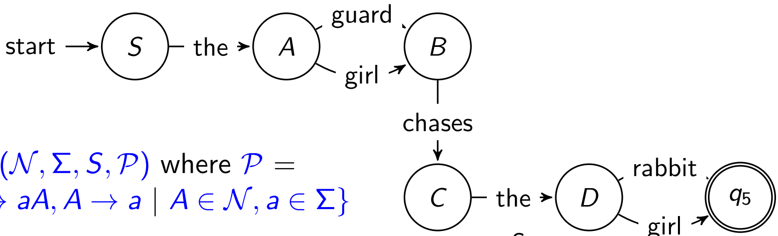
Formal Models of Language

Paula Buttery

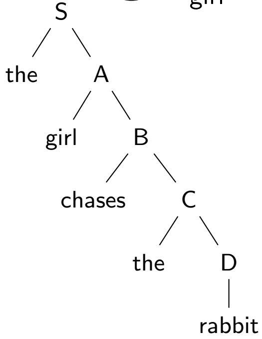
Dept of Computer Science & Technology, University of Cambridge

Regular grammars give us **linear** trees

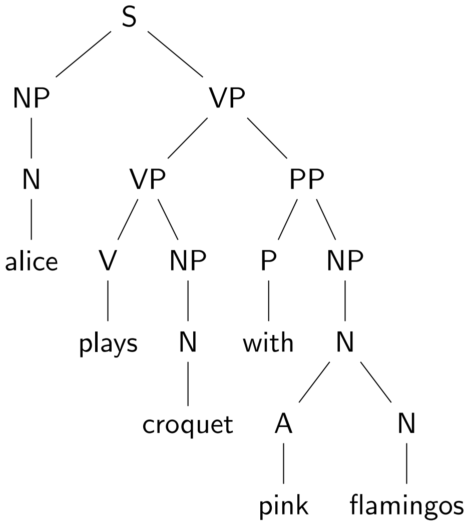


$G = (\mathcal{N}, \Sigma, S, \mathcal{P})$ where $\mathcal{P} = \{A \rightarrow aA, A \rightarrow a \mid A \in \mathcal{N}, a \in \Sigma\}$

- $\mathcal{N} = \{S, A, B, C, D, q_5\}$
- $\Sigma = \{the, girl, guard, \ldots\}$
- $\mathcal{S} = S$
- $\mathcal{P} = \{S \rightarrow the\ \ A,$
  $A \rightarrow guard\ \ B \mid girl\ \ B,$
  $B \rightarrow chases\ \ C,$
  $C \rightarrow the\ \ D,$
  $D \rightarrow girl \mid rabbit\}$



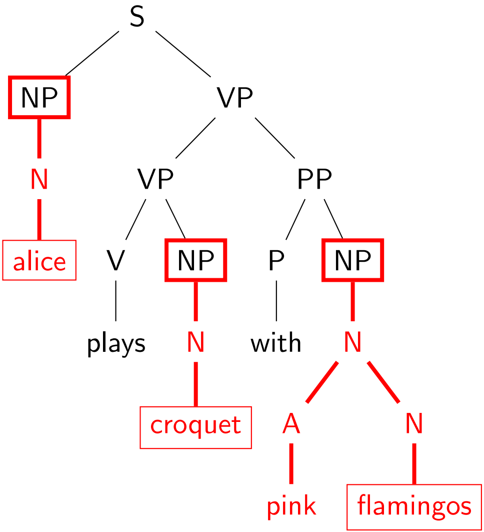
Context-free grammars capture **phrase structure**



$G = (\mathcal{N}, \Sigma, S, \mathcal{P})$ where
$\mathcal{P} = \{A \rightarrow \alpha \mid$
$A \in \mathcal{N}, \alpha \in (\mathcal{N} \cup \Sigma)^*\}$

A brief excursion into linguistic terminology...

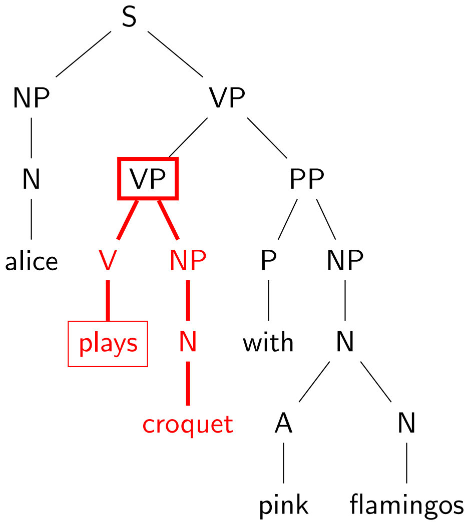
Context-free grammars capture **phrase structure**



When modelling natural language, linguists label the non-terminal symbols with names that encode the most *influential* word in the phrase. They call this influential word the **head**.

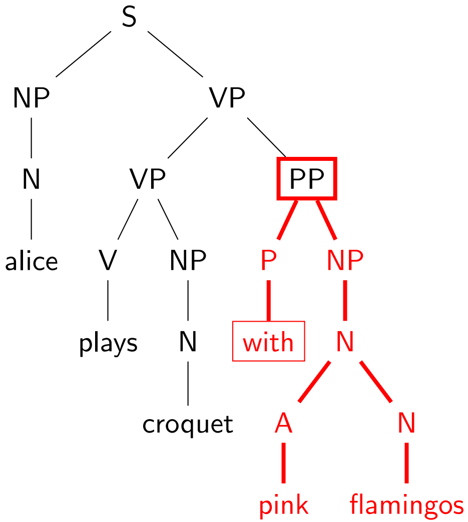
- noun phrases, *NP*, have a head noun

Context-free grammars capture **phrase structure**



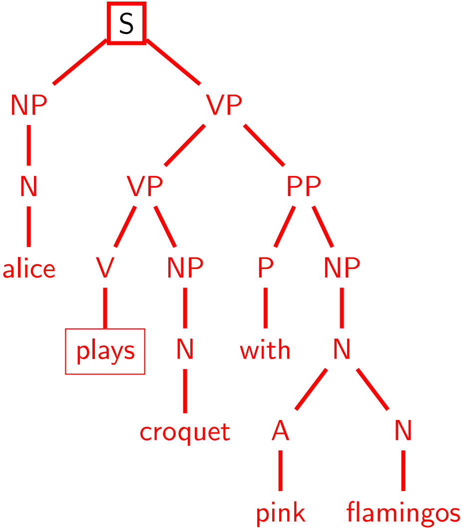
- verb phrases, *VP*, have a head verb

Context-free grammars capture **phrase structure**



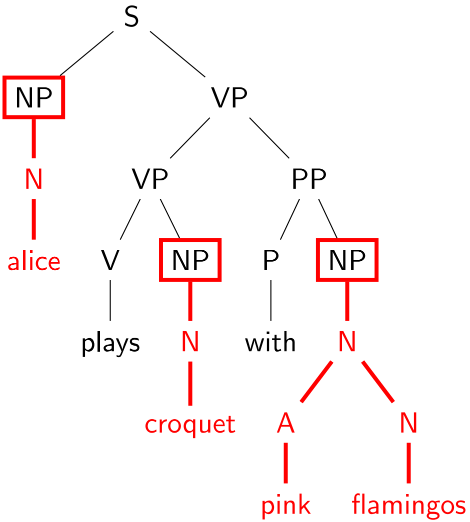
- prepositional phrases, *PP*, have a head preposition

Context-free grammars capture **phrase structure**



- the head of the whole string, S, is always the main verb

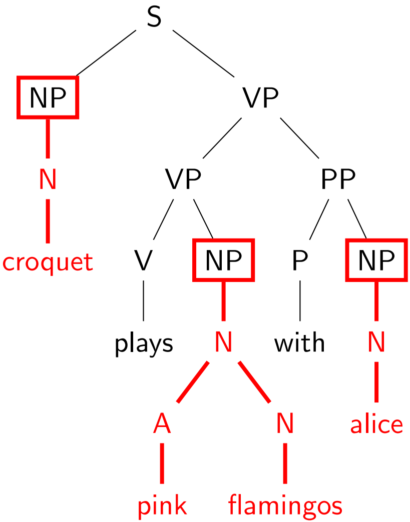
Context-free grammars capture **phrase structure**



Trees below nodes of the same type are interchangeable to yield another string in the language:

- $NP \rightarrow N$
- $N \rightarrow A\ N$
- $N \rightarrow alice|croquet|...$

Context-free grammars capture **phrase structure**



Trees below nodes of the same type are interchangeable to yield another string in the language:

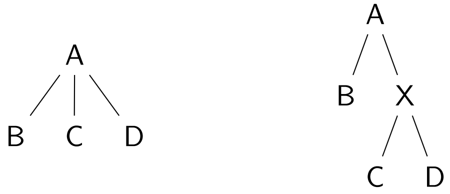
- $NP \rightarrow N$
- $N \rightarrow A\ N$
- $N \rightarrow alice | croquet | \ldots$

CFGs are often written in **Chomsky Normal Form**

Chomsky normal form: every production rule has the form, $A \rightarrow BC$, or, $A \rightarrow a$ where $A, B, C \in \mathcal{N}$, and, $a \in \Sigma$.

Conversion to Chomsky Normal Form

For every CFG there is a weakly equivalent CNF alternative.
$A \rightarrow BCD$ may be rewritten as the two rules, $A \rightarrow BX$, and, $X \rightarrow CD$.



CNF is a requirement for some parsing algorithms.

Context-free languages are accepted by **push down automata**

A PDA is defined as $M = (\mathcal{Q}, \Sigma, \Gamma, \Delta, s, \perp, \mathcal{F})$ where:

- $\mathcal{Q} = \{q_0, q_1, q_2 ...\}$ is a finite set of states.
- Σ is the input alphabet.
- Γ is the stack alphabet.
- $\Delta \subseteq (\mathcal{Q} \times (\Sigma \cup \epsilon) \times \Gamma) \times (\mathcal{Q} \times \Gamma^*)$ is a relation $(\mathcal{Q} \times (\Sigma \cup \epsilon) \times \Gamma) \rightarrow (\mathcal{Q} \times \Gamma^*)$ which we write as δ. Given $q \in \mathcal{Q}$, $i \in \Sigma$ and $A \in \Gamma$ then $\delta(q, i, A)$ returns (q', α), that is, a new state $q' \in \mathcal{Q}$ and replaces A at the top of the stack with $\alpha \in \Gamma^*$
- s is the starting state
- $\perp$ is the initial stack symbol
- $\mathcal{F}$ is the set of all end states

Moving from one state to the next we may **push** or **pop**

- in state q_x on encountering transition symbol a transition to state q_y popping A from the top of the stack and pushing B onto the stack

$$q_x \xrightarrow{a : A/B} q_y$$

BEFORE	AFTER
A	B
z_0	z_0

- in state q_x transition to state q_y pushing A onto the stack

$$q_x \xrightarrow{\epsilon : \epsilon/A} q_y$$

BEFORE	AFTER
z_0	A
	z_0

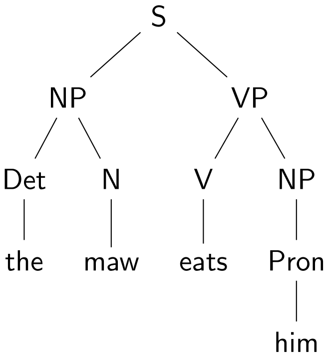
- in state q_x transition to state q_y popping A from the stack

$$q_x \xrightarrow{\epsilon : A/\epsilon} q_y$$

BEFORE	AFTER
A	z_0
z_0	

A toy context-free grammar

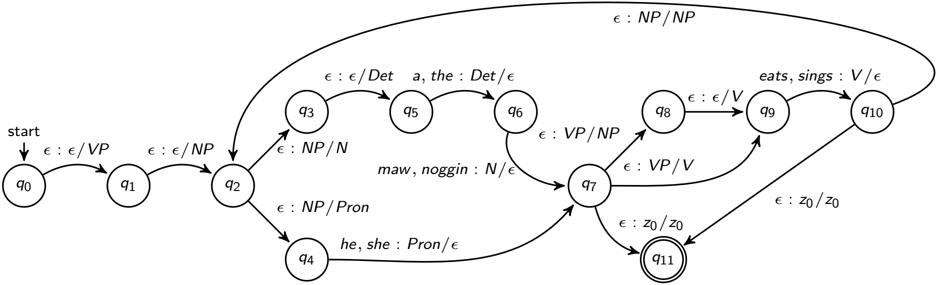
$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow Pron \\
NP &\rightarrow Det\ N \\
VP &\rightarrow V \\
VP &\rightarrow V\ NP \\
Det &\rightarrow \{a, the\} \\
N &\rightarrow \{maw, noggin, \ldots\} \\
Pron &\rightarrow \{he, she, him, her\} \\
V &\rightarrow \{eats, sings\}
\end{aligned}
$$



Recognising a string with a push down automaton

$$\begin{array}{rcl} S & \rightarrow & NP\ VP \\ NP & \rightarrow & Pron \\ NP & \rightarrow & Det\ N \\ VP & \rightarrow & V \\ VP & \rightarrow & V\ NP \end{array}$$

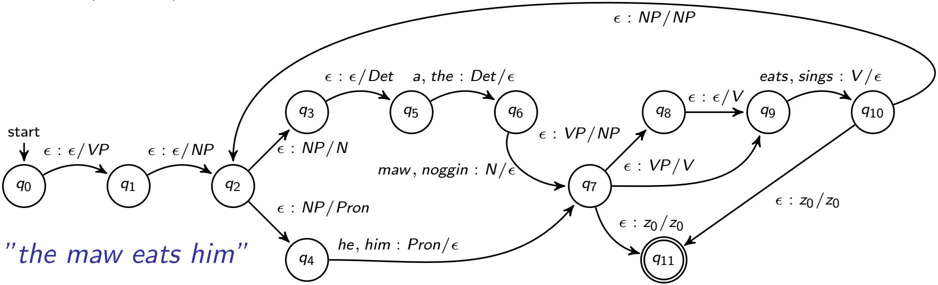
$$\begin{array}{rcl} Det & \rightarrow & \{a, the\} \\ N & \rightarrow & \{maw, noggin, ...\} \\ Pron & \rightarrow & \{he, him, her\} \\ V & \rightarrow & \{eats, sings\} \end{array}$$



Is *'the maw eats him'* a string in the language?

the	q_0	z_0			
the	q_0-q_1	VP	z_0		
the	q_1-q_2	NP	VP	z_0	
the	q_2-q_3	N	VP	z_0	
the	q_3-q_5	Det	N	VP	z_0
maw	q_5-q_6	N	VP	z_0	
eats	q_6-q_7	VP	z_0		
eats	q_7-q_8	NP	z_0		
eats	q_8-q_9	V	NP	z_0	
him	q_9-q_{10}	NP	z_0		
him	q_{10}-q_2	NP	z_0		
him	q_2-q_4	Pron	z_0		
him	q_4-q_7	z_0			
ϵ	q_7-q_{11}	z_0			

S	$\rightarrow$	NP VP
NP	$\rightarrow$	Pron
NP	$\rightarrow$	Det N
VP	$\rightarrow$	V
VP	$\rightarrow$	V NP
Det	$\rightarrow$	{a,the}
N	$\rightarrow$	{maw, noggin, ...}
Pron	$\rightarrow$	{he, him, her}
V	$\rightarrow$	{eats, sings}

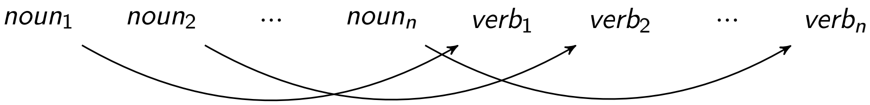


"the maw eats him"

Can context-free grammars model natural language?

Cross Serial Dependencies

A small number of languages exhibit strings of the form



Zurich dialect of Swiss German

mer d'chind em Hans es huus haend wele laa hälfe aastriiche.

we the children Hans the house have wanted to let help paint.

we have wanted to let the children help Hans paint the house

Such expressions, i.e. of the form $/a^n b^m c^n d^m/$, may not be derivable by a context-free grammar.

mer d'chind^n em Hans^m es huus haend wele laa^n hälfe^m aastriiche.

$\rightarrow$ $/wa^n b^m x c^n d^m y/$

Use the **pumping lemma** to prove **not** context-free

The pumping lemma for context-free languages (CFLs) is used to show that a language is not context-free. The pumping lemma property for CFLs is:

All $w \in \mathcal{L}$ with $|w| \geq k$ can be expressed as a concatenation of five strings, $w = u_1 y u_2 z u_3$, where u_1, y, u_2, z and u_3 satisfy:

- $|yz| \geq 1$ (i.e. we cannot have $y = \epsilon$ and $z = \epsilon$)
- $|yu_2z| \leq k$
- for all $n \geq 0$, $u_1 y^n u_2 z^n u_3 \in \mathcal{L}$
  (i.e. $u_1 u_2 u_3 \in \mathcal{L}$, $u_1 y u_2 z u_3 \in \mathcal{L}$, $u_1 yy u_2 zz u_3 \in \mathcal{L}$ etc.)

To prove that Swiss German is not context-free, similar proof as for **centre embeddings** (last lecture). Except that you need to remember that:
$\mathcal{L}_{reg1} \cap \mathcal{L}_{cfg1} = \mathcal{L}_{cfg2}$

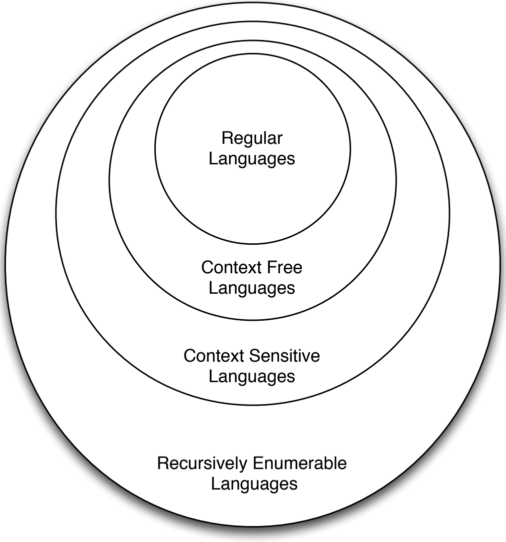
Are **CSGs** required to model natural languages?

Remember the **complexity** of a language class was defined in terms of the **recognition problem**.

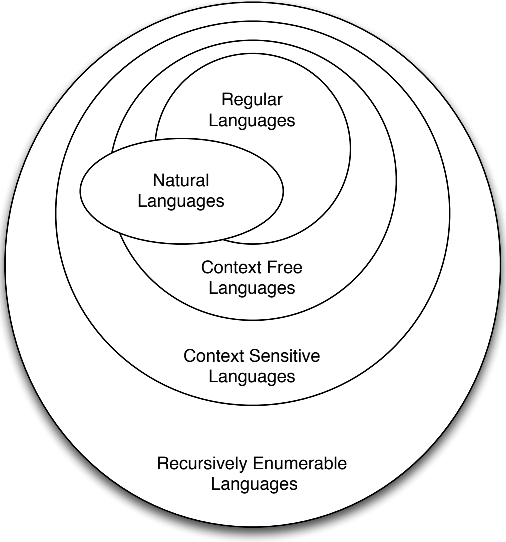
Type	Language Class	Complexity	machine
3	regular	$O(n)$	DFA
2	context-free	$O(n^c)$	PDA
1	context-sensitive	$O(c^n)$	LBA
0	recursively enumerable	*undecidable*	Turing

- Modelling natural languages using context-sensitive grammars is very expensive. In practice we don't have to because only very limited constructions are not captured by context-free grammars.
- However, it is still fun to place a limit on the complexity of natural languages — we are not limited to discussing language classes only in terms of the Chomsky hierarchy.

We are not limited to the **Chomsky hierarchy**



We are not limited to the **Chomsky hierarchy**



The mildly context-sensitive grammars

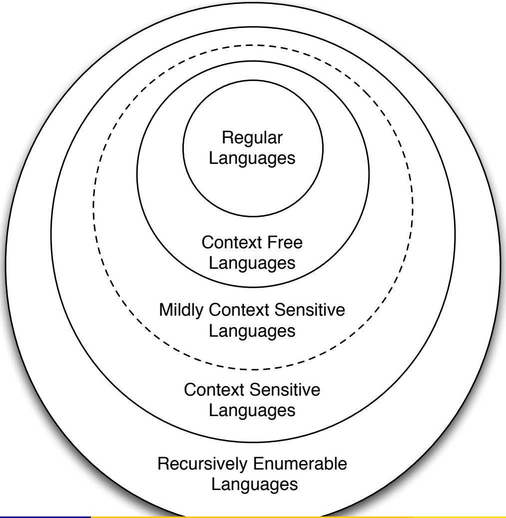
Joshi defined a class of languages that is more expressive than context-free languages, less expressive than context-sensitive languages and also sits neatly in the Chomsky hierarchy.

MILDLY CONTEXT-SENSITIVE languages

An abstract language class has the following properties:

- it includes all the context-free languages;
- members of the languages in the class may be recognised in polynomial time;
- the languages in the class account for all the constructions in natural language that context-free languages fail to account for (such as cross-serial dependencies).

Mildly CSGs are a **subset** of CSGs that account for natural language



In **Tree Adjoining Grammars** trees are rewritten as trees.

In phrase structure grammar symbols were rewritten with other symbols

In **Tree Adjoining Grammars** trees are rewritten as other trees.

The grammar consists of sets of two types of elementary tree:

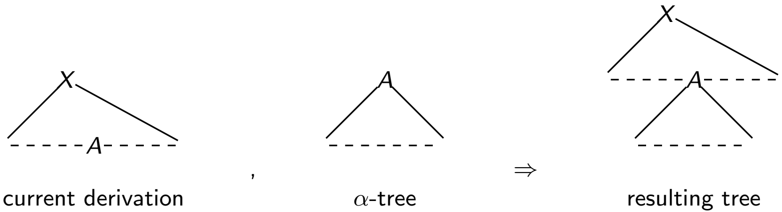
- **initial trees** or α trees
- **auxiliary trees** or β trees

A derivation is the result of recursive composition of elementary trees via one of two operations:

- **substitution**
- **adjunction**.

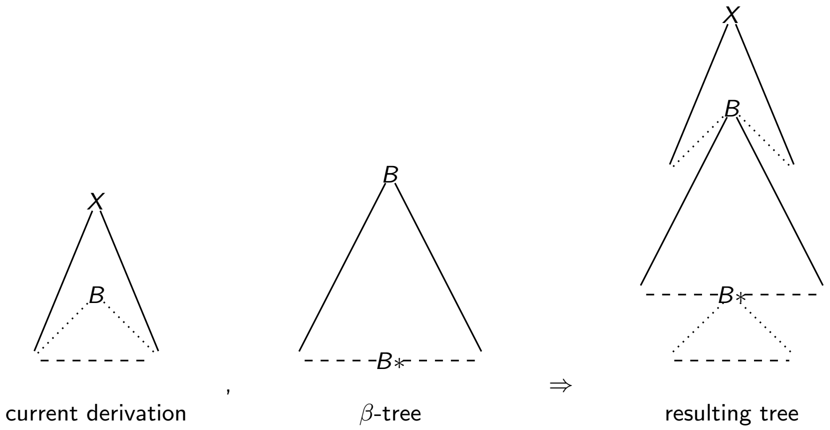
Tree adjoining grammars: the **substitution** operation

- SUBSTITUTION: a substitution may occur when a non-terminal leaf (that is, some $A \in \mathcal{N}$) of the current derivation tree is replaced by an α-tree that has A at its root.



Tree adjoining grammars: the **adjunction** operation

- ADJUNCTION:an adjunction may occur when an internal non-terminal node of the current derivation (some $B \in \mathcal{N}$) tree is replaced by a β tree that has a B at its root and *foot*.



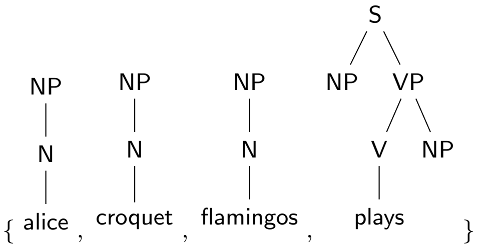
Tree adjoining grammars: definition

- $\mathcal{N}$ is the set of non-terminals
- Σ is the set of terminals
- S is a distinguished non-terminal $S \in \mathcal{N}$ that will be the root of complete derivations
- $\mathcal{I}$ is a set of **initial trees** (also known as α trees). Internal nodes of an α tree are drawn from $\mathcal{N}$ and the leaf nodes from $\Sigma \cup \mathcal{N} \cup \epsilon$.
- $\mathcal{A}$ is a set of **auxiliary trees** (also know as β trees). Internal nodes of an β-tree are drawn from $\mathcal{N}$ and the leaf nodes from $\Sigma \cup \mathcal{N} \cup \epsilon$. One leaf of a β-tree is distinguished as the **foot** and will be the same non-terminal as at its root (the foot is often indicated with an asterisk).

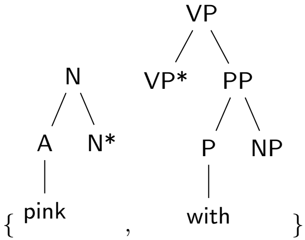
Tree adjoining grammars: natural language example

$G_{tag} = (\mathcal{N}, \Sigma, S, \mathcal{I}, \mathcal{A})$ where:

$\mathcal{I} =$

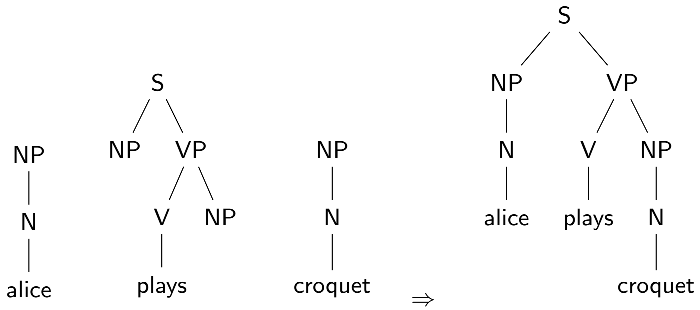


$\mathcal{A} =$



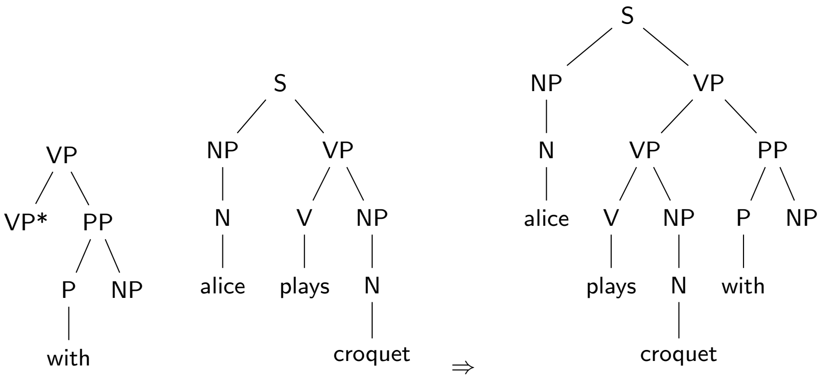
Tree adjoining grammars: natural language example

Deriving: *Alice plays croquet with pink flamingos*



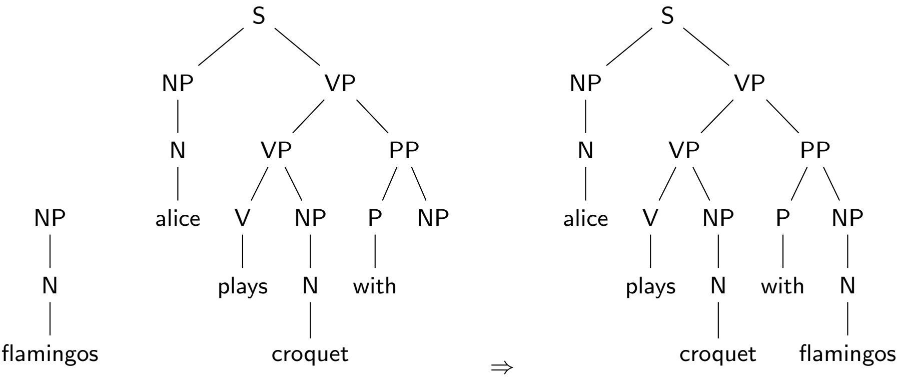
Tree adjoining grammars: natural language example

Deriving: *Alice plays croquet with pink flamingos*



Tree adjoining grammars: natural language example

Deriving: *Alice plays croquet with pink flamingos*



Tree adjoining grammars: natural language example

Deriving: *Alice plays croquet with pink flamingos*