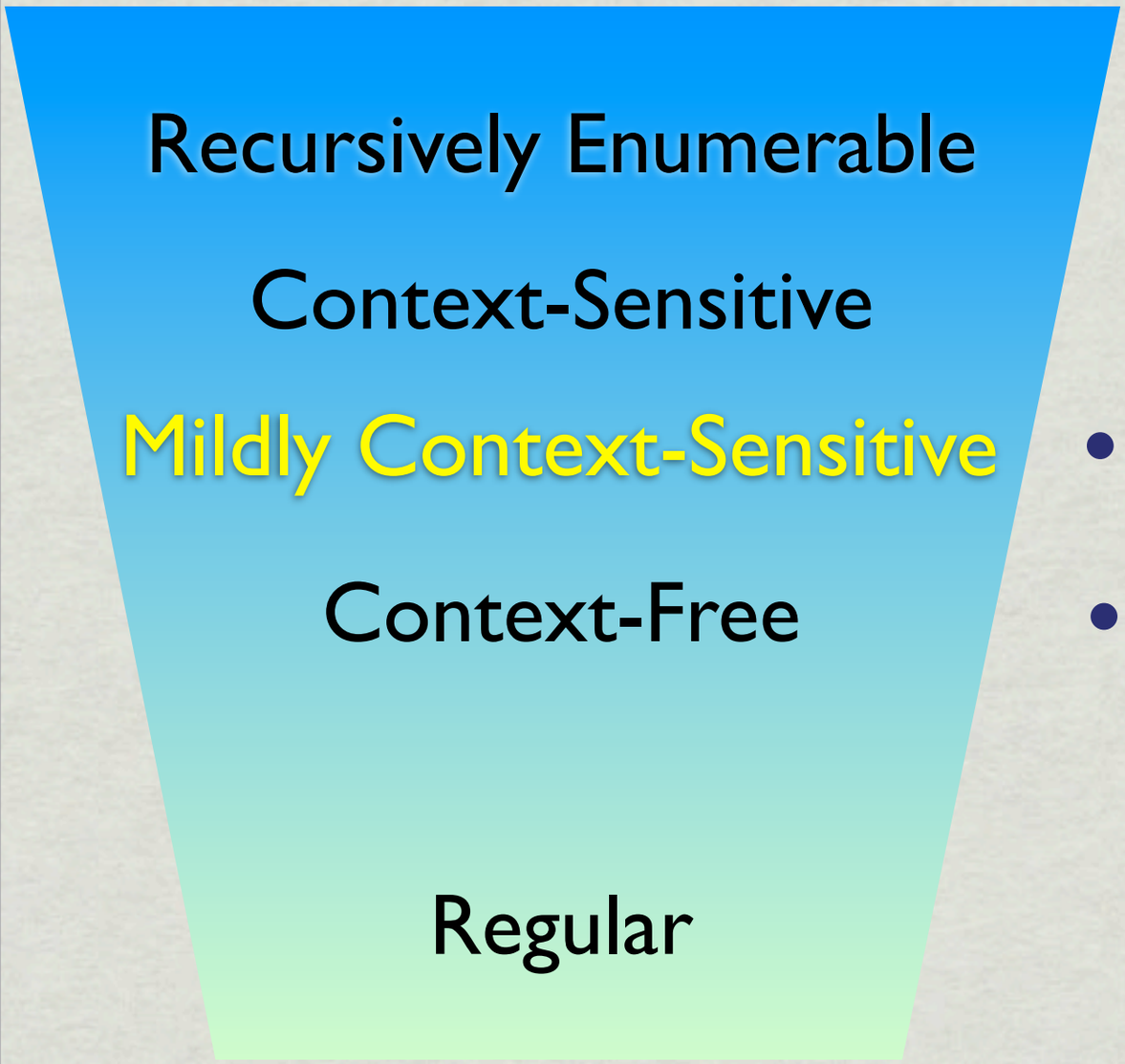
© Sakurambo via Wikimedia Commons

CGACU ... GCGAC ... AGUCG ... CGCUG

Mildly Context-Sensitiveness



Mildly Context-Sensitiveness



- Cross-serial Dependencies
- Polynomial-time Parsable

- ***Multiple*** *Context-Free Grammars*
- *Context-Free* ***Tree*** *Grammars*

Learning of Mildly Context-Sensitiveness

Learning of Mildly Context-Sensitiveness

Learning of Mildly Context-Sensitiveness

Multiple Context-Free

Multi-word

↕

Multi-Context

Context-Free Tree

Stub

↕

Tree-Context

Distributional Learning

of

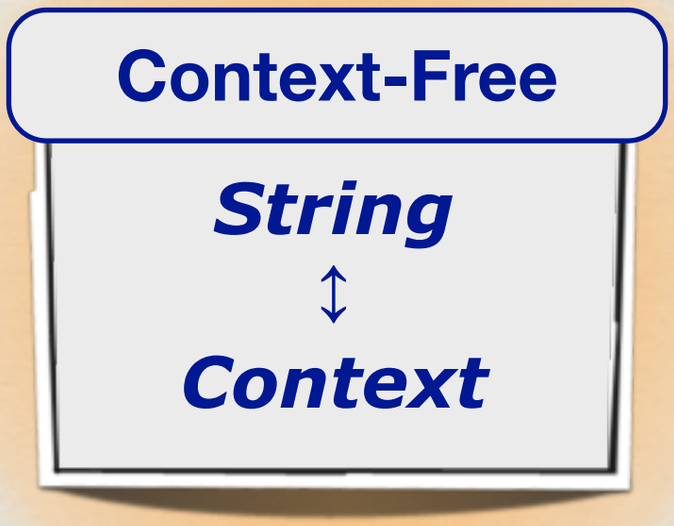
Mildly Context-Sensitive

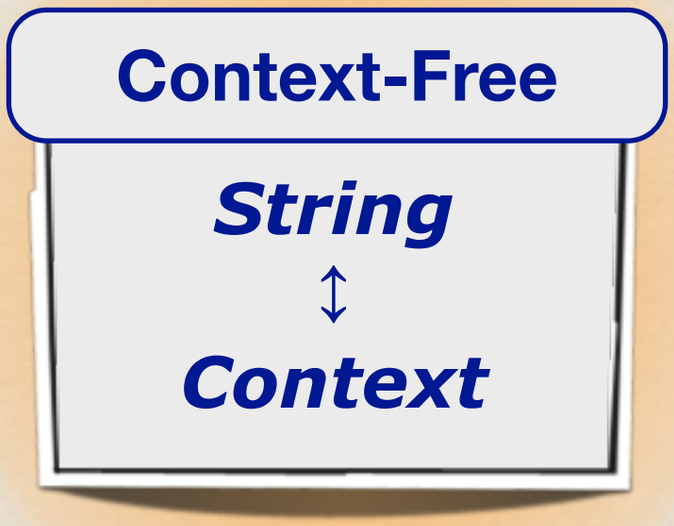
Context-Free

String

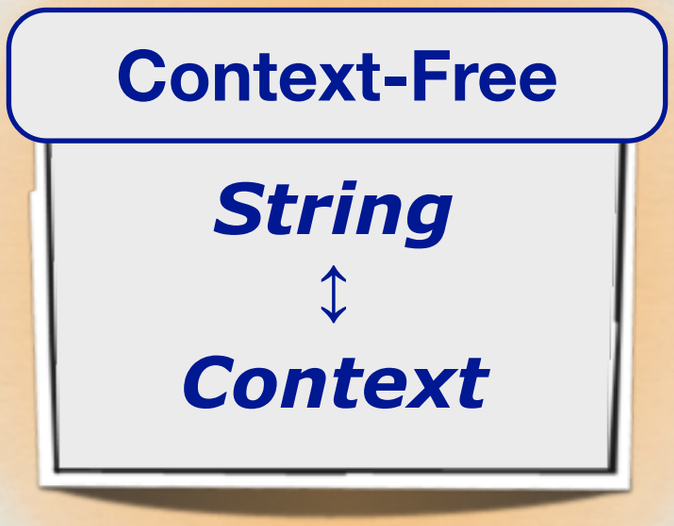
↕

Context


Context-Free
String
Context


Context-Free
String
Context

Learning of Substitutable Context-Free Languages from Positive Data



Languages

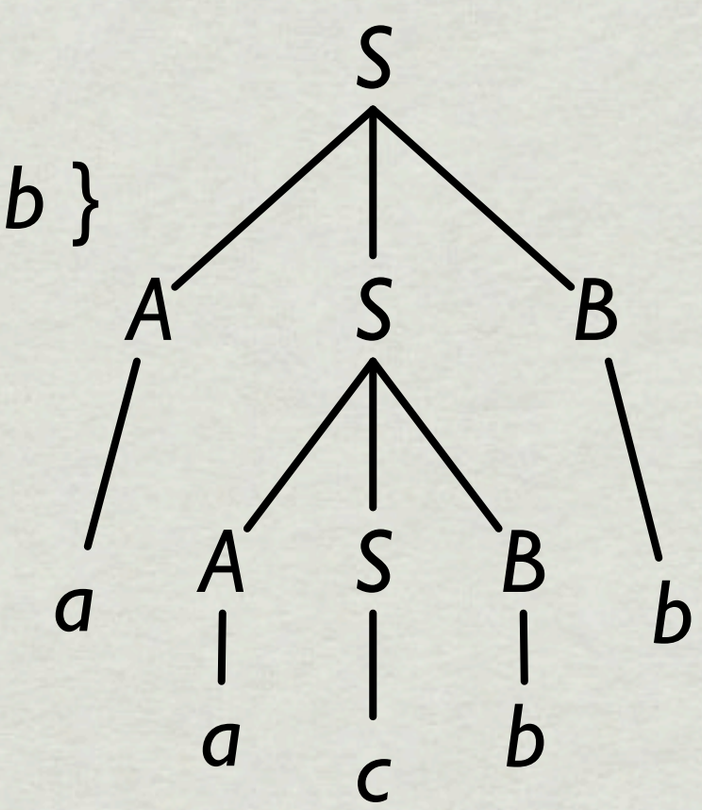
- $\Sigma = \{a,b,c,...\}$: finite set of symbols
- Σ^*: the set of *strings* over Σ
  a, abc, $caab$ ... $\in \Sigma^*$
- Any subset of Σ^* is called a *language*
- *Context*: pair of strings $(u,v) \in \Sigma^* \times \Sigma^*$
- Grammar: finite description for an (infinite) language

Context-Free Grammars

- $G = (N, \Sigma, P, I)$
  - N: nonterminal symbols
  - Σ: terminal symbols
  - $P \subseteq N \times (N \cup \Sigma)^*$: production rules ($A \rightarrow \alpha$)
  - $I \subseteq N$: initial symbols
- Derivation ($\Rightarrow$):
  - If $A \rightarrow \alpha \in P$, then $A \Rightarrow \alpha$
  - If $A \Rightarrow \alpha B \gamma$ and $B \Rightarrow \beta$, then $A \Rightarrow \alpha\beta\gamma$
- Language: $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow w \text{ for some } S \in I \}$

Example

- $G = (N, \Sigma, P, I)$
  - $\Sigma = \{ a,b,c \}$
  - $N = \{ S, A, B \}$
  - $I = \{ S \}$
  - $P = \{ S \rightarrow ASB,\ S \rightarrow c,\ A \rightarrow a,\ B \rightarrow b \}$
    - $S \Rightarrow ASB \Rightarrow AASBB \Rightarrow aacbb$
- $L(G) = \{ a^n c b^n \mid n \geqq 0 \}$



Substitutable CFLs

- Clark and Eyraud ('05,'07)
- L is **substitutable** iff

$$\forall v_1, v_2 \in \Sigma^+ \; [\exists \langle u_1, w_1 \rangle . \; u_1 v_1 w_1, u_1 v_2 w_1 \in L] \Rightarrow [\forall \langle u_2, w_2 \rangle . \; u_2 v_1 w_2 \in L \Leftrightarrow u_2 v_2 w_2 \in L]$$

Substitutable CFLs

- Clark and Eyraud ('05,'07)
- L is **substitutable** iff

  $\forall v_1, v_2 \in \Sigma^+ \; [\exists \langle u_1, w_1 \rangle . \; u_1 v_1 w_1, u_1 v_2 w_1 \in L]$

  $\Rightarrow [\forall \langle u_2, w_2 \rangle . \; u_2 v_1 w_2 \in L \Leftrightarrow u_2 v_2 w_2 \in L]$

- Positive data:

Substitutable CFLs

- Clark and Eyraud ('05,'07)
- L is **substitutable** iff

$$\forall v_1, v_2 \in \Sigma^+ \; [\exists \langle u_1, w_1 \rangle . \; u_1 v_1 w_1, u_1 v_2 w_1 \in L]$$
$$\Rightarrow [\forall \langle u_2, w_2 \rangle . \; u_2 v_1 w_2 \in L \Leftrightarrow u_2 v_2 w_2 \in L]$$

- Positive data:
  - A man gave John chocolate.
  - A man gave a little girl chocolate.

Substitutable CFLs

- Clark and Eyraud ('05,'07)
- L is **substitutable** iff

$$\forall v_1, v_2 \in \Sigma^+ \; [\exists \langle u_1, w_1 \rangle . \; u_1 v_1 w_1, u_1 v_2 w_1 \in L]$$
$$\Rightarrow [\forall \langle u_2, w_2 \rangle . \; u_2 v_1 w_2 \in L \Leftrightarrow u_2 v_2 w_2 \in L]$$

- Positive data:
  - A man gave John chocolate.
  - A man gave a little girl chocolate.
  - They like John.

Substitutable CFLs

- Clark and Eyraud ('05,'07)
- L is **substitutable** iff

  $\forall v_1, v_2 \in \Sigma^+ \; [\exists \langle u_1, w_1 \rangle . \; u_1 v_1 w_1, u_1 v_2 w_1 \in L]$

  $\Rightarrow [\forall \langle u_2, w_2 \rangle . \; u_2 v_1 w_2 \in L \Leftrightarrow u_2 v_2 w_2 \in L]$

- Positive data:
  - A man gave John chocolate.
  - A man gave a little girl chocolate.
  - They like John.
- Generalization: They like a little girl.


Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

was the man who was hungry ordering dinner ?

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

was the man who was hungry ordering dinner ?

* was the man who hungry was ordering dinner ?

Identification in the Limit from Positive Data

- Gold (1967)

Identification in the Limit from Positive Data

- Gold (1967)

Learning Target

L_0

Identification in the Limit from Positive Data

- Gold (1967)
- Learner
  - gets a positive example w_1 w_2 w_3 w_4 ...
  - updates the conjecture G_1 G_2 G_3 G_4 ...
  - $L_0 = \{ w_1, w_2, w_3, \ldots \}$

Learning Target

L_0

Identification in the Limit from Positive Data

- Gold (1967)
- Learner
  - gets a positive example w_1 w_2 w_3 w_4 ...
  - updates the conjecture G_1 G_2 G_3 G_4 ...
  - $L_0 = \{ w_1, w_2, w_3, \ldots \}$

- Identification in the Limit:
  - convergence to a grammar for the target
    $G_n = G_{n+1} = G_{n+2} \ldots$ and $L(G_n) = L_0$
- Learner should uniformly learn a rich class of languages

Clark & Eyraud's Algorithm

let G := vacuous grammar;
For n = 1,2,3,...
  let D := $\{w_1, w_2, \ldots, w_n\}$;
  If $D \not\subseteq L(G)$
    then **update** G by D;
  End if
  output G
End for

Learner's Conjecture

- $D = \{ w_1, ..., w_n \}$: positive data
- G : conjecture
  - $N = Sub(D)$: all substrings from D
    $= \{ [\![v]\!] \mid \exists \langle u_1, u_2 \rangle \; u_1 v u_2 \in D \}$
  - ★ $[\![v]\!] \Rightarrow v$ for all $[\![v]\!] \in N$,
  - Initial Symbols: $\{ [\![v]\!] \in N \mid v \in D \}$
  - Rules
    - Type I: $[\![uv]\!] \rightarrow [\![u]\!][\![v]\!]$ for all $[\![uv]\!] \in N$
    - Type II: $[\![a]\!] \rightarrow a$ for all $a \in \Sigma$
    - Type III: $[\![v]\!] \rightarrow [\![w]\!]$ if $\exists \langle u_1, u_2 \rangle$ s.t. $u_1 v u_2, u_1 w u_2 \in D$

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

was the man who was hungry ordering dinner ?

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

⟦was the man hungry ?⟧ ⇒ ⟦was the man⟧⟦hungry ?⟧

was the man who was hungry ordering dinner ?

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

⟦was the man hungry ?⟧ ⇒ ⟦was the man⟧⟦hungry ?⟧

⇒ ⟦was the man⟧⟦hungry⟧⟦?⟧ ⇒ ⟦was⟧⟦the⟧⟦man⟧⟦hungry⟧⟦?⟧

was the man who was hungry ordering dinner ?

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

⟦was the man hungry ?⟧ ⇒ ⟦was the man⟧⟦hungry ?⟧

⇒ ⟦was the man⟧⟦hungry⟧⟦?⟧ ⇒ ⟦was⟧⟦the⟧⟦man⟧⟦hungry⟧⟦?⟧

⇒ ⟦was⟧⟦the⟧⟦man⟧⟦ordering dinner⟧⟦?⟧

was the man who was hungry ordering dinner ?

Learner

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

⟦was the man hungry ?⟧ ⇒ ⟦was the man⟧⟦hungry ?⟧

⇒ ⟦was the man⟧⟦hungry⟧⟦?⟧ ⇒ ⟦was⟧⟦the⟧⟦man⟧⟦hungry⟧⟦?⟧

⇒ ⟦was⟧⟦the⟧⟦man⟧⟦ordering dinner⟧⟦?⟧

⇒ ⟦was⟧⟦the⟧⟦man who was hungry⟧⟦ordering dinner⟧⟦?⟧

was the man who was hungry ordering dinner ?

the man who was hungry died .
the man ordered dinner .
the man died .
the man was hungry .
was the man hungry ?
the man was ordering dinner .

Grammar

⟦man⟧ → ⟦man who was hungry⟧

⟦hungry⟧ → ⟦ordering dinner⟧

⟦was the man hungry ?⟧ ⇒ ⟦was the man⟧⟦hungry ?⟧

⇒ ⟦was the man⟧⟦hungry⟧⟦?⟧ ⇒ ⟦was⟧⟦the⟧⟦man⟧⟦hungry⟧⟦?⟧

⇒ ⟦was⟧⟦the⟧⟦man⟧⟦ordering dinner⟧⟦?⟧

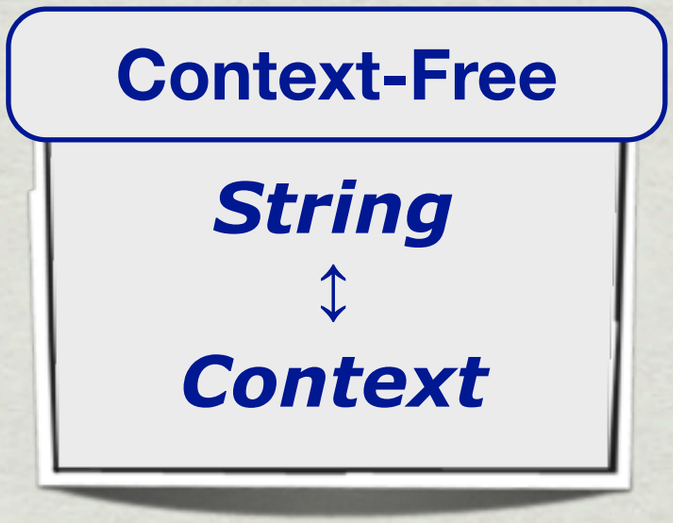
⇒ ⟦was⟧⟦the⟧⟦man who was hungry⟧⟦ordering dinner⟧⟦?⟧

⇒ was the man who was hungry ordering dinner ?

Theorem

- Clark and Eyraud's algorithm identifies every Substitutable CFL in the limit from positive data
- Polynomial-time update
- Polynomially many examples are enough for convergence w.r.t. the size of the grammar to be learnt

- A rare example of a class of CFLs that is efficiently learnable from positive data
- Explaining an aspect of natural language phenomena


Context-Free
String
Context

Multiple Context-Free

Multi-word

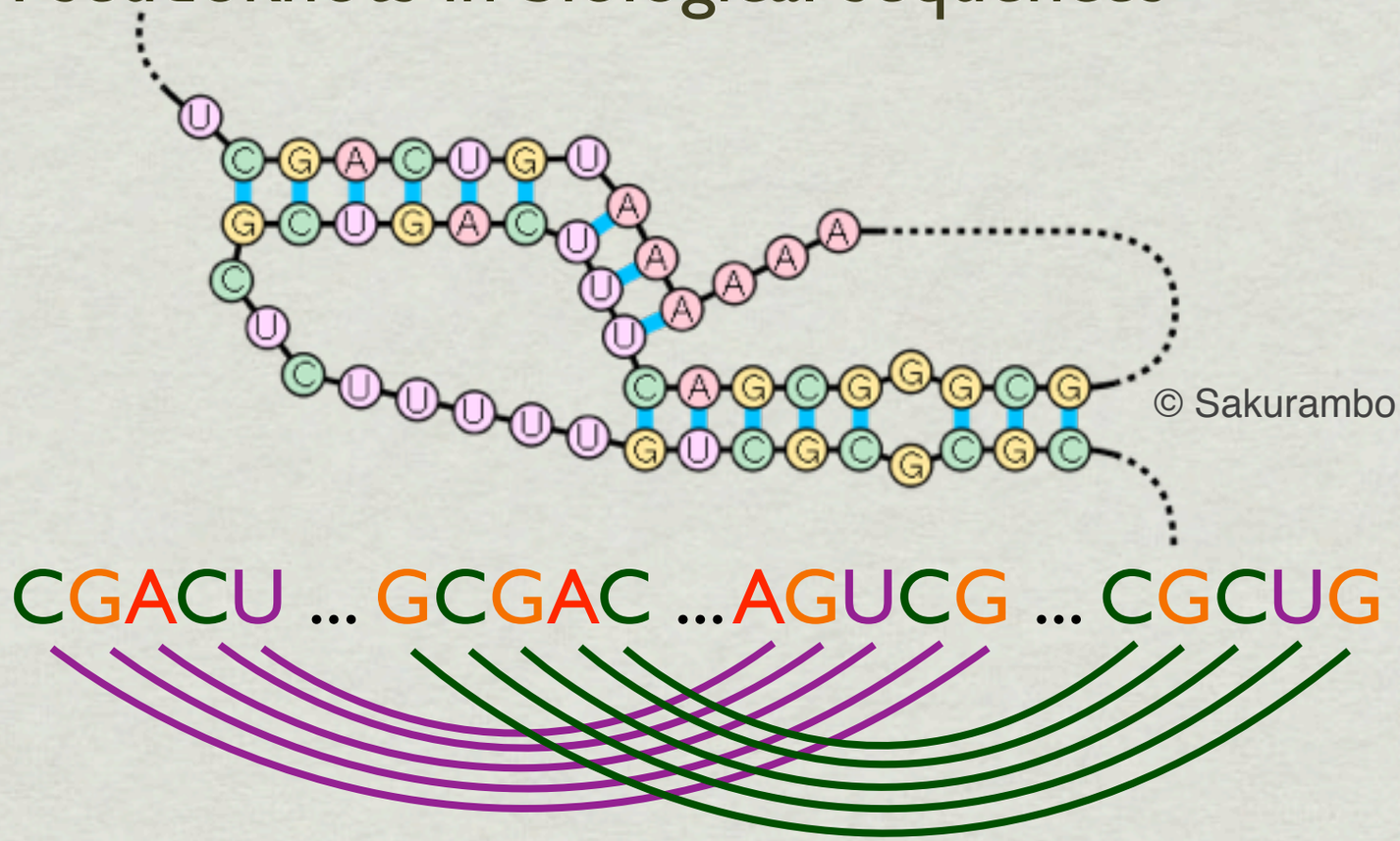
↕

Multi-Context

Multiple Context-Free Grammars

Non-context-free phenomena

- Pseudoknots in biological sequences



- $\{ a^m b^n c^m d^n \mid m, n > 0 \}$ is not context-free

Non-context-free phenomena

- $\{ a^m b^n c^m d^n \mid m, n > 0 \}$ is not context-free

Non-context-free phenomena

- $\{ a^m b^n c^m d^n \mid m, n > 0 \}$ is not context-free
- $A \Rightarrow a^m c^m$ for $m > 0$ by $A \to aAc$, $A \to ac$,

  $B \Rightarrow b^n d^n$ for $n > 0$ by $B \to bBd$, $B \to bd$.

Non-context-free phenomena

- { $a^m b^n c^m d^n$ | $m, n > 0$ } is not context-free
- $A \Rightarrow a^m c^m$ for $m > 0$ by $A \rightarrow aAc$, $A \rightarrow ac$,

  $B \Rightarrow b^n d^n$ for $n > 0$ by $B \rightarrow bBd$, $B \rightarrow bd$.

- { $a^m b^n c^m d^n$ | $m, n > 0$ } is *Multiple Context-Free*
- $A \Rightarrow \langle a^m, c^m \rangle$ for $m > 0$

  $B \Rightarrow \langle b^n, d^n \rangle$ for $n > 0$

  $S \Rightarrow a^m b^n c^m d^n$ for $m, n > 0$

- **M**CFG: nonterminals generate ***tuples*** of strings

Multiple CFGs

- CFG: nonterminals generate strings
  MCFG: nonterminals generate ***tuples*** of strings

- Each nonterminal $A \in N$ is assigned a dimension $\dim(A) = m$

- A generates m-tuples of strings for $\dim(A)=m$:
  $A \Rightarrow \langle v_1, v_2, ..., v_m \rangle \in (\Sigma^*)^m$

Multiple CFGs

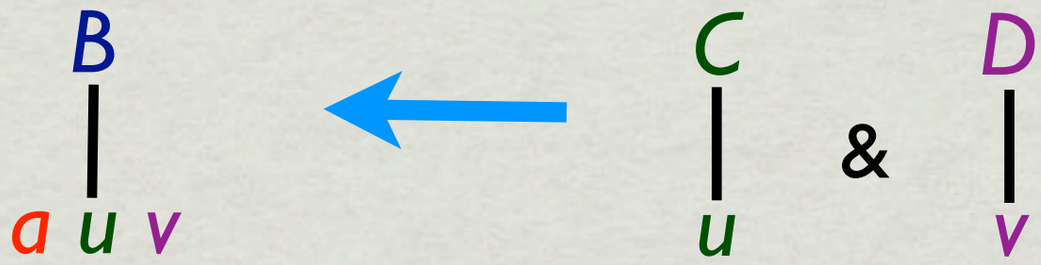
- $B \rightarrow a\ C\ D$ (context-free rule)

★ a : terminal symbol

Multiple CFGs

- $B \to a\ C\ D$ (context-free rule)

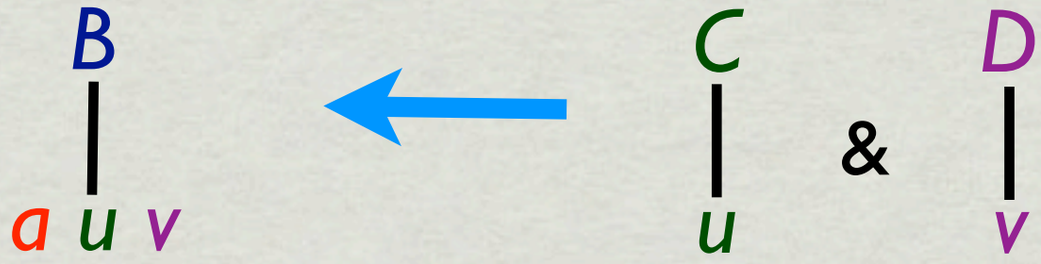
★ a : terminal symbol



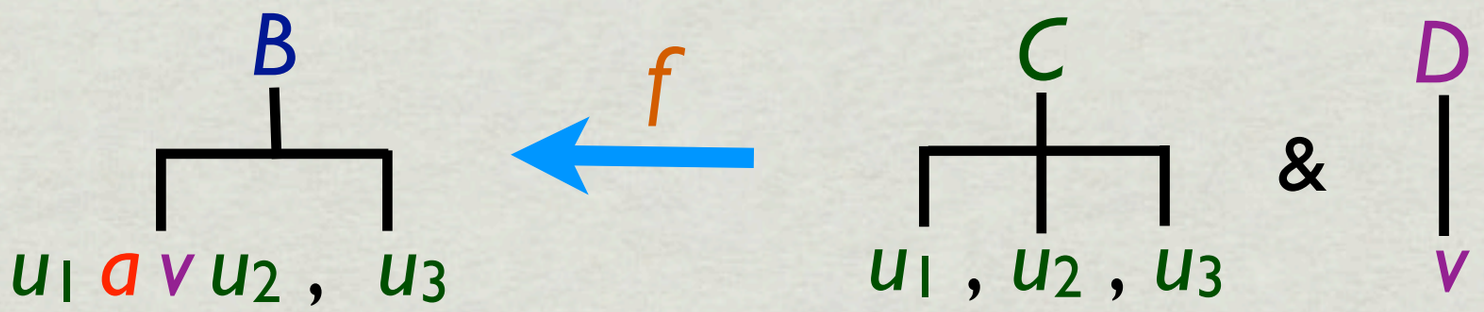
Multiple CFGs

- $B \to a\ C\ D$ (context-free rule)

★ a : terminal symbol



- $B \to f(C,D)$ (multiple cf rule)



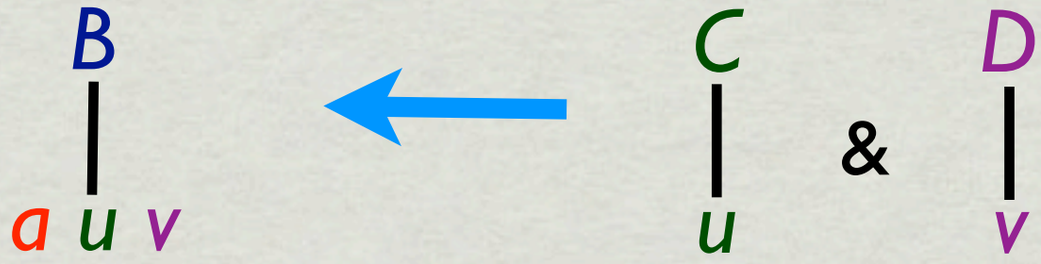
★ dim(B) = 2,
dim(C) = 3,
dim(D) = 1.

★ $f(\langle x_1,x_2,x_3\rangle, \langle y\rangle) = \langle x_1 a y x_2, x_3\rangle$

f uses each argument exactly once

Multiple CFGs

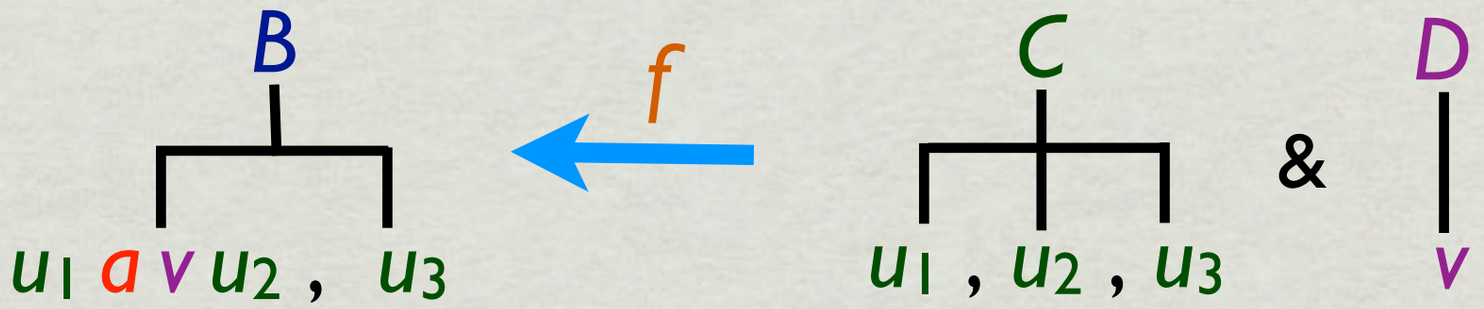
- $B \to a\ C\ D$ (context-free rule)



★ a : terminal symbol

- $B \to g(C,D)$
  - ★ $g(x, y) = axy$

- $B \to f(C,D)$ (multiple cf rule)



★ $\dim(B) = 2$,
$\dim(C) = 3$,
$\dim(D) = 1$.

★ $f(\langle x_1,x_2,x_3\rangle, \langle y\rangle) = \langle x_1 a y x_2, x_3\rangle$

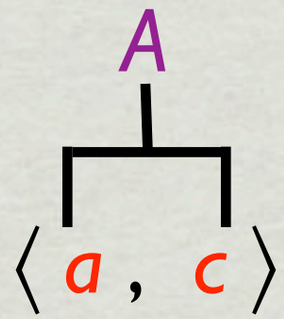
f uses each argument exactly once

Example: $L(G) = \{ a^m\ b^n\ c^m\ d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1y_1x_2y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle ax_1, cx_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1b, y_2d\rangle$.

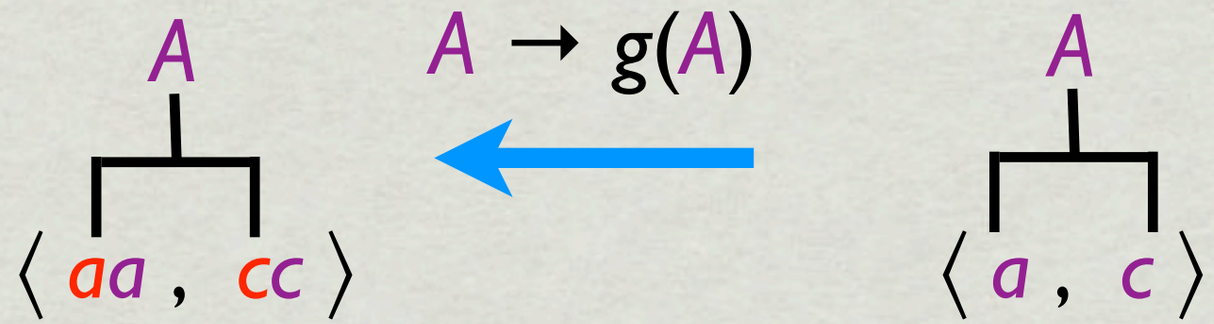
Example: $L(G) = \{ a^m b^n c^m d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1 y_1 x_2 y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle a x_1, c x_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1 b, y_2 d\rangle$.



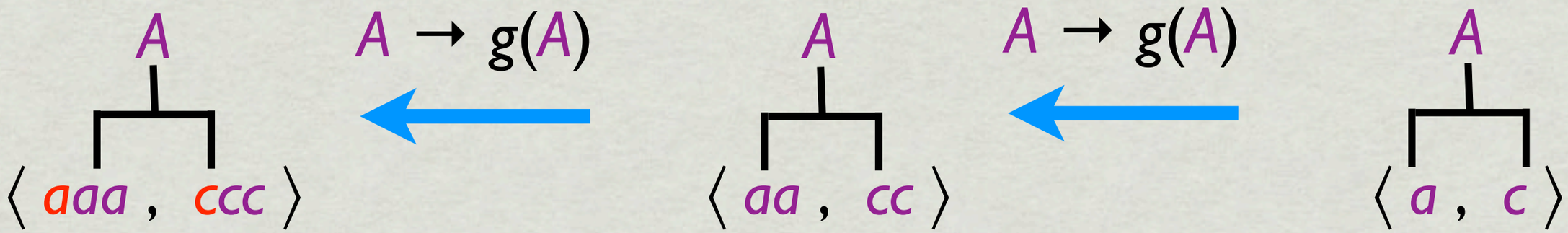
Example: $L(G) = \{ a^m\ b^n\ c^m\ d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1 y_1 x_2 y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle a x_1, c x_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1 b, y_2 d\rangle$.



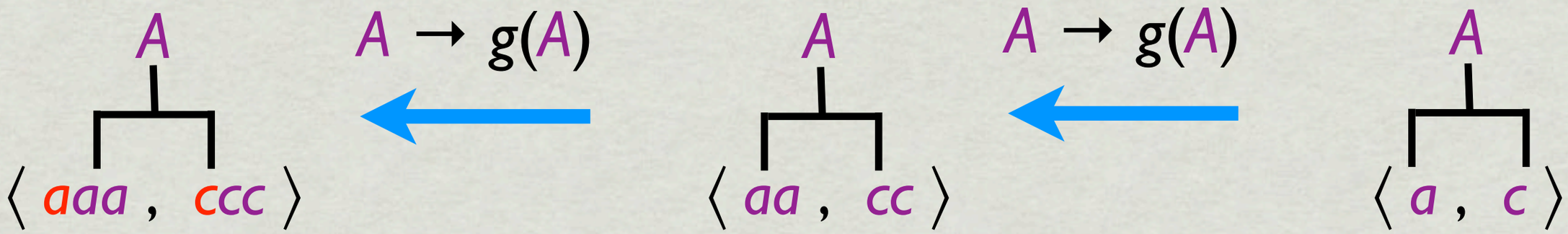
Example: $L(G) = \{ a^m\, b^n\, c^m\, d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1 y_1 x_2 y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle a x_1, c x_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1 b, y_2 d\rangle$.



Example: $L(G) = \{ a^m\, b^n\, c^m\, d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1 y_1 x_2 y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle a x_1, c x_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1 b, y_2 d\rangle$.

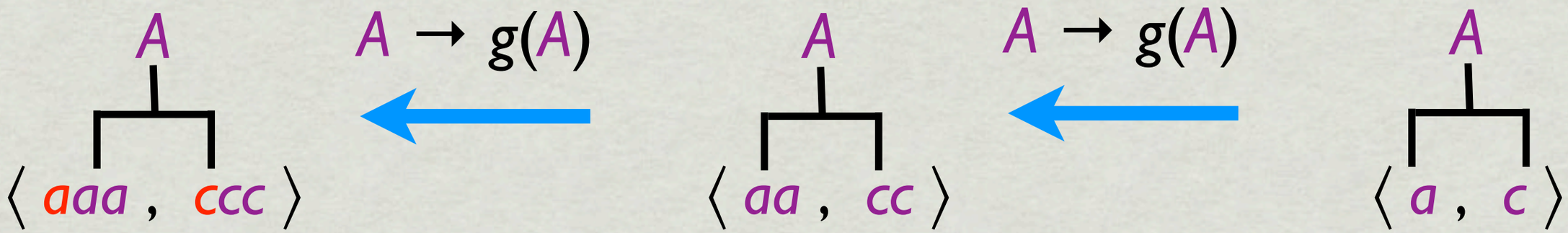


  - $A \Rightarrow \langle a^m, c^m\rangle$ for all $m > 0$

    $B \Rightarrow \langle b^n, d^n\rangle$ for all $n > 0$

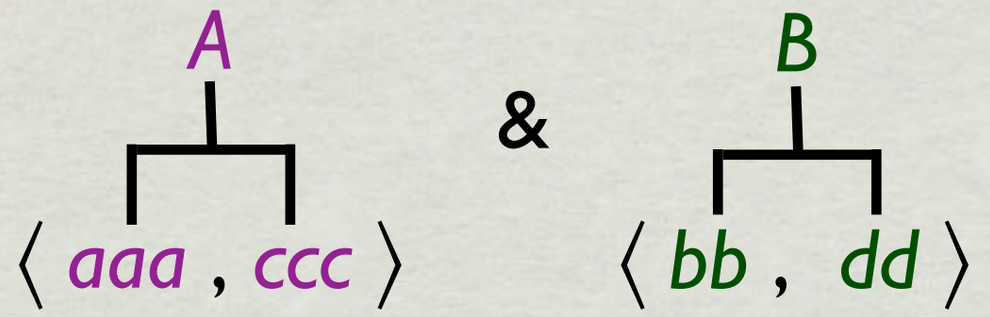
Example: $L(G) = \{ a^m\, b^n\, c^m\, d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1y_1x_2y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle ax_1, cx_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1b, y_2d\rangle$.



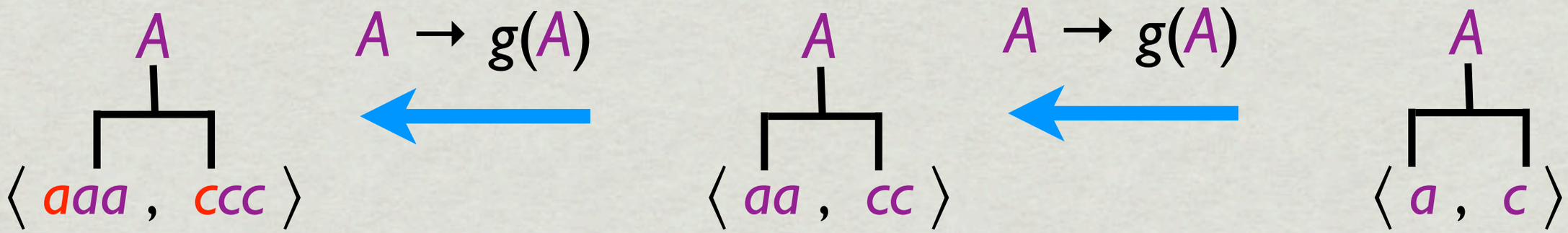
  - $A \Rightarrow \langle a^m, c^m\rangle$ for all $m > 0$

    $B \Rightarrow \langle b^n, d^n\rangle$ for all $n > 0$



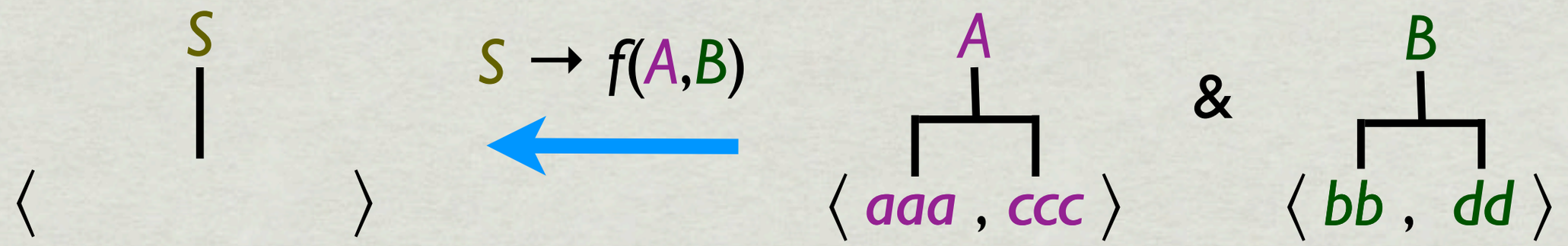
Example: $L(G) = \{ a^m\, b^n\, c^m\, d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1 y_1 x_2 y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle a x_1, c x_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1 b, y_2 d\rangle$.



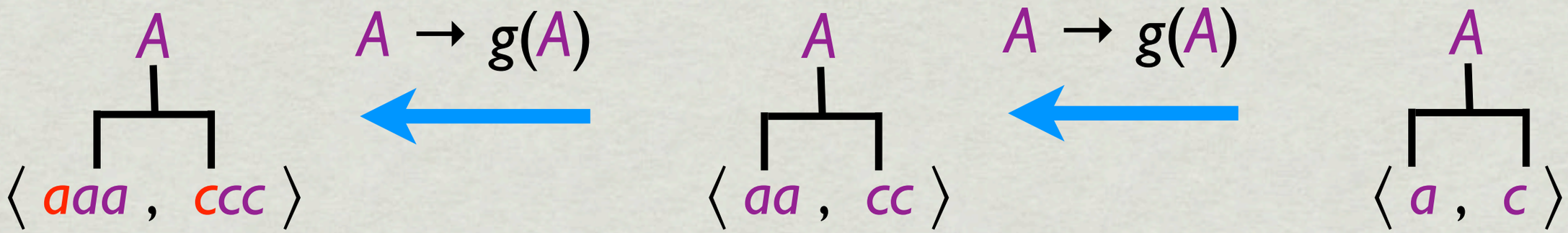
  - $A \Rightarrow \langle a^m, c^m\rangle$ for all $m > 0$

    $B \Rightarrow \langle b^n, d^n\rangle$ for all $n > 0$



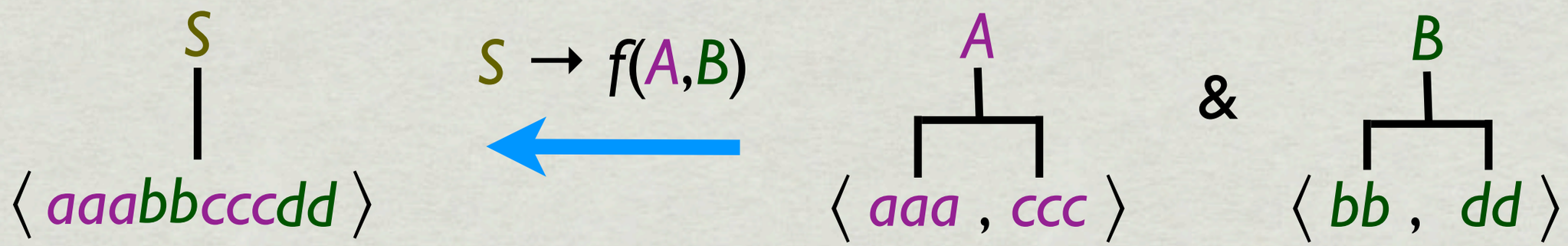
Example: $L(G) = \{ a^m\ b^n\ c^m\ d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1y_1x_2y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle ax_1, cx_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1b, y_2d\rangle$.



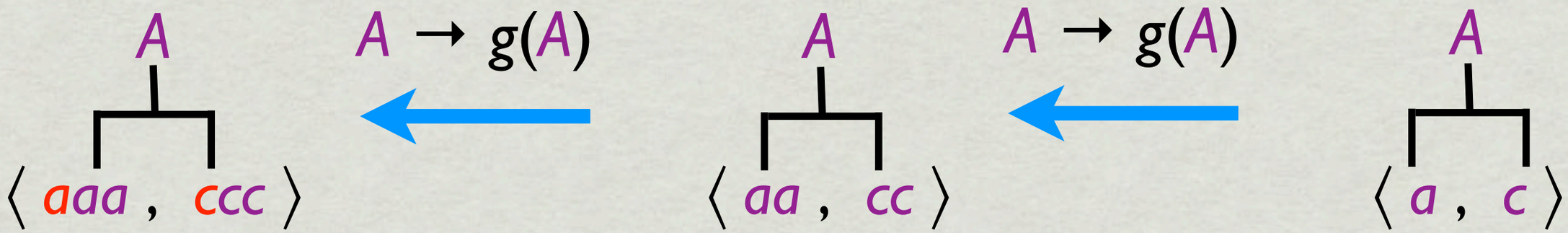
  - $A \Rightarrow \langle a^m, c^m\rangle$ for all $m > 0$

    $B \Rightarrow \langle b^n, d^n\rangle$ for all $n > 0$



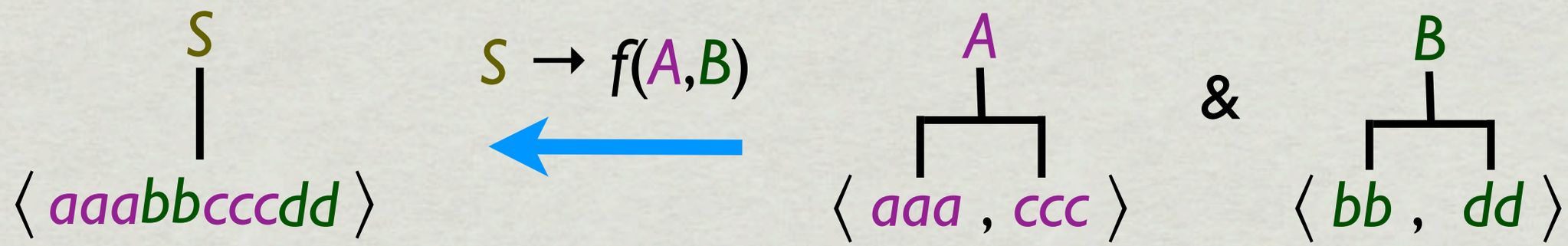
Example: $L(G) = \{ a^m\ b^n\ c^m\ d^n \mid m, n > 0 \}$

- $S \to f(A,B)$ with $f(\langle x_1,x_2\rangle,\langle y_1,y_2\rangle) = \langle x_1y_1x_2y_2\rangle$,
  $A \to \langle a, c\rangle$, $A \to g(A)$ with $g(\langle x_1,x_2\rangle) = \langle ax_1, cx_2\rangle$,
  $B \to \langle b, d\rangle$, $B \to h(B)$ with $h(\langle y_1,y_2\rangle) = \langle y_1b, y_2d\rangle$.



- $A \Rightarrow \langle a^m, c^m\rangle$ for all $m > 0$

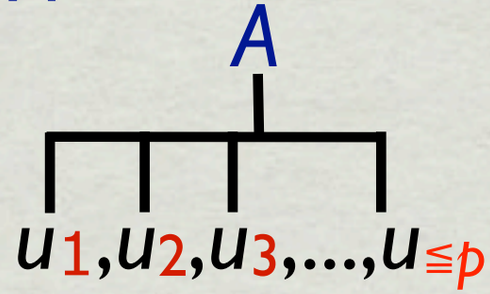
  $B \Rightarrow \langle b^n, d^n\rangle$ for all $n > 0$



- $S \Rightarrow a^m\ b^n\ c^m\ d^n$ for all $m, n > 0$

Hierarchy of p,q-MCFLs

- p: the maximum of $\dim(A)$ for nonterminals A
  (CFG: $p = 1$)



- q: maximum number of nonterminals on rhs of rules

$$A \to f(B_1, B_2, \ldots, B_{\leqq q})$$

- $p\text{-}\mathcal{MCFL}(q) \subseteq p\text{-}\mathcal{MCFL}(q+1)$
  $p\text{-}\mathcal{MCFL}(q) \subseteq (p+1)\text{-}\mathcal{MCFL}(q)$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ m-***context*** (multi-context):

$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ m-***word*** (multi-word):

$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ m-***context*** (multi-context):

$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ m-***word*** (multi-word):

$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

- $\boldsymbol{u} \otimes \boldsymbol{v} = u_0 \; u_1 \; \ldots \; u_m = v_1 \; \ldots \; v_m = w$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ m-***context*** (multi-context):
$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ m-***word*** (multi-word):
$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

- $\boldsymbol{u} \otimes \boldsymbol{v} = u_0 v_1 u_1 \ldots v_m u_m = w$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ *m*-***context*** (multi-context):
$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ *m*-***word*** (multi-word):
$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

- $\boldsymbol{u} \otimes \boldsymbol{v} = u_0 v_1 u_1 \ldots v_m u_m = w$

- $L/\boldsymbol{v} = \{ \boldsymbol{u} \mid \boldsymbol{u} \otimes \boldsymbol{v} \in L \}$:
the set of multi-contexts for $\boldsymbol{v}$

e.g.)
$\{ abcd^n \mid n > 0 \}/\langle b, d \rangle =$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ m-***context*** (multi-context):
$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ m-***word*** (multi-word):
$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

- $\boldsymbol{u} \otimes \boldsymbol{v} = u_0 v_1 u_1 \ldots v_m u_m = w$

- $L/\boldsymbol{v} = \{ \boldsymbol{u} \mid \boldsymbol{u} \otimes \boldsymbol{v} \in L \}$:
the set of multi-contexts for $\boldsymbol{v}$

e.g.)
$\{ abcd^i dd^j \mid i,j \geqq 0 \} / \langle \ , \ \rangle = \{ \langle \ , \ , \ \rangle \mid i,j \geqq 0 \}$

Substructure/Context Decomposition

CFGs

- $u_0 v u_1 \in L$

★ $\langle u_0, u_1 \rangle$: **context**

★ v : **substring**

Multiple Context-Free Grammars

- $w = u_0 v_1 u_1 \ldots v_m u_m \in L$

★ m-***context*** (multi-context):

$\boldsymbol{u} = \langle u_0, u_1, \ldots, u_m \rangle$

★ m-***word*** (multi-word):

$\boldsymbol{v} = \langle v_1, v_2, \ldots, v_m \rangle$

- $\boldsymbol{u} \otimes \boldsymbol{v} = u_0 v_1 u_1 \ldots v_m u_m = w$

- $L/\boldsymbol{v} = \{ \boldsymbol{u} \mid \boldsymbol{u} \otimes \boldsymbol{v} \in L \}$:

the set of multi-contexts for $\boldsymbol{v}$

e.g.)

$\{ abcd^i dd^j \mid i,j \geqq 0 \} / \langle b, d \rangle = \{ \langle a, cd^i, d^j \rangle \mid i,j \geqq 0 \}$

Multidimensionally Substitutable Multiple Context-Free Languages

p-dimensional Substitutability

cf. context-free case: $\forall v, v' \in \Sigma^+$

$[\exists \langle u,w \rangle \,.\, uvw, uv'w \in L] \Rightarrow [\forall \langle u',w' \rangle \,.\, u'vw' \in L \Leftrightarrow u'v'w' \in L]$

- L is ***p*D-substitutable** iff $\forall \boldsymbol{v}, \boldsymbol{v'} \in (\Sigma^+)^{\leqq p}$,

$[\exists \boldsymbol{u} \,.\, \boldsymbol{u} \otimes \boldsymbol{v}, \boldsymbol{u} \otimes \boldsymbol{v'} \in L] \Rightarrow [\forall \boldsymbol{u'} \,.\, \boldsymbol{u'} \otimes \boldsymbol{v} \in L \Leftrightarrow \boldsymbol{u'} \otimes \boldsymbol{v'} \in L]$

p-dimensional Substitutability

cf. context-free case: $\forall v, v' \in \Sigma^+$

$[\exists \langle u,w \rangle \,.\, uvw, uv'w \in L] \Rightarrow [\forall \langle u',w' \rangle \,.\, u'vw' \in L \Leftrightarrow u'v'w' \in L]$

- L is ***p*D-substitutable** iff $\forall \boldsymbol{v}, \boldsymbol{v'} \in (\Sigma^+)^{\leqq p}$,

  $[\exists \boldsymbol{u} \,.\, \boldsymbol{u} \otimes \boldsymbol{v}, \boldsymbol{u} \otimes \boldsymbol{v'} \in L] \Rightarrow [\forall \boldsymbol{u'} \,.\, \boldsymbol{u'} \otimes \boldsymbol{v} \in L \Leftrightarrow \boldsymbol{u'} \otimes \boldsymbol{v'} \in L]$

- $L/\boldsymbol{v} \cap L/\boldsymbol{v'} \neq \emptyset \quad \Rightarrow \quad L/\boldsymbol{v} = L/\boldsymbol{v'}$ where $|\boldsymbol{v}| = |\boldsymbol{v'}| \leqq p$

p-dimensional Substitutability

cf. context-free case: $\forall v, v' \in \Sigma^+$

$[\exists \langle u,w \rangle \,.\, uvw, uv'w \in L] \Rightarrow [\forall \langle u',w' \rangle \,.\, u'vw' \in L \Leftrightarrow u'v'w' \in L]$

- L is pD-**substitutable** iff $\forall \boldsymbol{v}, \boldsymbol{v'} \in (\Sigma^+)^{\leqq p}$,

  $[\exists \boldsymbol{u} \,.\, \boldsymbol{u} \otimes \boldsymbol{v}, \boldsymbol{u} \otimes \boldsymbol{v'} \in L] \Rightarrow [\forall \boldsymbol{u'} \,.\, \boldsymbol{u'} \otimes \boldsymbol{v} \in L \Leftrightarrow \boldsymbol{u'} \otimes \boldsymbol{v'} \in L]$

- $L/\boldsymbol{v} \cap L/\boldsymbol{v'} \neq \emptyset \quad \Rightarrow \quad L/\boldsymbol{v} = L/\boldsymbol{v'}$ where $|\boldsymbol{v}| = |\boldsymbol{v'}| \leqq p$

★ **Our Learning Target:**

pD-substitutable p,q-MCFGs for fixed p,q

Learning Algorithm

let G := vacuous grammar;
For n = 1,2,3,...
  let D := $\{w_1, w_2, \ldots, w_n\}$;
  If $D \not\subseteq L(G)$
    then **update** G by D;
  End if
  output G
End for

Learner's Conjecture

- $D = \{ w_1, ..., w_n \}$: positive data
- G : conjecture
  - $N = Sub(D)$: all sub-multi-words from D
    $= \{ [\![v]\!] \mid \exists w.\ w \otimes v \in D, |v| \leqq p \}$

★ $[\![v]\!] \Rightarrow v$ for all $[\![v]\!] \in N$, ($\dim([\![v]\!]) = |v|$)

  - Initial Symbols: $\{ [\![w]\!] \in N \mid w \in D \}$ ($\dim([\![w]\!]) = 1$)
  - Rules (Type I):
    - $[\![v_0]\!] \rightarrow f([\![v_1]\!],...,[\![v_k]\!])$ where $v_0 = f(v_1,...,v_k)$, $k \leqq q$

cf. context-free case: $[\![v_1 v_2]\!] \rightarrow [\![v_1]\!][\![v_2]\!]$ & $[\![a]\!] \rightarrow a$

Type I - example

context-free case $\begin{cases} [\![v_1 v_2]\!] \to [\![v_1]\!][\![v_2]\!] \\ [\![a]\!] \to a \end{cases}$

- Rules (Type I):
  - $[\![\boldsymbol{v}_0]\!] \to f([\![\boldsymbol{v}_1]\!],\ldots,[\![\boldsymbol{v}_k]\!])$ where $\boldsymbol{v}_0 = f(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)$, $k \leqq q$
- $\langle abc, de\rangle, \langle a, e\rangle, \langle c\rangle \in \mathrm{Sub}(D)$
- $[\![abc, de]\!], [\![a, e]\!], [\![c]\!] \in N$
  - $[\![abc, de]\!] \to f([\![a, e]\!], [\![c]\!])$ with $f(\langle x_1, x_2\rangle, \langle y\rangle) = \langle x_1 b y, d x_2\rangle$,
  - $[\![abc, de]\!] \to \langle abc, de\rangle$

★ $[\![\boldsymbol{v}]\!] \Rightarrow \boldsymbol{v}$ for all $[\![\boldsymbol{v}]\!] \in N$, ($\dim([\![\boldsymbol{v}]\!]) = |\boldsymbol{v}|$)

Type II

- Rules (Type II):
  - $[\![u]\!] \rightarrow [\![v]\!]$ if $w \otimes u, w \otimes v \in D$ for some w

cf-case: $[\![u]\!] \rightarrow [\![v]\!]$ if $w_1 u w_2, w_1 v w_2 \in D$ for some w_1, w_2

- Substitutability:
  - $L_0/u \cap L_0/v \neq \emptyset \quad \Rightarrow \quad L_0/u = L_0/v$ for $|u| = |v| \leqq p$

Learning Efficiency

- Nonterminals: N = Sub(D)
  - $w = u_0 v_1 u_1 \ldots v_m u_m \Rightarrow \langle v_1, \ldots, v_m \rangle \in \text{Sub}(D),\ m \leqq p$

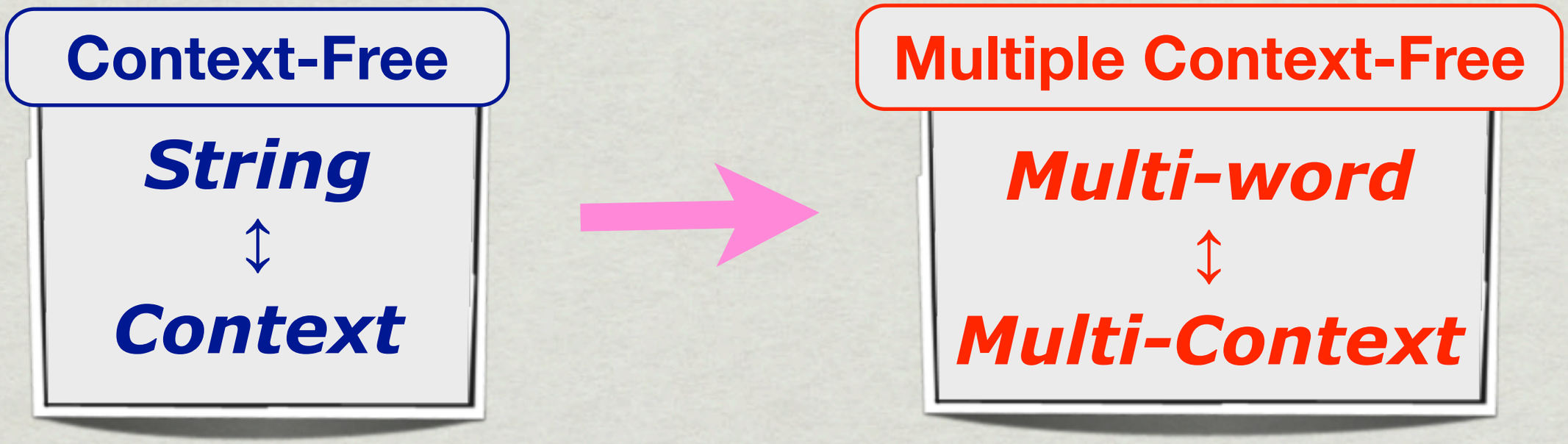
    At most $|w|^{2p}$ ways of decomposing w
- Rules (Type I):
  - $[\![\boldsymbol{u}]\!] \rightarrow f([\![\boldsymbol{v}_1]\!],\ldots,[\![\boldsymbol{v}_n]\!])$ where $\boldsymbol{u} = f(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n),\ |\boldsymbol{v}_i| \leqq p,\ n \leqq q$

    At most $||\boldsymbol{u}||^{2pq}$ ways of decomposing $\boldsymbol{u}$
- Rules (Type II):
  - $[\![\boldsymbol{u}]\!] \rightarrow [\![\boldsymbol{v}]\!]$ if $L_0/\boldsymbol{u} \cap L_0/\boldsymbol{v} \neq \varnothing$
- PolyTime in $||D||$
- Data needed: D_0
  - $|D_0| \leqq |G_0|$ —— $|G_0|$: # of rules of minimum G_0 with $L_0 = L(G_0)$

Theorem

- ***p*D-Substitutable *p*,*q*-Multiple CFGs are Polynomial-Time identifiable in the limit from Positive Data**

Distributional Learning



- { $a^m\ b^n\ c^m\ d^n$ | $m, n > 0$ } is Not substitutable
- { a^m e b^n f c^m g d^n | $m, n > 0$ } is substitutable
- Substitutability is too much restrictive

- { $a^m\ b^n\ c^m\ d^n$ | $m, n > 0$ } is Not substitutable
- { a^m e b^n f c^m g d^n | $m, n > 0$ } is substitutable
- Substitutability is too much restrictive

- Substitutable CFGs/MCFG are learnable from positive data only

- { $a^m\ b^n\ c^m\ d^n$ | $m, n > 0$ } is Not substitutable
- { a^m e b^n f c^m g d^n | $m, n > 0$ } is substitutable
- Substitutability is too much restrictive

- Substitutable CFGs/MCFG are learnable from positive data only
- Congruential CFGs are learnable with queries

- { $a^m\ b^n\ c^m\ d^n$ | $m, n > 0$ } is Not substitutable
- { a^m e b^n f c^m g d^n | $m, n > 0$ } is substitutable
- Substitutability is too much restrictive

- Substitutable CFGs/MCFG are learnable from positive data only
- Congruential CFGs are learnable with queries
- So are Congruential MCFGs

- { $a^m\ b^n\ c^m\ d^n$ | $m, n > 0$ } is Not substitutable
- { a^m e b^n f c^m g d^n | $m, n > 0$ } is substitutable
- Substitutability is too much restrictive

- Substitutable CFGs/MCFG are learnable from positive data only
- Congruential CFGs are learnable with queries
- So are Congruential MCFGs

Learning of Congruential Context-Free **Tree** Grammars

Multiple Context-Free

Multi-word

↕

Multi-Context

Context-Free Tree

Stub

↕

Tree-Context

Context-Free

String

↕

Context

Multiple Context-Free

Multi-word

↕

Multi-Context

Context-Free Tree

Stub

↕

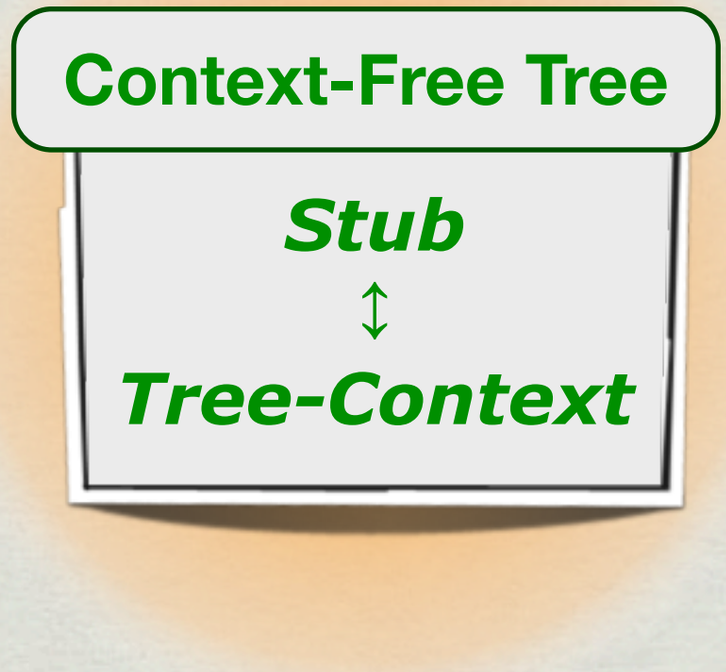
Tree-Context

Context-Free

String

↕

Context

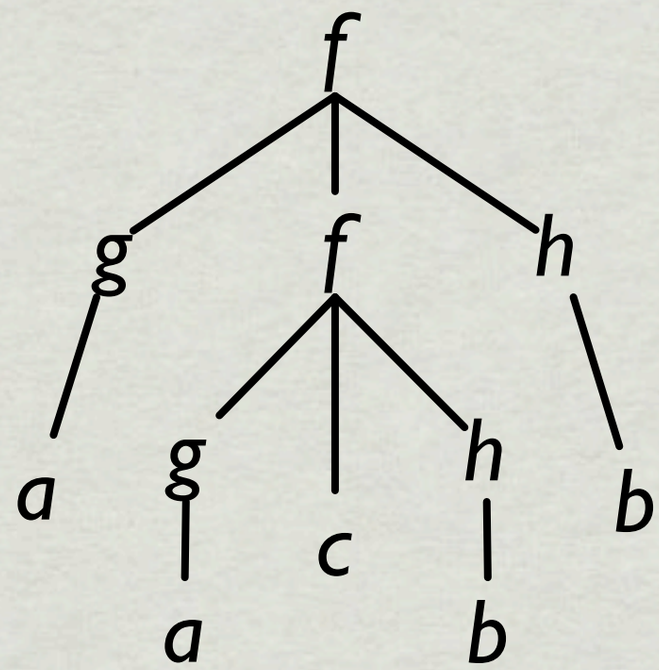

Context-Free Tree
Stub
↕
Tree-Context

Context-Free Tree Grammars

Trees

- CFGs define string languages
  CF**T**G define **tree** languages

Tree:
rooted, ordered,
labeled with ranked letters

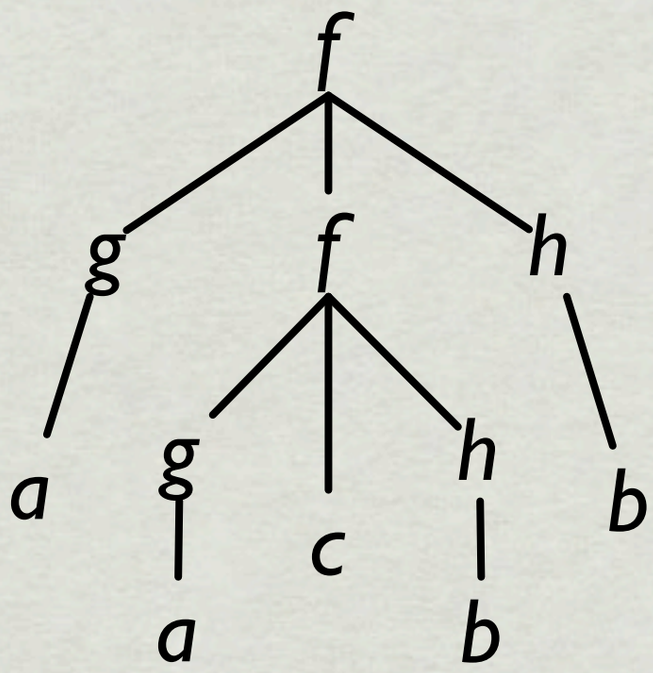


- ranked alphabet $\Sigma = \bigcup_k \Sigma_k$
  s.t. $\Sigma_i \cap \Sigma_j = \emptyset$ for $i \neq j$
- Σ_k: set of symbols of rank k
- $f \in \Sigma_k$ has k children

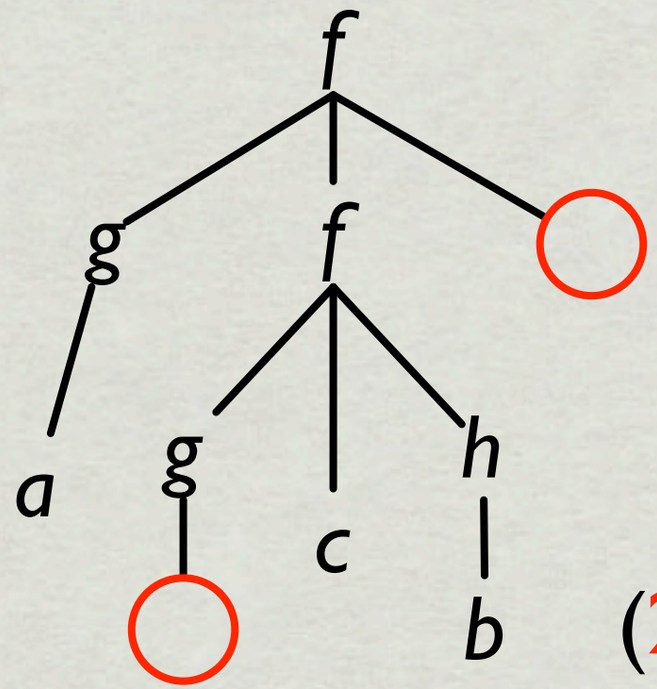
$\Sigma_0 = \{a,b,c\}$, $\Sigma_1 = \{g,h\}$, $\Sigma_3 = \{f\}$

Trees and Stubs

Tree (0-stub):
labeled, ordered, rooted



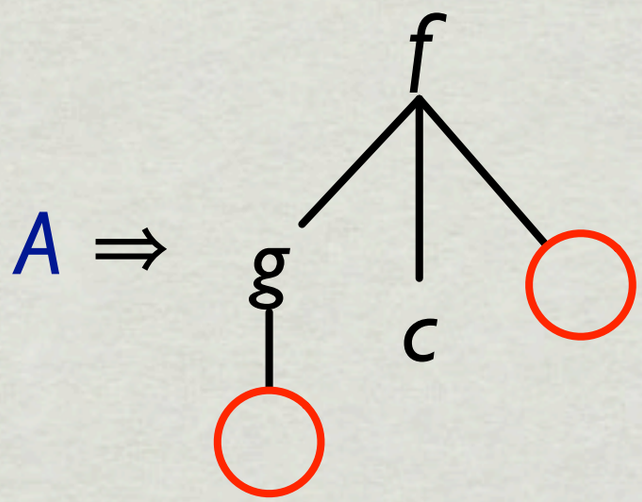
m-Stub:
tree with *m* "open leaves"



(2-Stub)

Context-Free Tree Grammars

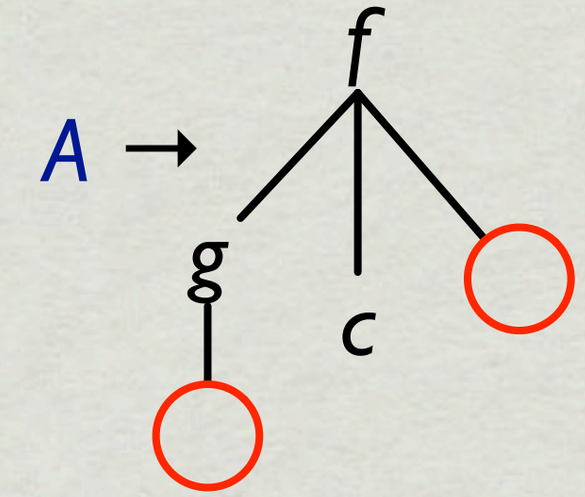
- CFGs define string languages
  CF**T**Gs define **tree** languages
- CFG: nonterminals generate substrings
  CF**T**G: nonterminals generate **stubs**
- Each nonterminal $A \in N$ is assigned a rank rnk(A) = m
- A generates rnk(A)-stubs:



rnk(A) = 2

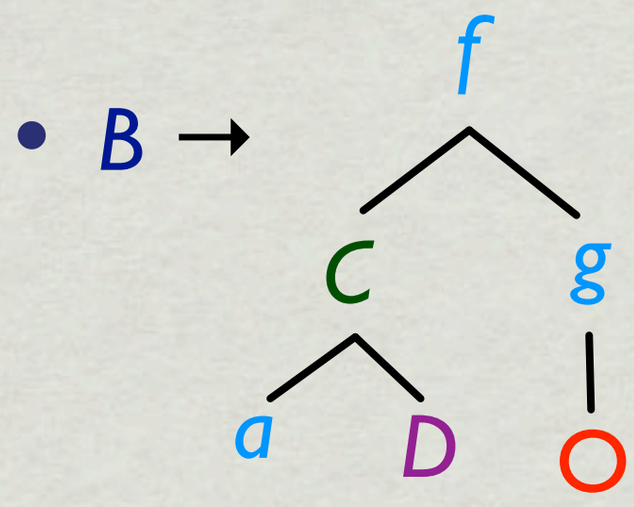
r-Context-Free Tree Grammars

- $G = (N, \Sigma, P, I)$
  - N, Σ: ranked nonterminal/terminal symbols
  - Rank is at most *r*
  - $P \subseteq \bigcup_k N_k \times (k\text{-Stubs})$: production rules
  - $I \subseteq N_0$: initial symbols of rank 0



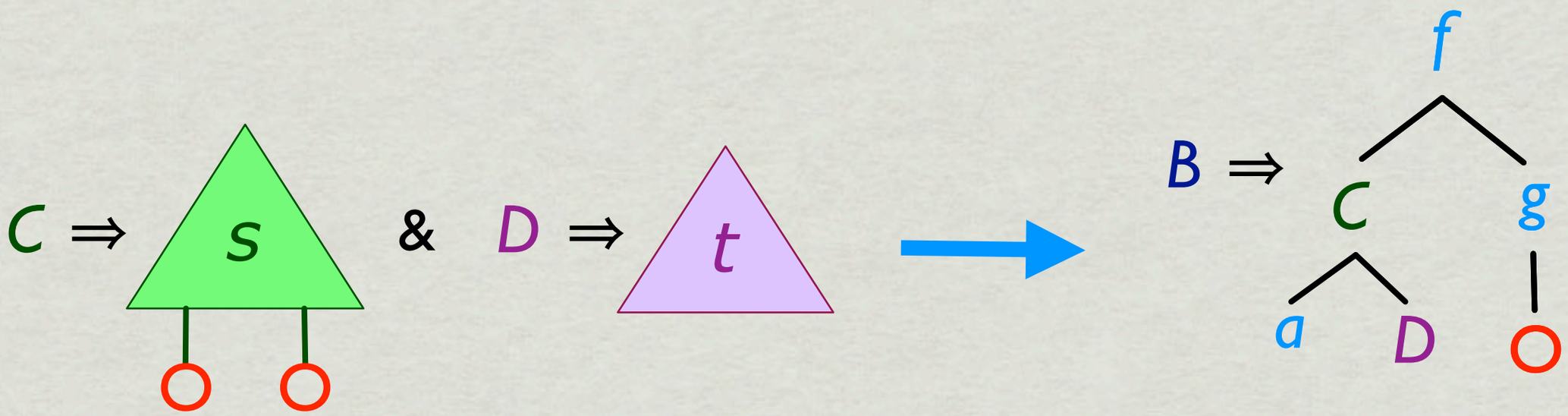
- $L(G) = \{ t \mid S \Rightarrow t$ for some $S \in I$ and t is a tree over $\Sigma \}$
- every 1-CFTG can be identified with a CFG

Context-Free Tree Grammars

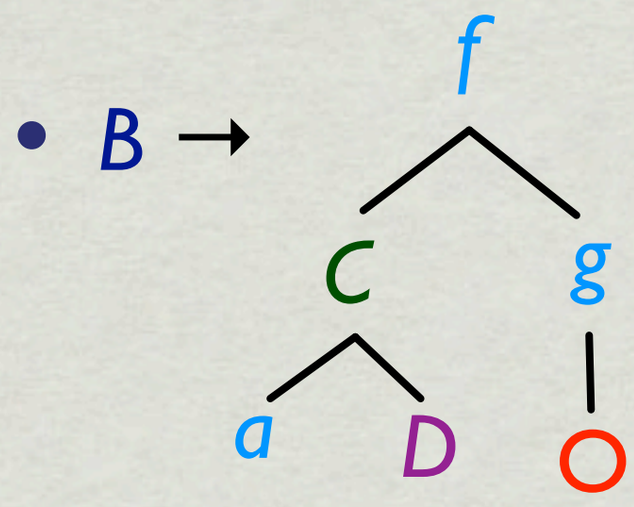


★ *a,f,g* : terminal symbol

rnk(*B*) = 1
rnk(*C*) = 2
rnk(*D*) = 0

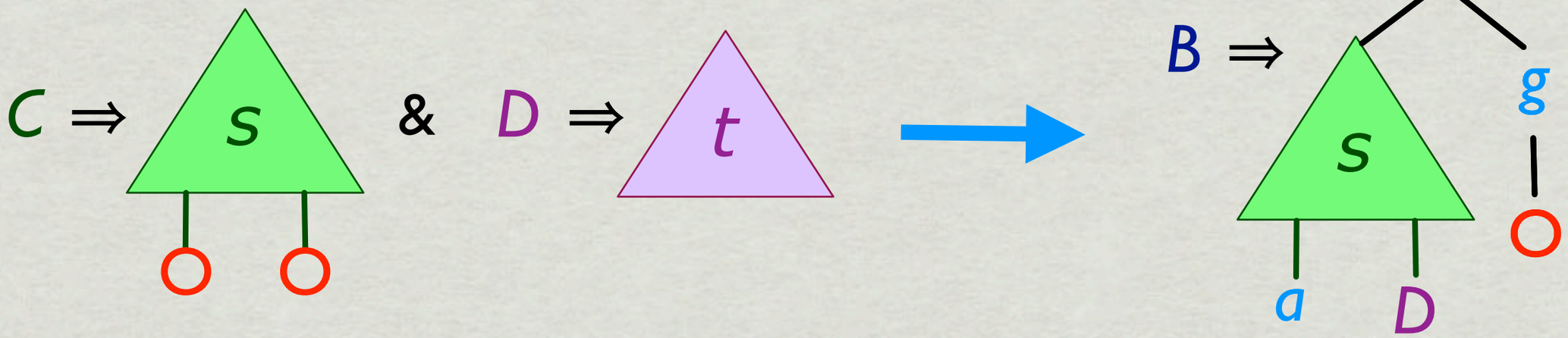


Context-Free Tree Grammars

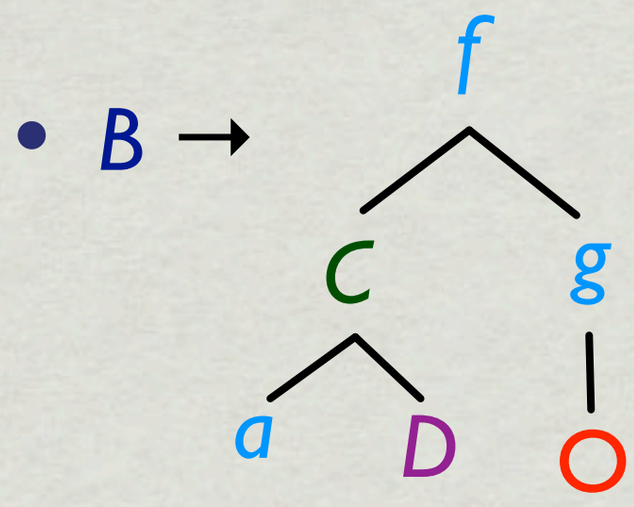


★ *a,f,g* : terminal symbol

rnk(*B*) = 1
rnk(*C*) = 2
rnk(*D*) = 0

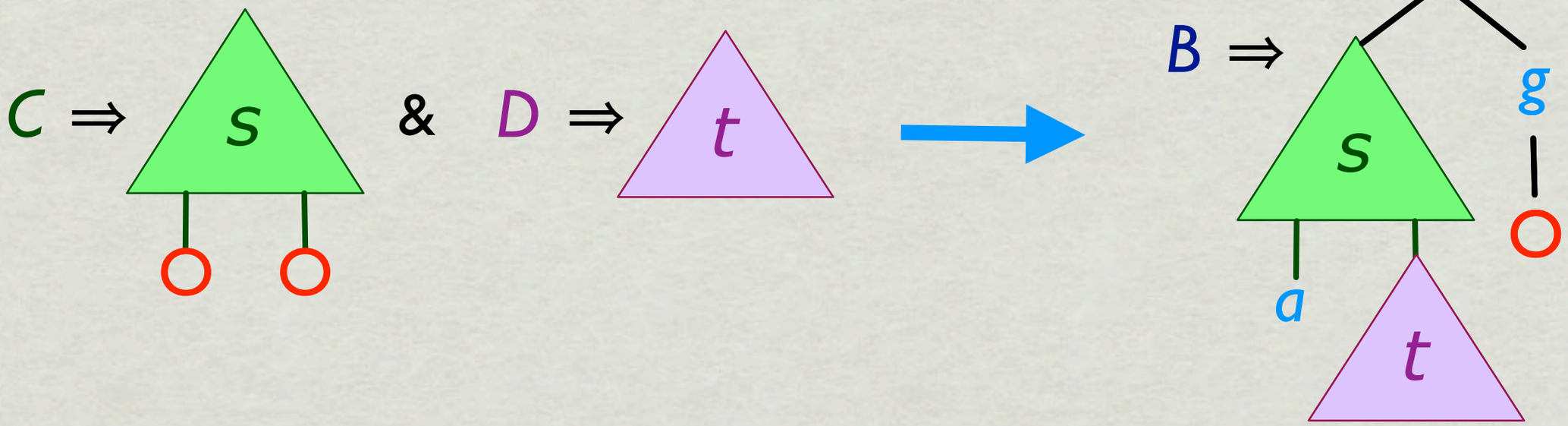


Context-Free Tree Grammars



★ *a,f,g* : terminal symbol

rnk(*B*) = 1
rnk(*C*) = 2
rnk(*D*) = 0



Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

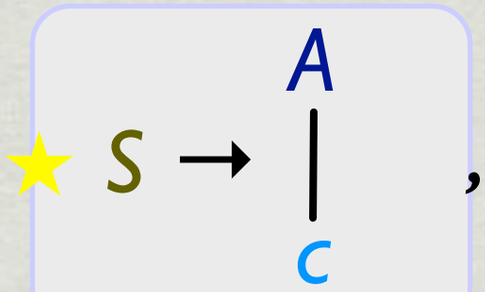
★ $S \to A$ | c ,

★ $A \to h$ | O ,

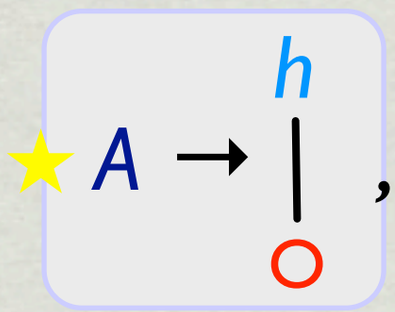
★ $A \to f(a, A, e)$, $A \to g(b, O, d)$

Example

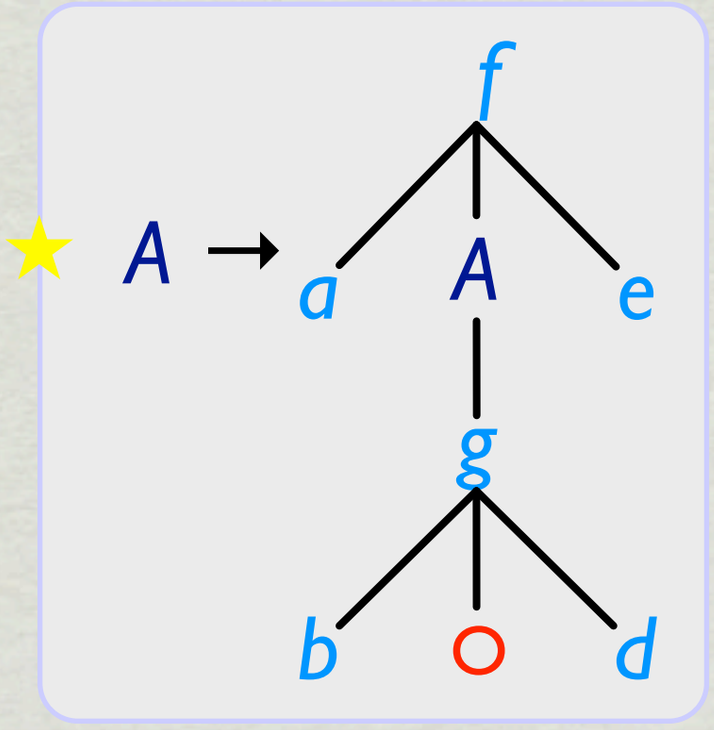
- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$



,



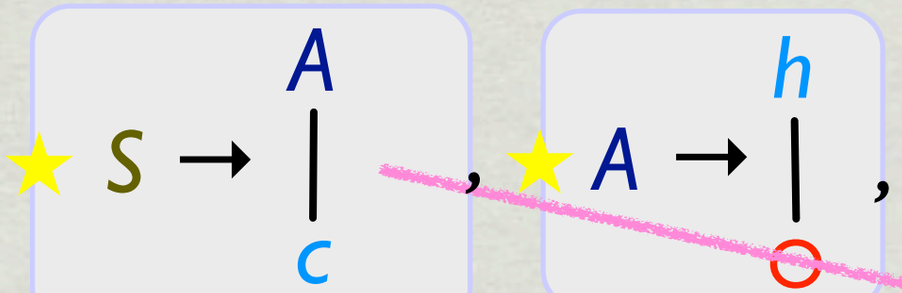
,



$S \Rightarrow$

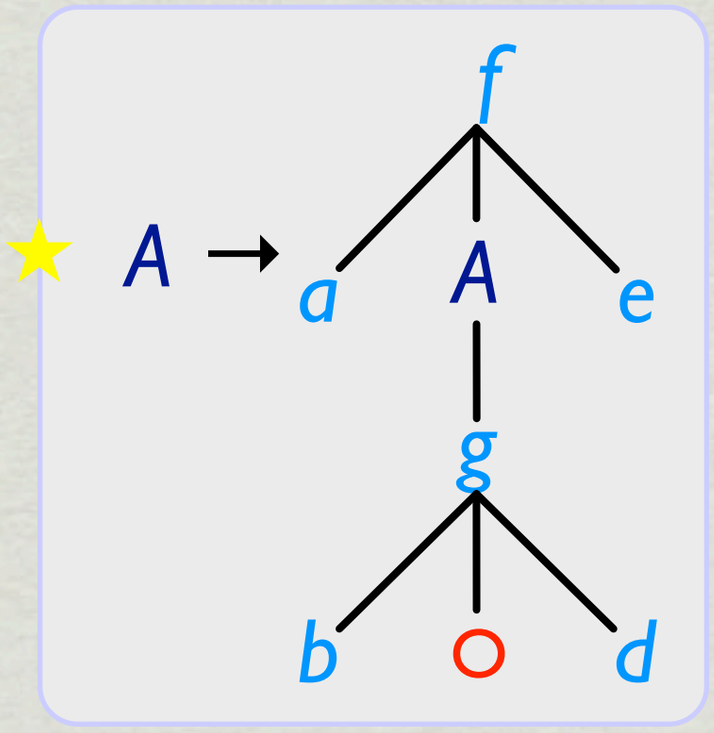
Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$



,

,



$S \Rightarrow$

Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

★ $S \to A(c)$,

★ $A \to h(0)$,

★ $A \to f(a, A(g(b, 0, d)), e)$

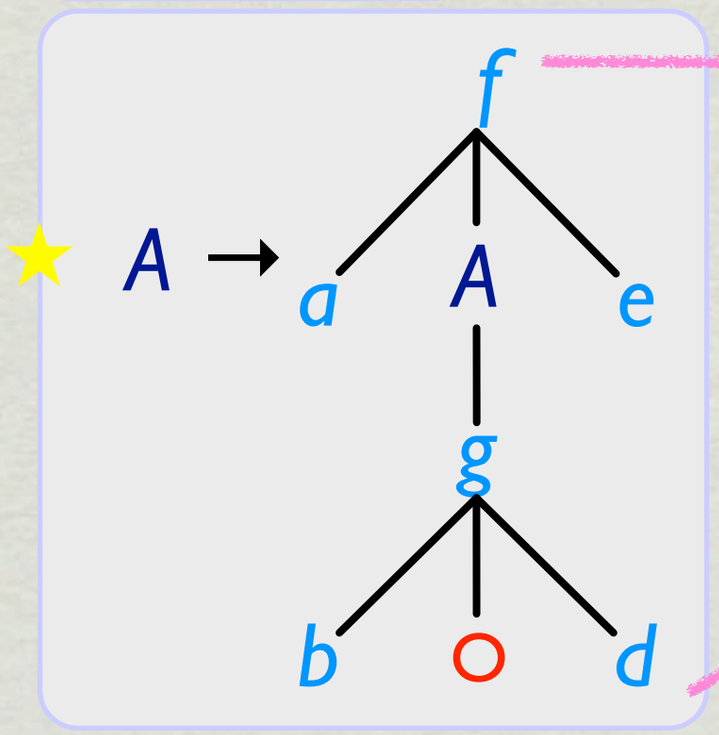
$S \Rightarrow A(c)$

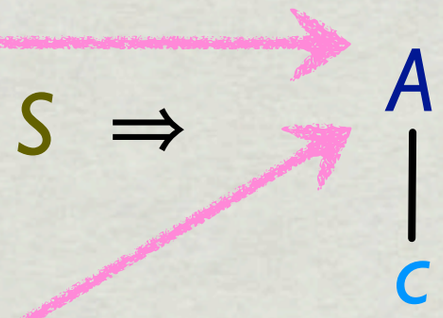
Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

,

,





Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

★ $S \to A$ | c ,

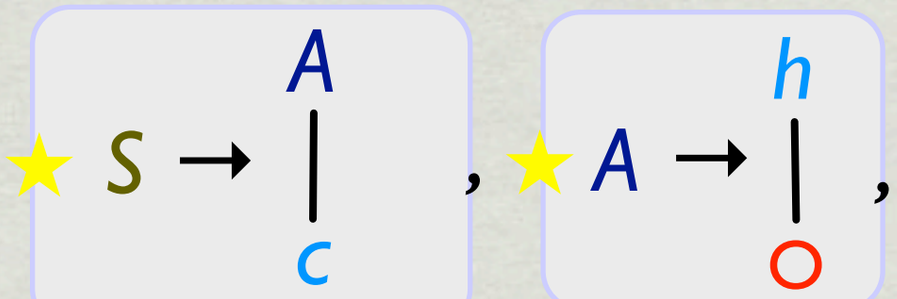
★ $A \to h$ | $\circ$,

★ $A \to f(a, A, e)$ with $A \to g(b, \circ, d)$

$S \Rightarrow f(a, A, e) \Rightarrow f(a, g(b, c, d), e)$

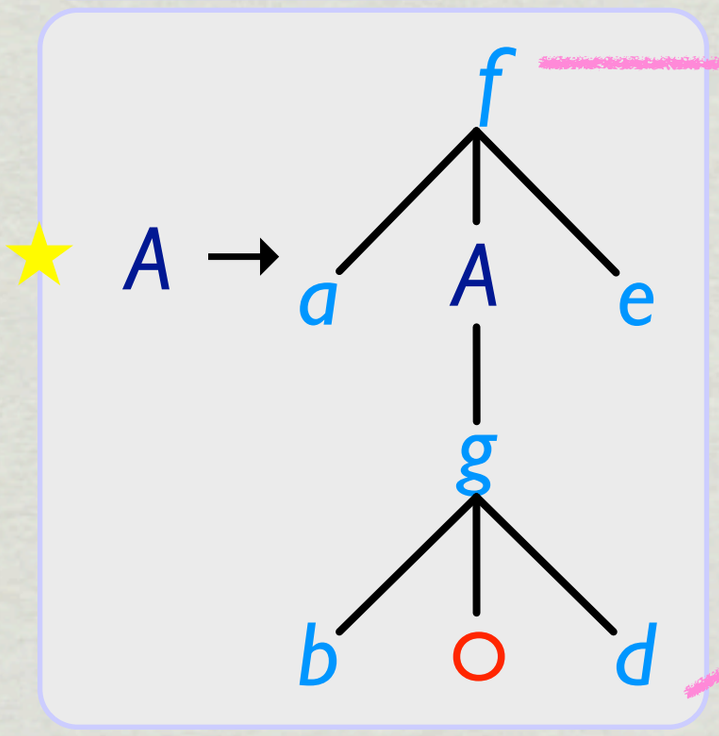
Example

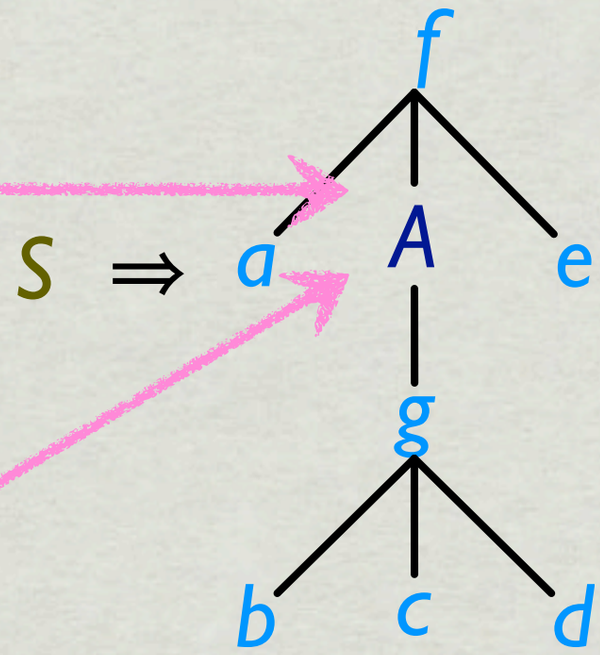
- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$



,

,





Example

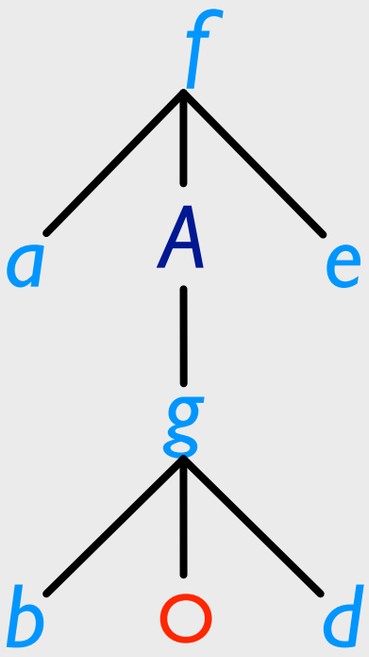
- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

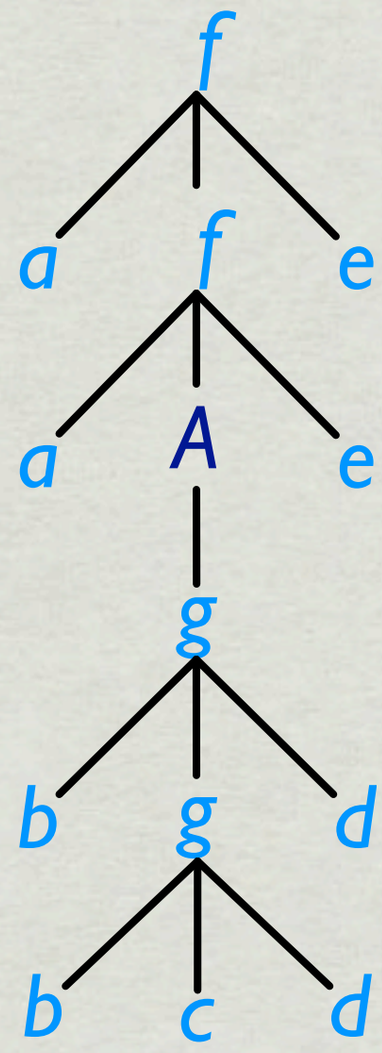
★ $S \to A$ 

,

★ $A \to h$ 

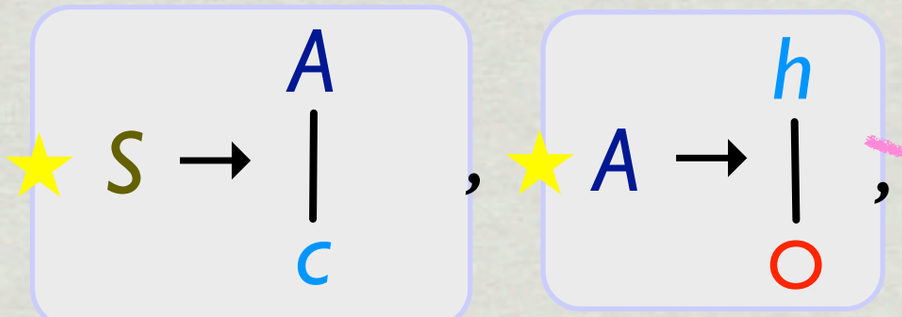
,

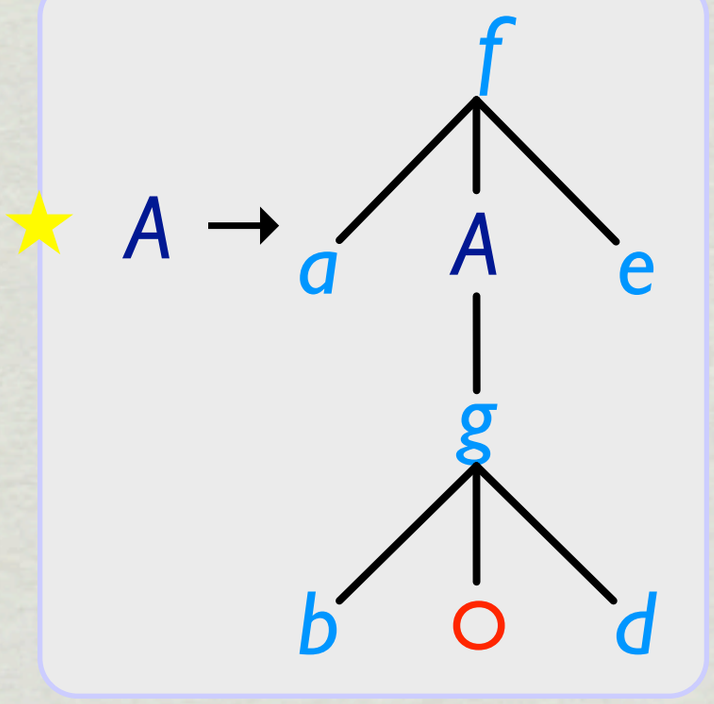
★ $A \to$ 

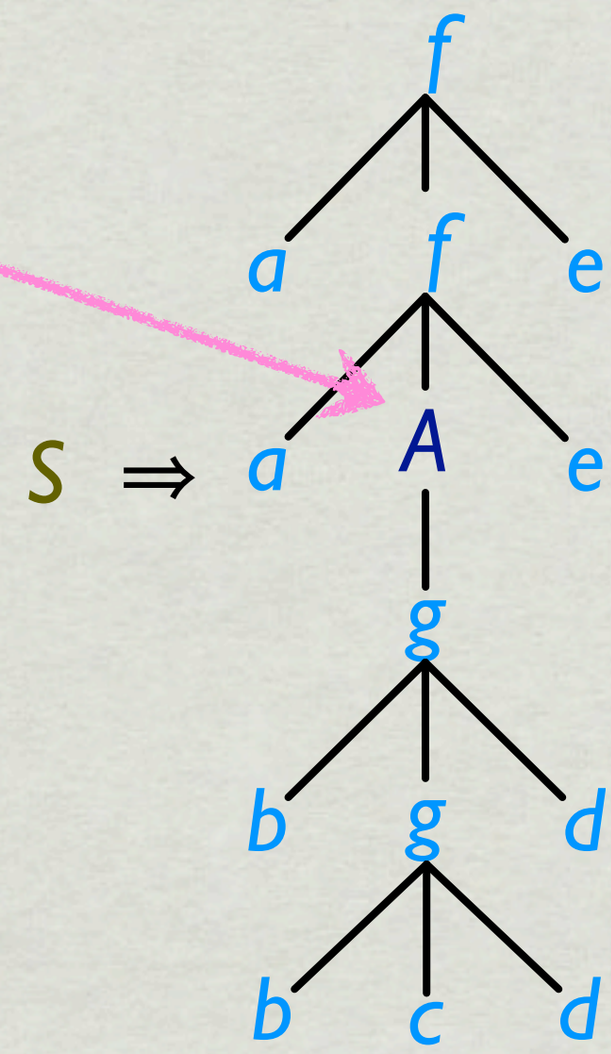
$S \Rightarrow$ 

Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

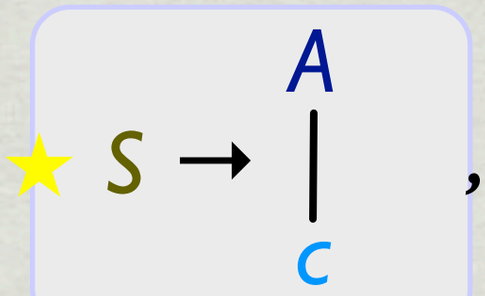


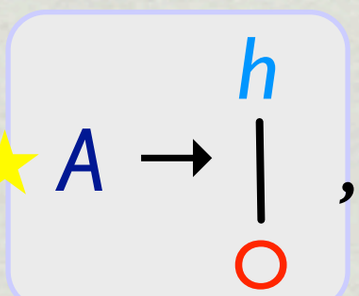


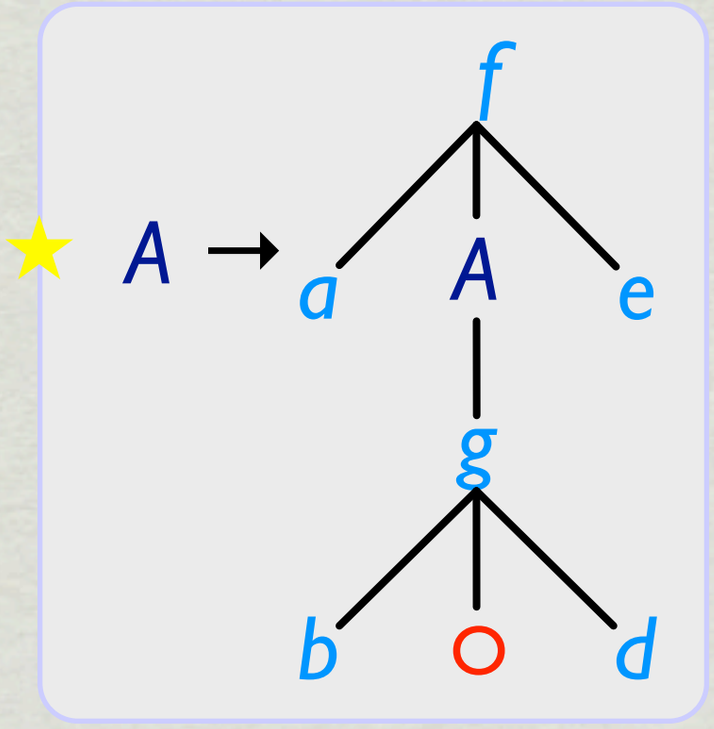


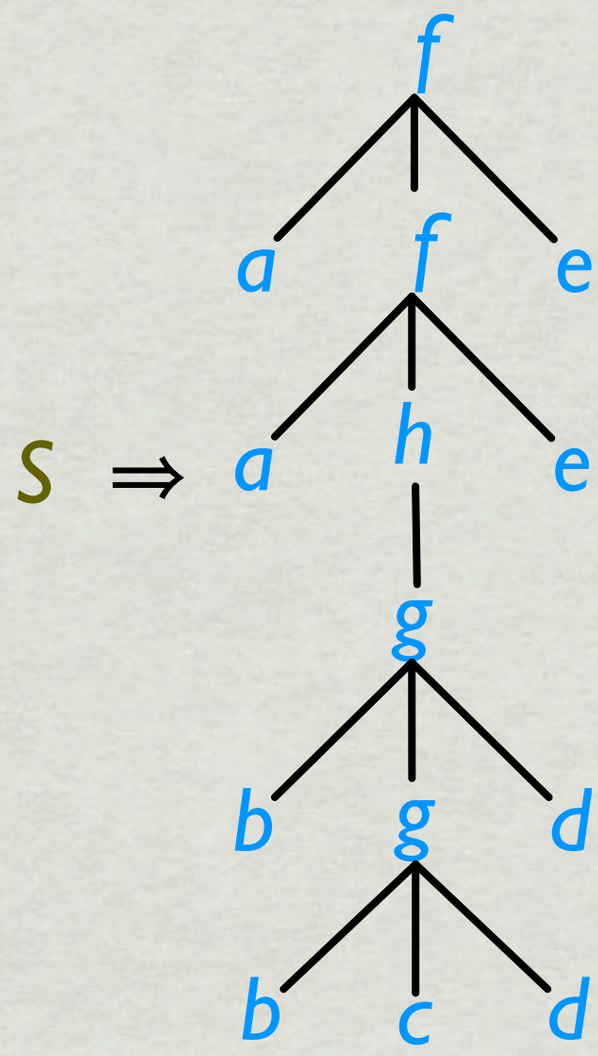
Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$









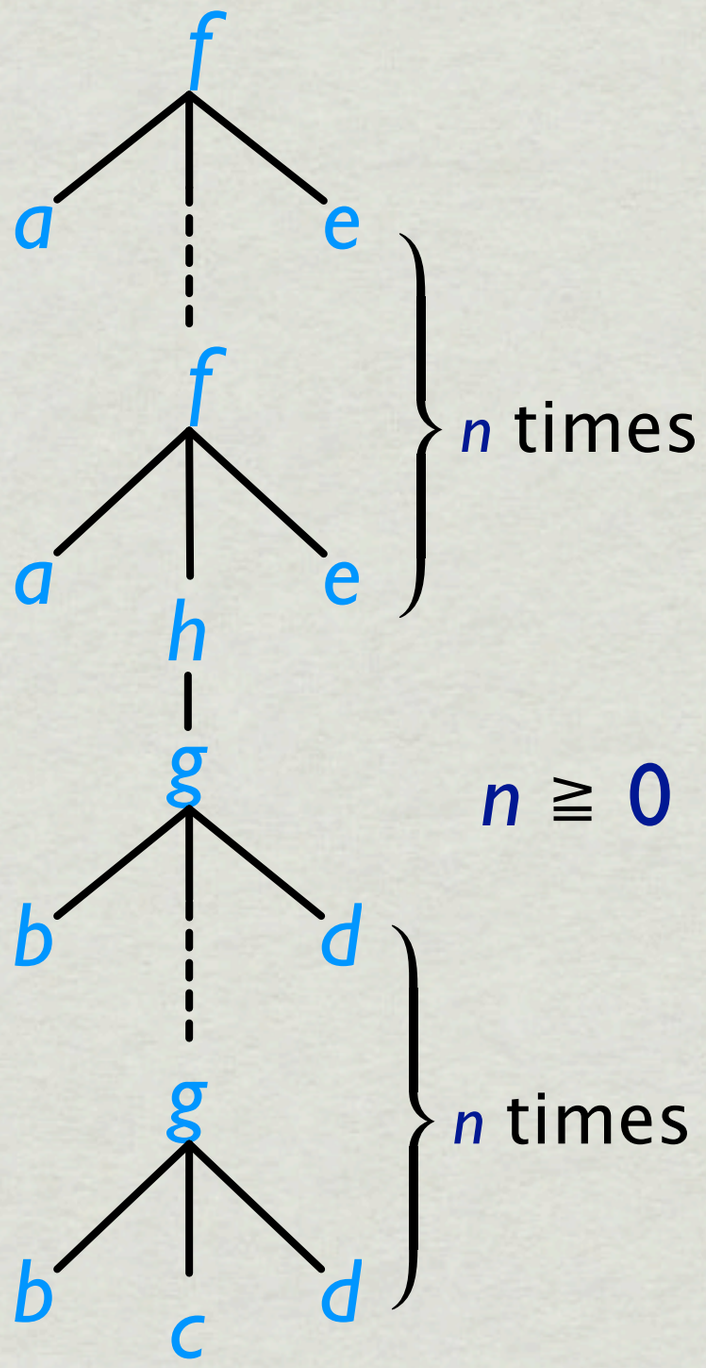
Example

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

★ $S \to A$, ★ $A \to h$,

★ $A \to f(a, A, e)$, $A \to g(b, c, d)$

$S \Rightarrow$



$n \geqq 0$

yield of $L(G)$:
$\{ a^n b^n c d^n e^n \mid n \geqq 0 \}$
non-context-free

Substructure/Context Decomposition

CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

(u,v): **context**

w : **substring**

Substructure/Context Decomposition

CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

  (u,v): **context**

  w : **substring**

Context-Free Tree Grammars

Substructure/Context Decomposition

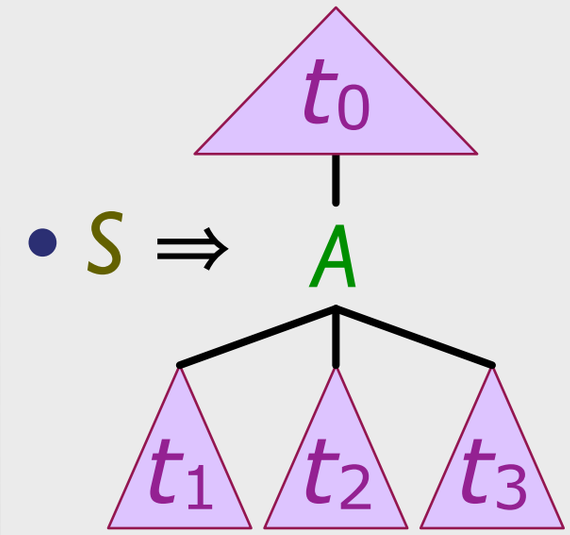
CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

  (u,v): **context**

  w : **substring**

Context-Free Tree Grammars



Substructure/Context Decomposition

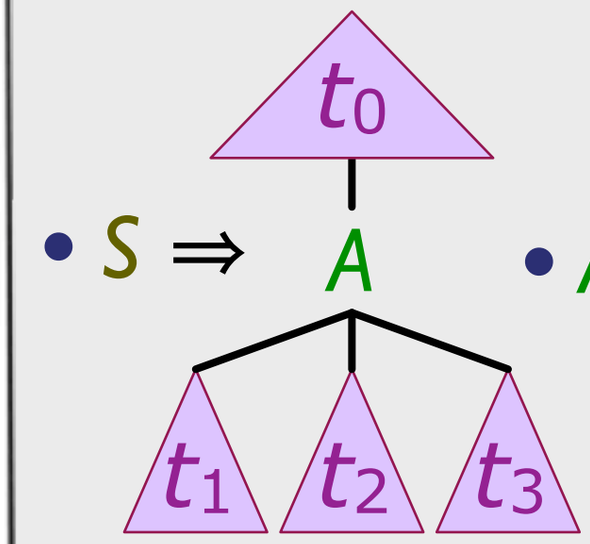
CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

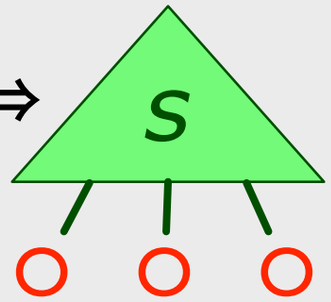
  (u,v): **context**

  w : **substring**

Context-Free Tree Grammars



- $S \Rightarrow$

- $A \Rightarrow$ 

Substructure/Context Decomposition

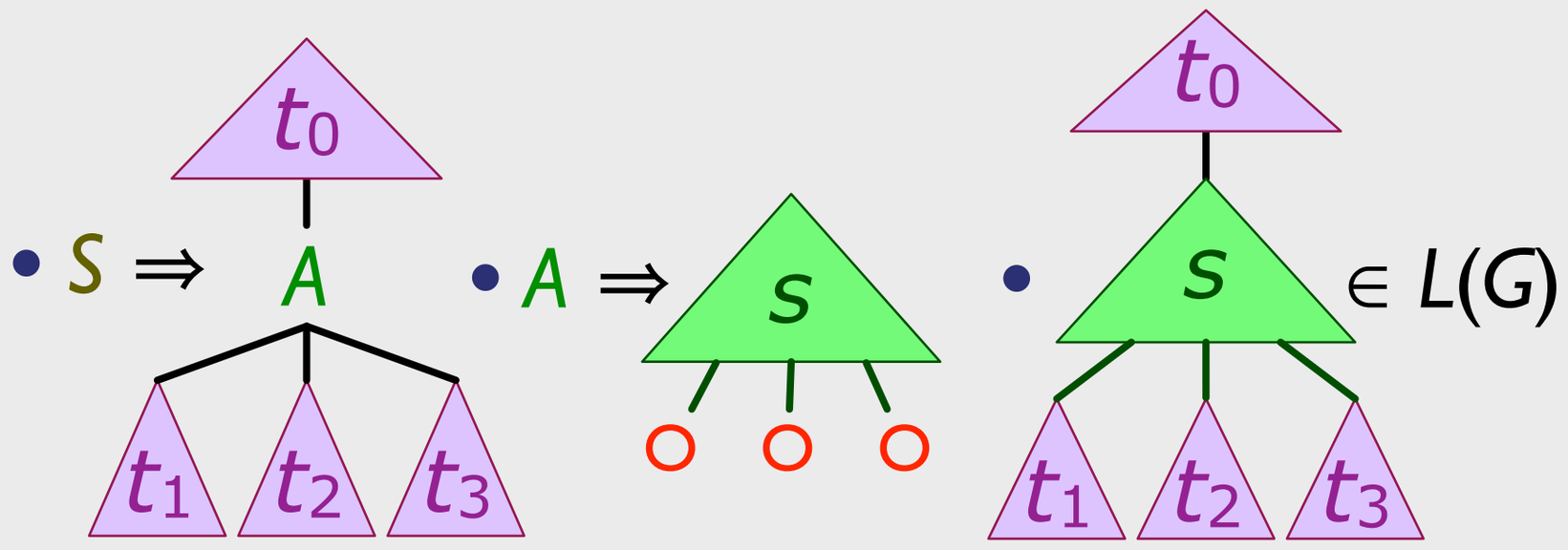
CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

(u,v): **context**

w : **substring**

Context-Free Tree Grammars



Substructure/Context Decomposition

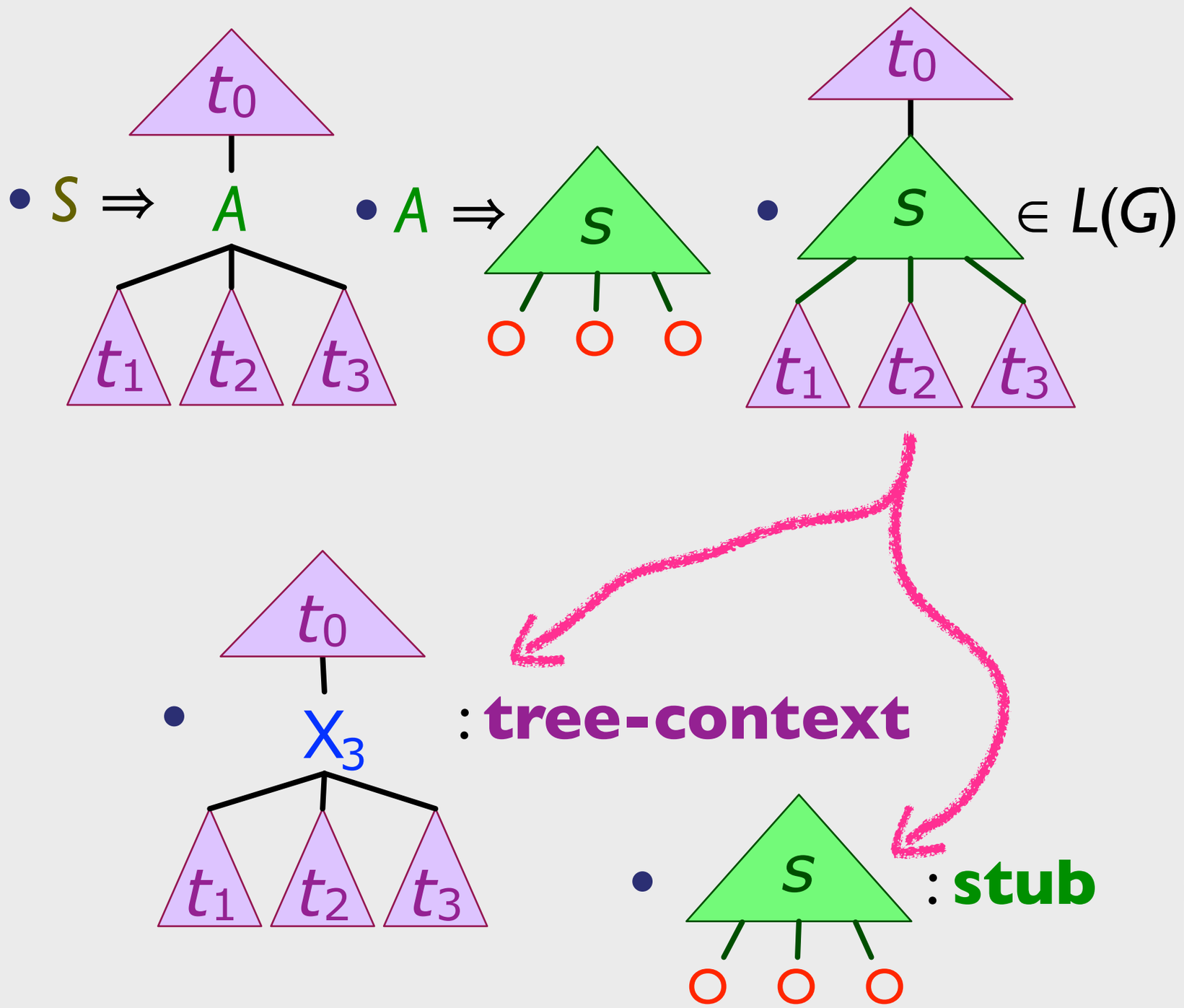
CFGs

- $S \Rightarrow uAv$
- $A \Rightarrow w$
- $uwv \in L(G)$

(u,v): **context**

w : **substring**

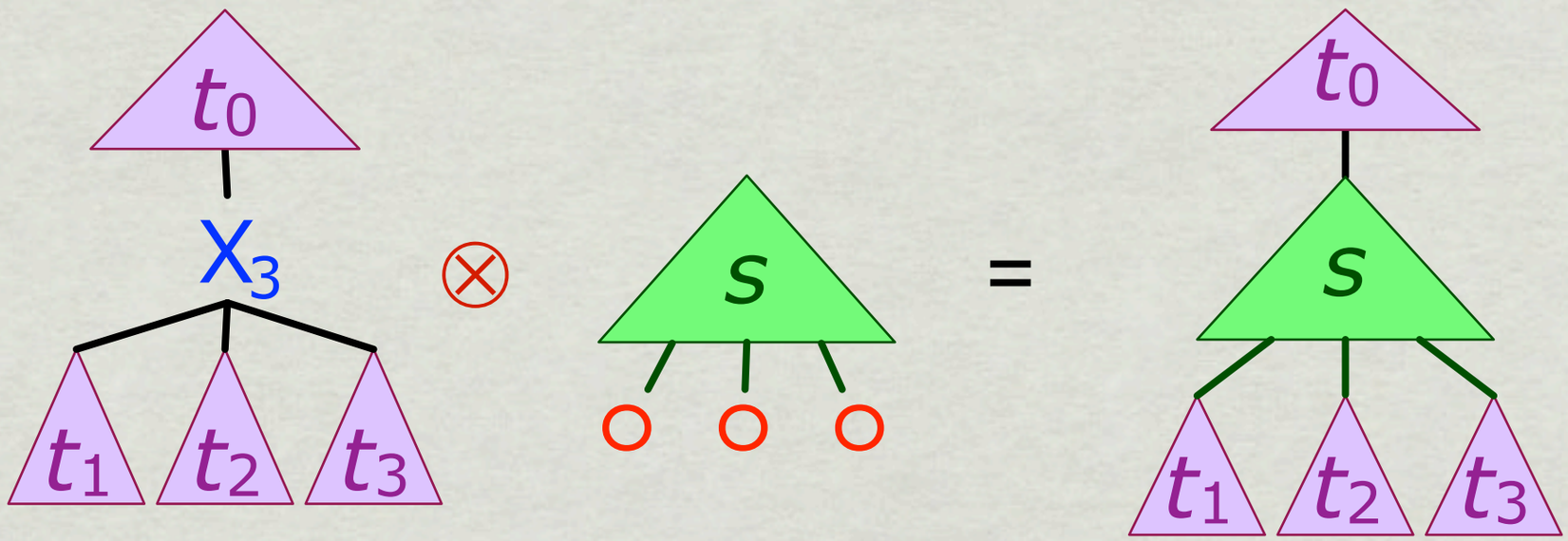
Context-Free Tree Grammars



Composition/Decomposition

- *m-tree-context* **c** ... tree with a special symbol X_m of rank *m*
- *m-stub* **s** ... tree with *m* open leaves ○
- **c** ⊗ **s** ..... substitute **s** for X_m in **c**

cf. context-free case:
$\langle u,w \rangle \otimes v = uvw$

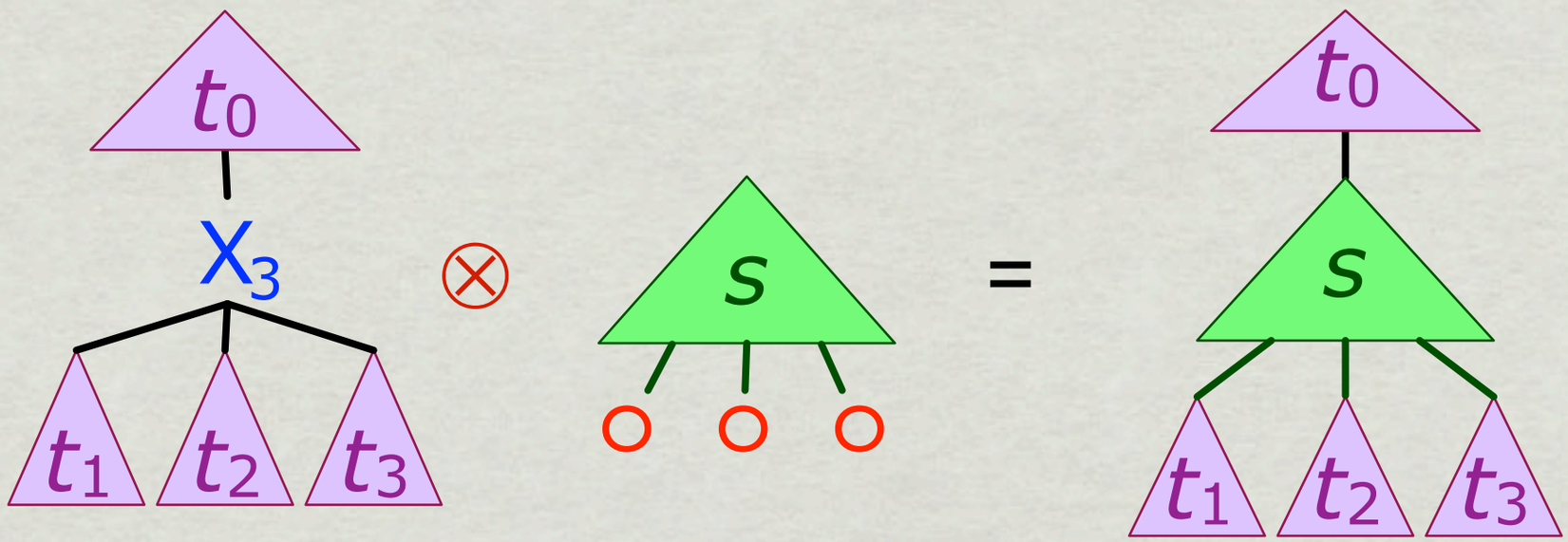


Composition/Decomposition

- *m-tree-context* **c** ... tree with a special symbol X_m of rank *m*
- *m-stub* **s** ... tree with *m* open leaves ○
- **c** ⊗ **s** ..... substitute **s** for X_m in **c**

cf. context-free case:
$\langle u,w \rangle \otimes v = uvw$

- $L_0/\mathbf{s} = \{ \mathbf{c} \mid \mathbf{c} \otimes \mathbf{s} \in L_0 \}$: the set of tree-contexts for **s**



Learning of Congruential *r*-CFTGs

- Learning Congruential CFGs with queries (Clark '10)

- Learning Congruential CF**T**Gs from positive data and membership queries

Congruential CFTGs

Congruential CFTGs

- CFTG G is **Congruential** iff

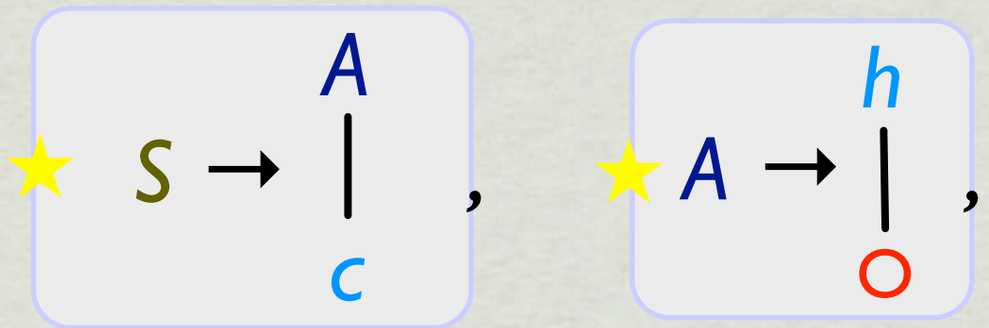
$\forall A \in N,\ \forall \boldsymbol{s}, \boldsymbol{t} (A \Rightarrow \boldsymbol{s}, A \Rightarrow \boldsymbol{t})$,

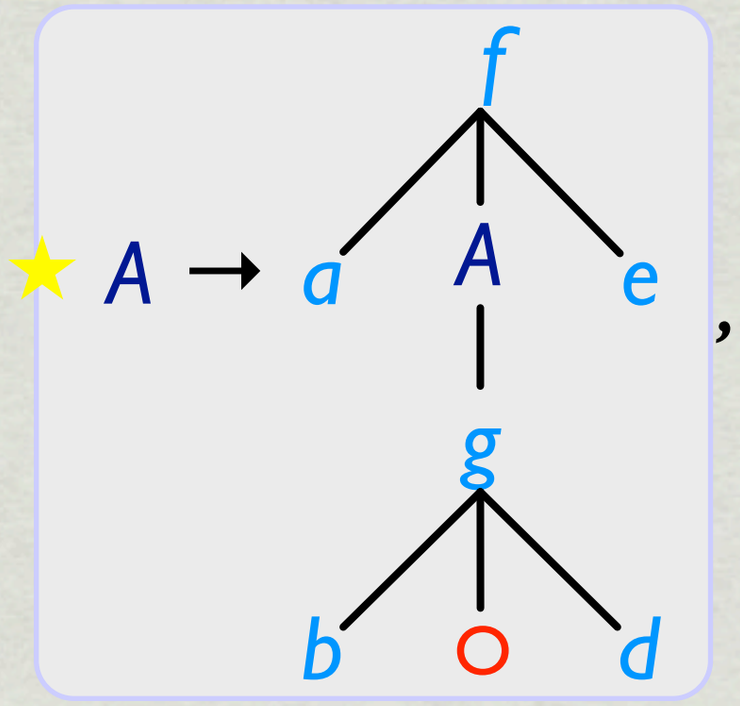
$$L(G)/\boldsymbol{s} = L(G)/\boldsymbol{t}$$

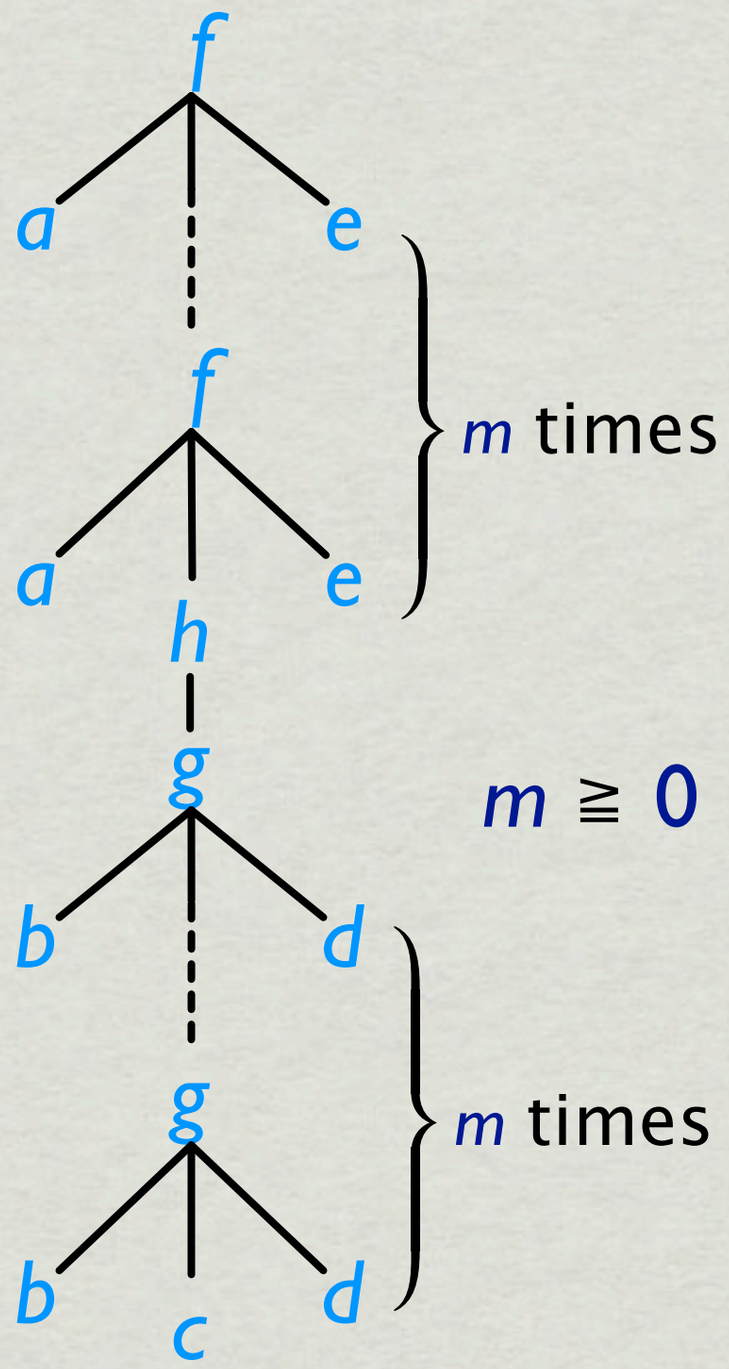
$$(\forall \boldsymbol{c}\ ,\ \boldsymbol{c} \otimes \boldsymbol{s} \in L(G) \ \Leftrightarrow\ \boldsymbol{c} \otimes \boldsymbol{t} \in L(G))$$

Congruential CFTG

- $N_0 = \{ S \},\ N_1 = \{ A \}$
- $\Sigma_0 = \{ a,b,c,d,e \},\ \Sigma_1 = \{ h \},\ \Sigma_3 = \{ f,g \}$

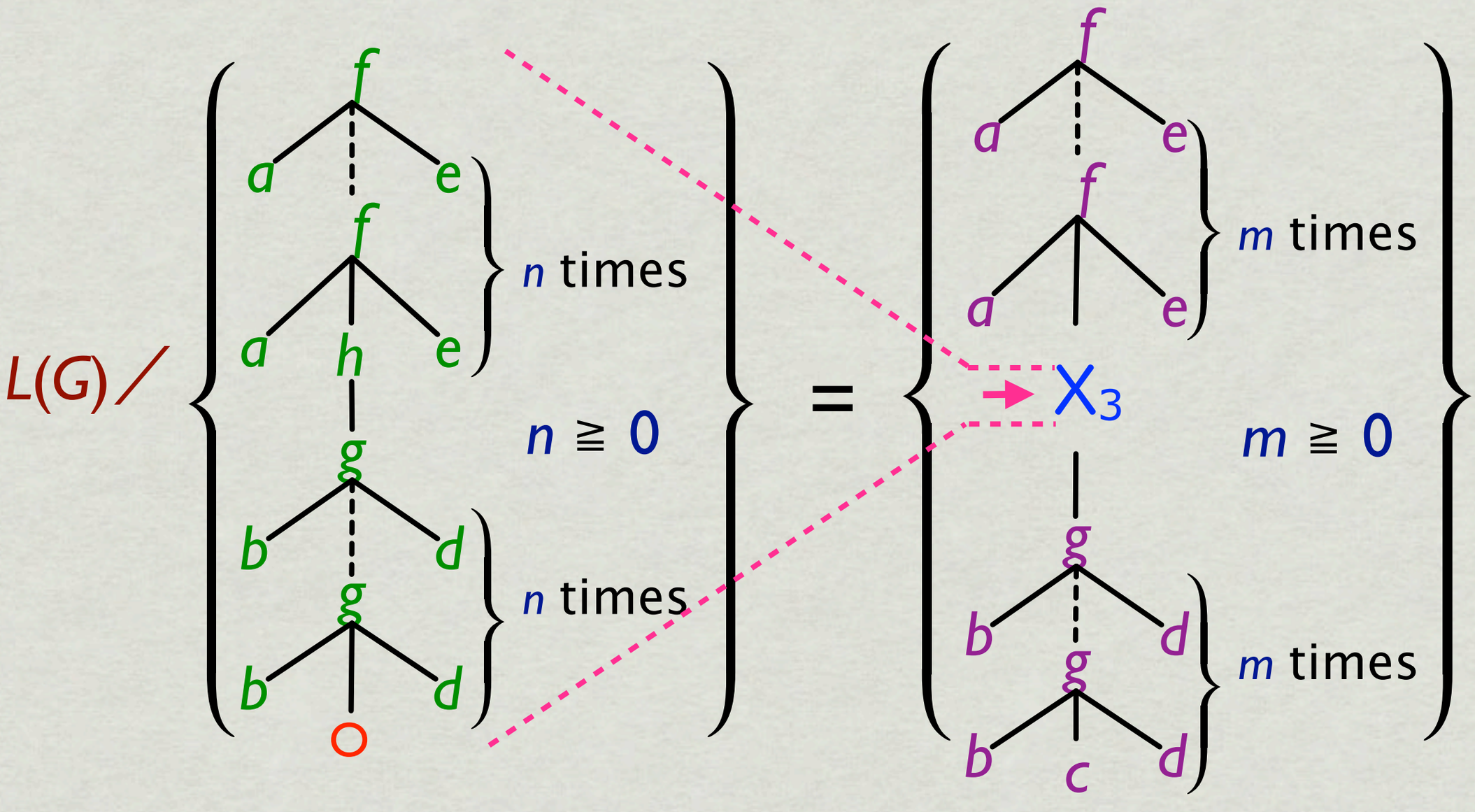






yield: $\{ a^m b^n c d^n e^m \mid n \geqq m \}$

- $\forall \boldsymbol{s},\boldsymbol{t}(A \Rightarrow \boldsymbol{s}, A \Rightarrow \boldsymbol{t}),\quad L(G)/\boldsymbol{s} = L(G)/\boldsymbol{t}$



Congruential *r*-CFTGs

- Congruential 1-CFTGs generate
  - All regular languages
  - Dyck language (well-bracketed parentheses)
- The yield languages of Congruential 2-CFTGs cover
  - $\{ \langle a^m\ b^n\ c^m\ d^n \rangle \mid m, n > 0 \}$
  - $\{ ww \mid w \in \Sigma^* \}$

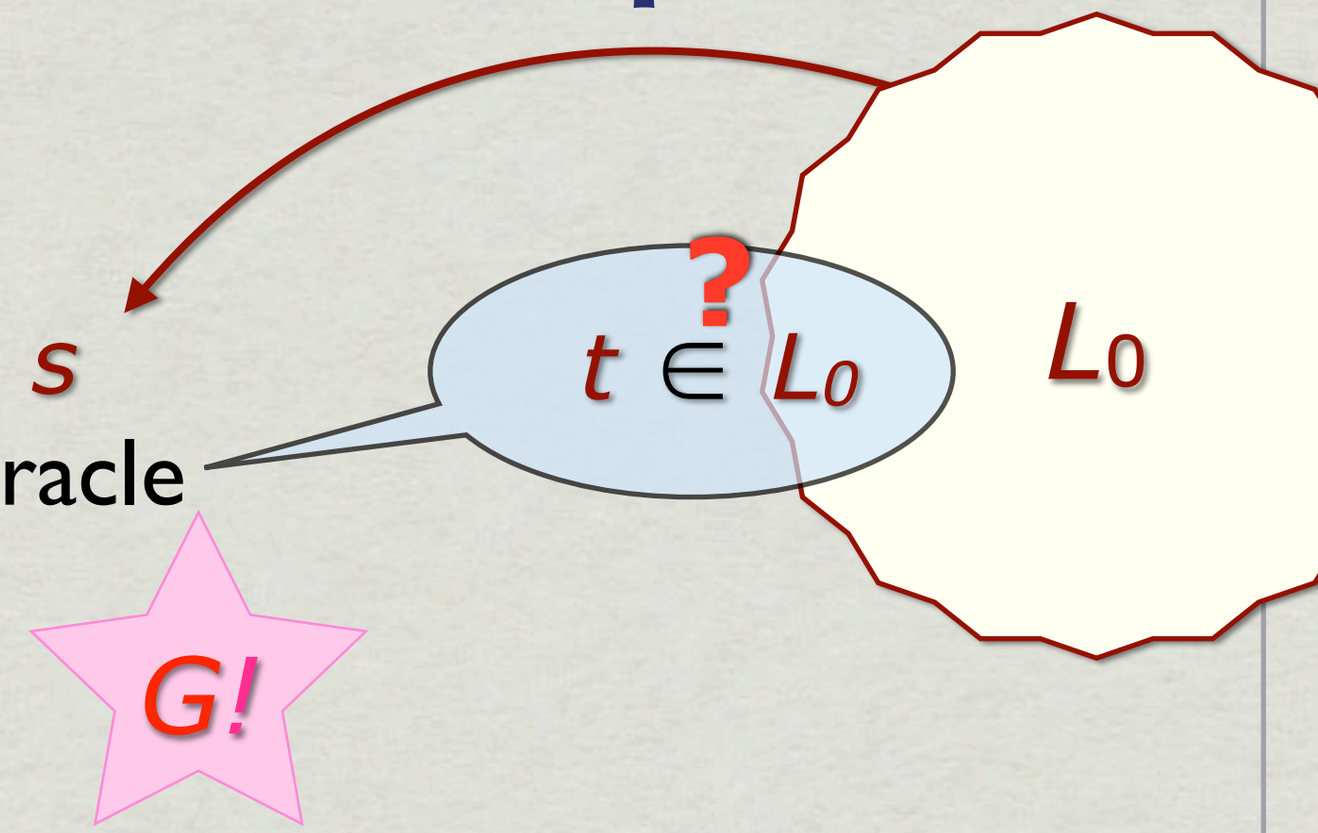
    ✳ *a b c a b c*

    ✳ d’chind em Hans es huus lönd hälfe aastriiche (Swiss-German)
  - etc.

Identification in the Limit from Positive Data and Membership Queries

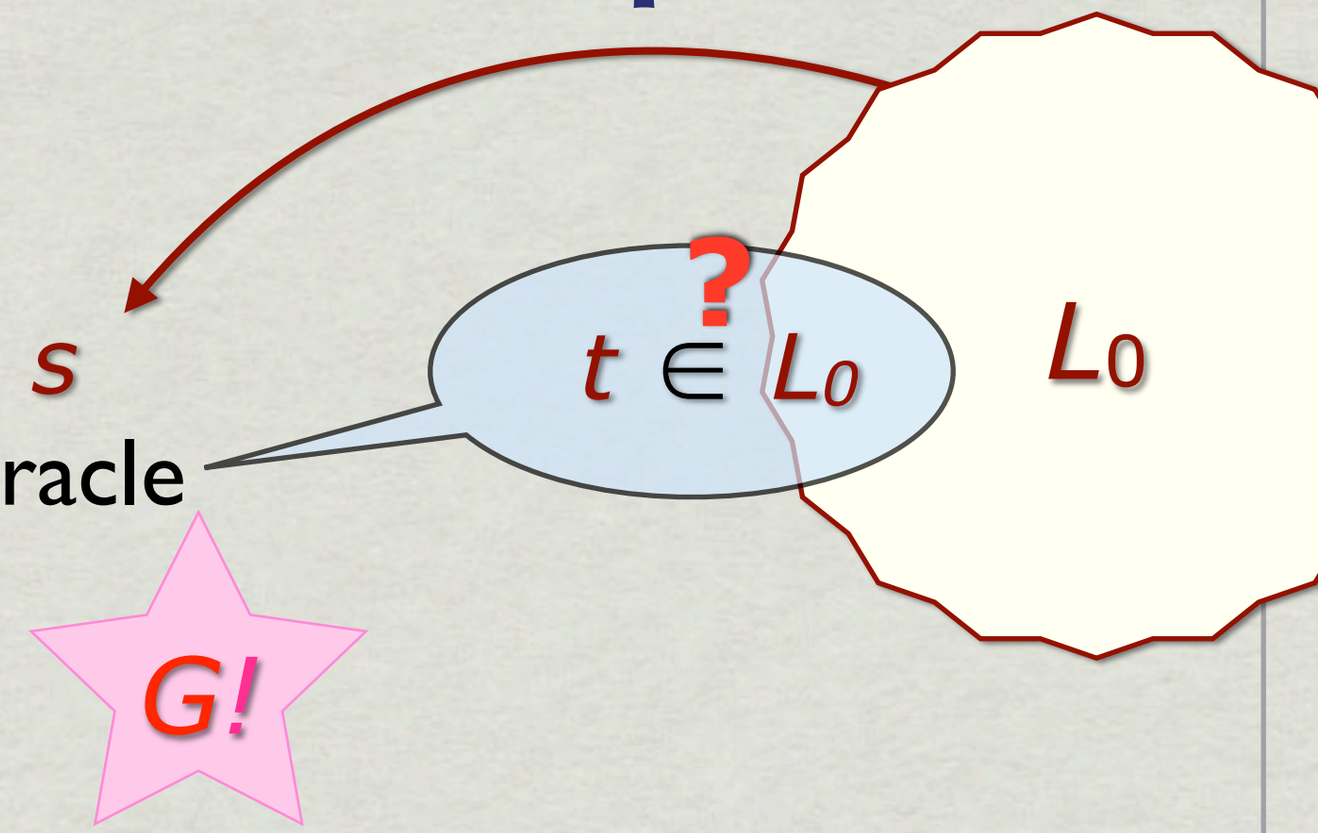
Identification in the Limit from Positive Data and Membership Queries

- Learner
  - gets a positive example s
  - calls the membership oracle
  - updates the conjecture



Identification in the Limit from Positive Data and Membership Queries

- Learner
  - gets a positive example s
  - calls the membership oracle
  - updates the conjecture



- Identification in the Limit:
  - convergence to a grammar for the target
  - not have to terminate

$$L(G) = L_0$$

Learner's Conjecture

Learner's Conjecture

- ***S*** : Finite set of k-stubs (for $0 \leqq k \leqq r$)
- ***C*** : Finite set of k-tree-contexts (for $0 \leqq k \leqq r$)

Learner's Conjecture

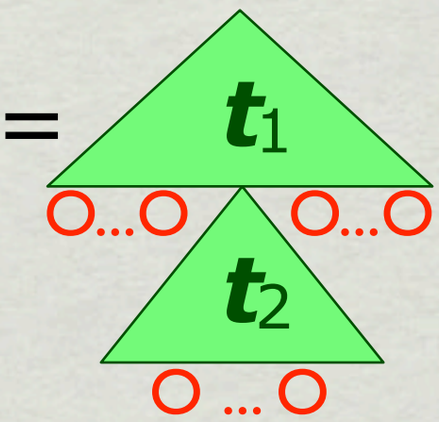
- $\boldsymbol{S}$: Finite set of k-stubs (for $0 \leqq k \leqq r$)
- $\boldsymbol{C}$: Finite set of k-tree-contexts (for $0 \leqq k \leqq r$)
- Nonterminals: $N = \boldsymbol{S}$
  - $[\![\boldsymbol{t}]\!] \Rightarrow \boldsymbol{t}$ for all $[\![\boldsymbol{t}]\!] \in N$, ($\boldsymbol{t}$ is a rnk($[\![\boldsymbol{t}]\!]$)-stub)
- Initial Symbols: $\{ [\![\boldsymbol{t}]\!] \in N \mid \boldsymbol{t} \in L_0 \cap \boldsymbol{S} \}$

Learner's Conjecture

- $\boldsymbol{S}$: Finite set of k-stubs (for $0 \leqq k \leqq r$)
- $\boldsymbol{C}$: Finite set of k-tree-contexts (for $0 \leqq k \leqq r$)
- Nonterminals: $N = \boldsymbol{S}$
  - $[\![\boldsymbol{t}]\!] \Rightarrow \boldsymbol{t}$ for all $[\![\boldsymbol{t}]\!] \in N$, ($\boldsymbol{t}$ is a rnk($[\![\boldsymbol{t}]\!]$)-stub)
- Initial Symbols: $\{\ [\![\boldsymbol{t}]\!] \in N \mid \boldsymbol{t} \in L_0 \cap \boldsymbol{S}\ \}$
- Rules (Type I, II):
  - $[\![\boldsymbol{t}]\!] \to [\![\boldsymbol{t}_1]\!]\langle o \ldots o, [\![\boldsymbol{t}_2]\!]\langle o \ldots o\rangle, o \ldots o\rangle$ if $\boldsymbol{t} = \boldsymbol{t}_1[o \ldots o, \boldsymbol{t}_2[o \ldots o], o \ldots o]$,
  - $[\![f\langle o \ldots o\rangle]\!] \to f\langle o \ldots o\rangle$ for $f \in \Sigma$

cf. string case:
$[\![uv]\!] \to [\![u]\!][\![v]\!]$ & $[\![a]\!] \to a$

$\boldsymbol{t} =$



Type III

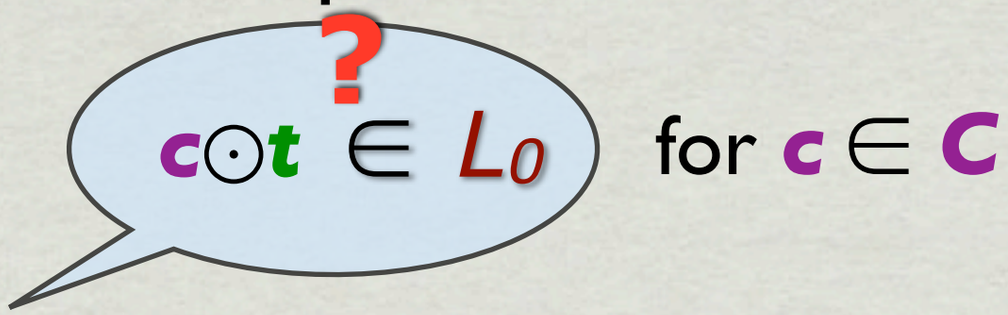
cf. Substitutability:

if $L_0/s \cap L_0/t \neq \emptyset$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s = L_0/t$ $(c \otimes s \in L_0 \Leftrightarrow c \otimes t \in L_0)$

Type III

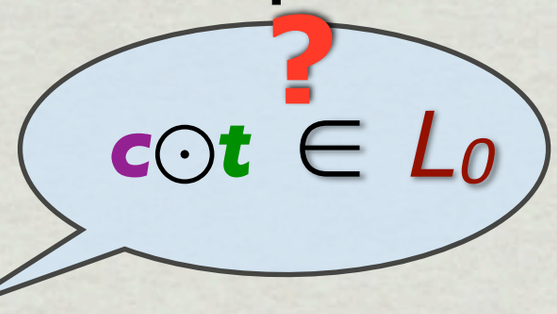
membership oracle



for $c \in C$

Type III

membership oracle

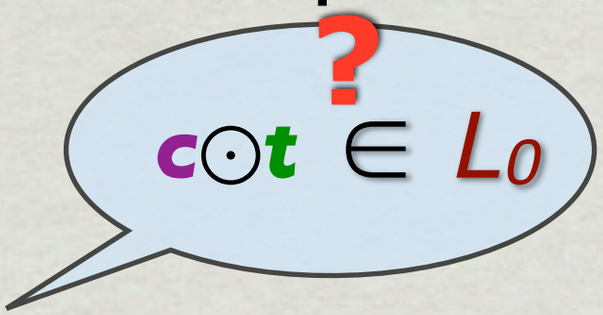


for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$

Type III

membership oracle

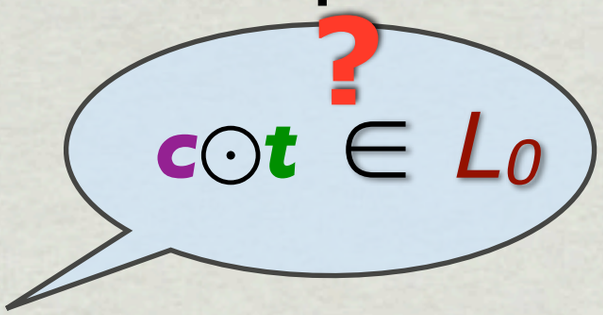


for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- $L_0/s = L_0/t$ implies $L_0/s \cap C = L_0/t \cap C$

Type III

membership oracle

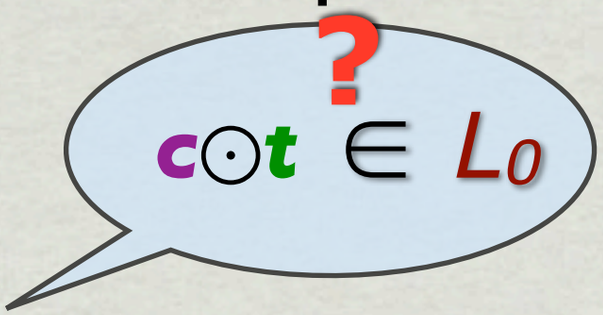


for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- $L_0/s = L_0/t$ implies $L_0/s \cap C = L_0/t \cap C$
  - $[\![s]\!] \rightarrow [\![t]\!]$ is **incorrect** if $L_0/s \neq L_0/t$

Type III

membership oracle

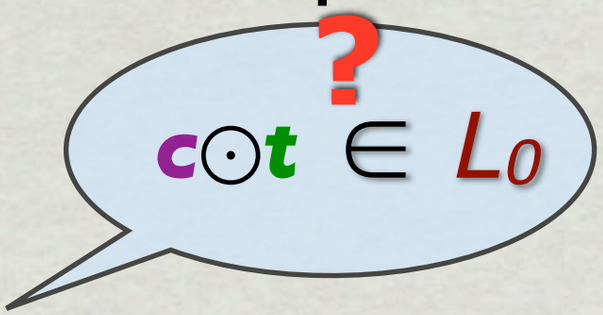


for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \to [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- $L_0/s = L_0/t$ implies $L_0/s \cap C = L_0/t \cap C$
  - $[\![s]\!] \to [\![t]\!]$ is **incorrect** if $L_0/s \neq L_0/t$
  - If C is rich enough, G has no incorrect rules

Type III

membership oracle

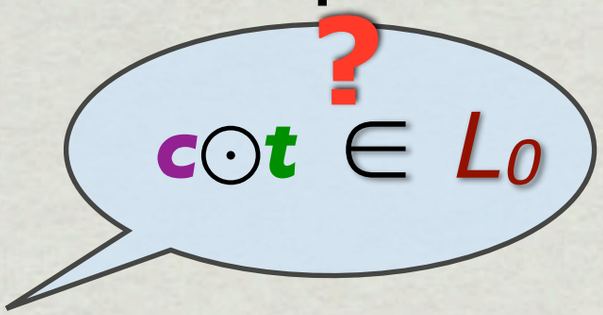


for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- $L_0/s = L_0/t$ implies $L_0/s \cap C = L_0/t \cap C$
- $[\![s]\!] \rightarrow [\![t]\!]$ is **incorrect** if $L_0/s \neq L_0/t$
- If C is rich enough, G has no incorrect rules
  - For each pair $[\![s]\!], [\![t]\!] \in N$ such that $L_0/s \neq L_0/t$, there should be $c \in C \cap ((L_0/s - L_0/t) \cup (L_0/t - L_0/s))$

Type III

membership oracle



for $c \in C$

- Rules (Type III):
  - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- $L_0/s = L_0/t$ implies $L_0/s \cap C = L_0/t \cap C$
- $[\![s]\!] \rightarrow [\![t]\!]$ is **incorrect** if $L_0/s \neq L_0/t$
- If C is rich enough, G has no incorrect rules
  - For each pair $[\![s]\!], [\![t]\!] \in N$ such that $L_0/s \neq L_0/t$, there should be $c \in C \cap ((L_0/s - L_0/t) \cup (L_0/t - L_0/s))$
  - $|C| \leqq |S|^2$ is enough

Monotonicity

- Nonterminals: $N = \boldsymbol{S}$
- Initial Symbols: $\{ [\![\boldsymbol{t}]\!] \in N \mid \boldsymbol{t} \in L_0 \}$
- Rules (Type I,II):
  - $[\![\boldsymbol{t}]\!] \rightarrow [\![\boldsymbol{t}_1]\!]\langle o...o, [\![\boldsymbol{t}_2]\!]\langle o...o\rangle, o...o\rangle$ if $\boldsymbol{t} = \boldsymbol{t}_1[o...o, \boldsymbol{t}_2[o...o], o...o]$
  - $[\![f\langle o...o\rangle]\!] \rightarrow f\langle o...o\rangle$ for $f \in \Sigma$
- Rules (Type III):
  - $[\![\boldsymbol{s}]\!] \rightarrow [\![\boldsymbol{t}]\!]$ if $L_0/\boldsymbol{s} \cap \boldsymbol{C} = L_0/\boldsymbol{t} \cap \boldsymbol{C}$

Expansion of $\boldsymbol{S}$ $\Rightarrow$ More Rules

Expansion of $\boldsymbol{C}$ $\Rightarrow$ Less Incorrect Rules

Algorithm

Let $D := \boldsymbol{S} := \emptyset$; $\boldsymbol{C} := \{X_0\}$; $G :=$ vacuous grammar;

For n = 1,2,3,...

  let $D := \{t_1, t_2, \ldots, t_n\}$

  If $D \not\subseteq L(G)$

    then expand $\boldsymbol{S}$ with all k-stubs ($k \leqq r$) extracted from D

      i.e., $\boldsymbol{S} := \{ \boldsymbol{s} : k\text{-stubs } (k \leqq r) \mid \boldsymbol{c} \otimes \boldsymbol{s} \in D \text{ for some } \boldsymbol{c} \}$

  End if

  expand $\boldsymbol{C}$ with all k-tree-contexts ($k \leqq r$) extracted from D

  i.e., $\boldsymbol{C} := \{ \boldsymbol{c} : k\text{-tree-context } (k \leqq r) \mid \boldsymbol{c} \otimes \boldsymbol{s} \in D \text{ for some } \boldsymbol{s} \}$

  update G by $\boldsymbol{S}$ and $\boldsymbol{C}$

End for

Learning Efficiency

- Expanding S and C from a positive datum t :
  $t = e \otimes s$
  root and variable leaves of s $\Leftarrow$ some nodes of t
  At most $|t|^{r+1}$ ways of decomposition
- Construction of G from S and E :
  - Rules (Type I, II):
    - $[\![t]\!] \rightarrow [\![t_1]\!]\langle o...o, [\![t_2]\!]\langle o...o\rangle, o...o\rangle$ if $t = t_1[o...o, t_2[o...o], o...o]$
    - $[\![f]\!] \rightarrow f\langle o...o\rangle$ for $f \in \Sigma$
  - Rules (Type III):
    - $[\![s]\!] \rightarrow [\![t]\!]$ if $L_0/s \cap C = L_0/t \cap C$
- PolyTime in $||S||$ and $||C||$, which are polynomial in the data size

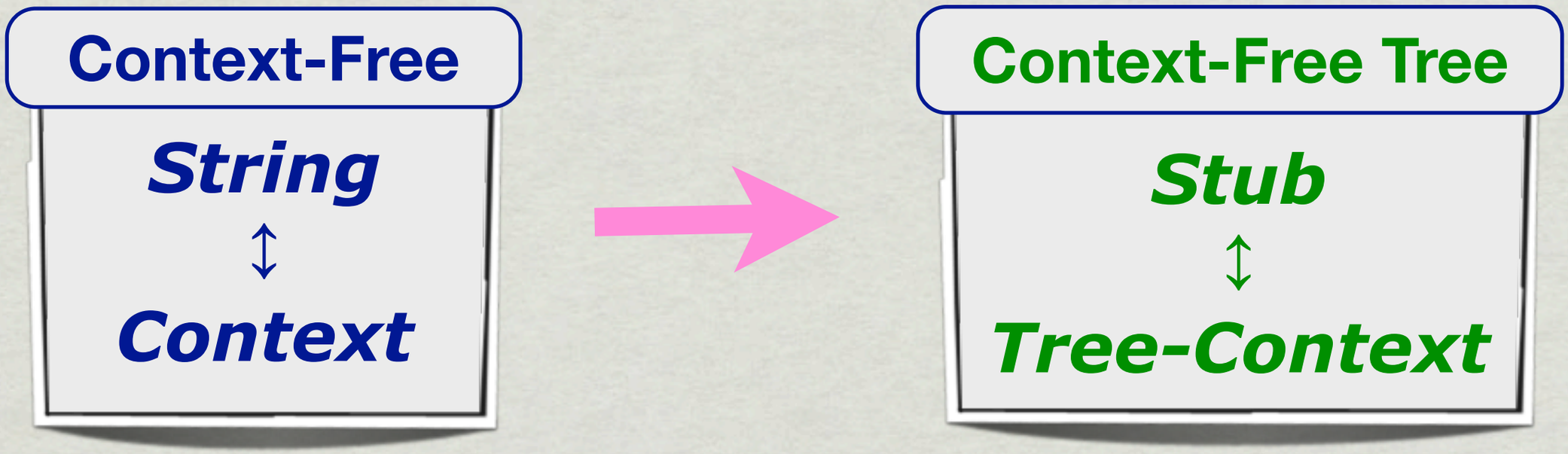
Learning Efficiency

- Let $L_0 = L(G_0)$
- When is $\boldsymbol{S}$ sufficient?
  - $A \Rightarrow \boldsymbol{s}_A$
  - $\boldsymbol{s}_A$, $\boldsymbol{s}_B[o...o,\boldsymbol{s}_C[o...o], o...o] \in \boldsymbol{S}$
    for each rule $A \rightarrow B\langle o...o, C\langle o...o\rangle, o...o\rangle$
  - $[\![\boldsymbol{s}_A]\!]$ simulates A
  - $|\boldsymbol{S}| \leqq 2|G_0|$
- When is $\boldsymbol{C}$ sufficient?
  - For each $[\![\boldsymbol{s}]\!], [\![\boldsymbol{t}]\!] \in N$ s.t. $L_0/\boldsymbol{s} \neq L_0/\boldsymbol{t}$,
    $\boldsymbol{C}$ should have $\boldsymbol{c} \in (L_0/\boldsymbol{s} - L_0/\boldsymbol{t}) \cup (L_0/\boldsymbol{t} - L_0/\boldsymbol{s})$
  - $|\boldsymbol{C}| \leqq |\boldsymbol{S}|^2$

Theorem

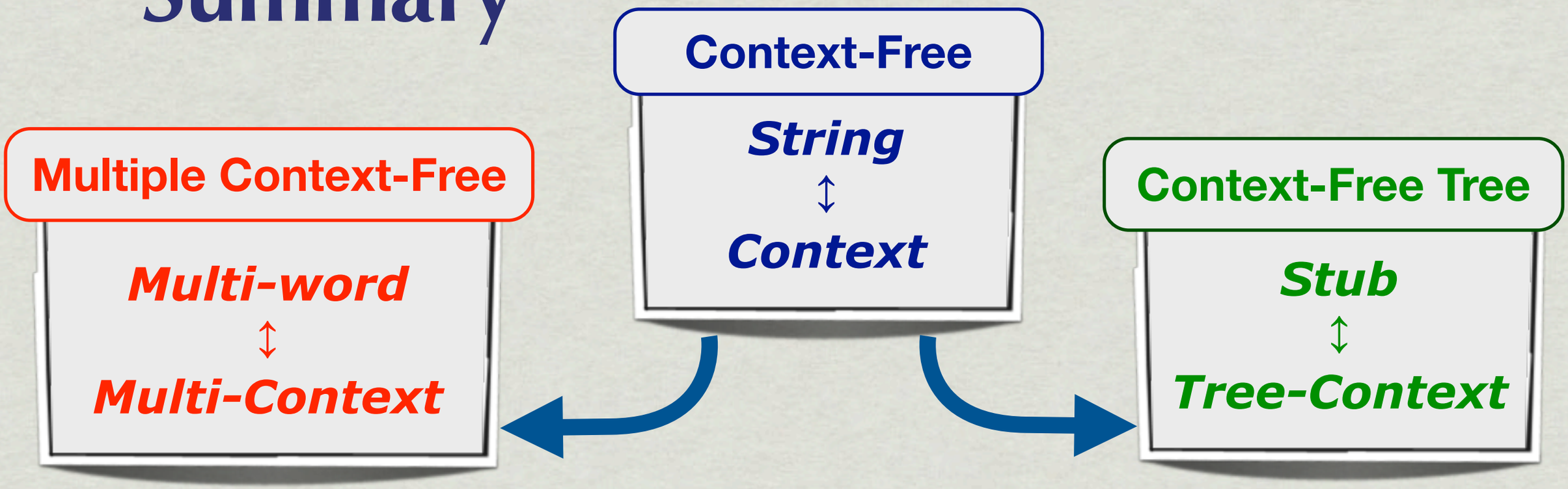
- **Congruential *r*-CFTGs are Polynomial-Time Identifiable in the Limit from Positive Data and Membership Queries**

Distributional Learning



Summary

Summary



- Generalization of substring/context decomposition
- Other formalisms such as *Hyperedge Replacement Grammars*
- Probabilistic Learning of CFGs, MCFGs, CFTGs,
- Applying our techniques to treebanks