Generative and Computational Power of Combinatory Categorial Grammar

Lena Katharina Schiffer

July 10th, 2024

Institute for Computer Science
Universität Leipzig

Combinatory Categorial Grammar

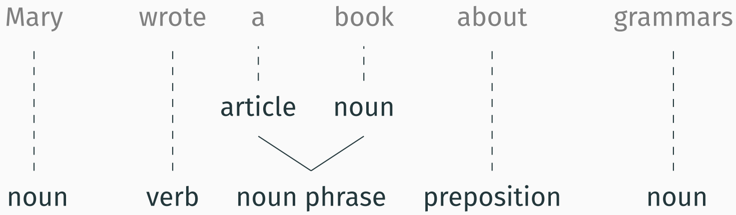
Constituents in Natural Language

Mary wrote a book about grammars

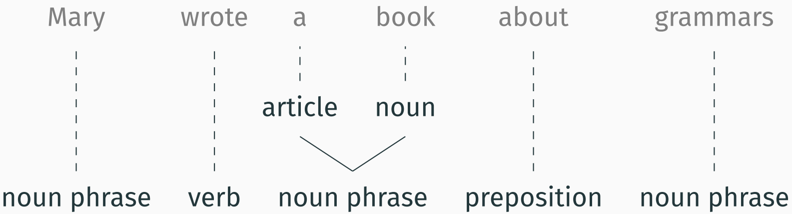
Constituents in Natural Language

Mary	wrote	a	book	about	grammars
noun	verb	article	noun	preposition	noun

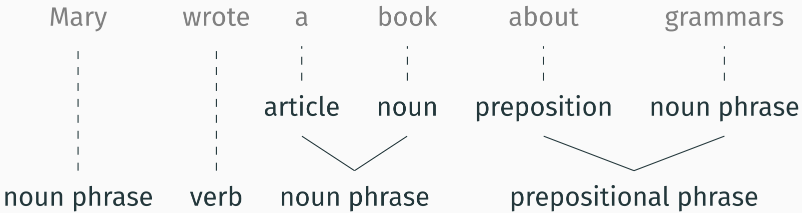
Constituents in Natural Language



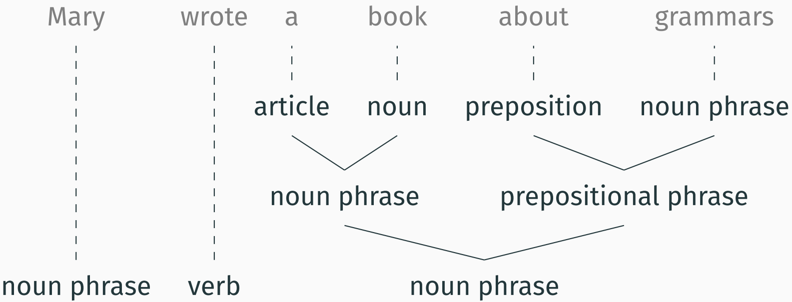
Constituents in Natural Language



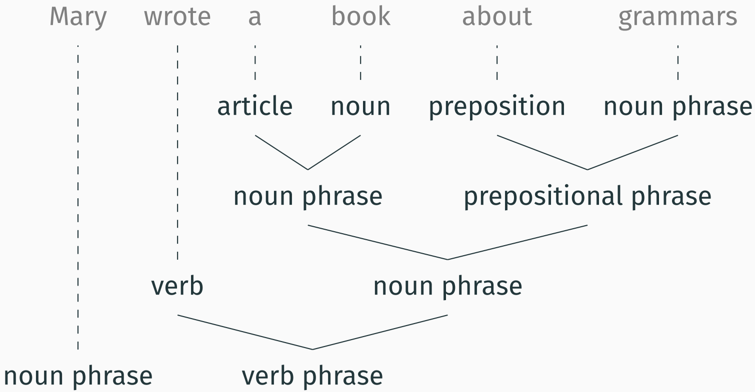
Constituents in Natural Language



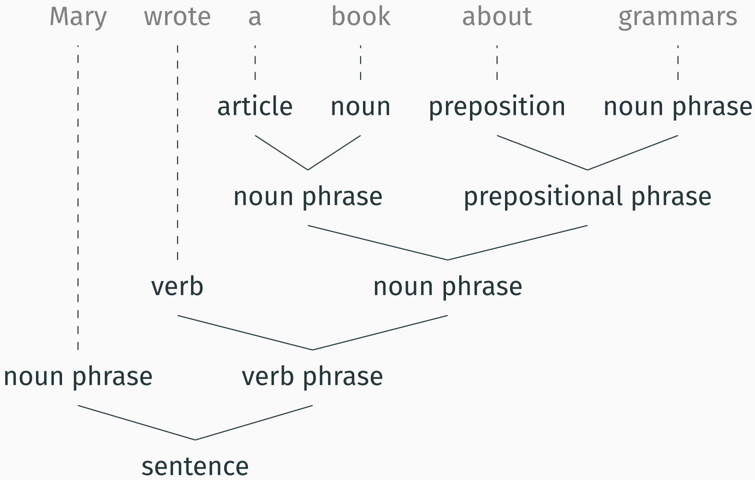
Constituents in Natural Language



Constituents in Natural Language



Constituents in Natural Language



Category Representation of Constituents

article noun

noun phrase

Category Representation of Constituents

article noun

noun phrase

NP/N N

NP

Category Representation of Constituents

article noun → noun phrase

NP/N N → NP

article category NP/N

"obtain a noun phrase if a noun is on the right side"

Category Representation of Constituents

article noun
noun phrase

NP/N N
NP

article category NP/N $f : \mathrm{N} \longrightarrow \mathrm{NP}$

"obtain a noun phrase if a noun is on the right side"

Category Representation of Constituents

article noun

noun phrase

NP/N N

NP

article category NP/N $\quad f : \mathrm{N} \rightarrow \mathrm{NP}$

"obtain a noun phrase if a noun is on the right side"

NP, N	atoms
NP	target
/N	argument

Category Representation of Constituents

noun phrase verb phrase

sentence

Category Representation of Constituents

noun phrase verb phrase

sentence

NP S\NP

S

Category Representation of Constituents

noun phrase   verb phrase

sentence

NP   S\NP

S

verb phrase category S\NP

“obtain a sentence if a noun phrase is on the left side”

Category Representation of Constituents

noun phrase   verb phrase

sentence

NP   S\NP

S

verb phrase category S\NP $f: \mathrm{NP} \to \mathrm{S}$

"obtain a sentence if a noun phrase is on the left side"

Category Representation of Constituents

noun phrase verb phrase

sentence

$\mathrm{NP} \quad \mathrm{S\backslash NP}$

S

verb phrase category $\mathrm{S\backslash NP}$ $f : \mathrm{NP} \rightarrow \mathrm{S}$

"obtain a sentence if a noun phrase is on the left side"

S, NP	atoms
S	target
\NP	argument

Category Representation of Constituents

transitive verb category (example: *likes*)

$S\backslash NP/NP$

Category Representation of Constituents

transitive verb category (example: *likes*)

$S\backslash NP/NP$

"obtain a sentence if there is

- a noun phrase on the right side and
- a noun phrase on the left side"

Category Representation of Constituents

transitive verb category (example: *likes*)

$\mathrm{S}\backslash\mathrm{NP}/\mathrm{NP}$ $\qquad f : \mathrm{NP} \times \mathrm{NP} \longrightarrow \mathrm{S}$

"obtain a sentence if there is

- a noun phrase on the right side and
- a noun phrase on the left side"

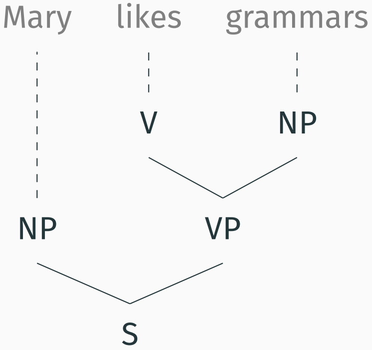
transitive verb category (example: *likes*)

$S\backslash NP/NP$ $\quad f: NP \times NP \longrightarrow S$ or $f: NP \longrightarrow (NP \longrightarrow S)$

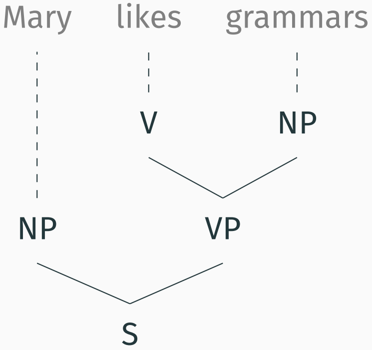
"obtain a sentence if there is

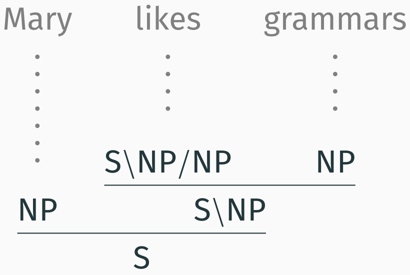
- a noun phrase on the right side and
- a noun phrase on the left side"

Constituency Tree VS CCG Derivation Tree

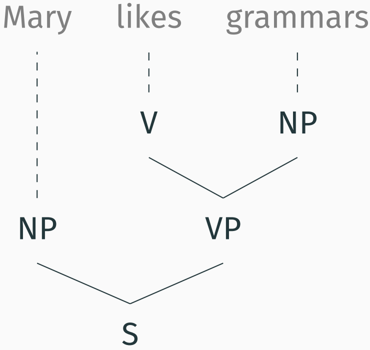


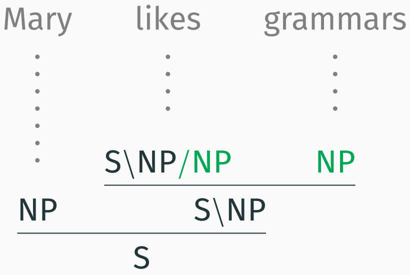
Constituency Tree VS CCG Derivation Tree



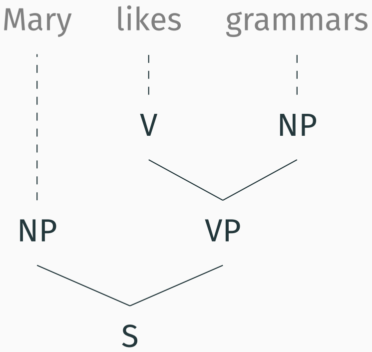


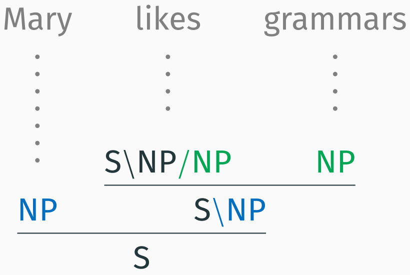
Constituency Tree VS CCG Derivation Tree



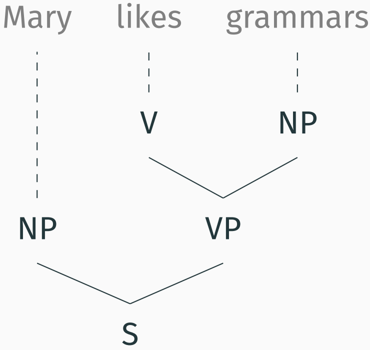


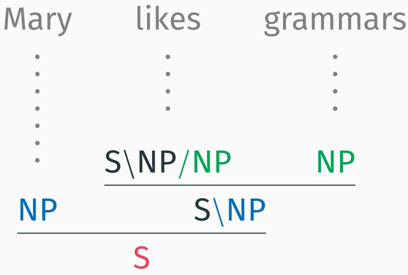
Constituency Tree VS CCG Derivation Tree





Constituency Tree VS CCG Derivation Tree





S initial category

forward application

$$\frac{c/b \quad b}{c}$$

backward application

$$\frac{b \quad c\backslash b}{c}$$

c/b, $c\backslash b$	primary category
b	secondary category
c	output category

Application Rules

forward application

$$\frac{c/b \quad b}{c}$$

backward application

$$\frac{b \quad c\backslash b}{c}$$

c/b, $c\backslash b$ primary category

b secondary category

c output category

$$\frac{\mathrm{S\backslash NP/NP} \quad \mathrm{NP}}{\mathrm{S\backslash NP}}$$

forward rule **backward rule**

$$\frac{c/b \quad b\beta}{c\beta} \qquad \frac{b\beta \quad c\backslash b}{c\beta}$$

β argument sequence

$|\beta|$ rule degree

Composition Rules

degree o

$$\frac{c/b \quad b}{c}$$

Composition Rules

degree 0

$$\frac{c/b \quad b}{c}$$

degree 1

$$\frac{c/b \quad b\backslash e}{c\backslash e}$$

Composition Rules

degree 0 **degree 1**

$$\frac{c/b \quad b}{c} \qquad \frac{c/b \quad b\backslash e}{c\backslash e}$$

$$f : e \longrightarrow b \qquad g : b \longrightarrow c$$

$$f \circ g : e \longrightarrow c$$

Composition Rules

degree 0

$$\frac{c/b \quad b}{c}$$

degree 1

$$\frac{c/b \quad b\backslash e}{c\backslash e}$$

degree 2

$$\frac{c\backslash b \quad b\backslash e/d}{c\backslash e/d}$$

$$f : e \longrightarrow b \qquad g : b \longrightarrow c$$

$$f \circ g : e \longrightarrow c$$

Composition Rules

degree 0	**degree 1**	**degree 2**
$\dfrac{c/b \quad b}{c}$	$\dfrac{c/b \quad b\backslash e}{c\backslash e}$	$\dfrac{c\backslash b \quad b\backslash e/d}{c\backslash e/d}$

$$f : e \longrightarrow b \qquad g : b \longrightarrow c$$

$$f \circ g : e \longrightarrow c$$

In a rule we may restrict

- the secondary category to a concrete category
- the target of the primary category to a concrete atom

$$\frac{c/b \quad b}{c} \qquad \longrightarrow \qquad \frac{S\,x/NP \quad NP}{S\,x}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha, \ \beta\}$
- atomic categories $A = \{D, \ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E, \ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c}, \frac{b \quad c\backslash b}{c} \right\}$$

$\alpha \quad \alpha \quad \beta \quad \beta$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc} \alpha & \alpha & \beta & \beta \\ & & \vdots & \vdots \\ & & E & E \end{array}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc} \alpha & \alpha & \beta & \beta \\ & \vdots & \vdots & \vdots \\ & D/E & E & E \end{array}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc} \alpha & \alpha & \beta & \beta \\ \vdots & \vdots & \vdots & \vdots \\ D/E/D & D/E & E & E \end{array}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc} \alpha & \alpha & \beta & \beta \\ \vdots & \vdots & \vdots & \vdots \\ D/E/D & D/E & E & E \end{array}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc} \alpha & \alpha & \beta & \beta \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & D/E & E & \vdots \\ \hline D/E/D & \multicolumn{2}{c}{D} & E \end{array}$$

EXAMPLE CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\begin{array}{cccc}
\alpha & \alpha & \beta & \beta \\
\vdots & \vdots & \vdots & \vdots \\
\vdots & D/E & E & \vdots \\
\cline{2-3}
D/E/D & \multicolumn{2}{c}{D} & E
\end{array}$$

EXAMPLE CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

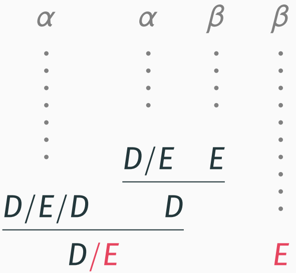
$$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$

$$\cfrac{\begin{matrix}\alpha\\ \vdots\\ D/E/D\end{matrix} \quad \cfrac{\begin{matrix}\alpha\\ \vdots\\ D/E\end{matrix} \quad \begin{matrix}\beta\\ \vdots\\ E\end{matrix}}{D}}{D/E} \qquad \begin{matrix}\beta\\ \vdots\\ E\end{matrix}$$

Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  all application rules

  $$R = \left\{ \frac{c/b \quad b}{c},\ \frac{b \quad c\backslash b}{c} \right\}$$



Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with

  $L(\alpha) = \{D/E,\ D/E/D\}$

  $L(\beta) = \{E\}$

- rule set includes all application rules

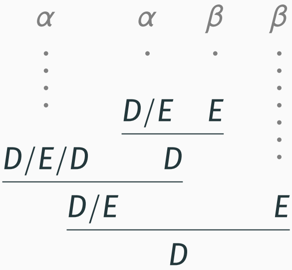
  $$R = \left\{ \frac{c/b \quad b}{c}, \frac{b \quad c\backslash b}{c} \right\}$$

$$\dfrac{\dfrac{\overset{\alpha}{D/E/D} \qquad \dfrac{\overset{\alpha}{D/E} \quad \overset{\beta}{E}}{D}}{D/E} \qquad \overset{\beta}{E}}{D}$$

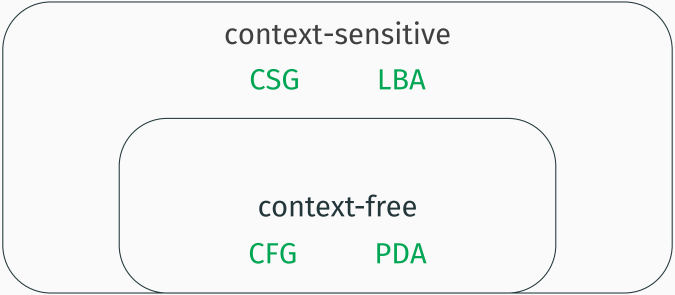
Example CCG $\mathcal{G} = (\Sigma, A, R, I, L)$

- input alphabet $\Sigma = \{\alpha,\ \beta\}$
- atomic categories $A = \{D,\ E\}$
- initial categories $I = \{D\}$
- lexicon L with
  $L(\alpha) = \{D/E,\ D/E/D\}$
  $L(\beta) = \{E\}$
- rule set includes
  restricted rules

$$R = \left\{ \frac{Dx/D \quad D}{Dx},\ \frac{Dx/E \quad E}{Dx} \right\}$$



Classical Language Classes



CSG	Context-Sensitive Grammar
LBA	Linear Bounded Automaton
CFG	Context-Free Grammar
PDA	Push-Down Automaton

Cross-Serial Dependencies

...	omdat	ik	Cecilia	Henk	de	nijlpaarden	zag	helpen	voeren.
...	because	I	Cecilia	Henk	the	hippopotamuses	saw	help	feed

'... because I saw Cecilia help Henk feed the hippopotamuses.'

Example from Steedman (1985)

Cross-Serial Dependencies

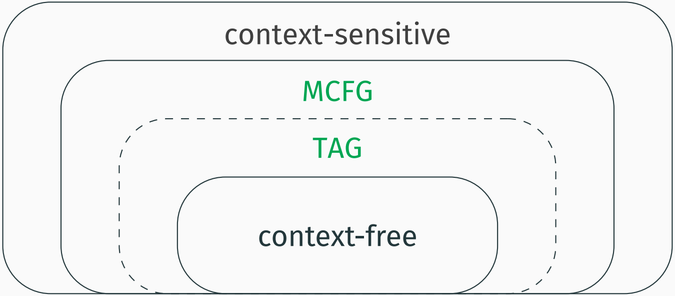
...	omdat	ik	Cecilia	Henk	de	nijlpaarden	zag	helpen	voeren.
...	because	I	Cecilia	Henk	the	hippopotamuses	saw	help	feed

'... because I saw Cecilia help Henk feed the hippopotamuses.'

$$\mathrm{COPY} = \{\, ww \mid w \in \Sigma^* \,\}$$

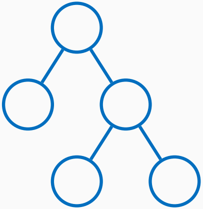
Example from Steedman (1985)

Mild Context-Sensitivity

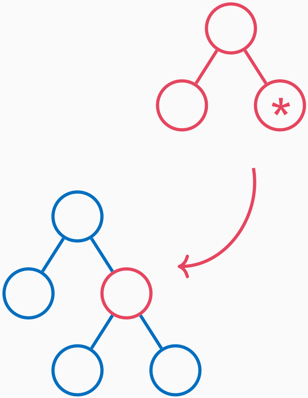


MCFG	Multiple Context-Free Grammar
TAG	Tree-Adjoining Grammar

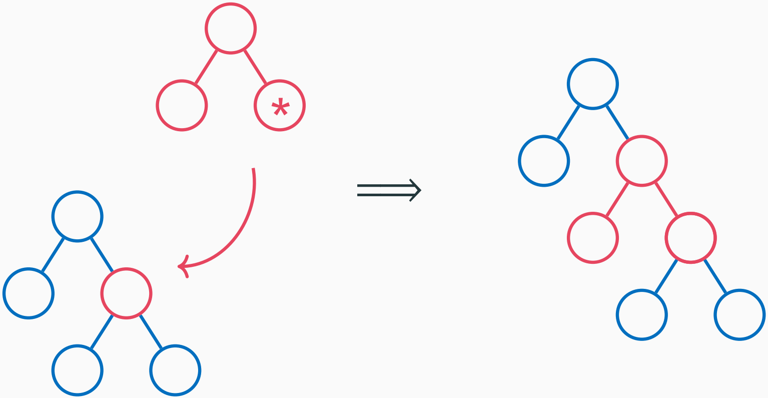
Tree-Adjoining Grammar



Tree-Adjoining Grammar



Tree-Adjoining Grammar



Mild Context-Sensitivity of CCG

Math. Systems Theory 27, 511–546 (1994)

Mathematical Systems Theory

The Equivalence of Four Extensions of Context-Free Grammars*

K. Vijay-Shanker[1] and D. J. Weir[2]

[1] Department of Computer and Information Science, University of Delaware, Newark, DE 19716, USA

[2] School of Cognitive and Computing Sciences, University of Sussex, Brighton, Sussex BN1 9QH, England

Abstract. There is currently considerable interest among computational linguists in grammatical formalisms with highly restricted generative power. This paper concerns the relationship between the class of string languages generated by several such formalisms, namely, combinatory categorial grammars, head grammars, linear indexed grammars, and tree adjoining grammars. Each of these formalisms is known to generate a larger class of languages than context-free grammars. The four formalisms under consideration were developed independently and appear superficially to be quite different from one another. The result presented in this paper is that all four of the formalisms under consideration generate exactly the same class of string languages.

1. Introduction

There is currently considerable interest among computational linguists in grammatical formalisms with highly restricted generative power. This is based on the argument that a grammar formalism should not merely be viewed as a notation, but as part of the linguistic theory [6]. It should make predictions about the structure of natural language and its value is lessened to the extent that it supports both good and bad analyses. In order for a grammar formalism to have such predictive power its generative capacity must be constrained. This has led to

* This work has been supported by NSF Grants MCS-82-19116-CER, MCS-82-07294, DCR-84-10413, IRI-8909810, ARO Grant DAA29-84-9-0027, and DARPA Grant N0014-85-K0018.

TAG = CCG = LIG = HG

Vijay-Shanker, Weir (1994)

LIG	Linear Indexed Grammar
HG	Head Grammar

Mild Context-Sensitivity of CCG

Parsing Some Constrained Grammar Formalisms

K. Vijay-Shanker*
University of Delaware

David J. Weir†
University of Sussex

In this paper we present a scheme to extend a recognition algorithm for Context-Free Grammars (CFG) that can be used to derive polynomial-time recognition algorithms for a set of formalisms that generate a superset of languages generated by CFG. We describe the scheme by developing a Cocke-Kasami-Younger (CKY)-like pure bottom-up recognition algorithm for Linear Indexed Grammars and show how it can be adapted to give algorithms for Tree Adjoining Grammars and Combinatory Categorial Grammars. This is the only polynomial-time recognition algorithm for Combinatory Categorial Grammars that we are aware of.

The main contribution of this paper is the general scheme we propose for parsing a variety of formalisms whose derivation process is controlled by an explicit or implicit stack. The ideas presented here can be suitably modified for other parsing styles or used in the generalized framework set out by Lang (1990).

1. Introduction

This paper presents a scheme to extend known recognition algorithms for Context-Free Grammars (CFG) in order to obtain recognition algorithms for a class of grammatical formalisms that generate a strict superset of the set of languages generated by CFG. In particular, we use this scheme to give recognition algorithms for Linear Indexed Grammars (LIG), Tree Adjoining Grammars (TAG), and a version of Combinatory Categorial Grammars (CCG). These formalisms belong to the class of *mildly context-sensitive grammar formalisms* identified by Joshi (1985) on the basis of some properties of their generative capacity. The parsing strategy that we propose can be applied to the formalisms listed as well as others that have similar characteristics (as outlined below) in their derivational process. Some of the main ideas underlying our scheme have been influenced by the observations that can be made about the constructions used in the proofs of the equivalence of these formalisms and Head Grammars (HG) (Vijay-Shanker 1987; Weir 1988; Vijay-Shanker and Weir 1993).

There are similarities between the TAG and HG derivation processes and that of Context-Free Grammars (CFG). This is reflected in common features of the parsing algorithms for HG (Pollard 1984) and TAG (Vijay-Shanker and Joshi 1985) and the CKY algorithm for CFG (Kasami 1965; Younger 1967). In particular, what can happen at each step in a derivation can depend only on which of a finite set of "states" the derivation is in (for CFG these states can be considered to be the nonterminal symbols). This property, which we refer to as the **context-freeness property**, is important because it allows one to keep only a limited amount of context during the recognition process,

* Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716. E-mail: vijay@udel.edu.
† School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, U.K. E-mail: davidw@cogs.susx.ac.uk.

CCG parsable in $\mathcal{O}(|w|^6)$

Vijay-Shanker, Weir (1994)

Computational Complexity

Membership Problem

- input: w
- question: $w \in \mathcal{L}(\mathcal{G})$?

can be solved in $\mathcal{O}(|w|^6)$ Vijay-Shanker, Weir (1994)

Parsing Decision Problems

Membership Problem

- input: w
- question: $w \in \mathcal{L}(\mathcal{G})$?

can be solved in $\mathcal{O}(|w|^6)$ Vijay-Shanker, Weir (1994)

Universal Recognition Problem

- input: w, $\mathcal{G}$
- question: $w \in \mathcal{L}(\mathcal{G})$?

EXPTIME-/NP-complete Kuhlmann, Satta, Jonsson (2018)

Lexicon Entries for the Empty Word

ε-entries

lexicon entries
for the
empty word ε

$$\begin{array}{c} \varepsilon \\ \vdots \\ B/D/C \end{array}$$

LEXICON ENTRIES FOR THE EMPTY WORD

ε-entries

lexicon entries for the empty word ε

$$\begin{array}{c} \varepsilon \\ \vdots \\ B/D/C \end{array}$$

classical proof for equivalence of TAG and CCG heavily relies on ε-entries Vijay-Shanker, Weir (1994)

Lexicon Entries for the Empty Word

ε-entries

lexicon entries for the empty word ε

$$\dfrac{\varepsilon}{B/D/C}$$

classical proof for equivalence of TAG and CCG heavily relies on ε-entries Vijay-Shanker, Weir (1994)

universal recognition problem

- with ε-entries EXPTIME-complete
- without them NP-complete

Kuhlmann, Satta, Jonsson (2018)

Research Question

CCG variant	complexity
with ε-entries	EXPTIME[†]
without ε-entries	NP[†]
?	PTIME

[†] Kuhlmann, Satta, Jonsson (2018)

Research Question

CCG variant	complexity
with ε-entries	EXPTIME†
without ε-entries	NP†
?	PTIME

- Can we restrict CCG such that parsing becomes polynomial in the grammar size?

†Kuhlmann, Satta, Jonsson (2018)

Research Question

CCG variant	complexity
with ε-entries	EXPTIME†
without ε-entries	NP†
?	PTIME

- Can we restrict CCG such that parsing becomes polynomial in the grammar size?
- Can we find a practically relevant formalism with this property?

†Kuhlmann, Satta, Jonsson (2018)

Approach

- inclusion of substitution rules

$$\frac{\boxed{c/b/e} \quad b/e}{c/e}$$

Approach

- inclusion of substitution rules

$$\frac{c/b/e \qquad b/e}{c/e} \qquad\qquad \frac{c/b/e \qquad b/e\backslash d}{c/e\backslash d}$$

Approach

- inclusion of substitution rules

$$\frac{c/b/e \quad b/e}{c/e} \qquad \frac{c/b/e \quad b/e\backslash d}{c/e\backslash d}$$

$\longrightarrow$ generalized rule notation

$$\frac{c/b\alpha \quad b\alpha\beta}{c\alpha\beta} \qquad \frac{b\alpha\beta \quad c\backslash b\alpha}{c\alpha\beta} \qquad \text{with } |\alpha| \leq 1$$

Approach

- inclusion of substitution rules

$$\frac{c/b/e \quad b/e}{c/e} \qquad \frac{c/b/e \quad b/e\backslash d}{c/e\backslash d}$$

$\longrightarrow$ generalized rule notation

$$\frac{c/b\alpha \quad b\alpha\beta}{c\alpha\beta} \qquad \frac{b\alpha\beta \quad c\backslash b\alpha}{c\alpha\beta} \qquad \text{with } |\alpha| \leq 1$$

- new parsing algorithm based on Kuhlmann, Satta (2014)

Approach

- inclusion of substitution rules

$$\frac{c/b/e \quad b/e}{c/e} \qquad \frac{c/b/e \quad b/e\backslash d}{c/e\backslash d}$$

$\longrightarrow$ generalized rule notation

$$\frac{c/b\alpha \quad b\alpha\beta}{c\alpha\beta} \qquad \frac{b\alpha\beta \quad c\backslash b\alpha}{c\alpha\beta} \qquad \text{with } |\alpha| \leq 1$$

- new parsing algorithm based on Kuhlmann, Satta (2014)
- complexity in terms of grammar size:
  new runtime exponential only in maximum rule degree k

Parsing is viewed as a deductive process:

- start from a set of axioms and derive new items

Shieber, Schabes, Pereira (1995)

Parsing as Deduction

Parsing is viewed as a deductive process:

- start from a set of axioms and derive new items
- use inference rules of the form

$$\frac{A_1 \quad \ldots \quad A_k}{B} \ \langle \text{side conditions} \rangle$$

Shieber, Schabes, Pereira (1995)

Parsing as Deduction

Parsing is viewed as a deductive process:

- start from a set of axioms and derive new items
- use inference rules of the form

$$\frac{A_1 \quad \ldots \quad A_k}{B} \ \langle\text{side conditions}\rangle$$

- input is accepted if goal item is derived

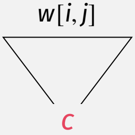
Shieber, Schabes, Pereira (1995)

Item Types

Tree Items

$$[c,\ i,\ j]$$

represents a derivation tree
with root category c



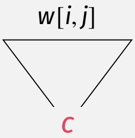
Kuhlmann, Satta (2014)

Item Types

Tree Items

$$[c,\ i,\ j]$$

represents a derivation tree with root category c

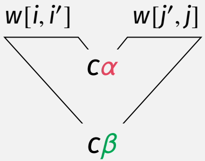


Context Items

$$[\alpha,\ \beta,\ i,\ i',\ j',\ j]$$

$1 \leq |\alpha| \leq 2$

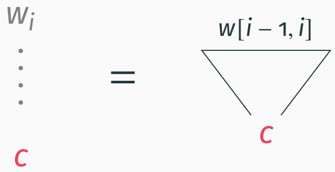
$|\beta| \leq$ maximum rule degree k



Kuhlmann, Satta (2014)

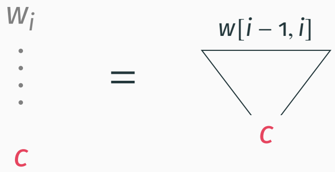
Parsing Algorithm – Axioms and Goal

Axioms: Lexicon Entry ⟶ Tree

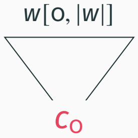


Parsing Algorithm – Axioms and Goal

Axioms: Lexicon Entry $\longrightarrow$ Tree



Goal: Tree over complete input with c_0 initial



Parsing Algorithm – Rule 1

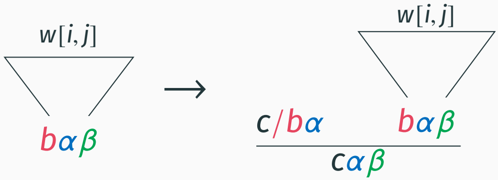
① Tree $\longrightarrow$ Context

$w[i,j]$

$b\alpha\beta$

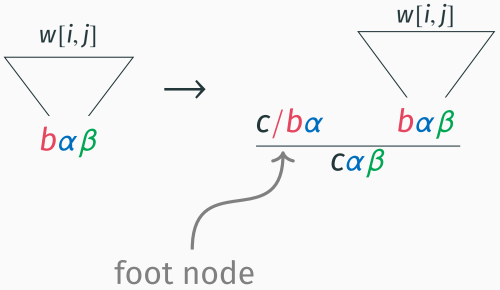
Parsing Algorithm – Rule 1

1. Tree $\longrightarrow$ Context



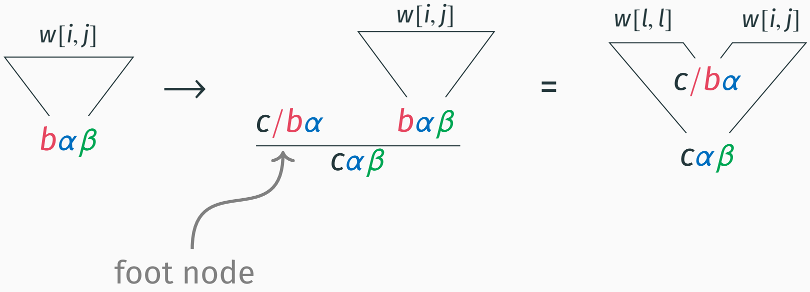
Parsing Algorithm – Rule 1

1. Tree $\longrightarrow$ Context



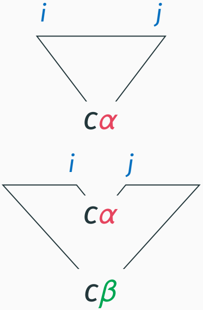
Parsing Algorithm – Rule 1

1. Tree $\longrightarrow$ Context



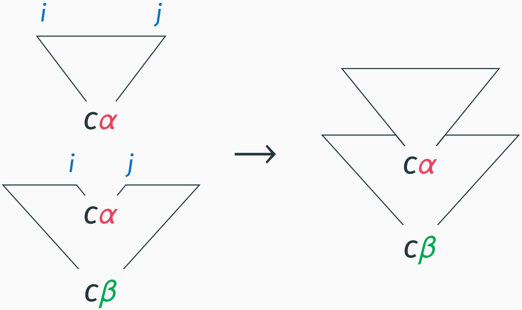
Parsing Algorithm – Rule 2

② Tree + Context ⟶ Tree



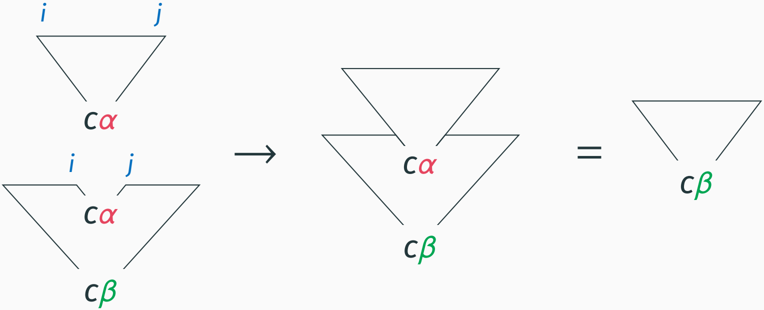
Parsing Algorithm – Rule 2

② Tree + Context $\longrightarrow$ Tree



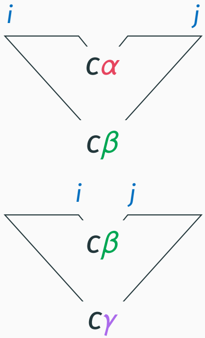
Parsing Algorithm – Rule 2

② Tree + Context ⟶ Tree



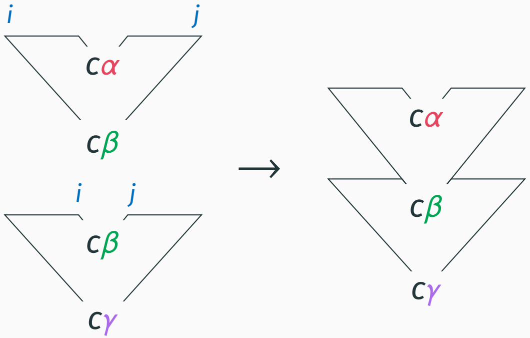
Parsing Algorithm – Rule 3

③ Context + Context ⟶ Context



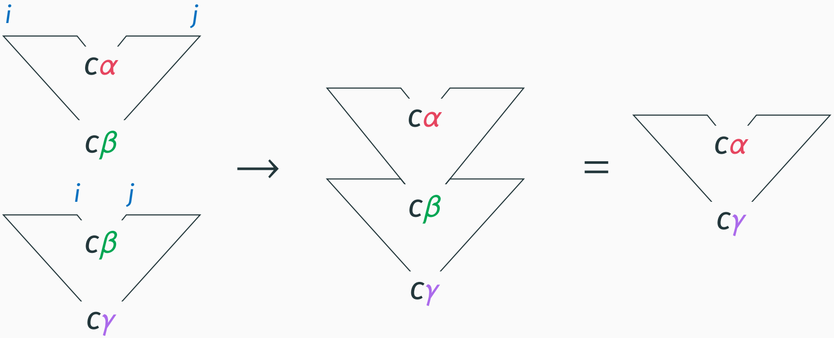
Parsing Algorithm – Rule 3

③ Context + Context ⟶ Context



Parsing Algorithm – Rule 3

③ Context + Context ⟶ Context



What is New?

- generalization to substitution rules

What is New?

- generalization to substitution rules
- improve complexity by restricting the tree items

What is New?

- generalization to substitution rules
- improve complexity by restricting the tree items

secondary category

$c\alpha$

OR

lexical category

$c\alpha$

tree item category

$c\beta$

What is New?

- generalization to substitution rules
- improve complexity by restricting the tree items

secondary category

$c\alpha$

OR

lexical category

$c\alpha$

tree item category

$c\beta$

with $|\beta| \leq 2$

and $|\beta| \leq |\alpha|$

What is New?

- generalization to substitution rules
- improve complexity by restricting the tree items

secondary category

$c\alpha$

OR

lexical category

$c\alpha$

tree item category

$c\beta$

with $|\beta| \leq 2$

and $|\beta| \leq |\alpha|$

$\longrightarrow$ number of items (and deduction rules!) exponential in k

Theorem

The universal recognition problem for k-CCG with ε-entries and substitution rules can be solved in time and space $\mathcal{O}(|\mathcal{G}|^{k+5} \cdot |w|^6)$.

Complexity Result

Theorem

The universal recognition problem for k-CCG with ε-entries and substitution rules can be solved in time and space $\mathcal{O}(|\mathcal{G}|^{k+5} \cdot |w|^6)$.

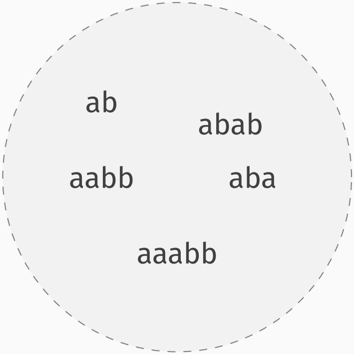
CCG variant	complexity
with ε-entries	EXPTIME[†]
without ε-entries	NP[†]
bounded degree	PTIME

[†] Kuhlmann, Satta, Jonsson (2018)

Generative Capacity

Strong Generative Capacity

weak
generative capacity

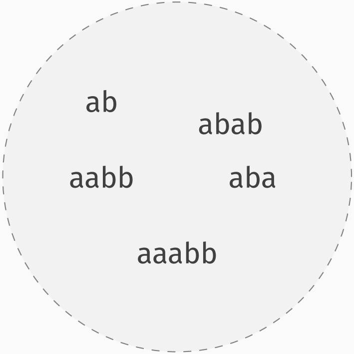


string languages
= sets of strings

Strong Generative Capacity

weak

generative capacity

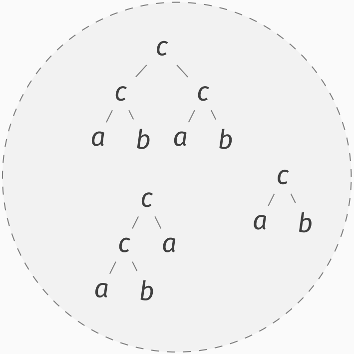


string languages

= sets of strings

strong

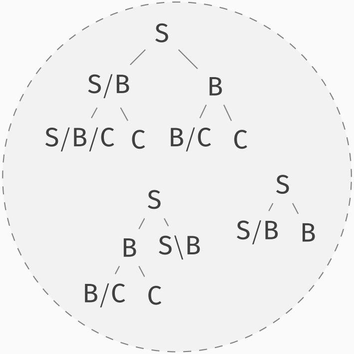
generative capacity



tree languages

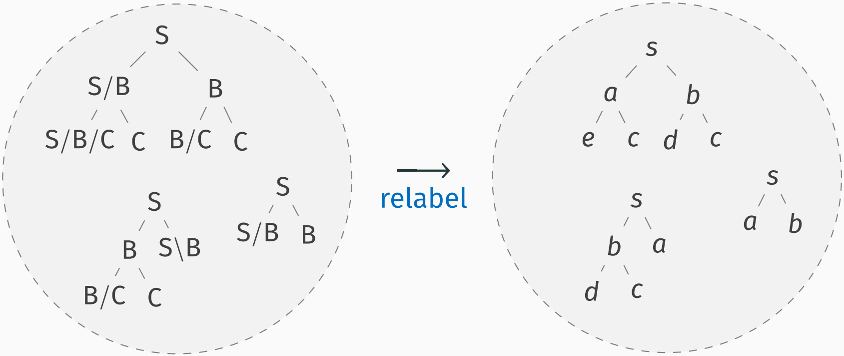
= sets of trees

Tree Language of CCG



CCG derivation tree set
(root category = initial)

Tree Language of CCG



CCG derivation tree set
(root category = initial)

generated
tree language

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$=$ CFG[a]	$=$ CFG[b]			$=$ TAG[c]
trees	$\subsetneq$ RTG[d]				

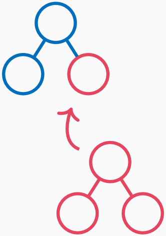
RTG Regular Tree Grammar

[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$=$ CFG[a]	$=$ CFG[b]			$=$ TAG[c]
trees	$\subsetneq$ RTG[d]				

RTG   Regular Tree Grammar

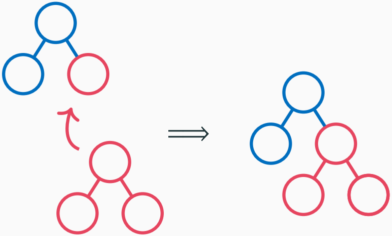


[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$=$ CFG[a]	$=$ CFG[b]			$=$ TAG[c]
trees	$\subsetneq$ RTG[d]				

RTG Regular Tree Grammar



[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Research Questions

What is the generative capacity of CCG without ε-entries?

Research Questions

What is the generative capacity of CCG without ε-entries?

What class of tree languages does CCG generate?

Research Questions

What is the generative capacity of CCG without ε-entries?

What class of tree languages does CCG generate?

How does the rule degree affect the generative capacity?

Strong Equivalence of TAG and CCG

$$\text{TAG} \overset{\dagger}{=} \text{sCFTG}$$

simple monadic

context-free tree grammar

$$\cup\!\!\!\nmid$$

RTG

[†]Kepser, Rogers (2011)

Strong Equivalence of TAG and CCG

$$\begin{array}{c} \mathrm{TAG} \overset{\dagger}{=} \mathrm{sCFTG} \\ \rotatebox{90}{$\subseteq$} \\ k\text{-}\mathrm{CCG} \\ \rotatebox{90}{$\subsetneq$} \\ \mathrm{RTG} \end{array}$$

simple monadic

context-free tree grammar

† Kepser, Rogers (2011)

Strong Equivalence of TAG and CCG

$$\text{2-CCG}$$

$$\cup\!\!|$$

$$\text{TAG} \overset{\dagger}{=} \text{sCFTG}$$

$$\cup\!\!|$$

$$\text{k-CCG}$$

$$\subsetneq$$

$$\text{RTG}$$

simple monadic

context-free tree grammar

† Kepser, Rogers (2011)

Strong Equivalence of TAG and CCG

$$\text{2-CCG}$$

$$\rotatebox{90}{$\subseteq$}$$

$$\text{TAG} \overset{\dagger}{=} \text{sCFTG} = \text{CCG}$$

$$\rotatebox{90}{$\subseteq$}$$

simple monadic

context-free tree grammar

$$\text{k-CCG}$$

$$\rotatebox{90}{$\subsetneq$}$$

$$\text{RTG}$$

† Kepser, Rogers (2011)

Strong Equivalence of TAG and CCG

$$\color{orange}{\text{2-CCG}}$$

$$\color{orange}{\cup\!\mid}$$

$$\text{TAG} \overset{\dagger}{=} \color{orange}{\text{sCFTG}} = \text{CCG}$$

$$\cup\!\mid$$

$$\text{k-CCG}$$

$$\not\supseteq$$

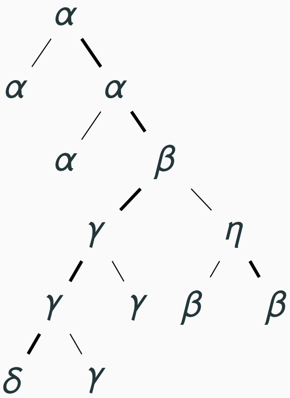
$$\text{RTG}$$

simple monadic

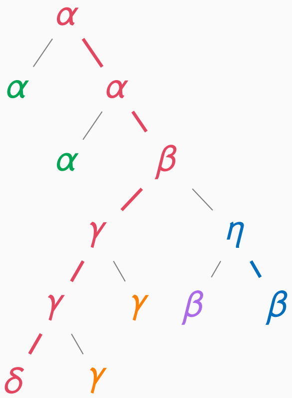
context-free tree grammar

[†] Kepser, Rogers (2011)

Decomposition into Spines



Decomposition into Spines



Decomposition into Spines

$\delta - \gamma - \gamma - \beta - \alpha - \alpha$

$\beta - \eta$

α

γ

β

$\vdots$

Decomposition into Spines

$$\left.\begin{array}{l} \delta - \gamma - \gamma - \beta - \alpha - \alpha \\ \beta - \eta \\ \alpha \\ \gamma \\ \beta \\ \vdots \end{array}\right\} \quad \text{context-free}$$

Decomposition into Spines

$\delta - \gamma - \gamma - \beta - \alpha - \alpha$

$\beta - \eta$

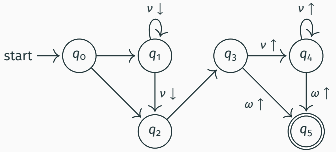
α

γ

β

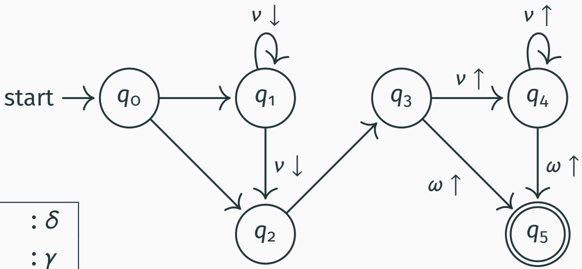
$\vdots$

context-free



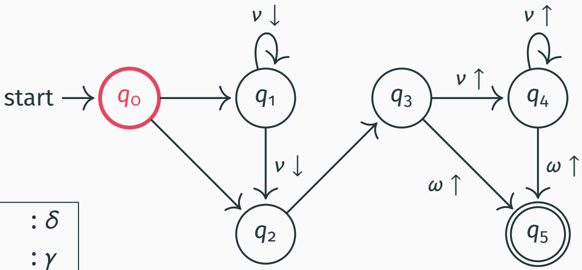
Moore PDA

Moore PDA



q_0	$:\delta$
$q_1,\ q_2$	$:\gamma$
q_3	$:\beta$
$q_4,\ q_5$	$:\alpha$

Moore PDA

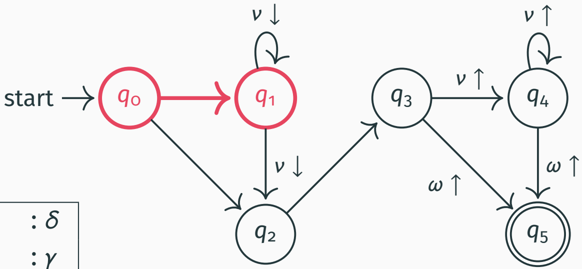


q_0	$: \delta$
q_1, q_2	$: \gamma$
q_3	$: \beta$
q_4, q_5	$: \alpha$

$\langle q_0, \boxed{\omega} \rangle$

δ

Moore PDA

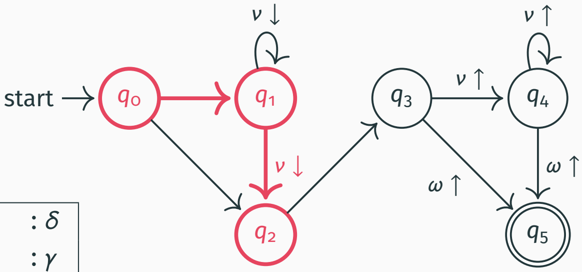


q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle$$

δ γ

Moore PDA

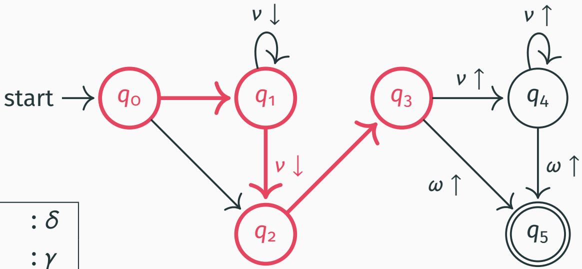


q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle$$

δ γ γ

Moore PDA

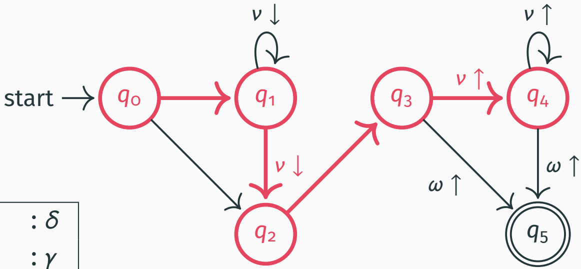


q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle$$

$\delta \qquad \gamma \qquad \gamma \qquad \beta$

Moore PDA

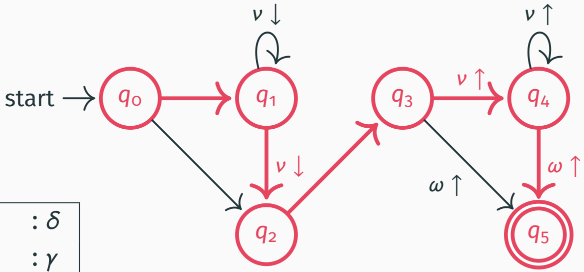


q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle$$

$$\delta \qquad \gamma \qquad \gamma \qquad \beta \qquad \alpha$$

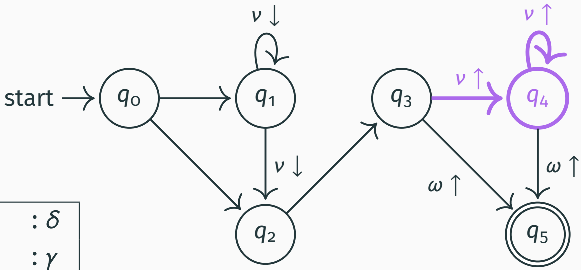
Moore PDA



q_0	$:\delta$
$q_1,\ q_2$	$:\gamma$
q_3	$:\beta$
$q_4,\ q_5$	$:\alpha$

$$\langle q_0, \boxed{\omega}\rangle \vdash \langle q_1, \boxed{\omega}\rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline\end{array}\rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline\end{array}\rangle \vdash \langle q_4, \boxed{\omega}\rangle \vdash \langle q_5, \_\_\rangle$$

$$\delta \qquad \gamma \qquad \gamma \qquad \beta \qquad \alpha \qquad \alpha$$



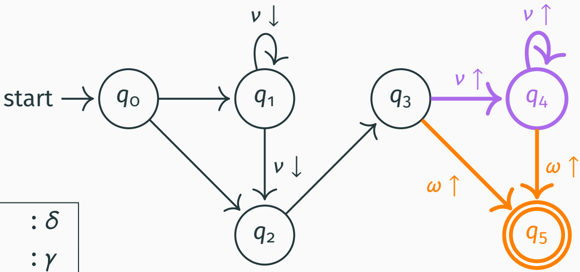
$\mathrm{pop}(\nu) = q_4$

q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

$$\delta \qquad \gamma \qquad \gamma \qquad \beta \qquad \alpha \qquad \alpha$$

Moore PDA



$\mathrm{pop}(\nu) = q_4$

$\mathrm{pop}(\omega) = q_5$

q_0	$: \delta$
$q_1,\ q_2$	$: \gamma$
q_3	$: \beta$
$q_4,\ q_5$	$: \alpha$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

$$\delta \qquad \gamma \qquad \gamma \qquad \beta \qquad \alpha \qquad \alpha$$

Simulate Moore PDA

- Moore PDA generates all spines (of length ≥ 2)

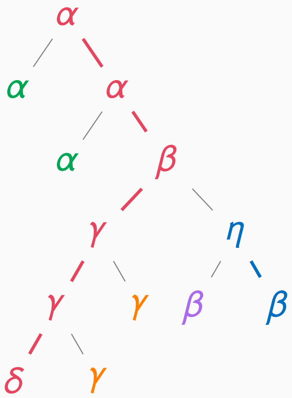
Simulate Moore PDA

- Moore PDA generates all spines (of length ≥ 2)
- primary category length can grow unbounded
  $\longrightarrow$ simulate Moore PDA in primary spines of CCG
  $\longrightarrow$ store stack in the argument sequence

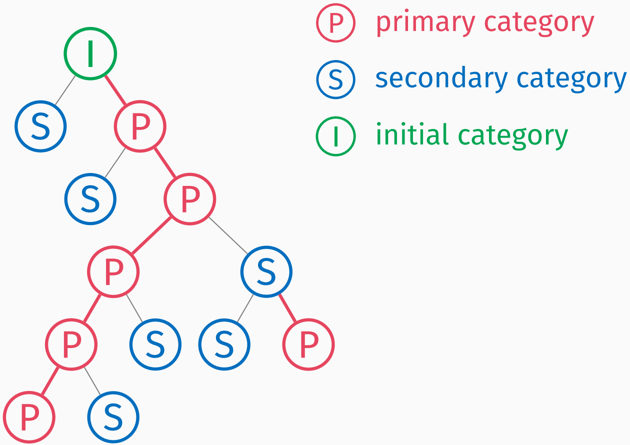
Simulate Moore PDA

- Moore PDA generates all spines (of length ≥ 2)
- primary category length can grow unbounded
  $\longrightarrow$ simulate Moore PDA in primary spines of CCG
  $\longrightarrow$ store stack in the argument sequence
- last argument of primary category stores
  - current state
  - topmost stack symbol

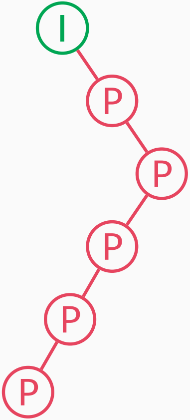
Primary Spines



Primary Spines

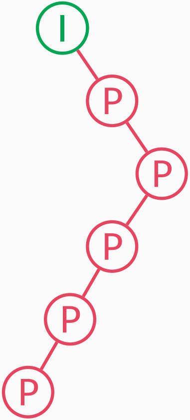


Simulate Moore PDA



$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

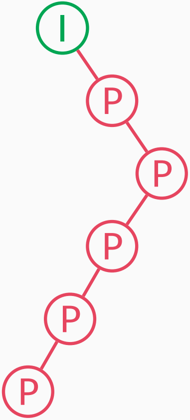
Simulate Moore PDA



$c \,/\, \begin{pmatrix} q_0 \\ \omega \end{pmatrix}$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA

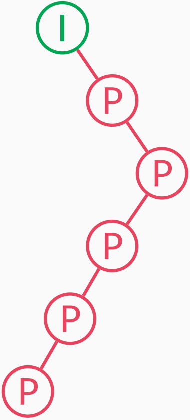


$c \,/ \begin{pmatrix} q_1 \\ \omega \end{pmatrix}$

$c \,/ \begin{pmatrix} q_0 \\ \omega \end{pmatrix}$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA



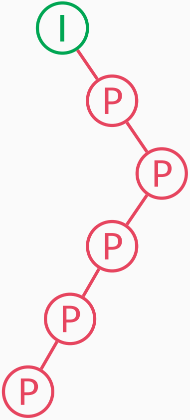
$$c \backslash \begin{pmatrix} q_4 \\ \omega \end{pmatrix} / \begin{pmatrix} q_2 \\ \nu \end{pmatrix} \qquad \mathrm{pop}(\nu) = q_4$$

$$c \,/ \begin{pmatrix} q_1 \\ \omega \end{pmatrix}$$

$$c \,/ \begin{pmatrix} q_0 \\ \omega \end{pmatrix}$$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \underline{\quad} \rangle$$

Simulate Moore PDA



$$c \setminus \binom{q_4}{\omega} \setminus \binom{q_3}{\nu}$$

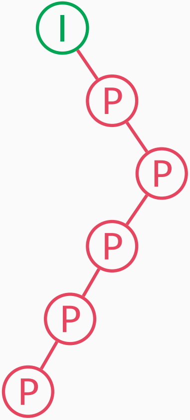
$$c \setminus \binom{q_4}{\omega} / \binom{q_2}{\nu} \qquad \mathrm{pop}(\nu) = q_4$$

$$c / \binom{q_1}{\omega}$$

$$c / \binom{q_0}{\omega}$$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \underline{\quad} \rangle$$

Simulate Moore PDA



$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix}$$

$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix} \setminus \begin{pmatrix} q_3 \\ \nu \end{pmatrix}$$

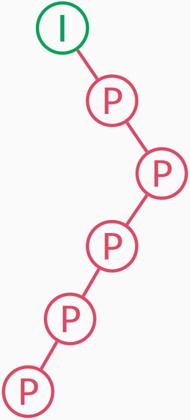
$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix} / \begin{pmatrix} q_2 \\ \nu \end{pmatrix} \qquad \mathrm{pop}(\nu) = q_4$$

$$c \,/ \begin{pmatrix} q_1 \\ \omega \end{pmatrix}$$

$$c \,/ \begin{pmatrix} q_0 \\ \omega \end{pmatrix}$$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA



c

$c \setminus \binom{q_4}{\omega}$

$c \setminus \binom{q_4}{\omega} \setminus \binom{q_3}{\nu}$

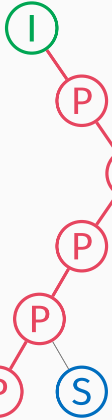
$c \setminus \binom{q_4}{\omega} / \binom{q_2}{\nu}$ $\qquad \mathrm{pop}(\nu) = q_4$

$c / \binom{q_1}{\omega}$

$c / \binom{q_0}{\omega}$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA



c

$c \backslash \binom{q_4}{\omega}$

$c \backslash \binom{q_4}{\omega} \backslash \binom{q_3}{\nu}$

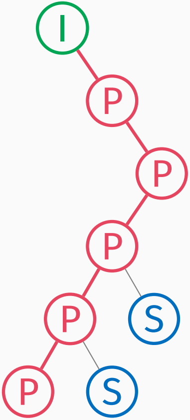
$c \backslash \binom{q_4}{\omega} / \binom{q_2}{\nu}$ $\quad$ $\mathrm{pop}(\nu) = q_4$

$c / \binom{q_1}{\omega}$

$c / \binom{q_0}{\omega}$ $\quad$ $\binom{q_0}{\omega} / \binom{q_1}{\omega}$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA



c

$c \backslash \binom{q_4}{\omega}$

$c \backslash \binom{q_4}{\omega} \backslash \binom{q_3}{\nu}$

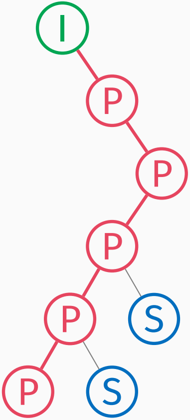
$c \backslash \binom{q_4}{\omega} / \binom{q_2}{\nu}$ $\quad$ $\mathrm{pop}(\nu) = q_4$

$c / \binom{q_1}{\omega} \quad \binom{q_1}{\omega} \backslash \binom{q_4}{\omega} / \binom{q_2}{\nu}$

$c / \binom{q_0}{\omega} \quad \binom{q_0}{\omega} / \binom{q_1}{\omega}$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Simulate Moore PDA



$$c = \begin{pmatrix} \bot \\ \varepsilon \\ q_5 \end{pmatrix}$$

$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix}$$

$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix} \setminus \begin{pmatrix} q_3 \\ \nu \end{pmatrix}$$

$$c \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix} / \begin{pmatrix} q_2 \\ \nu \end{pmatrix} \qquad \mathrm{pop}(\nu) = q_4$$

$$c / \begin{pmatrix} q_1 \\ \omega \end{pmatrix} \quad \begin{pmatrix} q_1 \\ \omega \end{pmatrix} \setminus \begin{pmatrix} q_4 \\ \omega \end{pmatrix} / \begin{pmatrix} q_2 \\ \nu \end{pmatrix}$$

$$c / \begin{pmatrix} q_0 \\ \omega \end{pmatrix} \quad \begin{pmatrix} q_0 \\ \omega \end{pmatrix} / \begin{pmatrix} q_1 \\ \omega \end{pmatrix}$$

$$\langle q_0, \boxed{\omega} \rangle \vdash \langle q_1, \boxed{\omega} \rangle \vdash \langle q_2, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_3, \begin{array}{|c|}\hline \nu \\ \hline \omega \\ \hline \end{array} \rangle \vdash \langle q_4, \boxed{\omega} \rangle \vdash \langle q_5, \_\_ \rangle$$

Strong Equivalence Result

TAG = sCFTG

$\rotatebox[origin=c]{90}{$\subseteq$}$

k-CCG

Strong Equivalence Result

$$\begin{array}{c} \text{2-CCG} \\ \cup\!\mid \\ \text{TAG} = \text{sCFTG} \\ \cup\!\mid \\ k\text{-CCG} \end{array}$$

Strong Equivalence Result

2-CCG

$\cup\!\!|$

TAG = sCFTG

$\cup\!\!|$

k-CCG with ε-entries

Strong Equivalence Result

2-CCG without ε-entries

$\cup\!\!\mid$

TAG = sCFTG

$\cup\!\!\mid$

k-CCG with ε-entries

Strong Equivalence Result

2-CCG without ε-entries

$\cup\!\!\mid$

TAG = sCFTG = 2-CCG without ε-entries

$\cup\!\!\mid$

k-CCG with ε-entries

Strong Equivalence Result

2-CCG without ε-entries

$\cup\!\!|$

TAG = sCFTG = 2-CCG without ε-entries

$\cup\!\!|$

k-CCG with ε-entries

Theorem

2-CCG without ε-entries generates the same class of tree languages as TAG.

Overview of Generative Capacity

<table>
<tr><th></th><th>0-CCG</th><th>1-CCG</th><th>2-CCG</th><th>k-CCG</th><th>k-CCG with ε-entries</th></tr>
<tr><td>strings</td><td>= CFG[a]</td><td>= CFG[b]</td><td colspan="2"></td><td>= TAG[c]</td></tr>
<tr><td>trees</td><td>$\subsetneq$ RTG[d]</td><td></td><td colspan="3"></td></tr>
</table>

[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$=$ CFG[a]	$=$ CFG[b]			$=$ TAG[c]
trees	$\subsetneq$ RTG[d]	$=$ RTG			

[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$=$ CFG[a]	$=$ CFG[b]			$=$ TAG[c]
trees	$\subsetneq$ RTG[d]	$=$ RTG	$=$ TAG		

[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Generative Capacity

	0-CCG	1-CCG	2-CCG	k-CCG	k-CCG with ε-entries
strings	$= \text{CFG}$[a]	$= \text{CFG}$[b]	$= \text{TAG}$		$= \text{TAG}$[c]
trees	$\subsetneq \text{RTG}$[d]	$= \text{RTG}$	$= \text{TAG}$		

[a] Bar-Hillel, Gaifman, Shamir (1964)
[b] Fowler, Penn (2010)
[c] Vijay-Shanker, Weir (1994)
[d] Buszkowski (1988)

Overview of Results

CCG variant	rule degree	ε-entries	string languages	tree languages
(pure) with application rules only	$k = 0$	yes/no	$=$ CFG	$\subsetneq$ RTG
pure with composition	$k = 1$	yes/no	$=$ CFG	$\subsetneq$ RTG
composition	$k = 1$	yes/no	$=$ CFG	$=$ RTG
pure with composition	$k \geq 2$	yes/no	$\subsetneq$ TAG	$\subsetneq$ TAG
prefix-closed, no target restrictions	$k \geq 2$	yes/no	$\subsetneq$ TAG	$\subsetneq$ TAG
prefix-closed	$k \geq 2$	yes	$=$ TAG	
composition	$k \geq 2$	no	$=$ TAG	$=$ TAG
composition	$k \geq 2$	yes	$=$ TAG	$=$ TAG
composition and substitution	$k \geq 2$	yes	$=$ TAG	
generalized composition	unlimited	no	$\supsetneq$ TAG	$\supsetneq$ TAG
generalized composition	unlimited	yes	$\supsetneq$ TAG	$\supsetneq$ TAG

Overview of Results

CCG variant	rule degree	ε-entries	string languages	tree languages
(pure) with application rules only	$k = 0$	yes/no	$=$ CFG	$\subsetneq$ RTG
pure with composition	$k = 1$	yes/no	$=$ CFG	$\subsetneq$ RTG
composition	$k = 1$	yes/no	$=$ CFG	$=$ RTG
pure with composition	$k \geq 2$	yes/no	$\subsetneq$ TAG	$\subsetneq$ TAG
prefix-closed, no target restrictions	$k \geq 2$	yes/no	$\subsetneq$ TAG	$\subsetneq$ TAG
prefix-closed	$k \geq 2$	yes	$=$ TAG	
composition	$k \geq 2$	no	$=$ TAG	$=$ TAG
composition	$k \geq 2$	yes	$=$ TAG	$=$ TAG
composition and substitution	$k \geq 2$	yes	$=$ TAG	
generalized composition	unlimited	no	$\supsetneq$ TAG	$\supsetneq$ TAG
generalized composition	unlimited	yes	$\supsetneq$ TAG	$\supsetneq$ TAG

CCG variant	complexity
with ε-entries	EXPTIME
without ε-entries	NP
bounded degree	PTIME

Publications

- Marco Kuhlmann, Andreas Maletti, and Lena K. Schiffer. **The tree-generative capacity of combinatory categorial grammars.** *FSTTCS*, 2019.
- Marco Kuhlmann, Andreas Maletti, and Lena K. Schiffer. **The tree-generative capacity of combinatory categorial grammars.** *Journal of Computer and System Sciences*, 2022.
- Lena K. Schiffer and Andreas Maletti. **Strong equivalence of TAG and CCG.** *Transactions of the Association for Computational Linguistics*, 2021.
- Lena K. Schiffer, Marco Kuhlmann, and Giorgio Satta. **Tractable Parsing for CCGs of Bounded Degree.** *Computational Linguistics*, 2022.
- Andreas Maletti and Lena K. Schiffer. **Combinatory categorial grammars as generators of weighted forests.** *Information and Computation*, 2023.