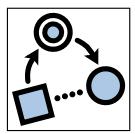
Grammar Implementation with Lexicalized Tree Adjoining Grammars and Frame Semantics

Introduction

Laura Kallmeyer, Timm Lichte, Rainer Osswald & Simon Petitjean

University of Düsseldorf

DGfS CL Fall School, September 11, 2017



What this course is about

“Working with Tree-Adjoining Grammar”

- Who is “working”?

What this course is about

“Working with Tree-Adjoining Grammar”

- Who is “working”? The linguist!

What this course is about

“Working with Tree-Adjoining Grammar”

- Who is “working”? The linguist!
- On what?

What this course is about

“Working with Tree-Adjoining Grammar”

- Who is “working”? The linguist!
- On what? Trying to implement$^{1/2}$ syntactic theories!
  - implement^{1}: general concepts → mathematical objects
  - implement^{2}: paper & pencil → electronic resource

What this course is about

"Working with Tree-Adjoining Grammar"

- Who is "working"? The linguist!
- On what? Trying to implement$^{1/2}$ syntactic theories!
  - implement^{1}: general concepts → mathematical objects
  - implement^{2}: paper & pencil → electronic resource
- Why implementation?

As is frequently pointed out but cannot be overemphasized, an important goal of formalization in linguistics is to enable subsequent researchers to see the defects of an analysis as clearly as its merits; only then can progress be made efficiently. (Dowty (1979): 322)

What this course is about

"Working with Tree-Adjoining Grammar"

- Who is "working"? The linguist!
- On what? Trying to implement$^{1/2}$ syntactic theories!
  - implement^{1}: general concepts → mathematical objects
  - implement^{2}: paper & pencil → electronic resource
- Why implementation?

As is frequently pointed out but cannot be overemphasized, an important goal of formalization in linguistics is to enable subsequent researchers to see the defects of an analysis as clearly as its merits; only then can progress be made efficiently. (Dowty (1979): 322)

- incentive for rigor
- check for consistency
- application (NLP)

What this course is about: Observations and theory

Observations

Theory

What this course is about: Observations and theory

Observations

set of pairs ⟨ **utterance , meaning** ⟩

Theory

What this course is about: Observations and theory

Observations

set of pairs ⟨ **utterance , meaning** ⟩

Theory

“**Grammar**”, that is

- formalized
- efficient (i.e., not NP-hard)
- comprehensible (i.e., no black-box technology)

(maybe characteristic for **computational linguistics**)

What this course is about: Observations and theory

Observations (more specific)

set of pairs ⟨ **sentence , semantic frame** ⟩

Theory (more specific)

“**Grammar**”, that is

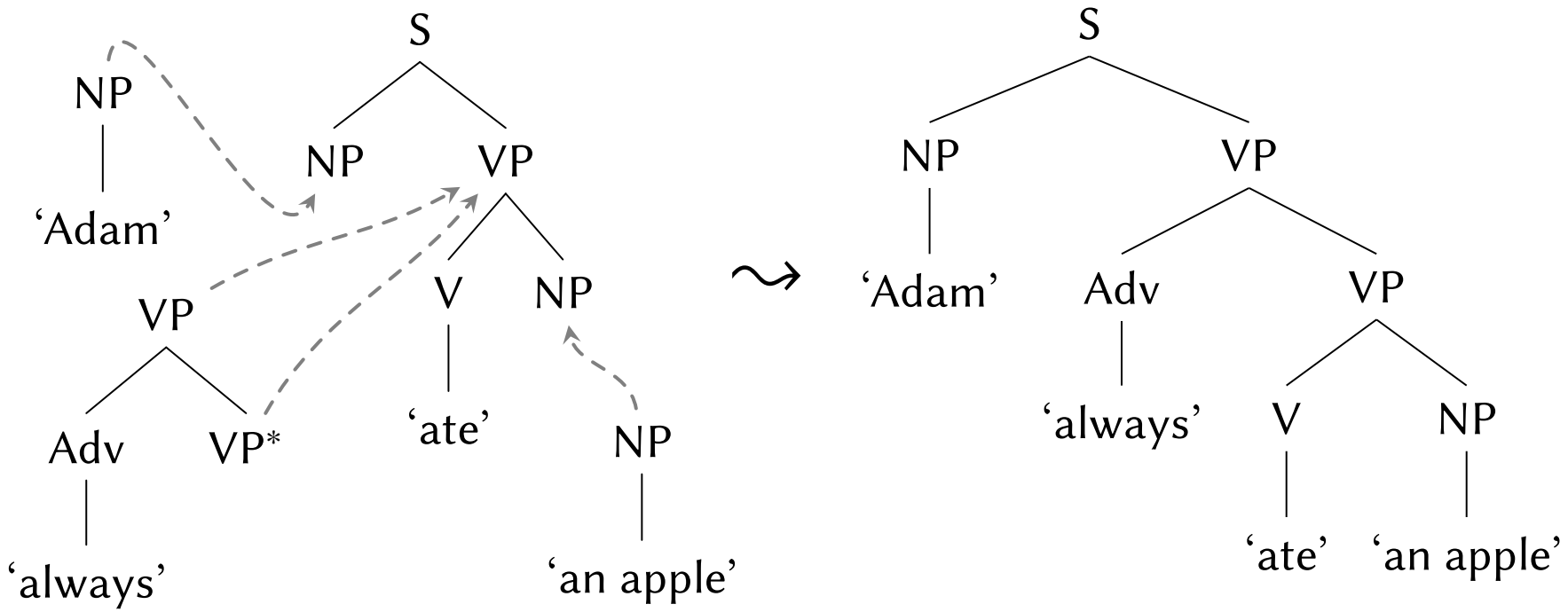
- formalized
- efficient (i.e., not NP-hard)
- comprehensible (i.e., no black-box technology)

⇒ based on **Tree-Adjoining Grammar (TAG)** and **Frame Semantics**

What this course is about: Example 1

A simple example: tree composition in LTAG

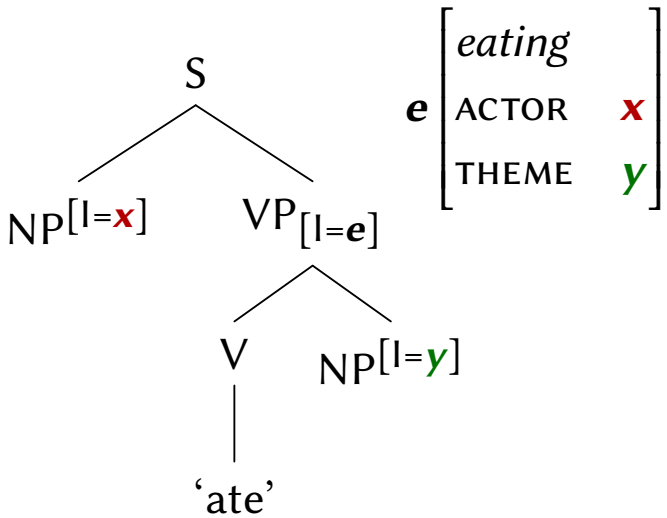
(1) Adam always ate an apple.



What this course is about: Example 2

A simple example: tree-frame pairs and composition

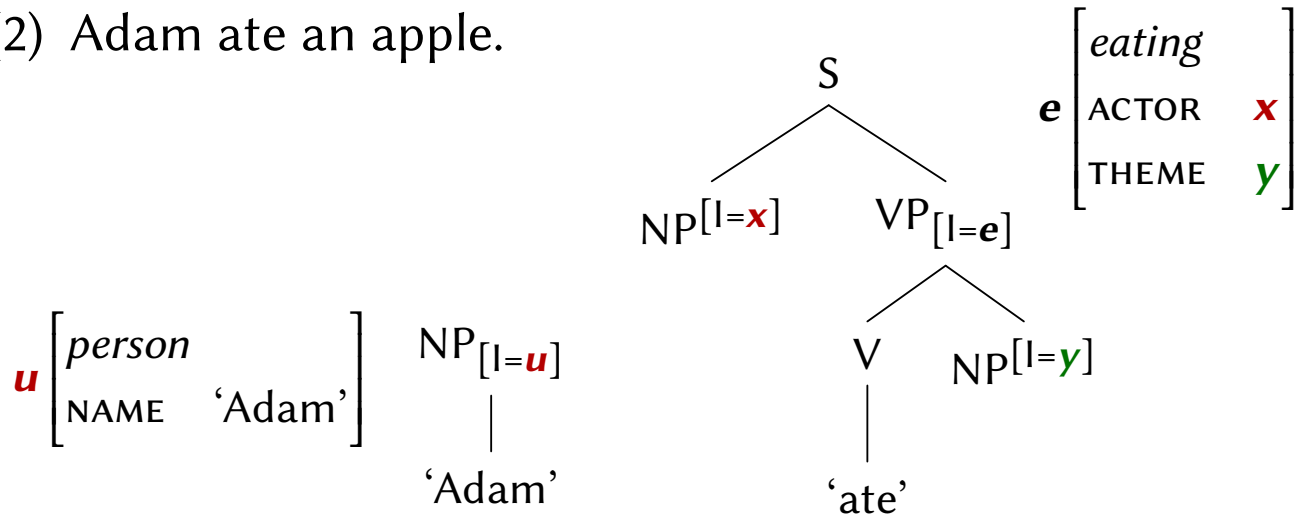
(2) Adam ate an apple.



What this course is about: Example 2

A simple example: tree-frame pairs and composition

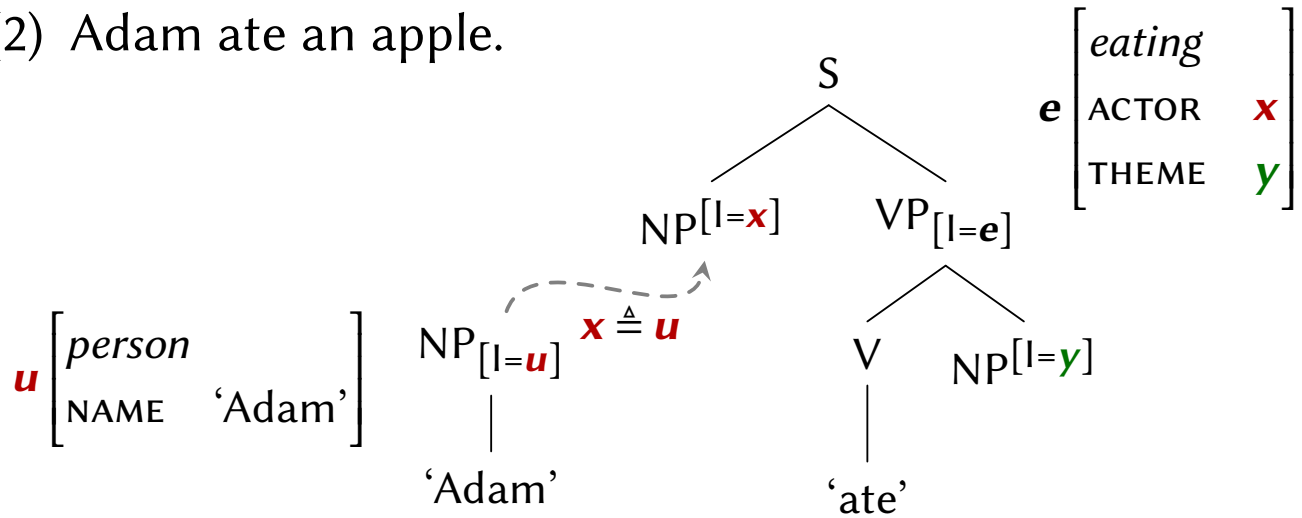
(2) Adam ate an apple.



What this course is about: Example 2

A simple example: tree-frame pairs and composition

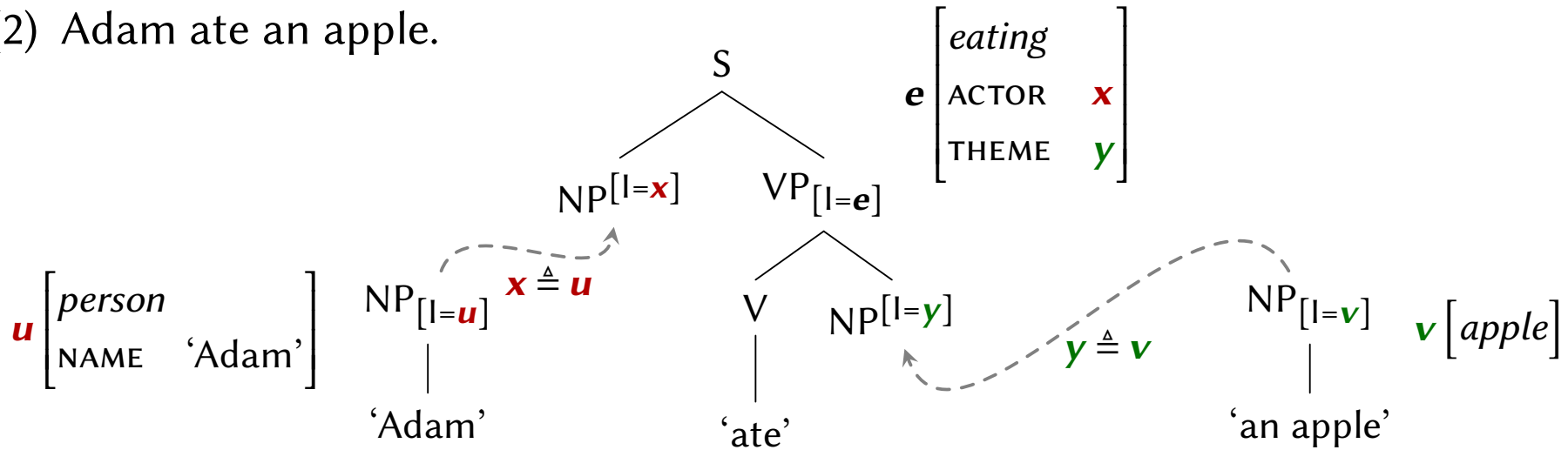
(2) Adam ate an apple.



What this course is about: Example 2

A simple example: tree-frame pairs and composition

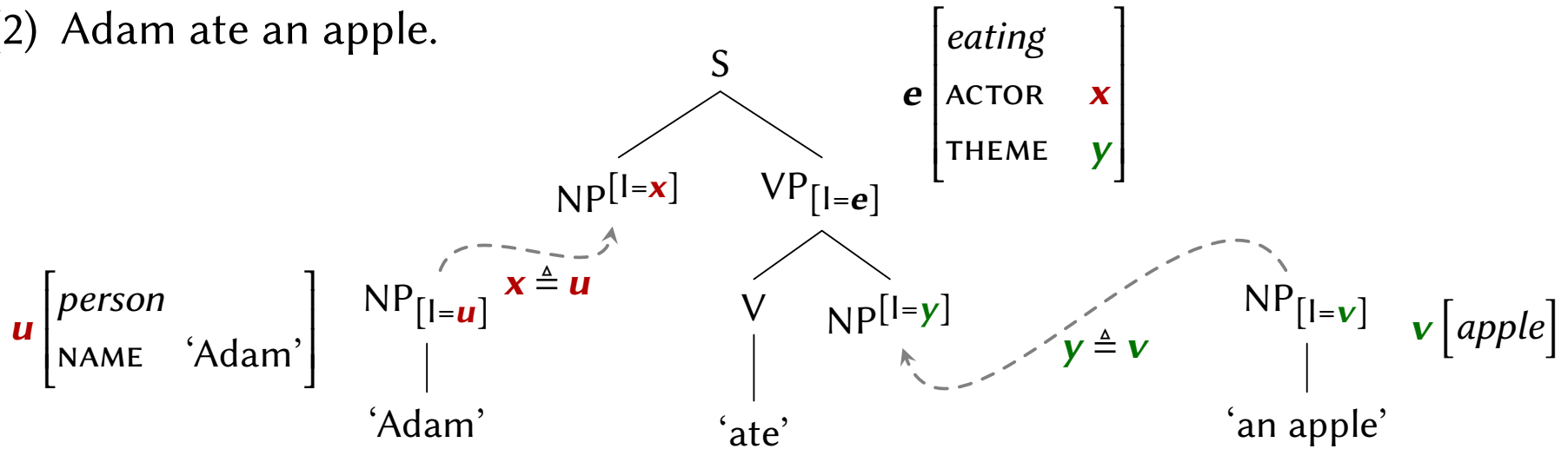
(2) Adam ate an apple.

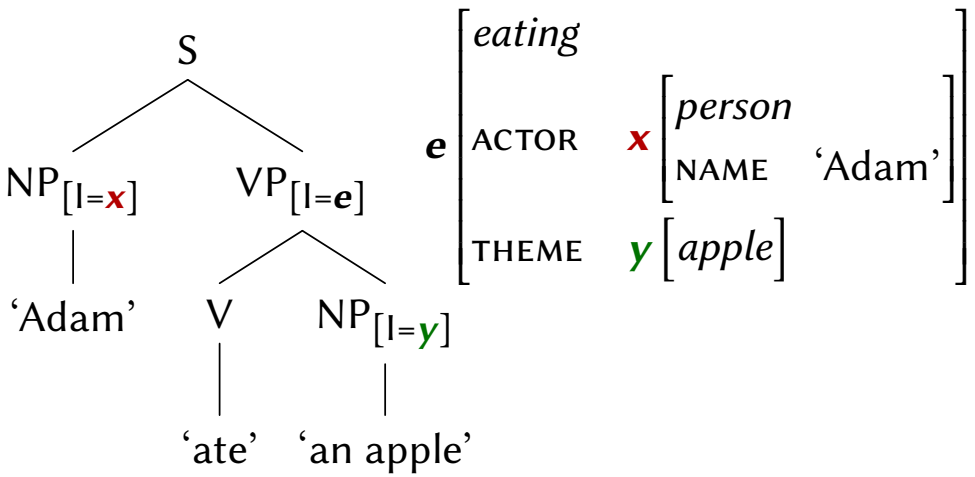


What this course is about: Example 2

A simple example: tree-frame pairs and composition

(2) Adam ate an apple.

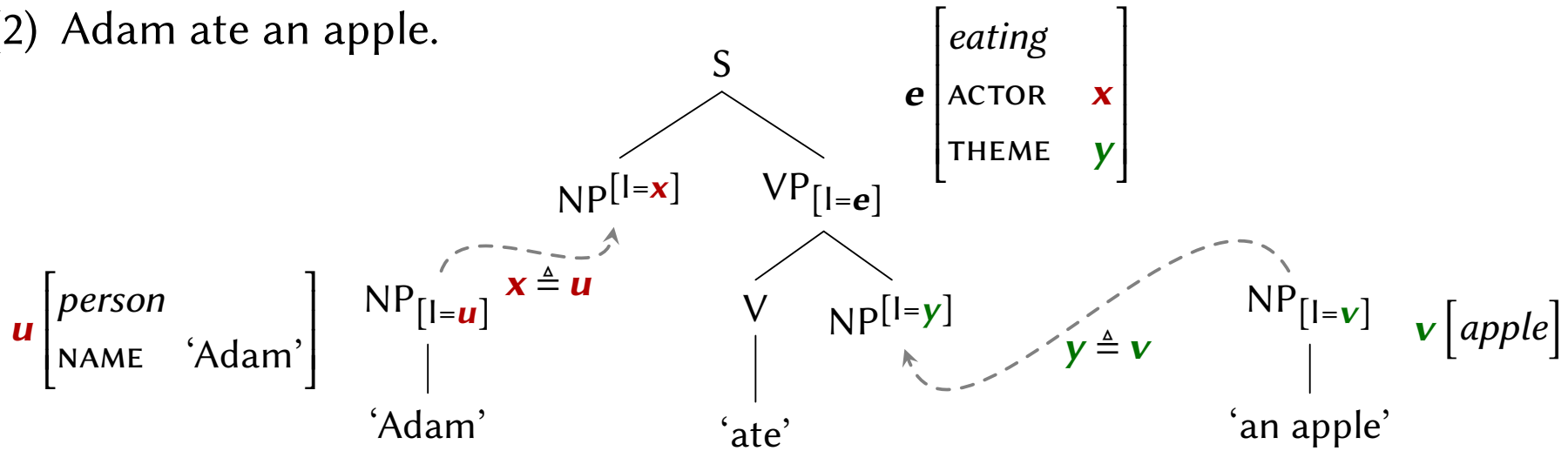


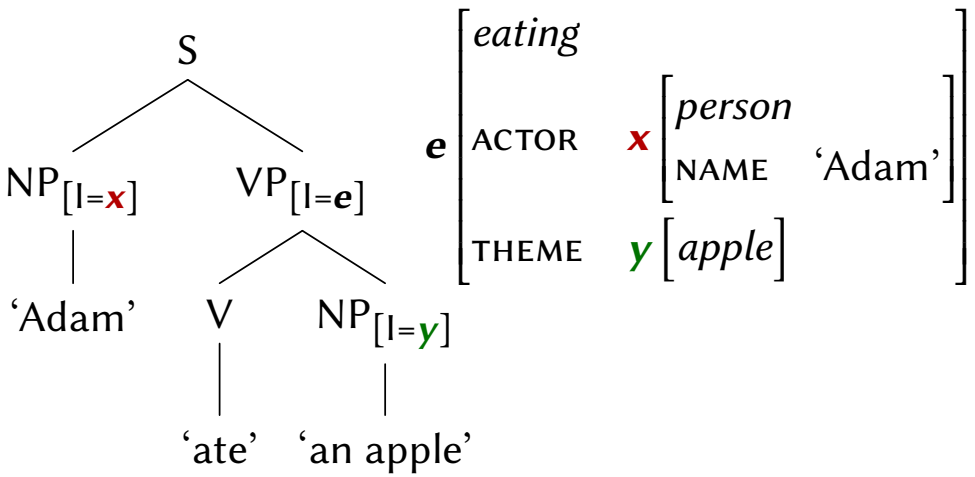


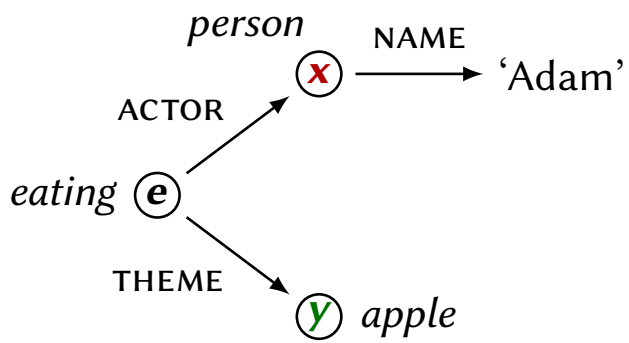
What this course is about: Example 2

A simple example: tree-frame pairs and composition

(2) Adam ate an apple.







This week

Mon: introduction to LTAG

Tue: syntactic analyses with LTAG I,
derivation trees, feature structures

Wed: syntactic analyses with LTAG II,
introduction to LTAG semantics

Thu: introduction to frame semantics

Fri: putting things together

(will be taught by Laura Kallmeyer and Rainer Osswald)

Next week

Mon: introduction to grammar engineering and XMG

Tue: implementing syntax with XMG

Wed: implementing semantics with XMG

Thu: parsing implemented grammars with TuLiPA

Fri: Conclusion

(will be taught by Timm Lichte and Simon Petitjean)

Some of the slides are taken from Kata Balogh and Timm Lichte's ESSLLI 2015 course.

Outline of today's course

1. Why "working" with TAG?
   - Formal reasons
   - Linguistic reasons
2. From CFG to TAG
   - Context-Free Grammars
   - Lexicalization
   - Tree Substitution Grammars (TSG)
   - Adding adjunction
3. Further related formalisms
4. Summary & outlook
5. Appendix: NL and the generative capacity of grammar formalisms

Outline of today's course

1. Why "working" with TAG?
   - Formal reasons
   - Linguistic reasons
2. From CFG to TAG
   - Context-Free Grammars
   - Lexicalization
   - Tree Substitution Grammars (TSG)
   - Adding adjunction
3. Further related formalisms
4. Summary & outlook
5. Appendix: NL and the generative capacity of grammar formalisms

Why “working” with TAG? Formal reasons

Hypothesis of the adequacy of expressive power

TAG exactly provides the expressive power needed to treat NL.

(The complexity of a language is determined by the weakest formal grammar that generates it.)

Why “working” with TAG? Formal reasons

Hypothesis of the adequacy of expressive power

TAG exactly provides the expressive power needed to treat NL.

(The complexity of a language is determined by the weakest formal grammar that generates it.)

Why is the formal complexity of natural languages interesting?
It allows one to gain insights into

- ⇒ the general structure of natural language
- ⇒ the general human language capacity
- ⇒ the adequacy of grammar formalisms
- ⇒ lower bound of the computational complexity of NLP tasks

Why “working” with TAG? Formal reasons

Expressive power in terms of a specific generative capacity:

Weak generative capacity (WGC)

The capacity to generate **string languages**.

Strong generative capacity (SGC)

The capacity to generate **tree languages**.

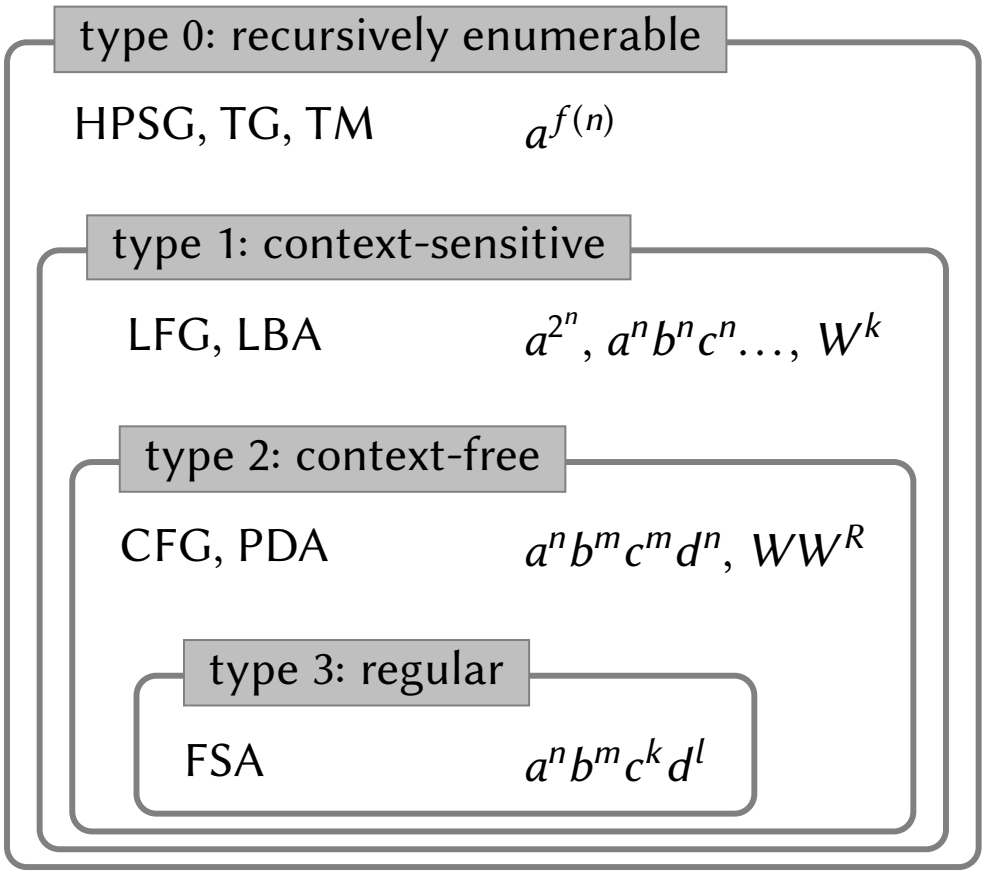
Derivational generative capacity (DGC)

The capacity to generate string languages in a certain way.

In what follows we will consider the **weak generative capacity**.

Why “working” with TAG? Formal reasons

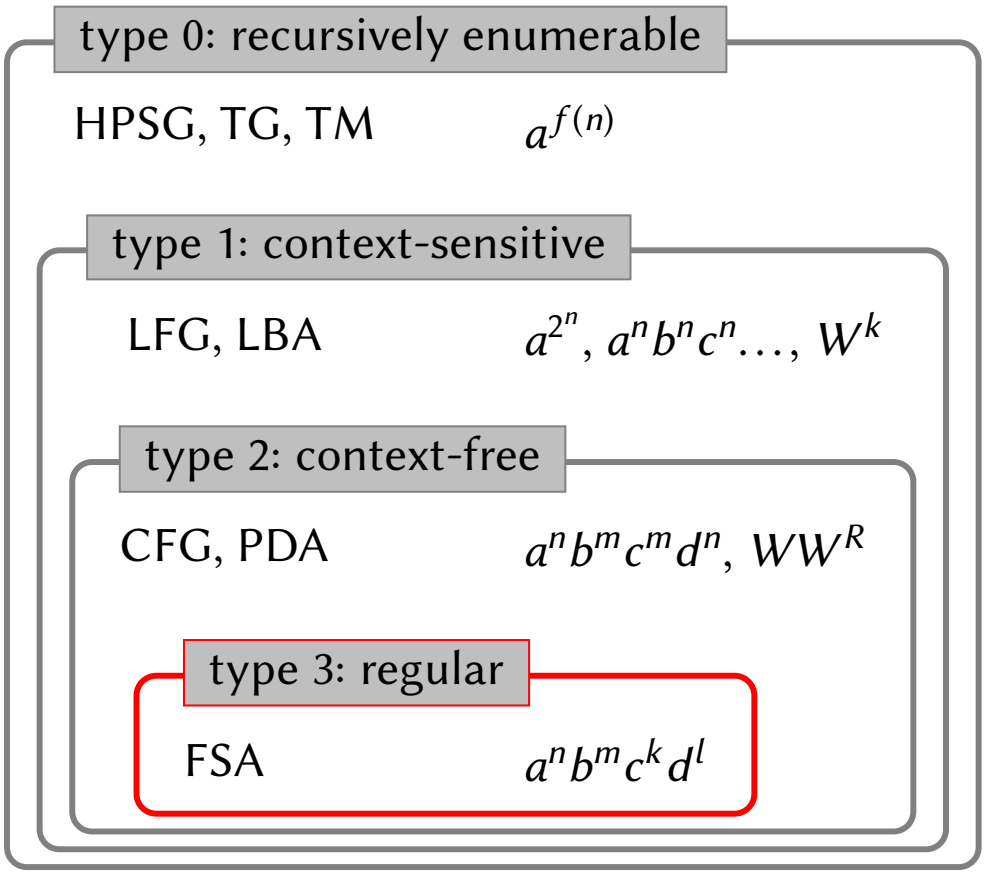
How much expressive power do we need to treat NL?



Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

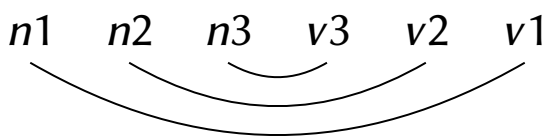
Why “working” with TAG? Formal reasons

How much expressive power do we need to treat NL?



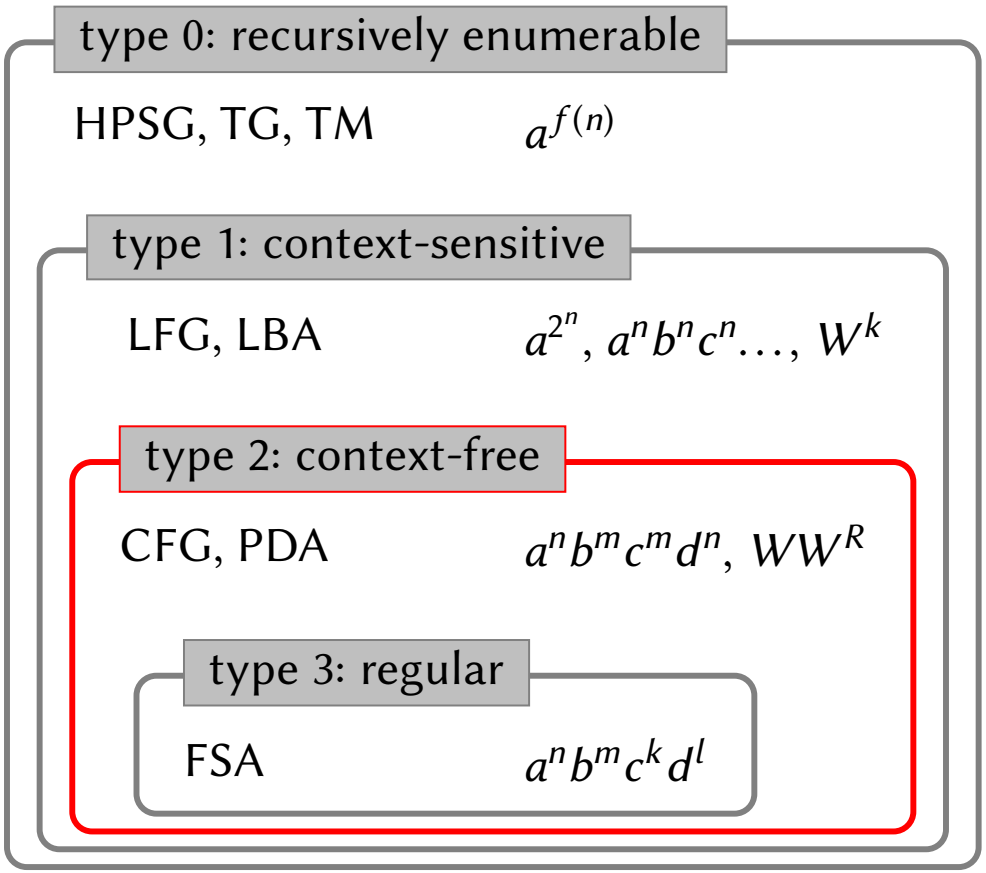
Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

NL is not regular!
(Chomsky 1956; 1957)
center embedding with relative clauses



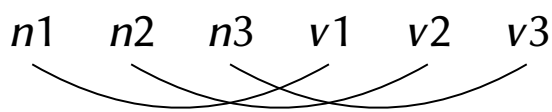
Why “working” with TAG? Formal reasons

How much expressive power do we need to treat NL?



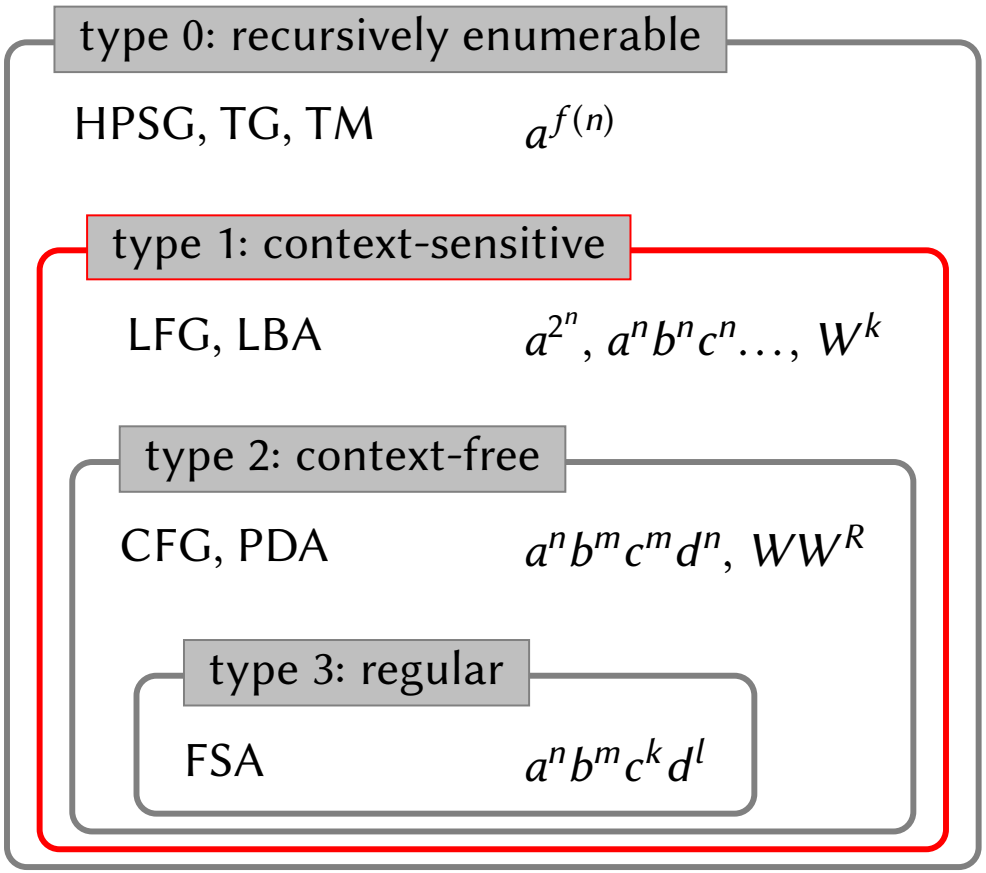
Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

NL is not context-free!
Shieber (1985)
cross serial dependencies in Dutch and Swiss-German



Why “working” with TAG? Formal reasons

How much expressive power do we need to treat NL?

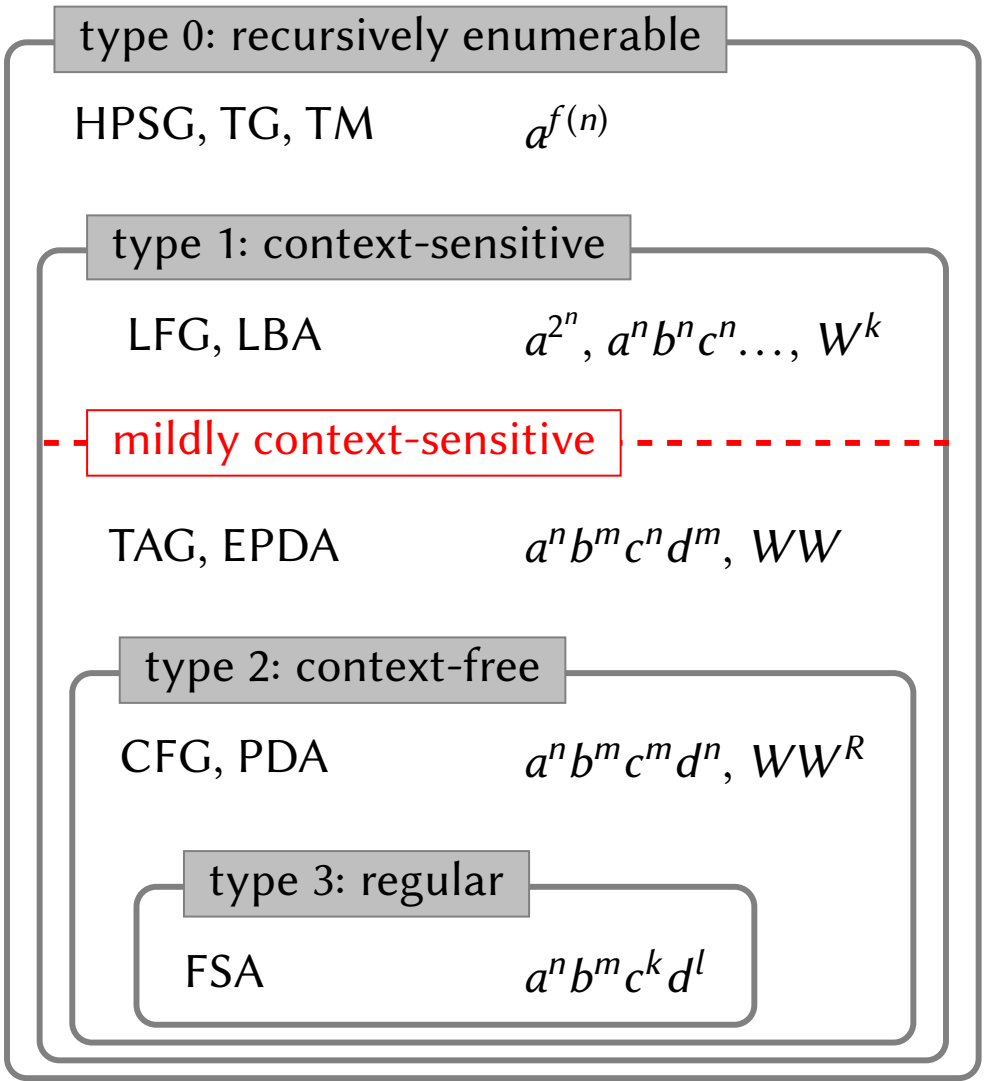


Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

NL is context-sensitive?

Why “working” with TAG? Formal reasons

How much expressive power do we need to treat NL?



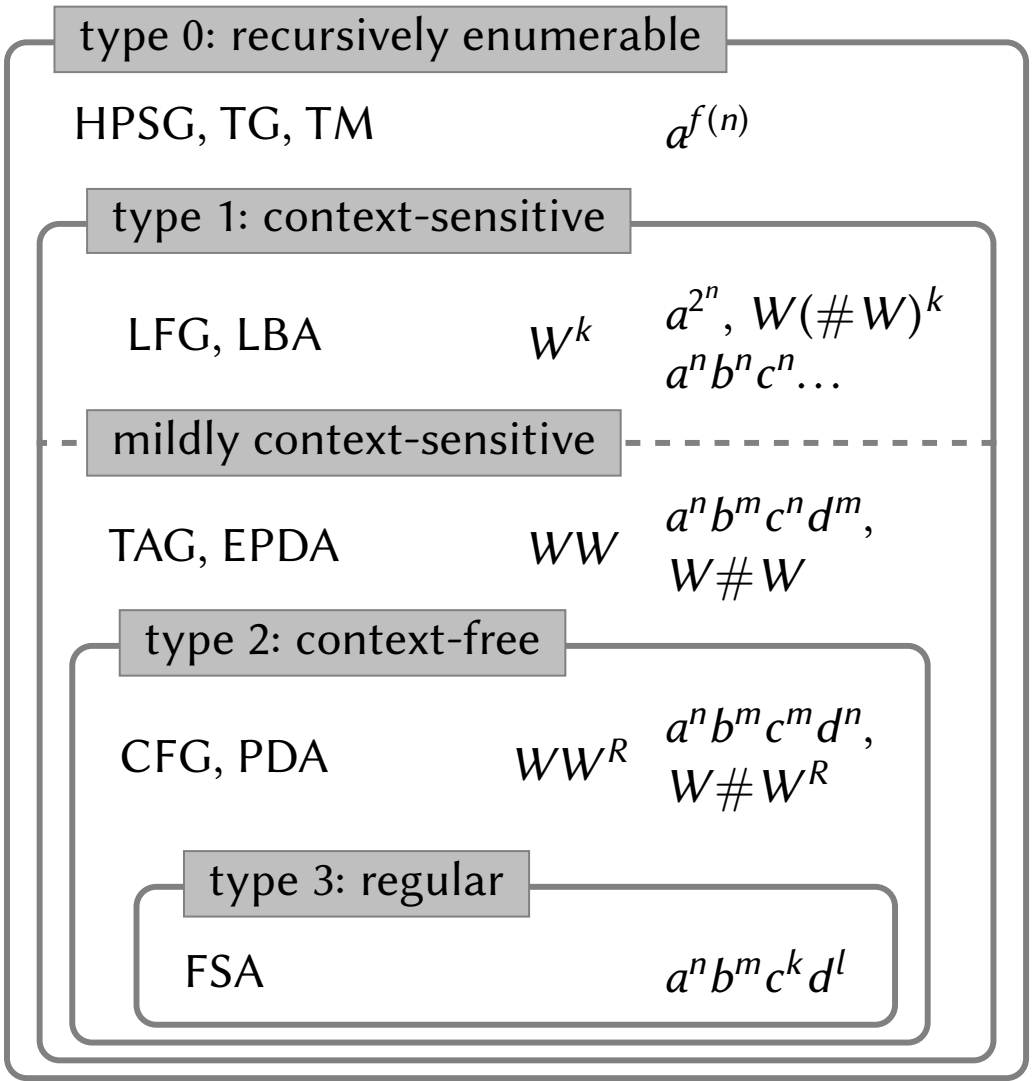
Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

NL is **mildly** context-sensitive? (Joshi 1985)

- $\supset$ CFL
- cross-serial dep.
- semi-linear
- in PTIME

Why “working” with TAG? Formal reasons

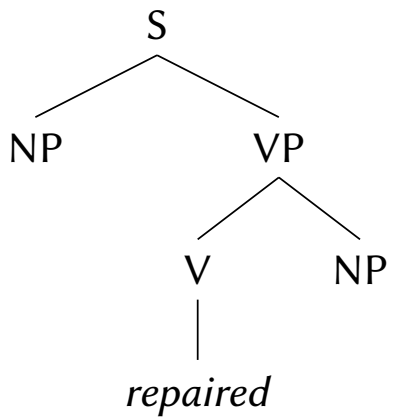
How much expressive power do we need to treat NL?



Chomsky(-Schützenberger) hierarchy
(Chomsky & Schützenberger 1963)

Why “working” with TAG? Linguistic reasons

- extended domain of locality



- long-distance dependencies / discontinuous constituents

  (3) Who did Mary say that Tom claimed . . . repaired the fridge?

- multi-word expressions

  (4) to kick the bucket (‘to die’)

- incarnation of Construction Grammar

Outline of today's course

1. Why "working" with TAG?
   - Formal reasons
   - Linguistic reasons
2. From CFG to TAG
   - Context-Free Grammars
   - Lexicalization
   - Tree Substitution Grammars (TSG)
   - Adding adjunction
3. Further related formalisms
4. Summary & outlook
5. Appendix: NL and the generative capacity of grammar formalisms

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow \textit{Peter} \mid \textit{fridge}$$
$$Det \rightarrow \textit{the}$$
$$A \rightarrow \textit{easily}$$
$$V \rightarrow \textit{repaired}$$

$\}$

Example derivation :

S

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation :

$$NP\ VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow \textit{Peter} \mid \textit{fridge}$$
$$Det \rightarrow \textit{the}$$
$$A \rightarrow \textit{easily}$$
$$V \rightarrow \textit{repaired}$$

$\}$

Example derivation :

$$N\ VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow Peter \mid fridge$$
$$Det \rightarrow the$$
$$A \rightarrow easily$$
$$V \rightarrow repaired$$

$\}$

Example derivation :

$$Peter\ VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow Peter \mid fridge$$
$$Det \rightarrow the$$
$$A \rightarrow easily$$
$$V \rightarrow repaired$$

$\}$

Example derivation :

$$Peter\ AP\ VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation :

$$Peter\ A\ VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation :

$$\textit{Peter easily } VP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow \textit{Peter} \mid \textit{fridge}$$
$$Det \rightarrow \textit{the}$$
$$A \rightarrow \textit{easily}$$
$$V \rightarrow \textit{repaired}$$

$\}$

Example derivation :

$$\textit{Peter easily } V\ NP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation :

$$\textit{Peter easily repaired } NP$$

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow \textit{Peter} \mid \textit{fridge}$$
$$Det \rightarrow \textit{the}$$
$$A \rightarrow \textit{easily}$$
$$V \rightarrow \textit{repaired}$$

$\}$

Example derivation :

Peter easily repaired Det N

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow Peter \mid fridge$$
$$Det \rightarrow the$$
$$A \rightarrow easily$$
$$V \rightarrow repaired$$

$\}$

Example derivation :

Peter easily repaired the N

From CFG to TAG: Context-Free Grammar

- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation :

Peter easily repaired the fridge

From CFG to TAG: Context-Free Grammar

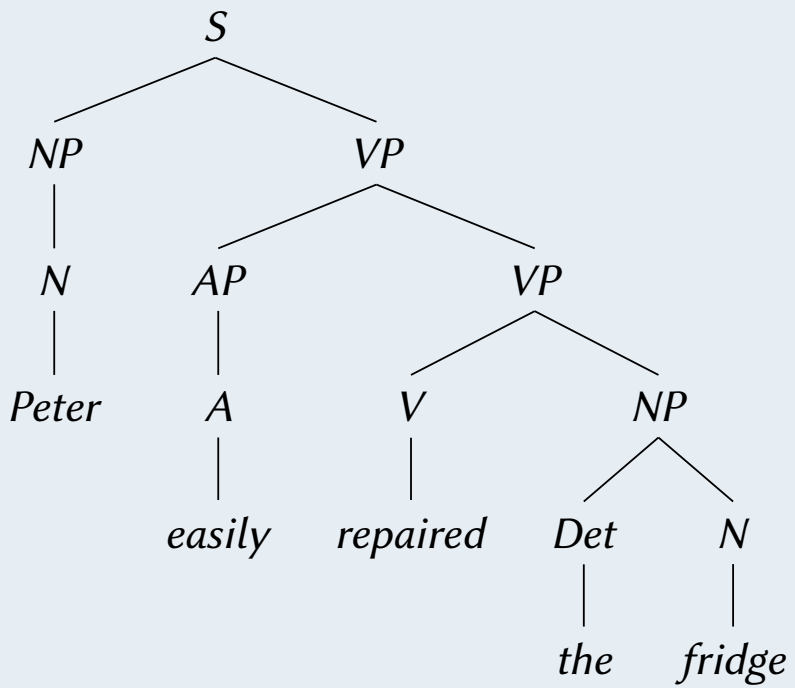
- string rewriting
- replace non-terminals by strings of terminals and non-terminals

$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow V\ NP \mid AP\ VP \\
NP &\rightarrow N \mid Det\ N \\
AP &\rightarrow A \\
N &\rightarrow \textit{Peter} \mid \textit{fridge} \\
Det &\rightarrow \textit{the} \\
A &\rightarrow \textit{easily} \\
V &\rightarrow \textit{repaired}
\end{aligned}
$$

$\}$

Example derivation history:



From CFG to TAG: Context-Free Grammar

Why not stick to CFGs (literally)?

- low generative capacity: cannot describe all NL phenomena; e.g.
  - cross-serial dependencies ($a^n b^m c^n d^m$) — Swiss German (Shieber 1985)
  - duplication ($w\#w$) — Bambara (Culy 1985)
  - multiple agreement ($a^n b^n c^n$) — Bantu languages

From CFG to TAG: Context-Free Grammar

Why not stick to CFGs (literally)?

- low generative capacity: cannot describe all NL phenomena; e.g.
  - cross-serial dependencies ($a^n b^m c^n d^m$) — Swiss German (Shieber 1985)
  - duplication ($w\#w$) — Bambara (Culy 1985)
  - multiple agreement ($a^n b^n c^n$) — Bantu languages
- poor support of expressing linguistic generalizations
  - Rules have a very limited domain of locality. ($\rightsquigarrow$ no strong lexicalization)
  - atomic non-terminals ($\rightsquigarrow$ massive proliferation)

From CFG to TAG: Context-Free Grammar

Why not stick to CFGs (literally)?

- low generative capacity: cannot describe all NL phenomena; e.g.
  - cross-serial dependencies ($a^n b^m c^n d^m$) — Swiss German (Shieber 1985)
  - duplication ($w\#w$) — Bambara (Culy 1985)
  - multiple agreement ($a^n b^n c^n$) — Bantu languages
- poor support of expressing linguistic generalizations
  - Rules have a very limited domain of locality.
    ($\rightsquigarrow$ no strong lexicalization)
  - atomic non-terminals ($\rightsquigarrow$ massive proliferation)

First step: turn strings into trees!

Lexicalization

lexicalization → each structure of the grammar has at least one non-terminal

Lexicalization

lexicalization → each structure of the grammar has at least one non-terminal

Lexicalized grammar

A lexicalized grammar consists of: (i) a finite set of structures each associated with a lexical item (anchor); and (ii) operation(s) for composing these structures.

Lexicalization

lexicalization → each structure of the grammar has at least one non-terminal

Lexicalized grammar

A lexicalized grammar consists of: (i) a finite set of structures each associated with a lexical item (anchor); and (ii) operation(s) for composing these structures.

Lexicalization

A formalism F can be lexicalized by another formalism F', if for any finitely ambiguous grammar G in F, there is a grammar G' in F', such that (i) G' is a lexicalized grammar; and (ii) G and G' generate the same tree set.

Lexicalization

lexicalization → each structure of the grammar has at least one non-terminal

Lexicalized grammar

A lexicalized grammar consists of: (i) a finite set of structures each associated with a lexical item (anchor); and (ii) operation(s) for composing these structures.

Lexicalization

A formalism F can be lexicalized by another formalism F', if for any finitely ambiguous grammar G in F, there is a grammar G' in F', such that (i) G' is a lexicalized grammar; and (ii) G and G' generate the same tree set.

weak vs. strong lexicalization

- weak lexicalization: preserve the string language
- strong lexicalization: preserve the tree structure

Lexicalization

- **Formally interesting:**
  - a finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous)

Lexicalization

- **Formally interesting:**
  - a finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous)
- **Linguistically interesting:**
  - syntactic properties of lexical items can be accounted for more directly
  - each lexical item comes with the possibility of certain partial syntactic constructions

Lexicalization

- **Formally interesting:**
  - a finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous)
- **Linguistically interesting:**
  - syntactic properties of lexical items can be accounted for more directly
  - each lexical item comes with the possibility of certain partial syntactic constructions
- **Computationally interesting:**
  - the search space during parsing can be delimited (grammar filtering)

Lexicalization of CFG's

- Greibach normal-form (Greibach 1965): $A \rightarrow aX$ or $A \rightarrow a$ ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$)

Lexicalization of CFG's

- Greibach normal-form (Greibach 1965): $A \rightarrow aX$ or $A \rightarrow a$ ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$)
- example:
  - a CFG G: $S \rightarrow SS$, $S \rightarrow a$

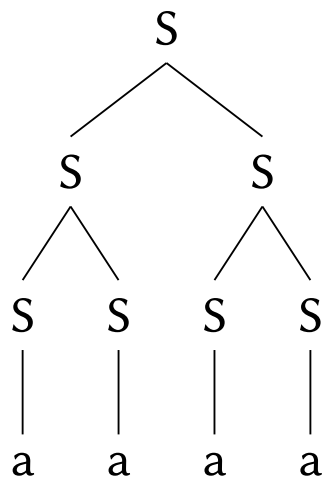
Lexicalization of CFG's

- Greibach normal-form (Greibach 1965): $A \rightarrow aX$ or $A \rightarrow a$ ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$)
- example:
  - a CFG G: $S \rightarrow SS$, $S \rightarrow a$
  - lexicalize $G \Rightarrow G'$ (Greibach): $S \rightarrow aS$, $S \rightarrow a$
- same string language, but not the same tree set

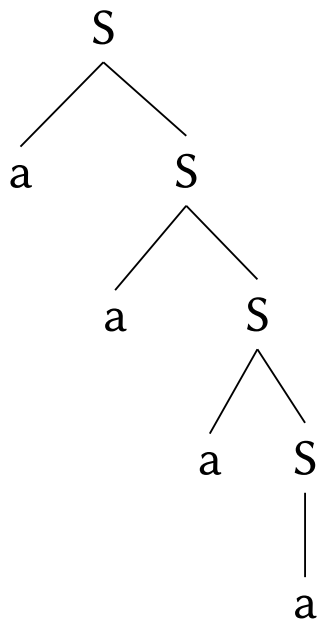
Lexicalization of CFG's

- Greibach normal-form (Greibach 1965): $A \to aX$ or $A \to a$ ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$)
- example:
  - a CFG G: $S \to SS$, $S \to a$
  - lexicalize $G \Rightarrow G'$ (Greibach): $S \to aS$, $S \to a$
- same string language, but not the same tree set

by G:



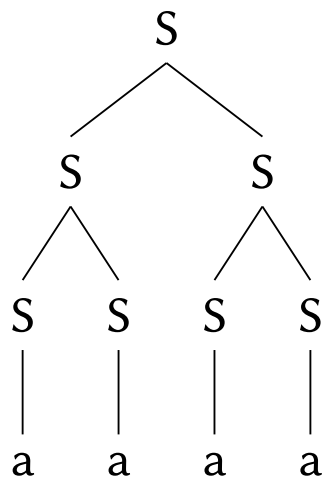
by G':



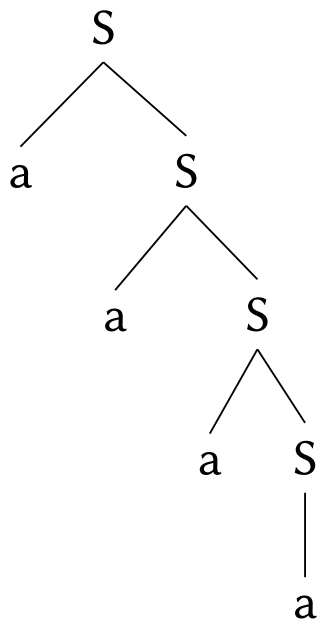
Lexicalization of CFG's

- Greibach normal-form (Greibach 1965): $A \to aX$ or $A \to a$ ($a \in V_T$; $A \in V_N$; $X \in (V_N)^*$)
- example:
  - a CFG G: $S \to SS$, $S \to a$
  - lexicalize $G \Rightarrow G'$ (Greibach): $S \to aS$, $S \to a$
- same string language, but not the same tree set

by G:



by G':



- $\rightsquigarrow$ only weak lexicalization possible

From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

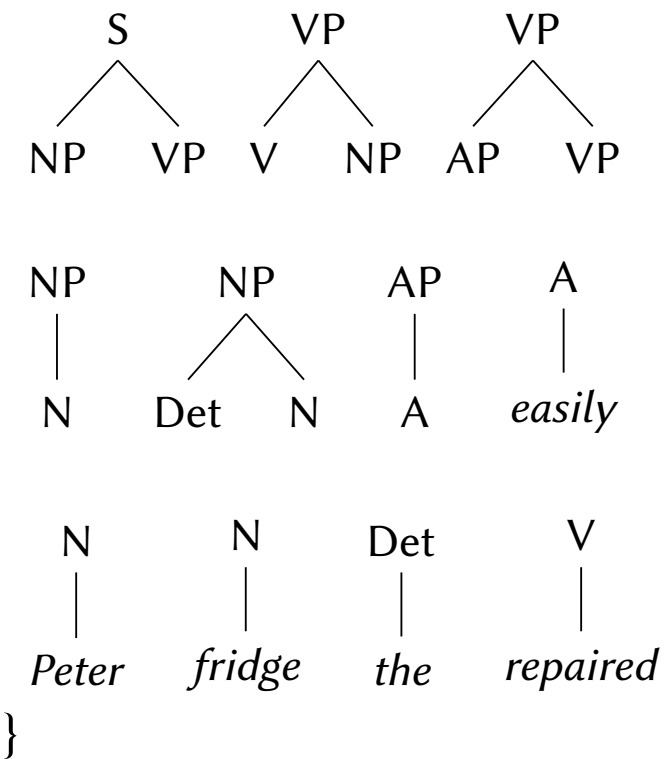
$G_{\text{CFG}} = \langle N, T, S, P \rangle$
$P = \{$

$$S \rightarrow NP\ VP$$
$$VP \rightarrow V\ NP \mid AP\ VP$$
$$NP \rightarrow N \mid Det\ N$$
$$AP \rightarrow A$$
$$N \rightarrow Peter \mid fridge$$
$$Det \rightarrow the$$
$$A \rightarrow easily$$
$$V \rightarrow repaired$$

$\}$

$\approx$

$G_{\text{TSG}} = \langle N, T, S, I \rangle$
$I = \{$



$\}$

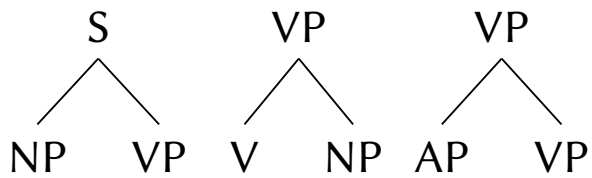
From CFG to TAG: Tree Substitution Grammar (TSG)

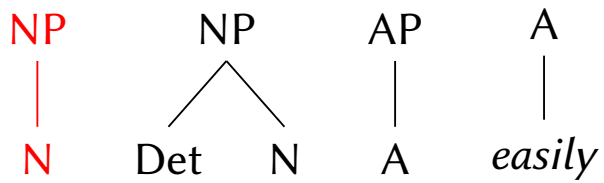
First step: turn strings into trees!

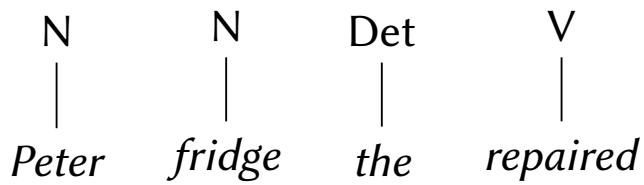
- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$

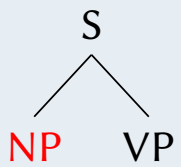






$\}$

Example derivation:



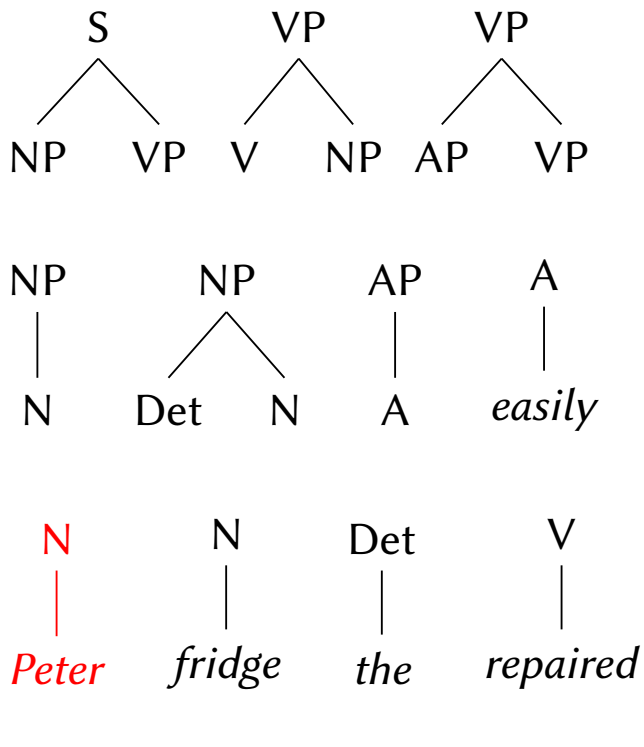
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

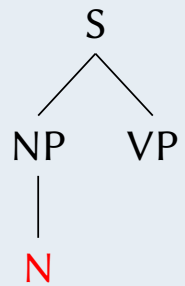
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



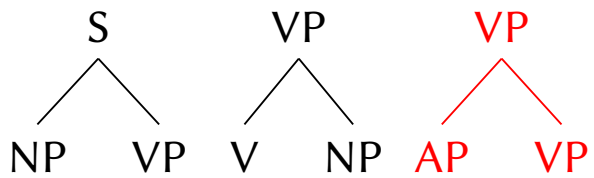
From CFG to TAG: Tree Substitution Grammar (TSG)

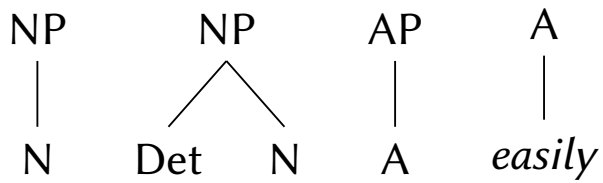
First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

$G_{\text{TSG}} = \langle N, T, S, I \rangle$

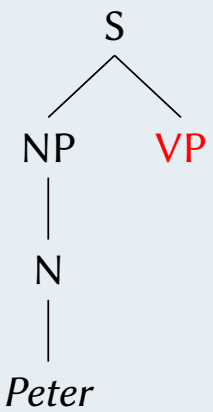
$I = \{$





$\}$

Example derivation:



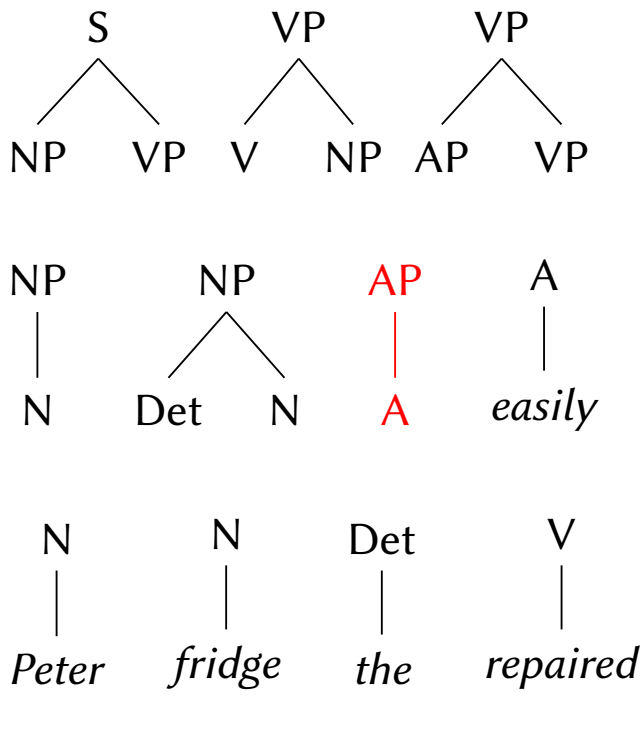
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

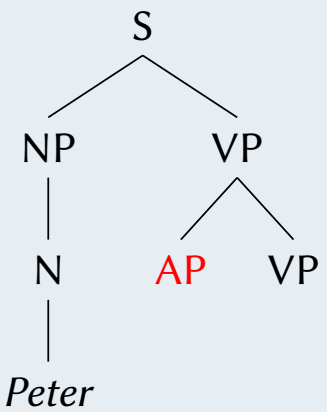
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



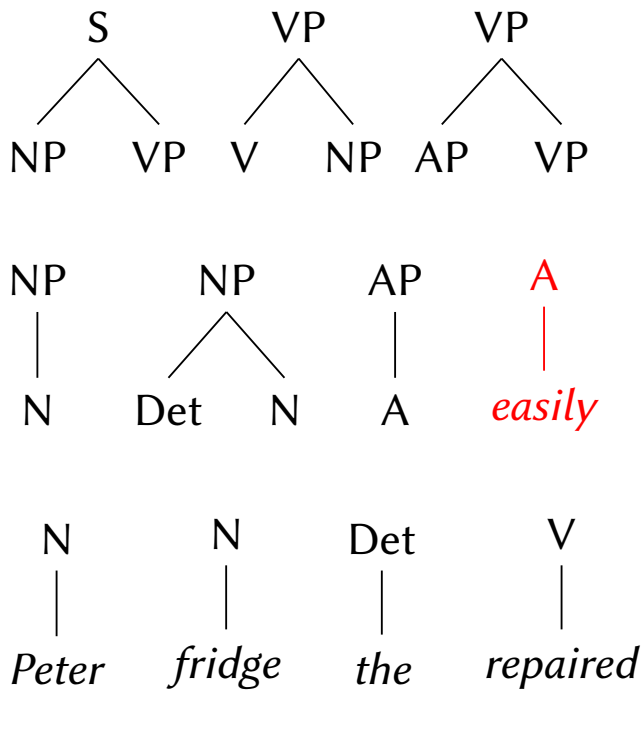
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

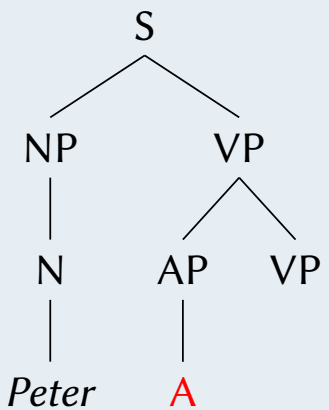
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



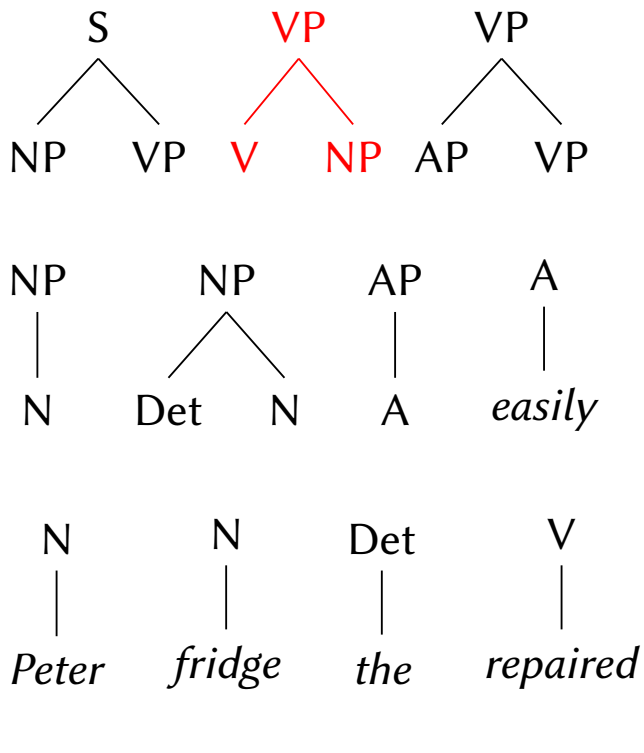
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

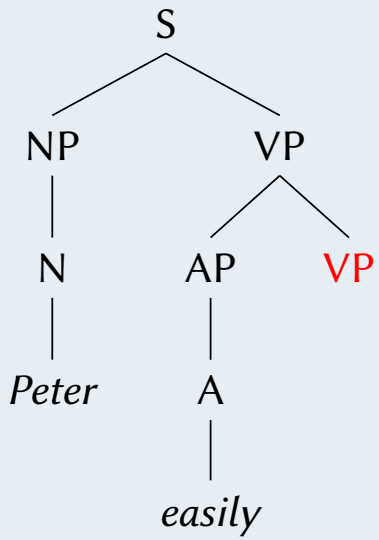
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



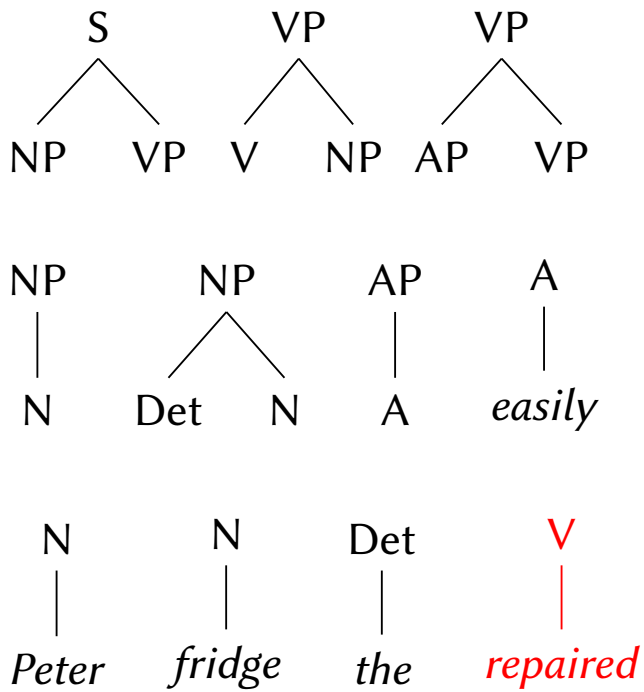
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

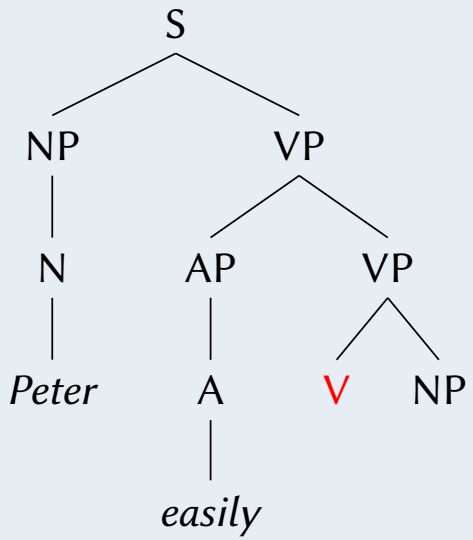
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



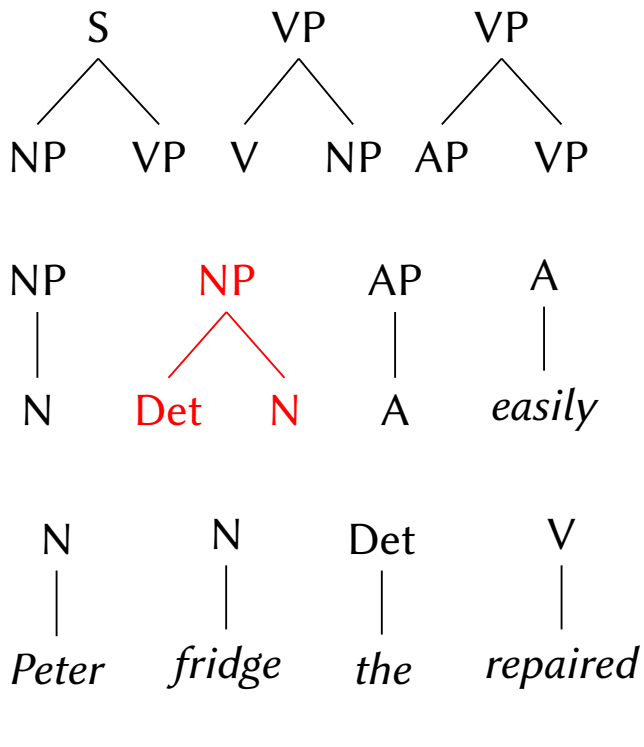
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

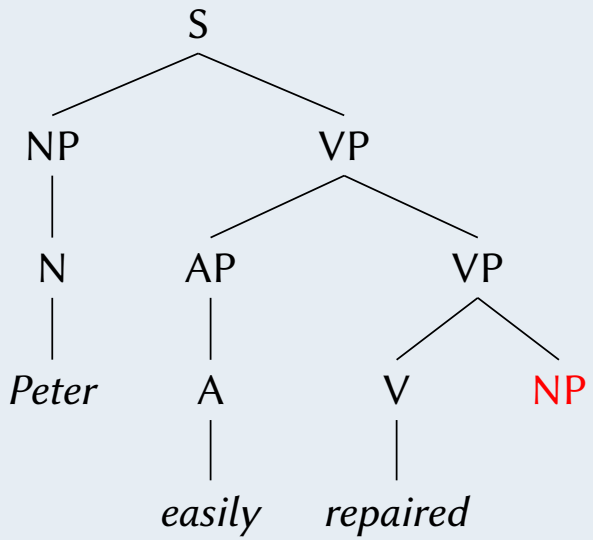
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



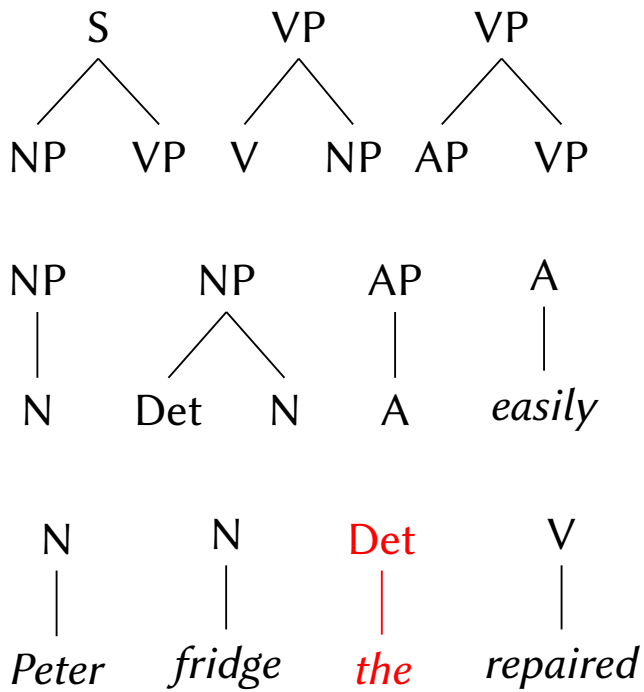
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

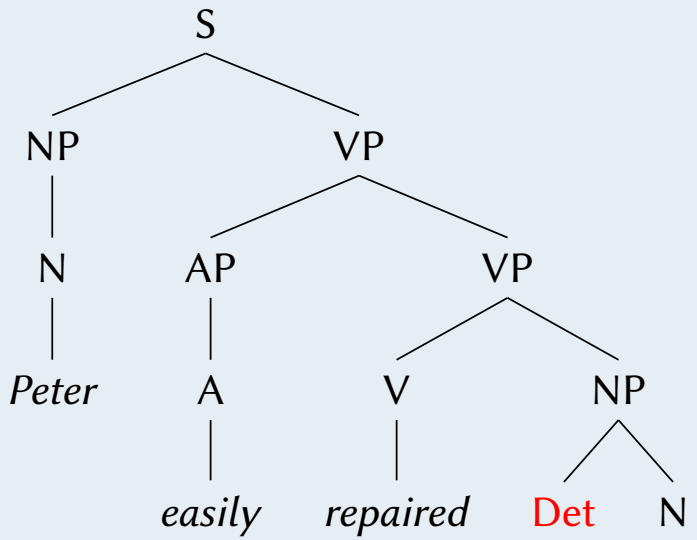
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



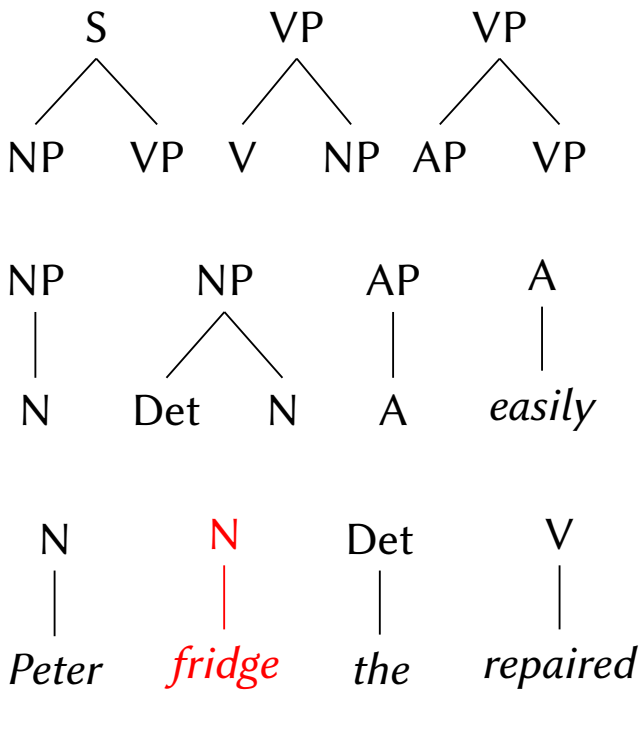
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

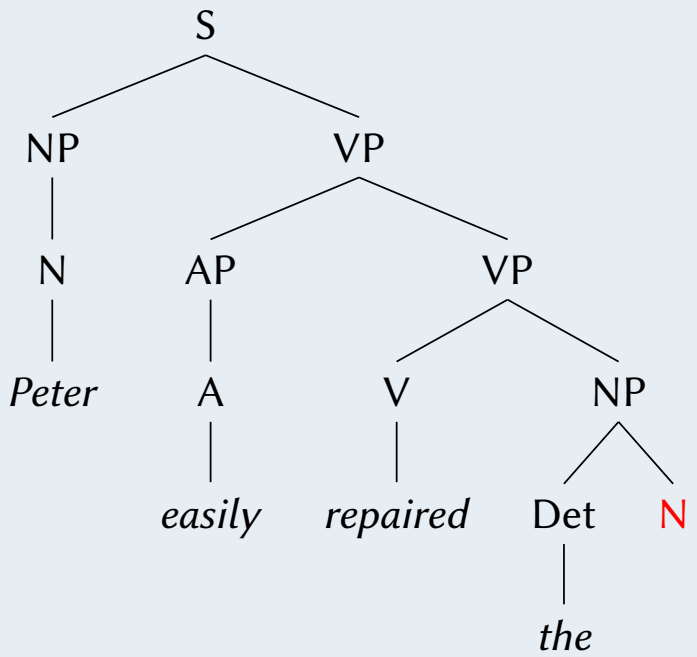
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

Example derivation:



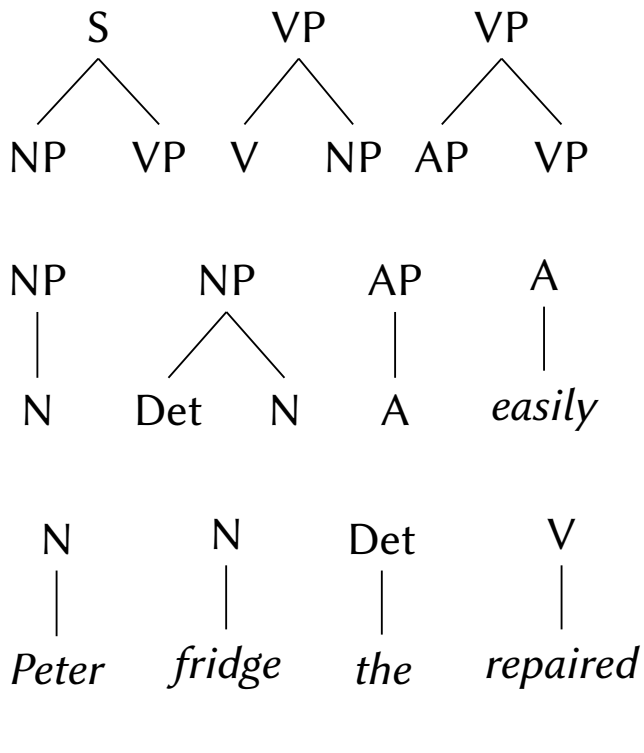
From CFG to TAG: Tree Substitution Grammar (TSG)

First step: turn strings into trees!

- tree rewriting
- **Substitution:** replace a non-terminal leaf with a tree

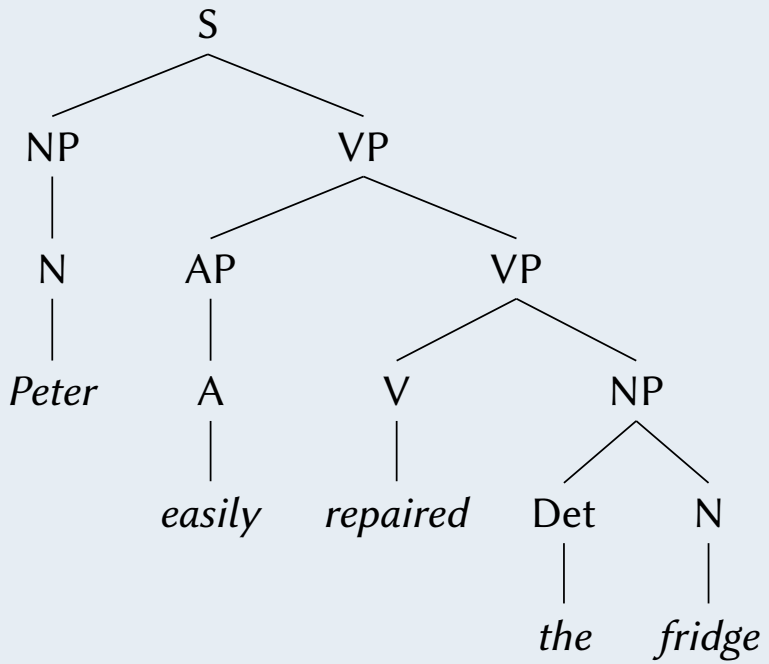
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



$\}$

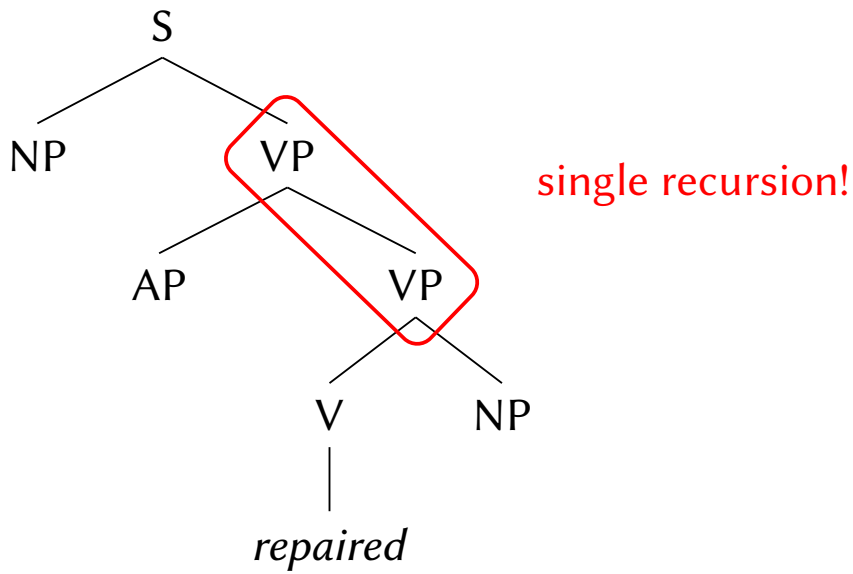
Example derivation:



From CFG to TAG: Tree Substitution Grammar (TSG)

TSG versus CFG:

- weakly equivalent (same string languages, but more tree languages)



- still no strong lexicalization of CFG, cross-serial dependencies etc.

Applications of TSG:

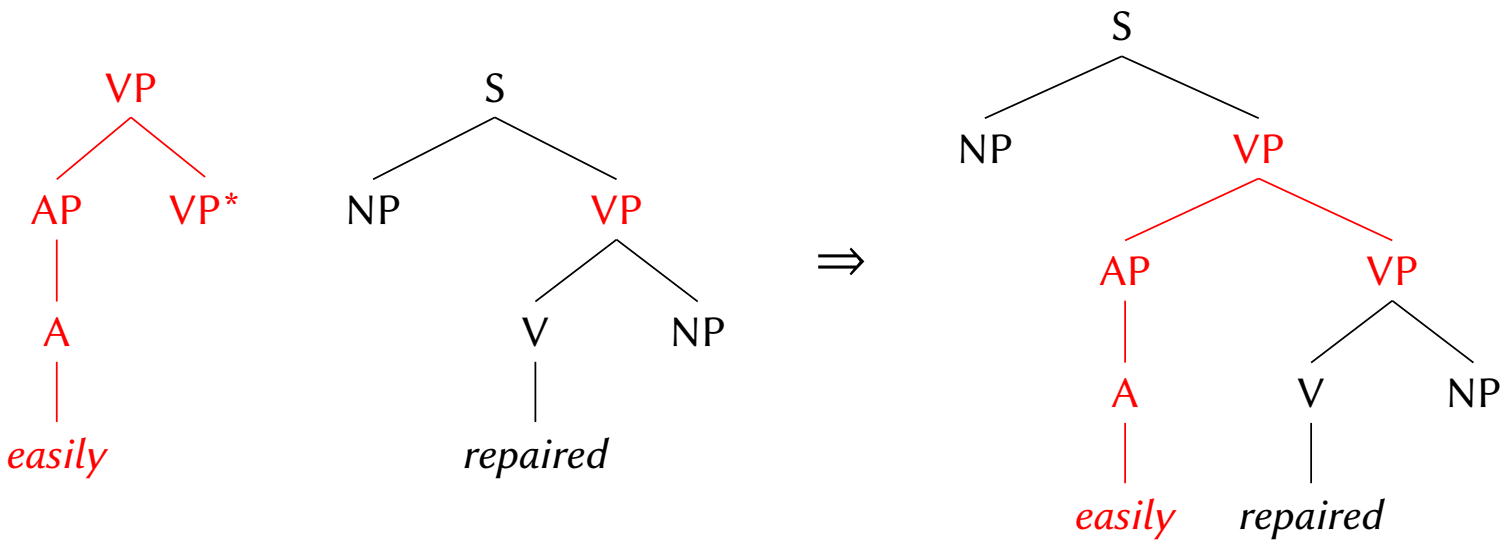
- Data-Oriented Parsing (DOP, Bod 2009)

Second step: add adjunction!

From CFG to TAG: Adding adjunction

Adjunction:

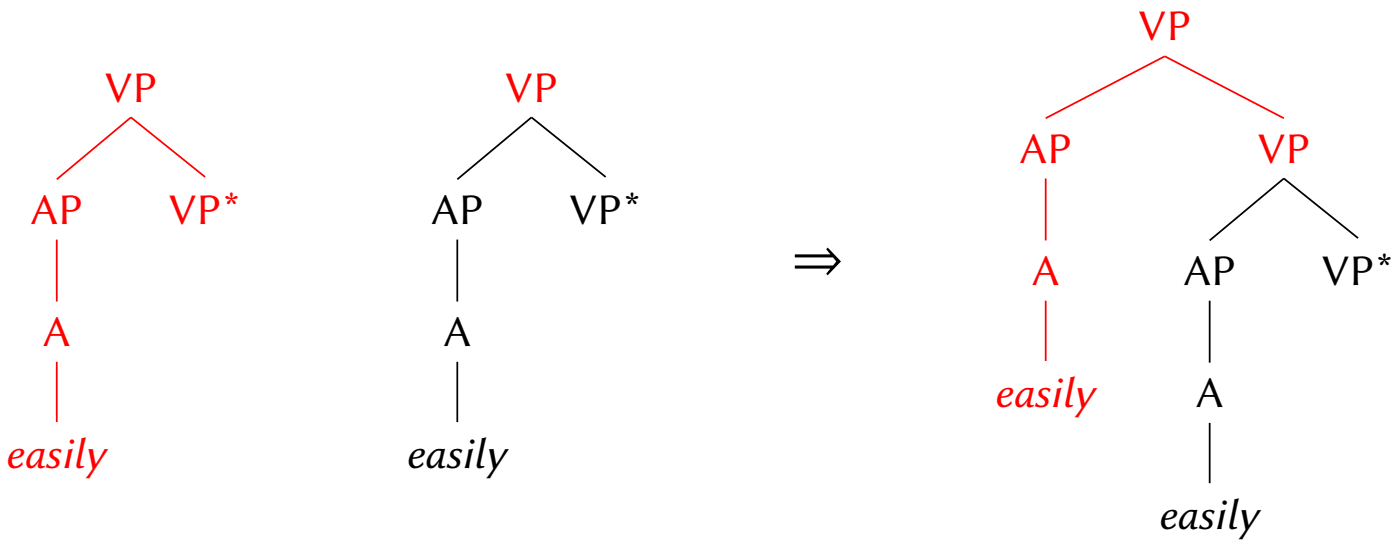
- replace a **non-terminal node** with an "auxiliary" tree
- put the subtree of the replaced node under the **footnode** (*)



From CFG to TAG: Adding adjunction

Adjunction:

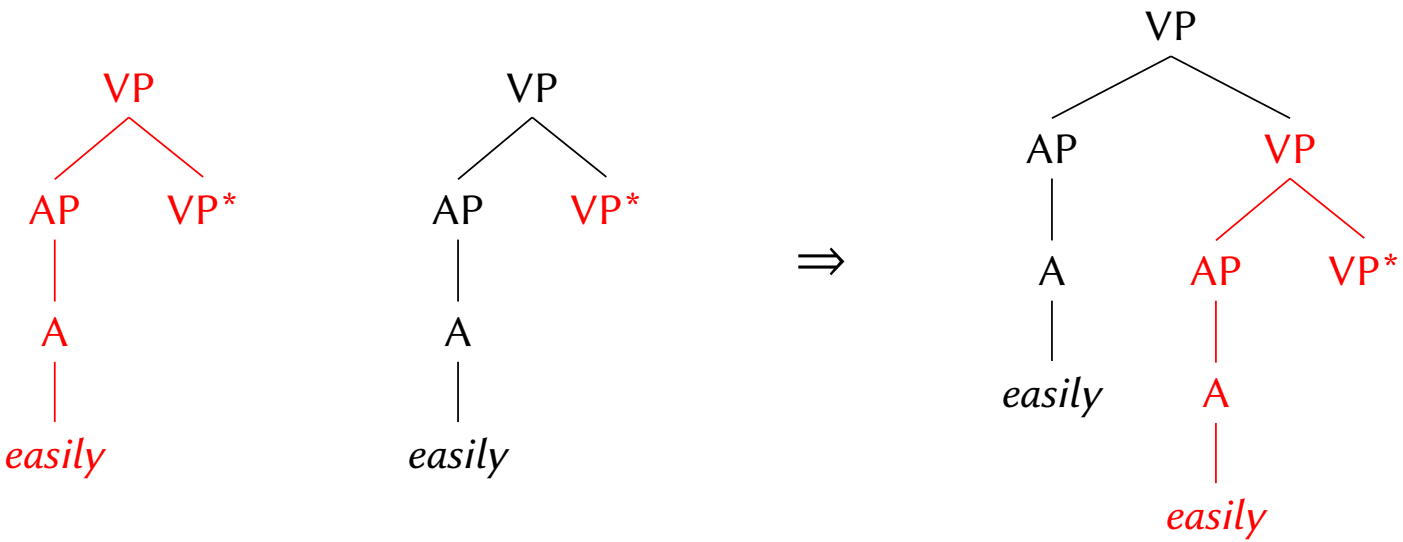
- replace a **non-terminal node** with an "auxiliary" tree
- put the subtree of the replaced node under the **footnode** (*)



From CFG to TAG: Adding adjunction

Adjunction:

- replace a **non-terminal node** with an "auxiliary" tree
- put the subtree of the replaced node under the **footnode** (*)



⇒ Adjunction at footnodes causes spurious ambiguities in derivations.

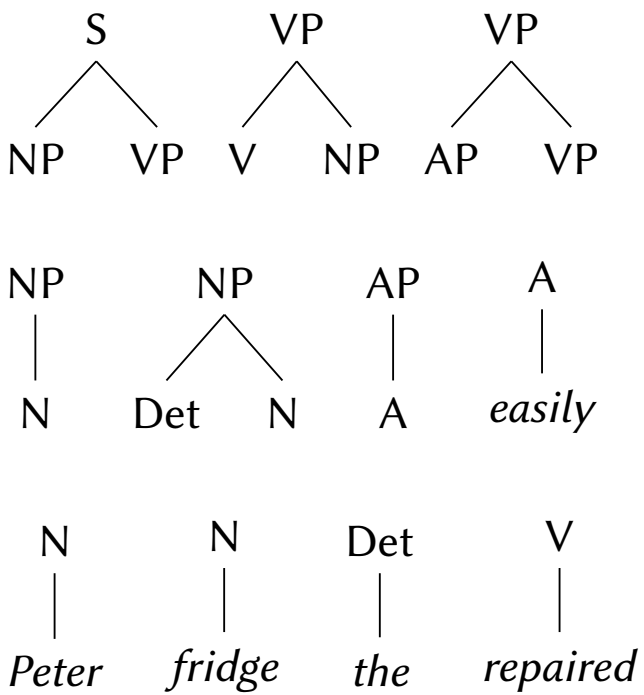
⇒ Therefore, this is usually forbidden.

From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree

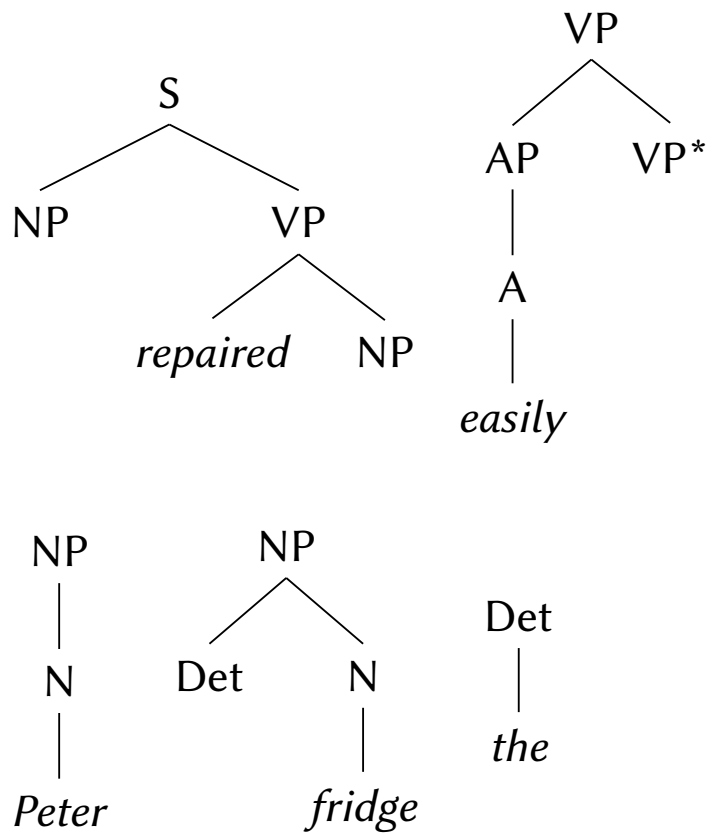
$G_{\text{TSG}} = \langle N, T, S, I \rangle$

$I = \{$



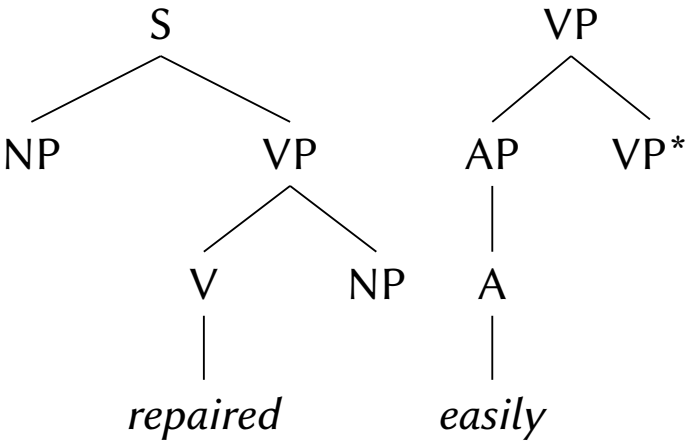
$\}$

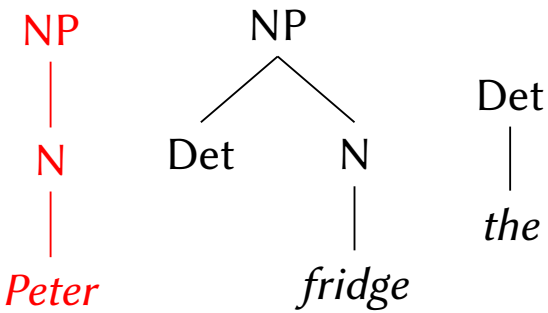
$\approx$



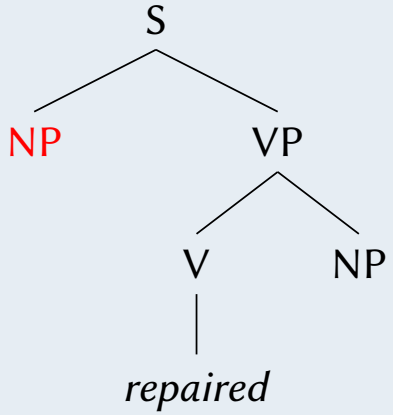
From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree



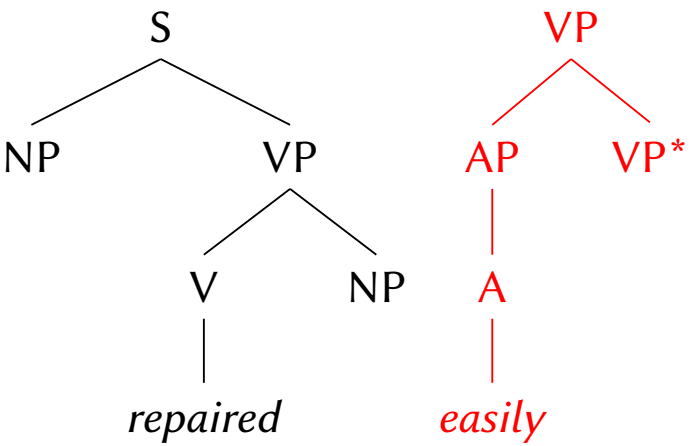


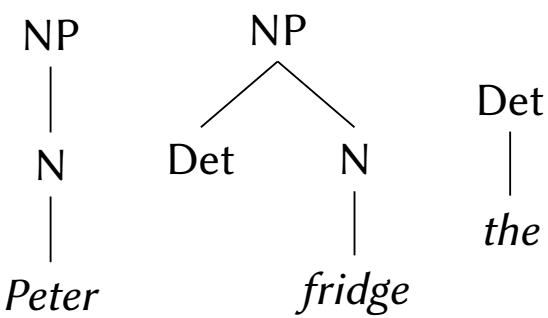
Example derivation:



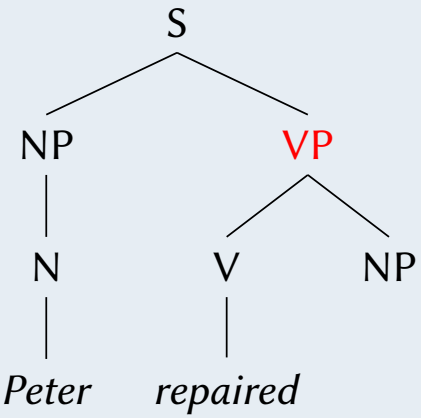
From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree



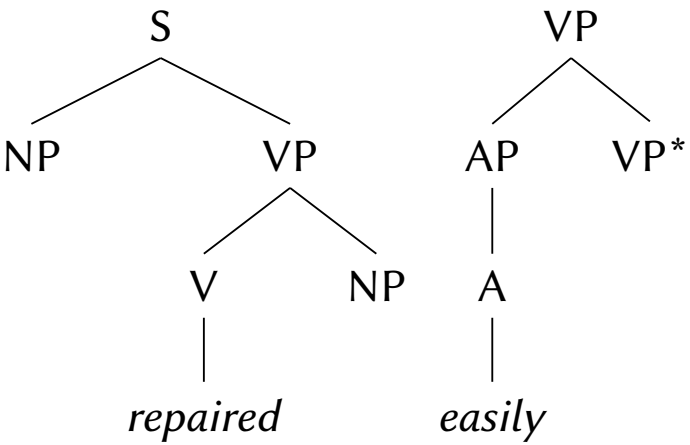


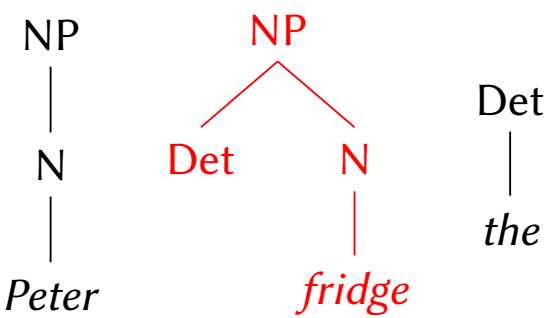
Example derivation:



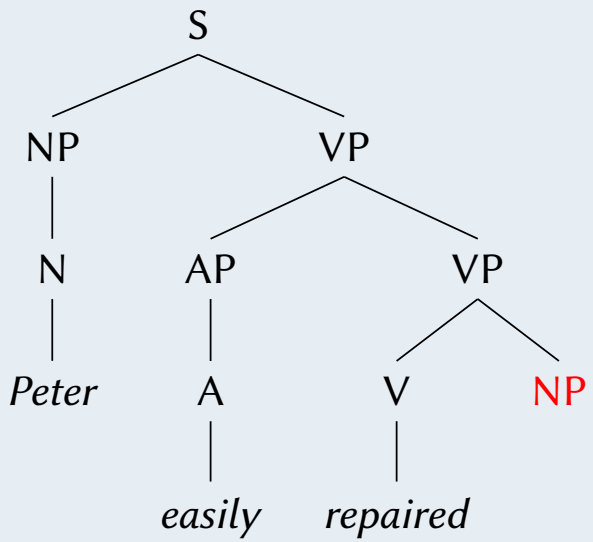
From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree



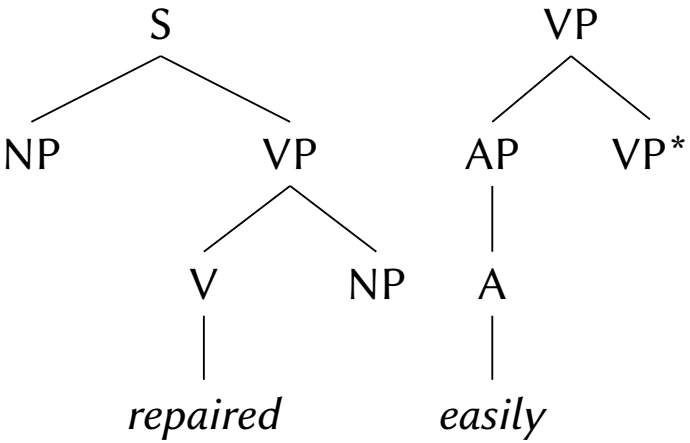


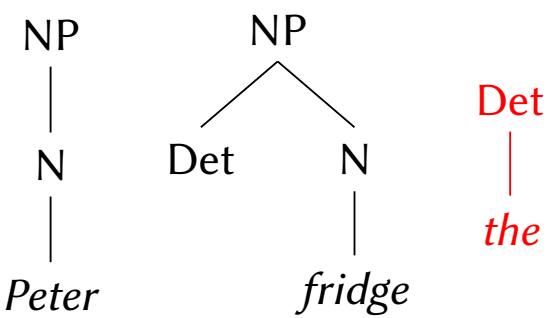
Example derivation:



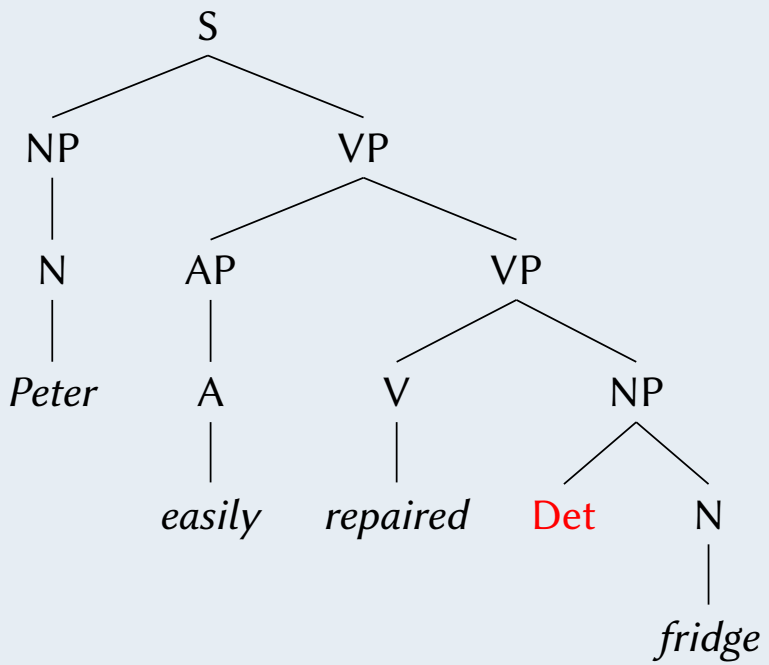
From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree



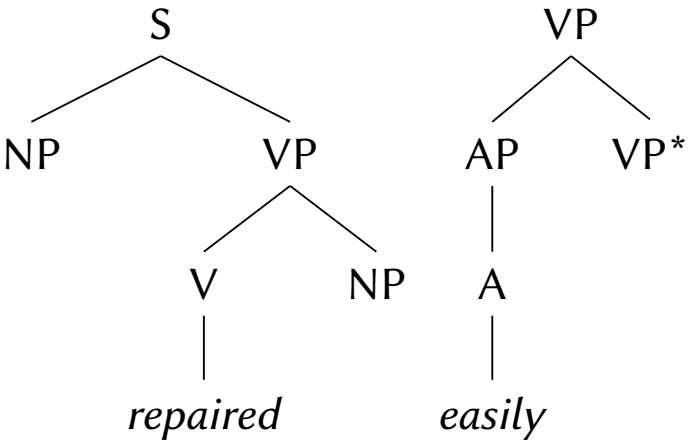


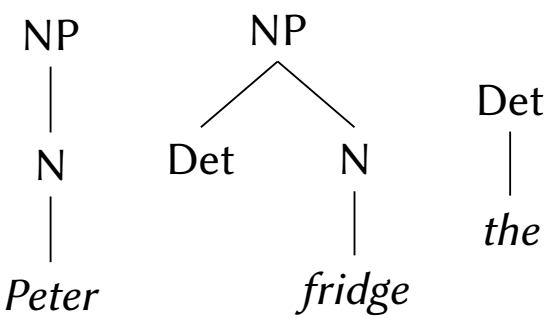
Example derivation:



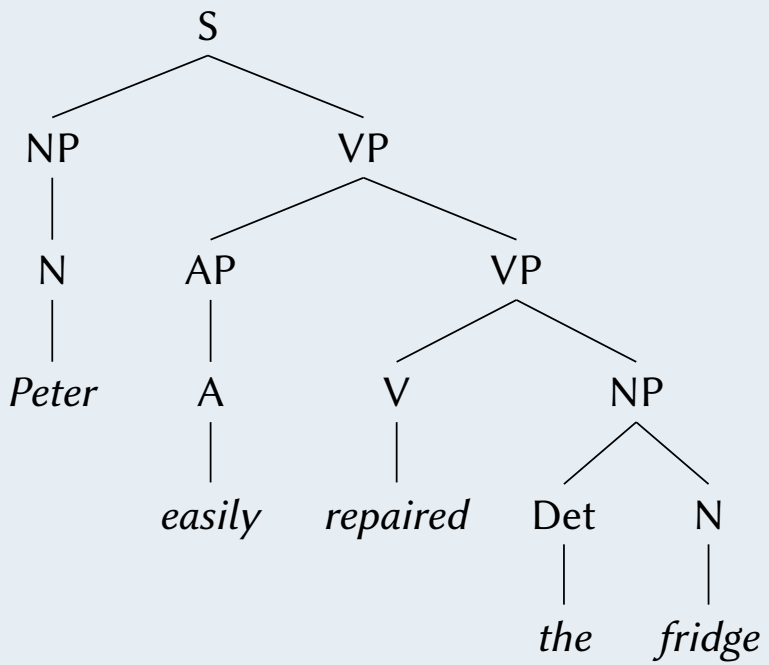
From CFG to TAG: Example with adjunction

- tree rewriting
- **Substitution:** replace a non-terminal **leaf** with a tree
- **Adjunction:** replace a non-terminal **node** with an "auxiliary" tree





Example derivation:



From CFG to TAG: Restrictions on adjunction (I)

Restrictions on the shape of auxiliary trees:

- The root node and the footnode must carry the same non-terminal.

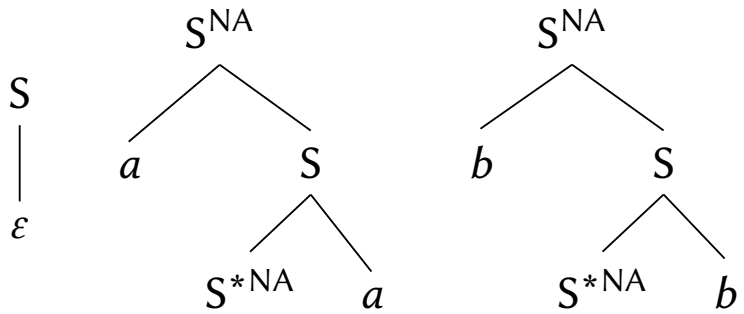
Specific adjunction constraints on target nodes:

- obligatory adjunction (OA): true/false
- null adjunction (NA): no adjoinable auxiliary tree
- selective adjunction (SA): a nonempty set of adjoinable auxiliary trees

Adjunction constraints are essential in generating non-context-free languages (e.g. the copy language $\{ww | w \in \{a, b\}^*\}$)!

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:

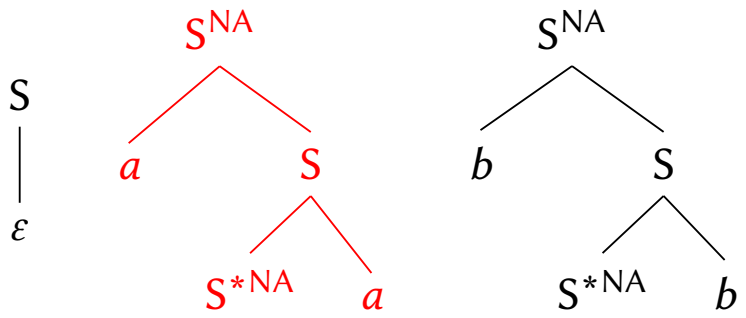


Example derivation of *abbabb*:

$\Rightarrow$ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:

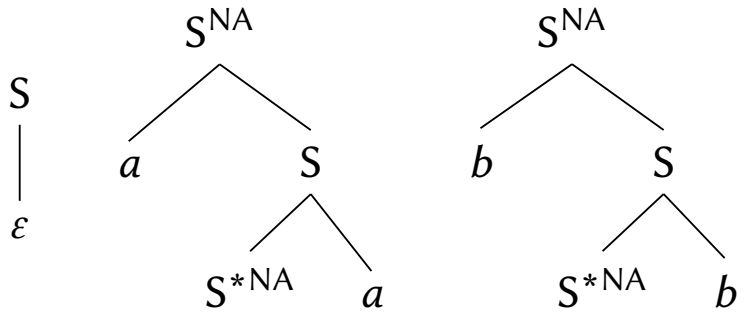


Example derivation of *abbabb*:

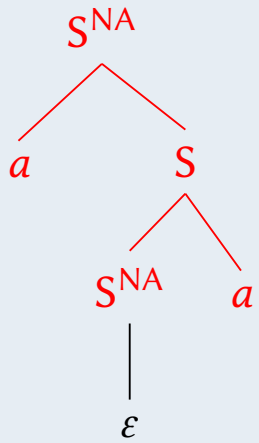
$\Rightarrow$ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



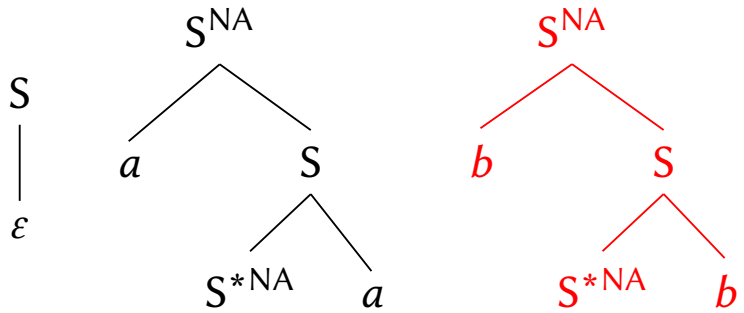
Example derivation of *abbabb*:



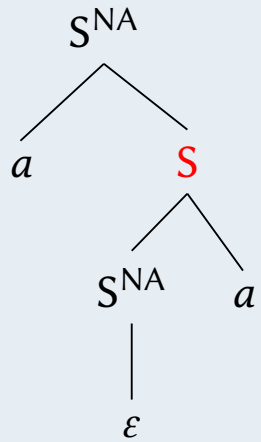
⇒ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



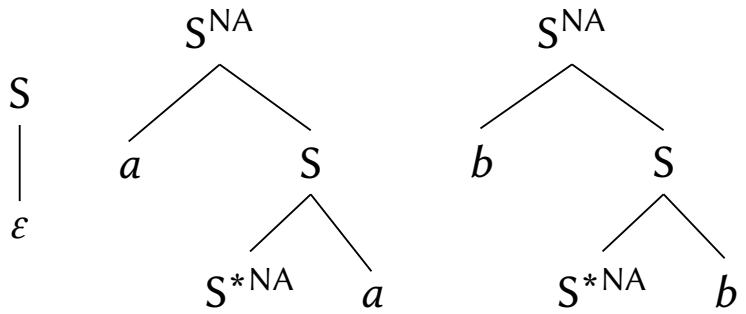
Example derivation of *abbabb*:



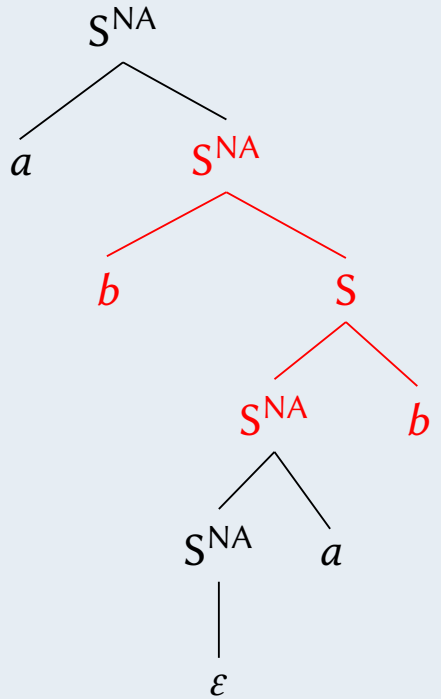
⇒ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



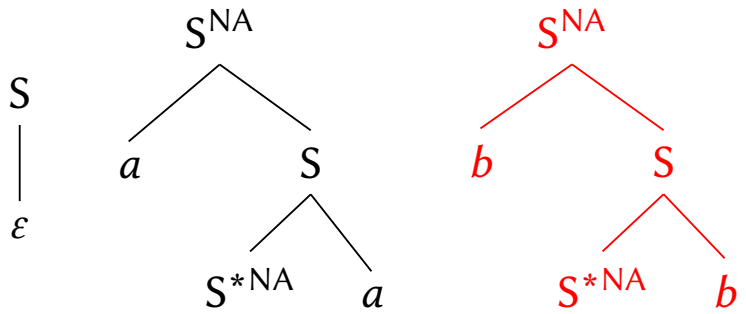
Example derivation of *abbabb*:



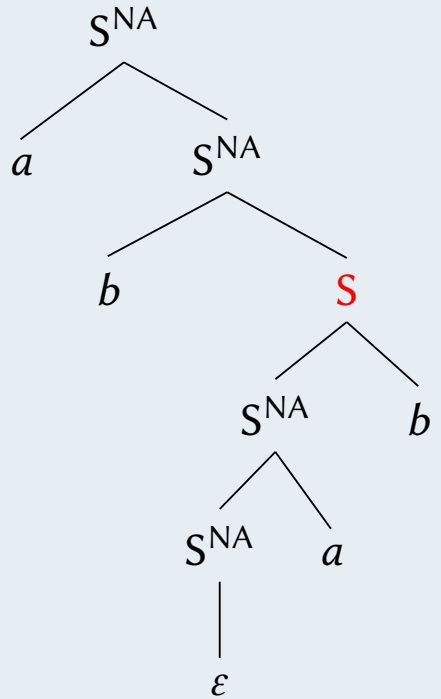
$\Rightarrow$ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



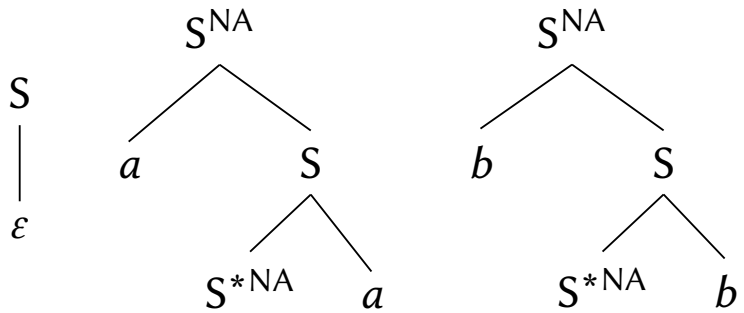
Example derivation of *abbabb*:



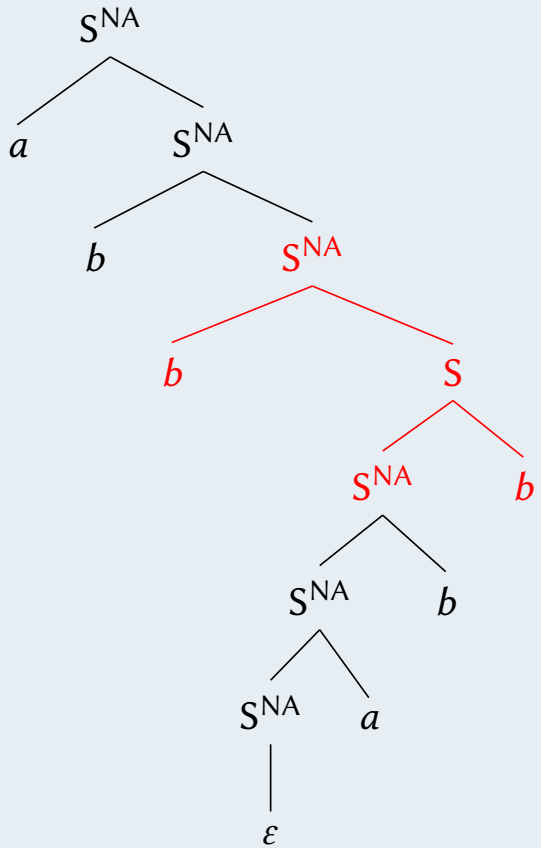
$\Rightarrow$ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Restrictions on adjunction (I)

Example grammar for the copy language $\{ww | w \in \{a, b\}^*\}$:



Example derivation of *abbabb*:



⇒ TAG = TSG + adjunction + adjunction constraints

From CFG to TAG: Tree-Adjoining Grammar

A **Tree Adjoining Grammar (TAG)** is a tuple $G = \langle N, T, I, A, O, C \rangle$:

- T and N are disjoint alphabets, the terminals and nonterminals,
- I is a finite set of **intial trees**, and
- A is a finite set of **auxiliary trees**.
- $O : \{v \mid v \text{ is a node in a tree in } I \cup A\} \to \{1, 0\}$ is a function, and
- $C : \{v \mid v \text{ is a node in a tree in } I \cup A\} \to P(A)$ is a function.

Let v be a node in $I \cup A$:

- **obligatory adjunction (OA):** $O(v) = 1$
- **null adjunction (NA):** $O(v) = 0$ and $C(v) = \emptyset$
- **selective adjunction (SA):** $O(v) = 0$ and $C(v) \neq \emptyset$ and $C(v) \neq A$

The trees in $I \cup A$ are called **elementary** trees.

From CFG to TAG: Tree-Adjoining Grammar

TAG is **mildly context-sensitive** (MCS, Joshi 1985)

- generates the context-free languages
- generates cross-serial dependencies (i.e. WW)
- constant growth (or semi linear, no a^{2^n})
- polynomial time parsing ($O(n^6)$) (Schabes 1990; Joshi & Schabes 1997; Kallmeyer 2010)

TAG can strongly lexicalize finitely ambiguous CFG. (Schabes 1990; Joshi & Schabes 1991)

- **Formally interesting:** a finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous).
- **Linguistically interesting:** syntactic properties of lexical items can be accounted for more directly.
- **Computationally interesting:** the search space during parsing can be delimited (grammar filtering).

Outline of today's course

Restricting TAG

Further adjunction constraints:

- no adjunction at the spine below the root node of auxiliary trees
  - off-spine TAG (osTAG, Swanson et al. (2013))

  $\Rightarrow$ WGC of CFG ($O(n^3)$)

  $\Rightarrow$ more compact grammars than CFG or TSG

  $\Rightarrow$ strongly lexicalizes CFG?

Restrictions on the shape of auxiliary trees:

- Footnodes are at the left or right edge of an ET.
  - Tree Insertion Grammar (TIG, Schabes & Waters (1995))
  - further constraint: no adjunction of left auxiliary trees to the spine of right auxiliary trees

  $\Rightarrow$ WGC of CFG ($O(n^3)$)

  $\Rightarrow$ more compact grammars than CFG (or TSG?)

  $\Rightarrow$ strongly lexicalizes CFG

MCS-alternatives to TAG and extensions

- Linear Indexed Grammar (LIG Gazdar 1988; Keller & Weir 1995)
- Head Grammar (HG Pollard 1984)
- Multicomponent TAG (MCTAG Seki et al. 1991)
- Minimalist Grammar (MG Stabler 1997)
- Combinatory Categorial Grammar (CCG Steedman 1984)
- Linear Context-Free Rewriting Systems (LCFRS Vijay-Shanker et al. 1987)

TAG, CCG (but not recent versions of CCG), LIG and HG are weakly equivalent. MCTAG and LCFRS subsume TAG, CCG, LIG and HG. (Kallmeyer 2010)

$\Rightarrow$ TAG cannot generate all MCSLs!

- $\{a^n b^n c^n d^n e^n \mid n \geq 1\}$, $\{www \mid w \in \{a, b\}^*\}$
- MIX := $\{w | w \in \{a, b, c\}^*, |w|_a = |w|_b = |w|_c\}$ (Bach 1988)
- SCR^{ind} := $\{\sigma(NP_1, \ldots, NP_m) V_m \ldots V_1 | m \geq 1$ and σ is a permutation$\}$ (Becker et al. 1992)

Outline of today's course

Summary & outlook

Summary

- motivation
- CFG → TSG → TSG+adjunction
  → TSG + adjunction + adjunction constraints = **TAG**

Summary & outlook

Summary

- motivation
- CFG → TSG → TSG+adjunction
  → TSG + adjunction + adjunction constraints = **TAG**

Tomorrow

- linguistic applications using LTAG
- the derivation tree
- subcategorization, extraction, modification
- adding feature structures

Outline of today's course

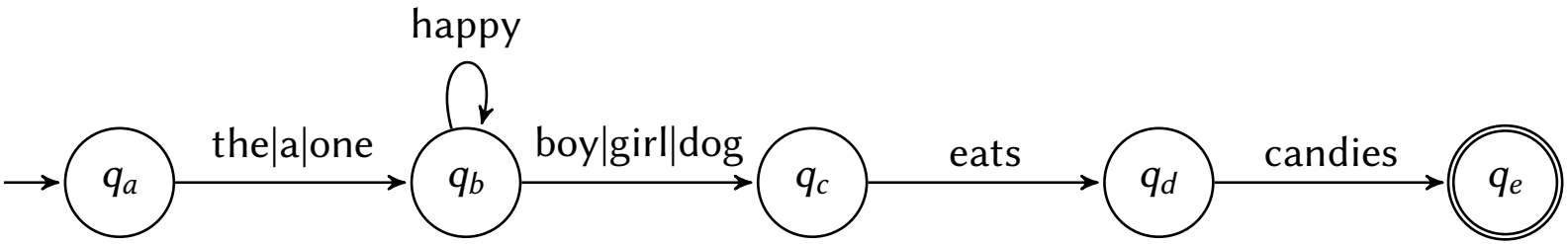
Appendix: NL is not regular

Hypothesis

All natural languages can be accepted by a finite state automaton (FSA).

FSA:

- finite set of states (including one start state and at least one end state)
- finite set of transitions between states
- On every transition, a word is read from the input.



Potential problems: recursion, constituency, long-distance dependencies . . . whenever we might need a storage.

How to **proof** the inadequacy of FSA on the level of string languages?

Appendix: NL is not regular

The case of nested dependency (Chomsky (1957)):

(5) a. a woman hired another woman
    b. a woman whom another woman hired hired another woman
    c. a woman whom another woman whom another woman hired hired hired another woman
    d. ...

Formal proof by contradiction (using closure properties and the Pumping Lemma):

- Every regular language satisfies the **Pumping Lemma**, hence the pattern wa^nz.
- homomorphism f: $f(\text{a woman}) = w$, $f(\text{whom another woman}) = a$, $f(\text{hired}) = b$, $f(\text{hired another woman}) = z$
  - wa^*b^*z is a regular language; and
  - f(English) $\cap$ $wa^*b^*z = wa^nb^nz$ should be regular as well.
- wa^nb^nz contradicts the Pumping Lemma for regular languages.

$\Rightarrow$ English is not regular!

back

Appendix: NL is not context-free

- a long time debate about the context-freeness of natural languages

Chomsky (1957):34

"Of course there are languages (in our general sense) that cannot be described in terms of phrase structure, but I do not know whether or not English is itself literally outside the range of such analysis."

- several wrong arguments (see Pullum & Gazdar 1982), e.g.:

Bresnan (1978):37–38

"in many cases the number of a verb agrees with that of a noun phrase at some distance from it ... this type of syntactic dependency can extend as memory or patience permits ... the distant type of agreement ... cannot be adequately described even by context-sensitive phrase-structure rules, for the possible context is not correctly describable as a finite string of phrases."

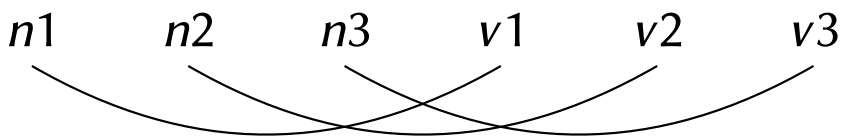
- right proof techniques: pumping lemma and closure properties
- What is a non context-free phenomenon in natural languages?

Appendix: NL is not context-free

A serious try: Syntax of Dutch (Bresnan et al. 1982)

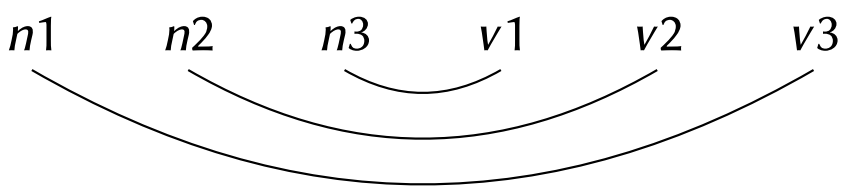
(6) dat Jan Piet de kinderen zag helpen zwemmen.
that Jan Piet the children saw help swim
'that Jan saw Piet helping the children to swim.'

Linguistic dependencies are cross-serial:



However: No reflection on the surface, i. e. in the string language!

⇒ In principle the string can be generated by a CFG, even though the dependencies will get lost.



Appendix: NL is not context-free

Another try by Culy (1985): Duplication in the morphology of Bambara

(7)	a.	wulu	*'dog'*
	b.	wulu-lela	*'dog watcher'*
	c.	wulu-lela-nyinila	*'dog watcher hunter'*
	d.	wulu-o-wulu	*'whatever dog'*
	e.	wulu-lela-o-wulu-lela	*'whatever dog watcher'*
	f.	wulu-lela-nyinila-o-wulu-lela-nyinila	*'whatever dog watcher hunter'*

Pattern: $a^n b^m a^n b^m$ or ww (copy language)

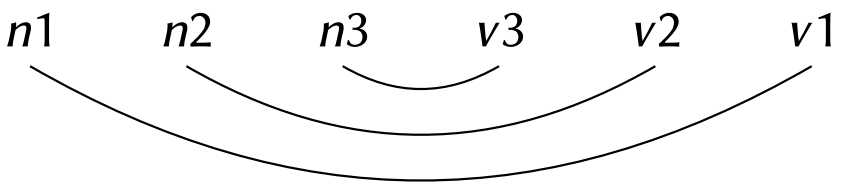
$\Rightarrow$ not context-free!

However: proof for morphology, not for syntax (the string language)!

Appendix: NL is not context-free

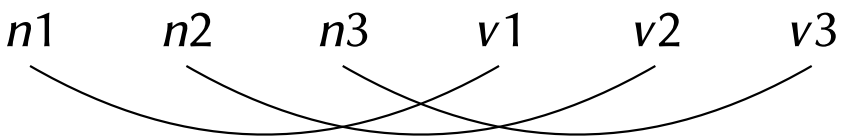
- German: **nested dependency** (subordinate clauses)

(8) (dass) er die Kinder dem Hans das Haus streichen helfen ließ
(that) he the children the Hans the house paint help let
'(that) he let the children help Hans paint the house'



- Schwyzerdütsch: **cross-serial dependency**

(9) …mer d'chind em Hans es huus lönd hälfe aastriiche
…we children.ACC the Hans.DAT the house.ACC let help paint
'…we let the children help Hans paint the house'



Appendix: NL is not context-free

Proof by Shieber (1985):

(10) Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.

- homomorphism f:

  $f(\text{d'chind}) = a$ $f(\text{em Hans}) = b$ $f(\text{lönd}) = c$ $f(\text{hälfe}) = d$
  $f(\text{Jan säit das mer}) = w$ $f(\text{es huus}) = x$ $f(\text{aastriiche}) = y$
  $f(s) = z$

- $f(\text{Schwyzerdütsch}) \cap wa^*b^*xc^*d^*y = wa^mb^nxc^md^ny$
  - CF languages are closed under intersection with regular languages
  - $wa^*b^*xc^*d^*y$ is regular
  - by Pumping Lemma: $wa^mb^nxc^md^ny$ is not regular

$\Rightarrow$ Schwyzerdütsch is not context-free

back

Appendix: Mildly context sensitive languages

Appendix: NL is not mildly context-sensitive?

1. A set L of languages is mildly context-sensitive iff
   a. L contains all context-free languages
   b. L can describe cross-serial dependencies: there is an $n \geq 2$ such that $\{w^k \mid w \in (V_T)^*\} \in L$ for all $k \geq n$
   c. the languages in L are polynomially parseable, i.e., $L \subset$ PTIME
   d. the languages in L have the constant growth property
2. A formalism F is mildly context-sensitive iff the set $\{L \mid L = L(G)$ for some $G \in F\}$ is mildly context-sensitive.

- constant growth property: if we order the words of a language according to their length, then the length grows in a linear way
- there is a finite figure n, that limits the maximum number of instantiations of cross serial dependencies in a sentence of L

Appendix: NL is not mildly context-sensitive?

Evidence brought forward against semi-linearity:

- case stacking ("Suffixaufnahme") in Old Georgian (Michaelis & Kracht (1997))
- Chinese number-names (Radzinski 1991)
- coordination in Dutch (Groenink 1997)
- relativized predicates in Yoruba (Kobele 2006)

back

Appendix: Indexed Grammar (IG)

Indexed Grammar (IG, Aho (1968)):

- $IG = \langle T, N, I, S, R \rangle$, where I = a set of indices
  - context-free rules extended with indices $\Rightarrow$ a kind of stack
  - indices can appear only on non-terminals, e.g. $A[ijkiij]$
  - adding or removing indices on the right-hand side of the rule
  - the index-string can be infinite long
- two kinds of rules in R:

(i) "push and copy": $A[..] \rightarrow B[i..]$ e.g. $\ldots A[jk] \ldots \Rightarrow \ldots B[ijk] \ldots$

(ii) "pop and copy": $A[i..] \rightarrow B[..]$ e.g. $\ldots A[ijk] \ldots \Rightarrow \ldots B[jk] \ldots$

Linear Indexed Grammar (LIG, Gazdar (1988)): The stack may be copied to at most one non-terminal per rule.

~~$A[..] \rightarrow B[..]C[..]$~~

Appendix: Indexed Grammar (IG)

LIG for the language $a^n b^n c^n$:

- $G = \langle T, N, I, S, R \rangle$, where

$$T = \{a, b, c\},\ N = \{S, Q\},\ I = \{i\}$$
$$R = \{\ S[..] \to aS[i..]c, \quad S[..] \to Q[..],$$
$$Q[i..] \to Q[..]b, \quad Q[\,] \to \epsilon \qquad \}$$

- example derivation:
  - $S[\,] \Rightarrow aS[i]c \Rightarrow aaS[ii]cc \Rightarrow aaaS[iii]ccc \Rightarrow aaaQ[iii]ccc \Rightarrow aaaQ[ii]bccc \Rightarrow aaaQ[i]bbccc \Rightarrow aaaQ[\,]bbbccc \Rightarrow aaabbbccc$
  - $S[\,] \overset{*}{\Rightarrow} a^n Q[i^n] c^n \overset{*}{\Rightarrow} a^n b^n c^n$

LIG for the language $\{ww | w \in \{a, b\}^*\}$:

- Try yourself!

back

References

Aho, Alfred V. 1968. Indexed Grammars. An extension of Context-Free Grammars. **Journal of the ACM** 15(4). 647–671. http://doi.acm.org/10.1145/321479.321488.

Bach, Emmon. 1988. Categorial grammars as theories of language. In Richard T. Oehrle, Emmon Bach & Deirdre Wheeler (eds.), **Categorial grammars and natural language structures** (Studies in Linguistics and Philosophy 32), 17–34. Dordrecht, Holland: D. Reidel Publishing Company.

Becker, Tilman, Owen Rambow & Michael Niv. 1992. The derivationel generative power of formal systems or scrambling is beyond LCFRS. IRCS Technical Report IRCS-92-38 Institute for Research in Cognitive Science, University of Pennsylvania Philadelphia, PA.

Bod, Rens. 2009. From exemplar to grammar: A probabilistic analogy-based model of language learning. **Cognitive Science** 33(5). 752–793. http://dx.doi.org/10.1111/j.1551-6709.2009.01031.x.

Bresnan, Joan. 1978. A realistic transformational grammar. In Morris Halle, Joan Bresnan & George A. Miller (eds.), **Linguistic theory and psychological reality**, 1–59. Cambridge, MA: The MIT Press.

Bresnan, Joan, Ronald M. Kaplan, Stanley Peters & Annie Zaenen. 1982. Cross-serial dependencies in dutch. **Linguistic Inquiry** 13(4). 613–634.

Chomsky, Noam. 1956. Three models for the description of language. **IRE Transactions on Information Theory** 2. 113–124.

Chomsky, Noam. 1957. **Syntactic structures**. Den Haag: Mouton.

References (cont.)

Chomsky, Noam & Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. In P. Braffort & D. Hirschberg (eds.), **Computer programming and formal systems** (Studies in Logic and the Foundations of Mathematics 35), 118–161. Elsevier.

Culy, Christopher. 1985. The complexity of the vocabulary of Bambara. **Linguistics and Philosophy** 8(3). 345–351. http://www.jstor.org/stable/25001211.

Dowty, David R. 1979. **Word meaning and Montague Grammar**. Dordrecht: D. Reidel Publishing Company. Reprinted 1991 by Kluwer Academic Publishers.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In Uwe Reyle & Christian Rohrer (eds.), **Natural language parsing and linguistic theories** (Studies in Linguistics and Philosophy 35), 69–94. Dordrecht: D. Reidel Publishing Company.

Greibach, Sheila A. 1965. A New Normal-Form Theorem for Context-Free Phrase Structure Grammars. **Journal of the ACM** .

Groenink, Annius V. 1997. Mild context-sensitivity and tuple-based generalizations of context-grammar. **Linguistics and Philosophy** 20(6). 607–636. doi:10.1023/A:1005376413354. http://dx.doi.org/10.1023/A%3A1005376413354.

Hausser, Roland. 1992. Complexity in Left-Associative Grammar. **Theoretical Computer Science** 106(2). 283–308. http://www.sciencedirect.com/science/article/pii/030439759290253C.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions. In David Dowty, Lauri Karttunen & Arnold Zwicky (eds.), **Natural language parsing**, 206–250. Cambridge University Press.

References (cont.)

Joshi, Aravind K. & Yves Schabes. 1991. Tree-Adjoining Grammars and lexicalized grammars. Tech. Rep. MS-CIS-91-22 Department of Computer and Information Science, University of Pennsylvania. http://repository.upenn.edu/cis_reports/445/.

Joshi, Aravind K. & Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg & A. Salomaa (eds.), **Handbook of formal languages**, vol. 3, 69–124. Berlin, New York: Springer.

Kallmeyer, Laura. 2010. **Parsing beyond Context-Free Grammars**. Berlin: Springer.

Keller, Bill & David J. Weir. 1995. A tractable extension of Linear Indexed Grammars. In **Eacl**, 75–82.

Kobele, Gregory M. 2006. **Generating copies: An investigation into structural identity in language and grammar**. Los Angeles: University of California Dissertation. http://home.uchicago.edu/~gkobele/files/Kobele06GeneratingCopies.pdf.

Michaelis, Jens & Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In Christian Retoré (ed.), **Logical aspects of computational linguistics** (Lecture Notes in Computer Science 1328), 329–345. Berlin: Springer. doi:10.1007/BFb0052165. http://dx.doi.org/10.1007/BFb0052165.

Pollard, Carl J. 1984. **GPSGs, Head Grammar, and natural language**. Stanford, CA: Stanford University Dissertation.

Pullum, Geoffrey K. & Gerald Gazdar. 1982. Natural languages and context-free languages. **Linguistics and Philosophy** 4(4). 471–504. http://www.jstor.org/stable/25001071.

Radzinski, Daniel. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. **Computational Linguistics** 17. 277–299.

References (cont.)

Rosenberg, Arnold L. 1967. Real-time definable languages. **Journal of the Association for Computing Machinery** 14(4). 645–662.

Schabes, Yves. 1990. **Mathematical and computational aspects of lexicalized grammars**: University of Pennsylvania dissertation.

Schabes, Yves & Richard C. Waters. 1995. Tree Insertion Grammar: A cubic-time parsable formalism that lexicalizes Context-Free Grammar without changing the trees produced. **Computational Linguistics** 21(4). 479–513.

Seki, Hiroyuki, Takahashi Matsumura, Mamoru Fujii & Tadao Kasami. 1991. On multiple context-free grammars. **Theoretical Computer Science** 88(2). 191–229.

Shieber, Stuart. 1985. Evidence against the context-freeness of natural language. **Linguistics and Philosophy** 8. 333–343.

Stabler, Edward. 1997. Derivational minimalism. In Christian Retoré (ed.), **Logical aspects of computational linguistics** (Lecture Notes in Computer Science 1328), 68–95. New York: Springer.

Steedman, Mark. 1984. A categorial theory of intersecting dependencies in Dutch infinitival complements. In Wim de Geest & Yvan Putseys (eds.), **Sentential complementation**, 215–226. Foris, Dordrecht.

Swanson, Ben, Elif Yamangil, Eugene Charniak & Stuart Shieber. 2013. A context free TAG variant. In **Proceedings of the 51st annual meeting of the Association for Computational Linguistics**, 302–310. Sofia, Bulgaria. `http://www.aclweb.org/anthology/P13-1030`.

References (cont.)

Vijay-Shanker, K., David J. Weir & Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In **Proceedings of acl**, .

Wurm, Christian. 2012. Regular Growth Automata: Properties of a class of finitely induced infinite machines. In Makoto Kanazawa, Markus Kracht, Hiroyuki Seki & Andras Kornai (eds.), **Proceedings of the 12th conference on the mathematics of language (MOL 2012)** (LNCS 6878), 192–208. Berlin: Springer.