Synchronous Grammars

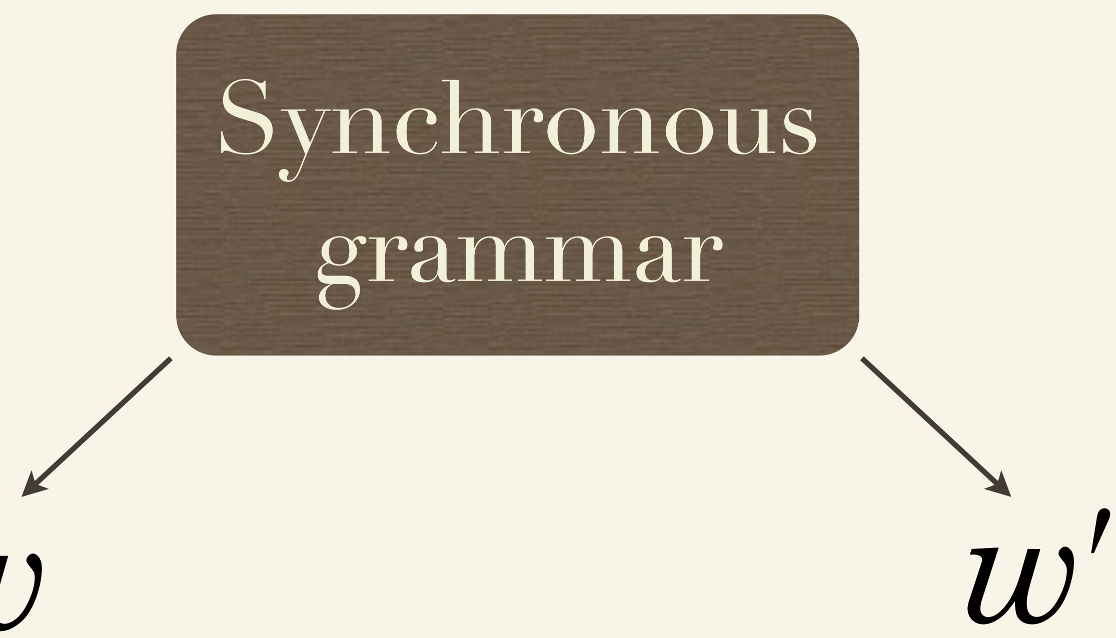
David Chiang

TAG+11

26 Sep 2012

Synchronous grammars

are a way of simultaneously generating
pairs of recursively related strings (or trees)



Synchronous grammars

were originally invented for
programming language compilation

Synchronous grammar

```
for i := 1 to 10 do
begin
    n := n + i
end
```

```
mov ax, 1
loop: add bx, ax
cmp ax, 10
jle loop
```

Synchronous grammars

have been used for syntax-based machine translation

Synchronous grammar

I open the box

watashi wa hako wo akemasu

Synchronous grammars

have been proposed as a way of doing semantic interpretation

Synchronous grammar

I open the box

$open'(me', box')$

Synchronous grammars

can do much fancier transformations
than finite-state methods

The boy stated that the student said that the teacher danced

shoonen ga gakusei ga sensei ga odotta to itta to hanashita

boy student teacher danced that said that stated

Synchronous grammars

can do much fancier transformations
than finite-state methods

…that John saw Peter help the children swim

…dat Jan Piet de kinderen zag helpen zwemmen

John Peter the children saw help swim

Overview

- Definitions
- Properties
- Algorithms
- Extensions

Definitions

Synchronous CFGs

$S \rightarrow NP\ \ VP$

$NP \rightarrow I$

$NP \rightarrow \text{the box}$

$VP \rightarrow V\ \ NP$

$V \rightarrow \text{open}$

$S \rightarrow NP\ \ VP$

$NP \rightarrow \text{watashi wa}$

$NP \rightarrow \text{hako wo}$

$VP \rightarrow NP\ \ V$

$V \rightarrow \text{akemasu}$

Synchronous CFGs

$$S \rightarrow NP_1\ VP_2,\ NP_1\ VP_2$$

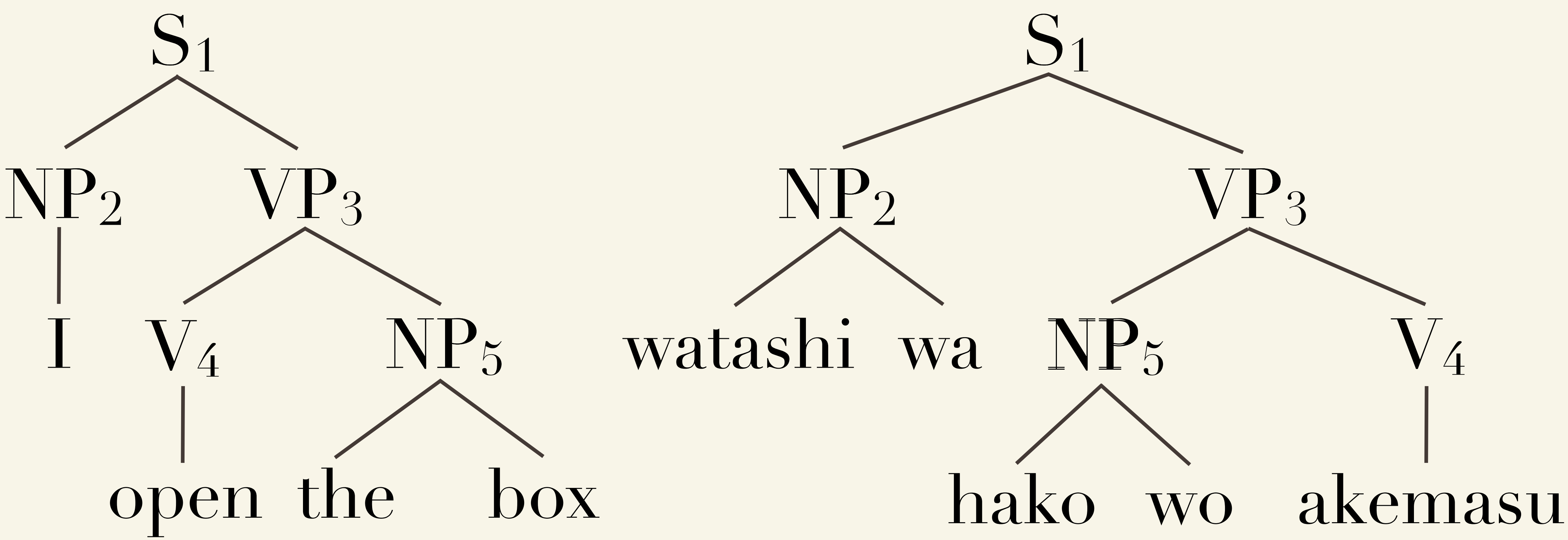
$$NP \rightarrow \text{I, watashi wa}$$

$$NP \rightarrow \text{the box, hako wo}$$

$$VP \rightarrow V_1\ NP_2,\ NP_2\ V_1$$

$$V \rightarrow \text{open, akemasu}$$

Synchronous CFGs



Other notations

$$VP \rightarrow (V_1\ NP_2\ ,\ NP_2\ V_1)$$

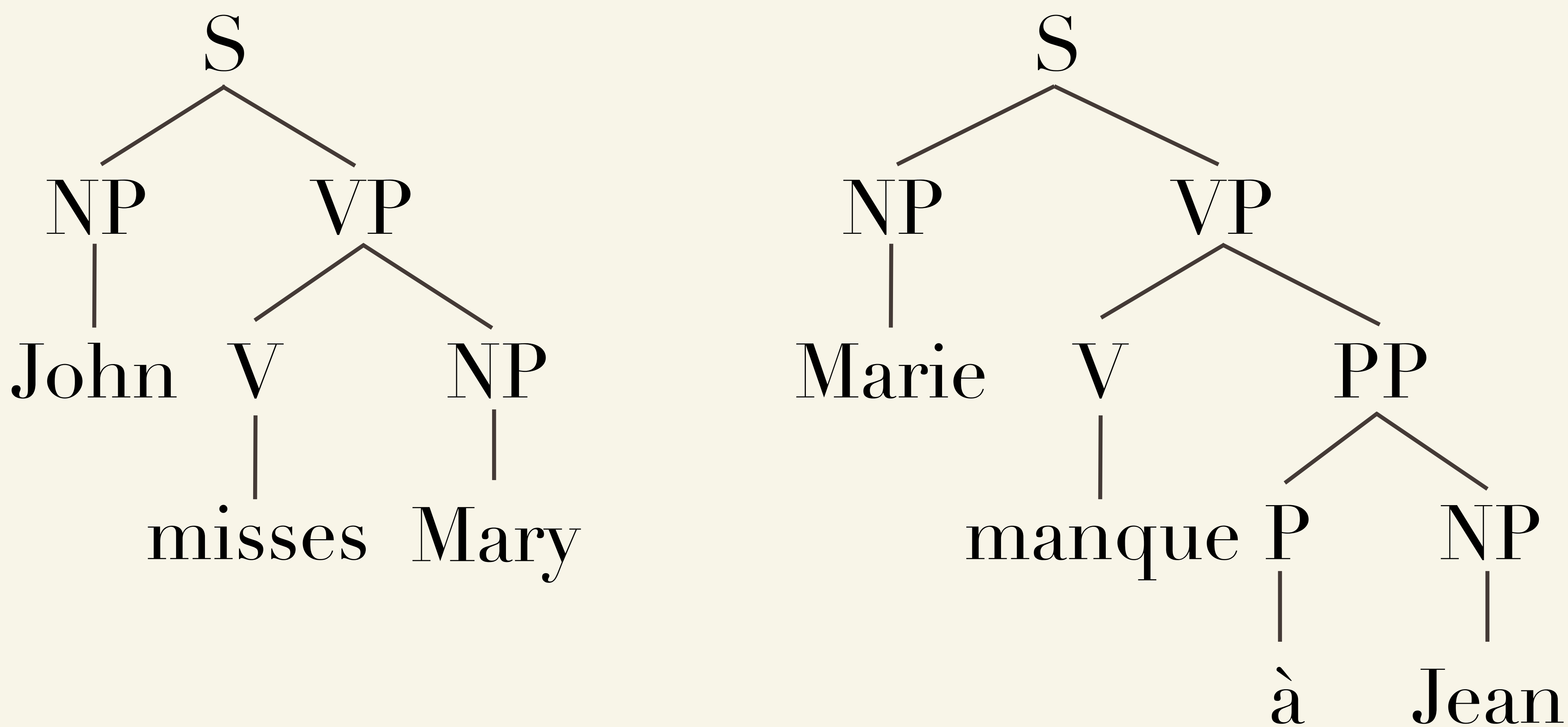
Syntax directed translation schema (Aho and Ullman; Lewis and Stearns)

$$(VP \rightarrow V_1\ NP_2\ ,\ VP \rightarrow NP_2\ V_1)$$

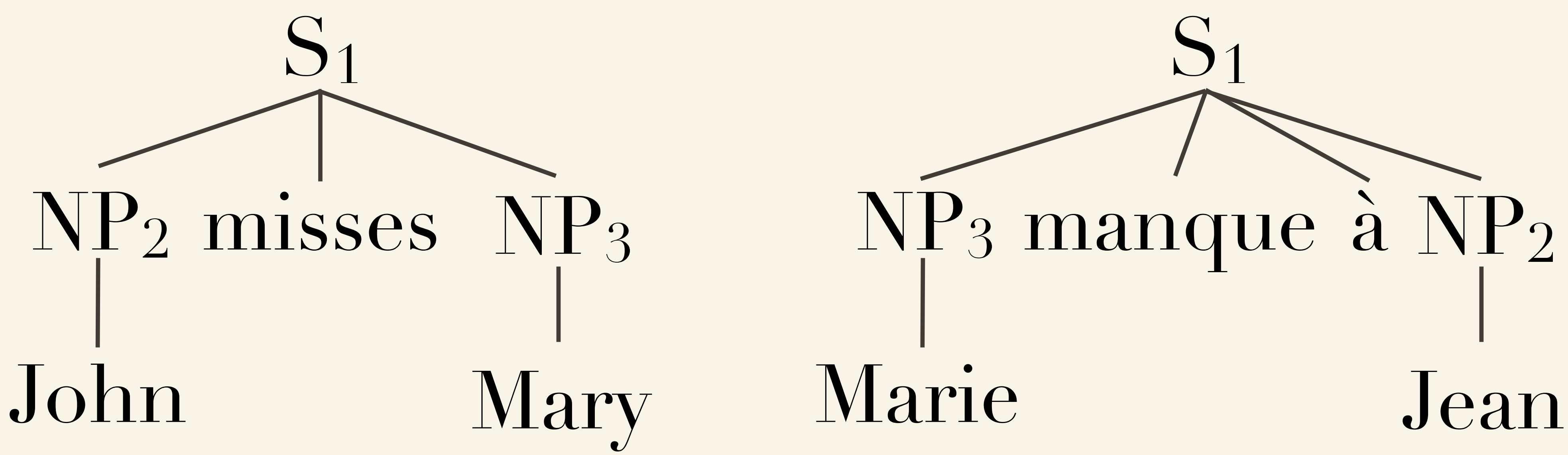
$$VP \rightarrow \langle V\ NP \rangle$$

Inversion transduction grammar (Wu)

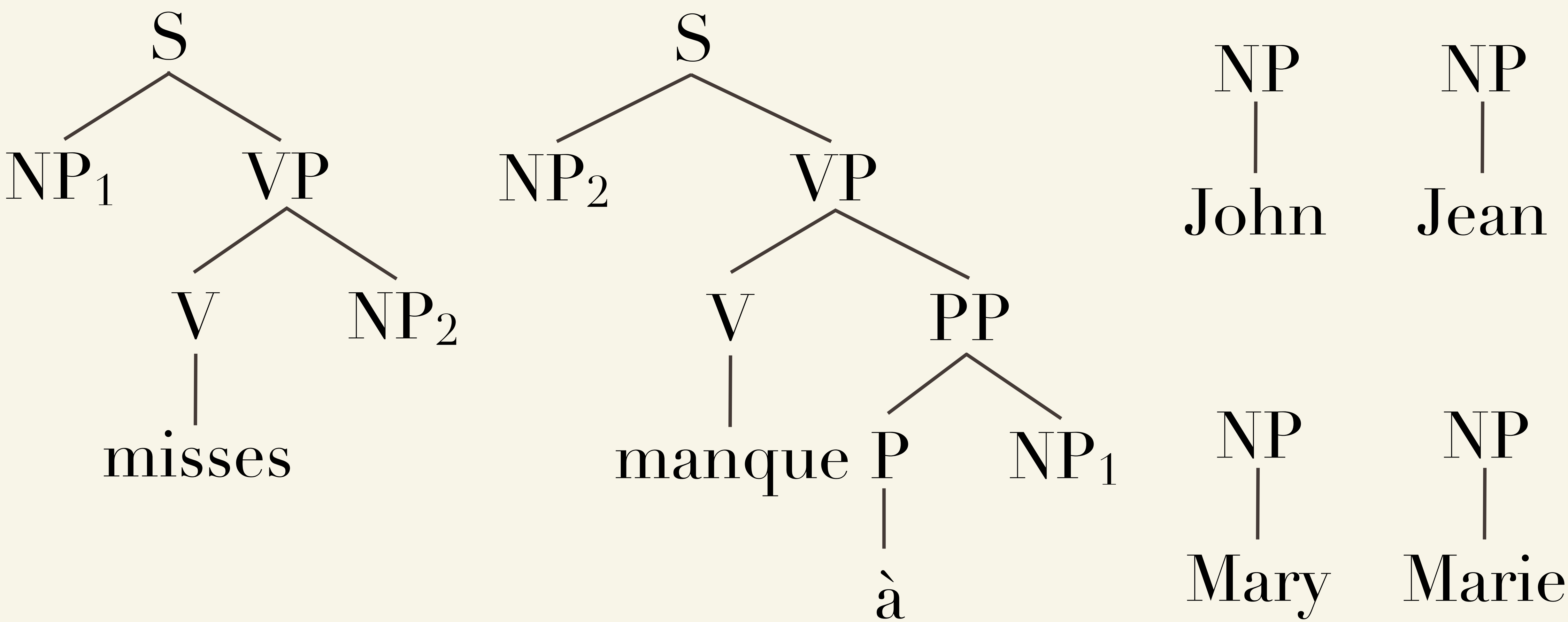
Limitations of synchronous CFGs



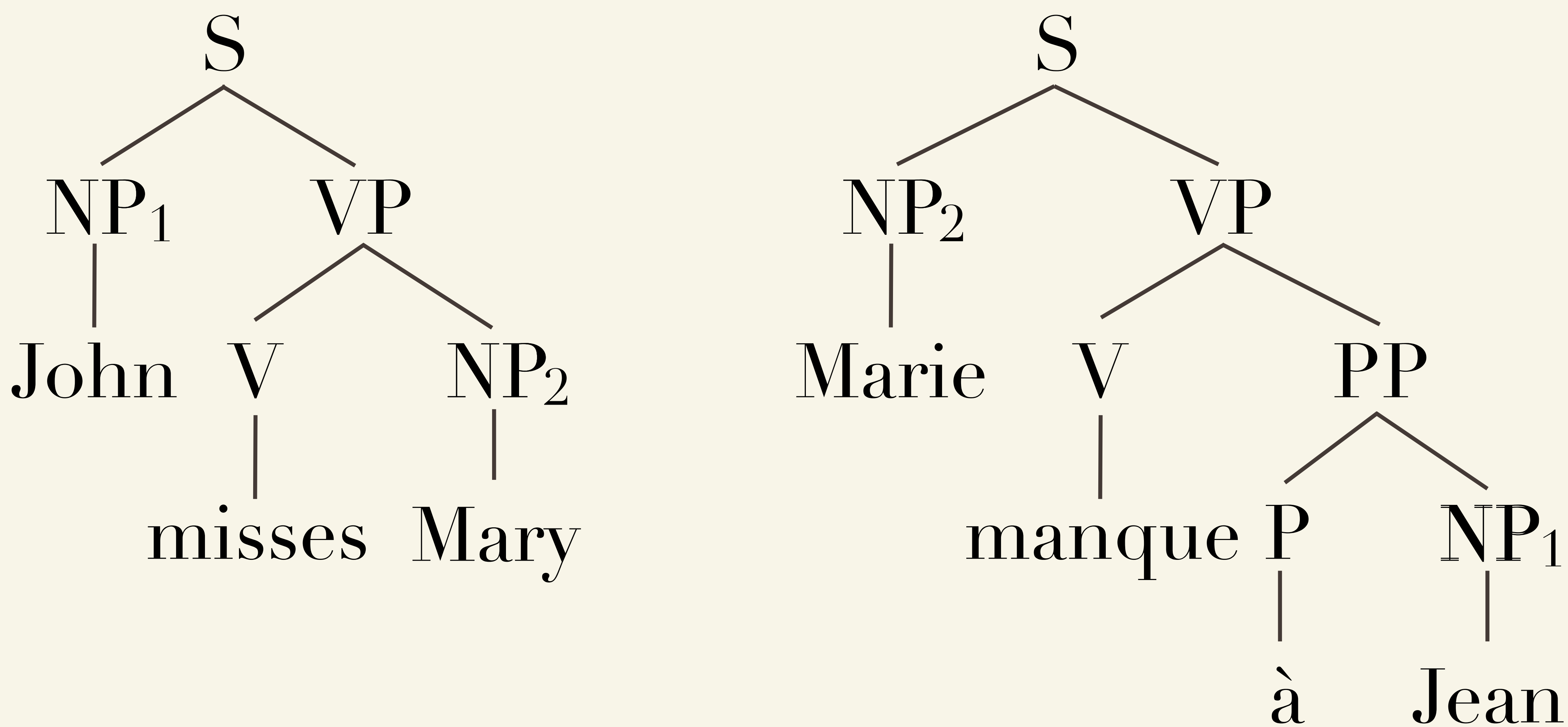
One solution



Synchronous tree substitution grammars



Synchronous tree substitution grammars



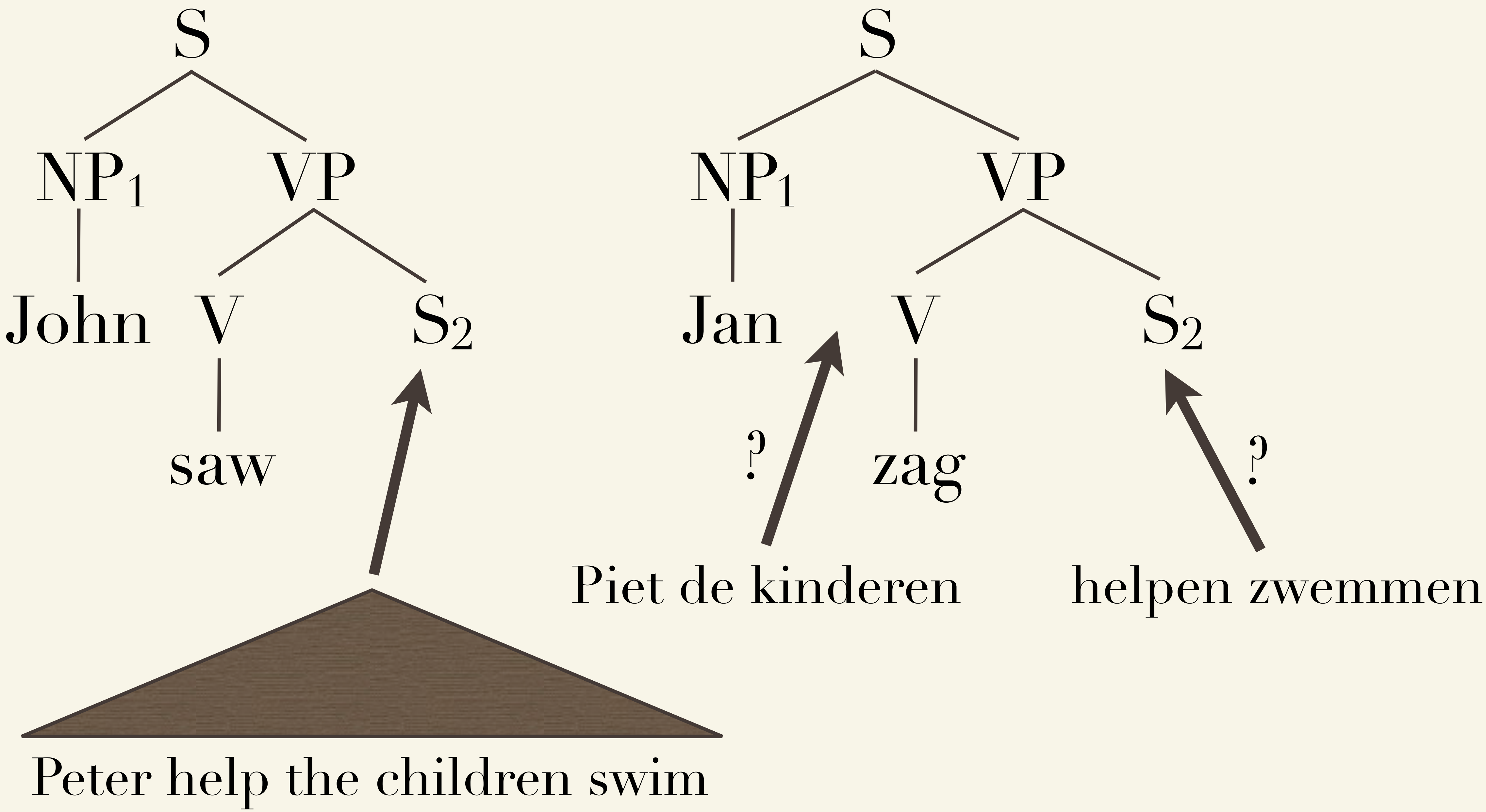
Limitations of synchronous TSGs

…dat Jan Piet de kinderen zag helpen zwemmen

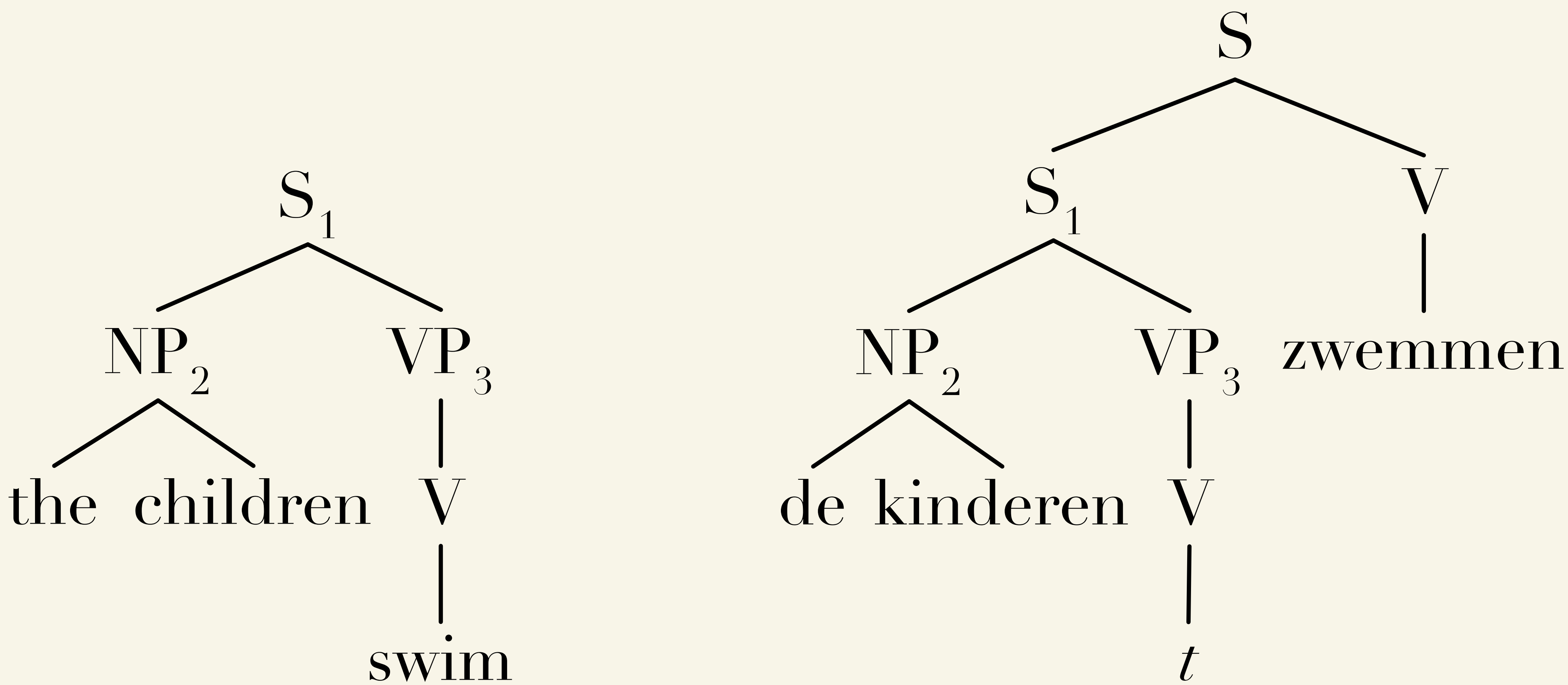
…that John saw Peter help the children swim

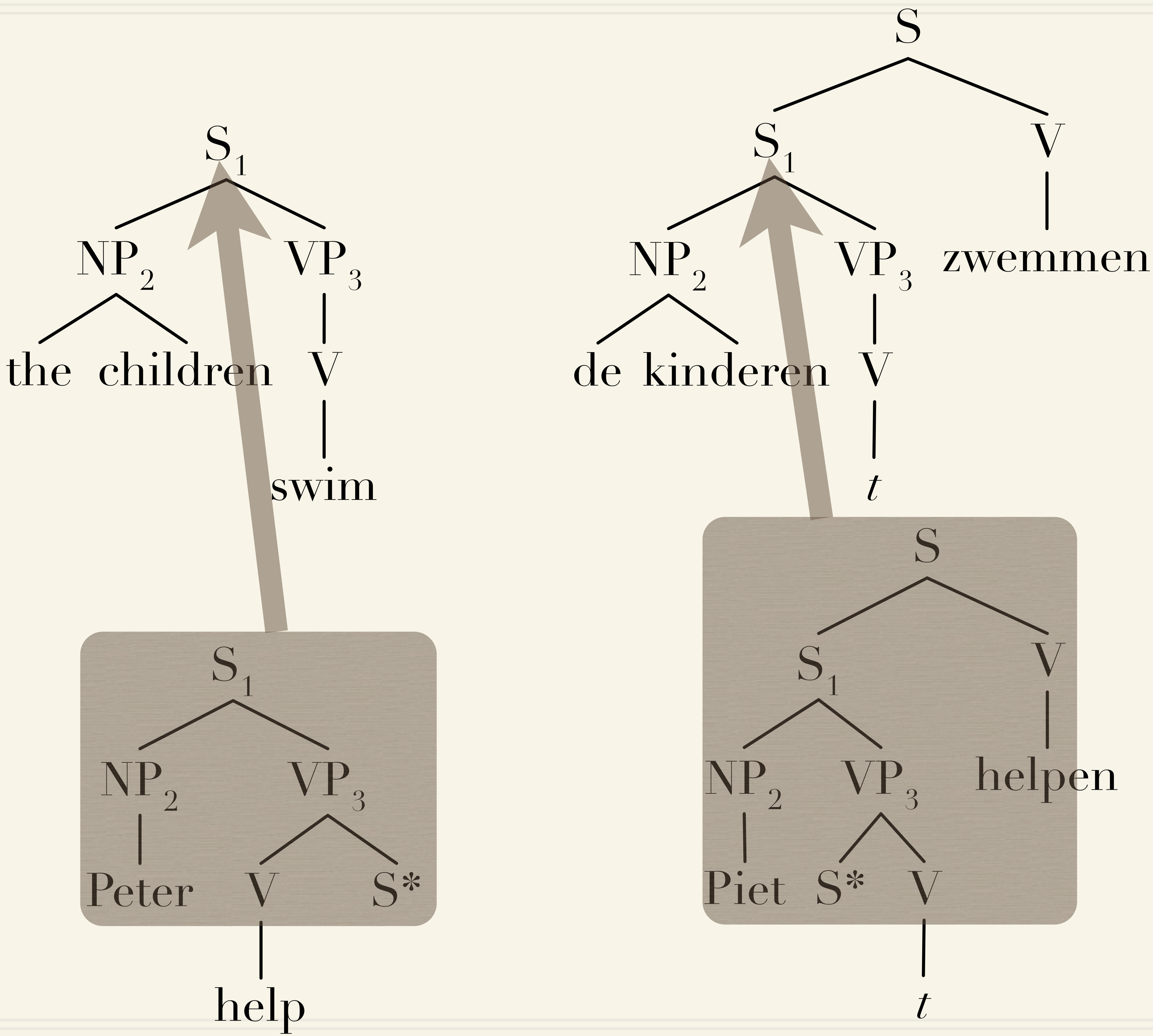
This pattern extends to *n* nouns and *n* verbs

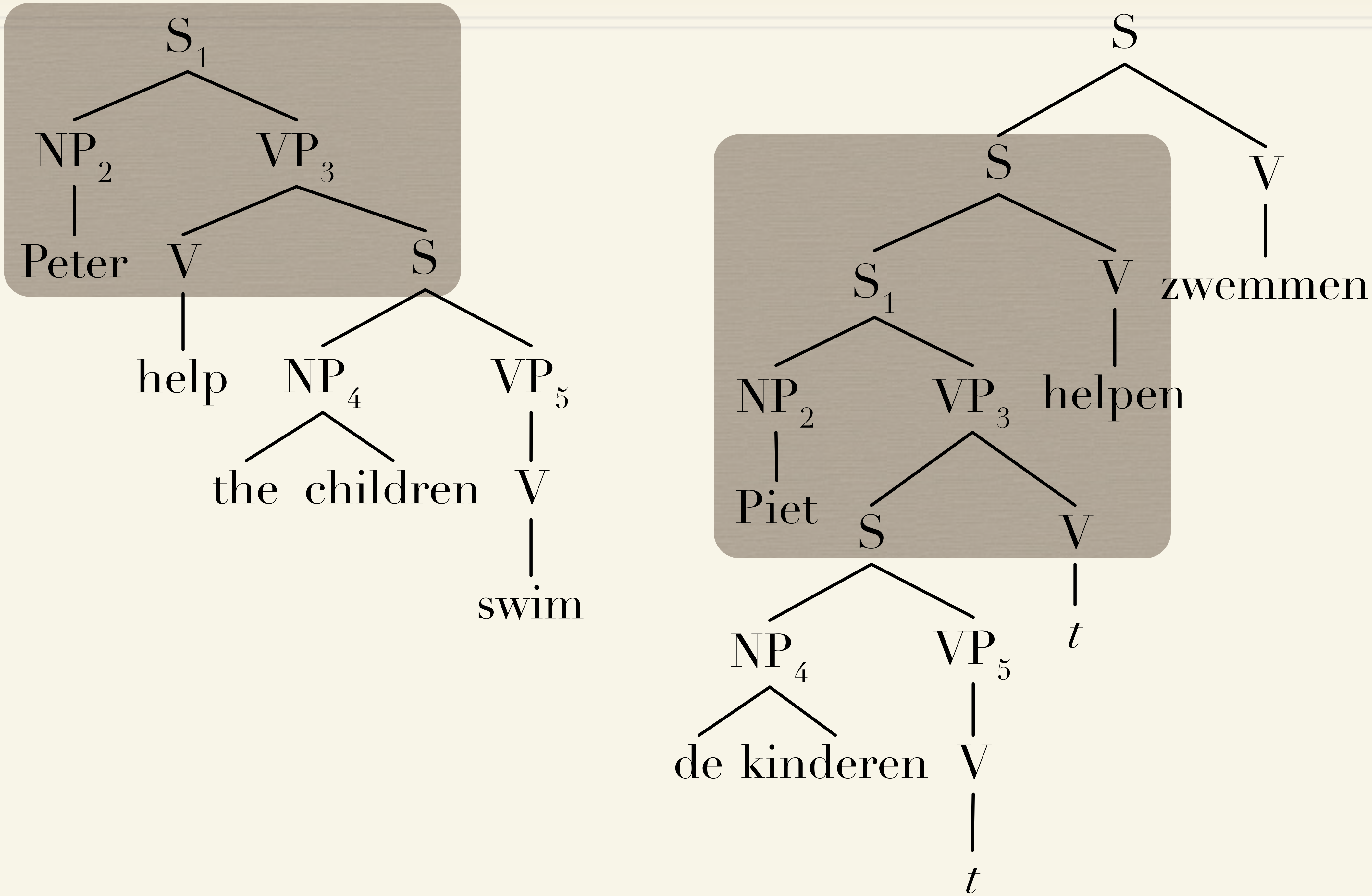
Limitations of synchronous TSGs

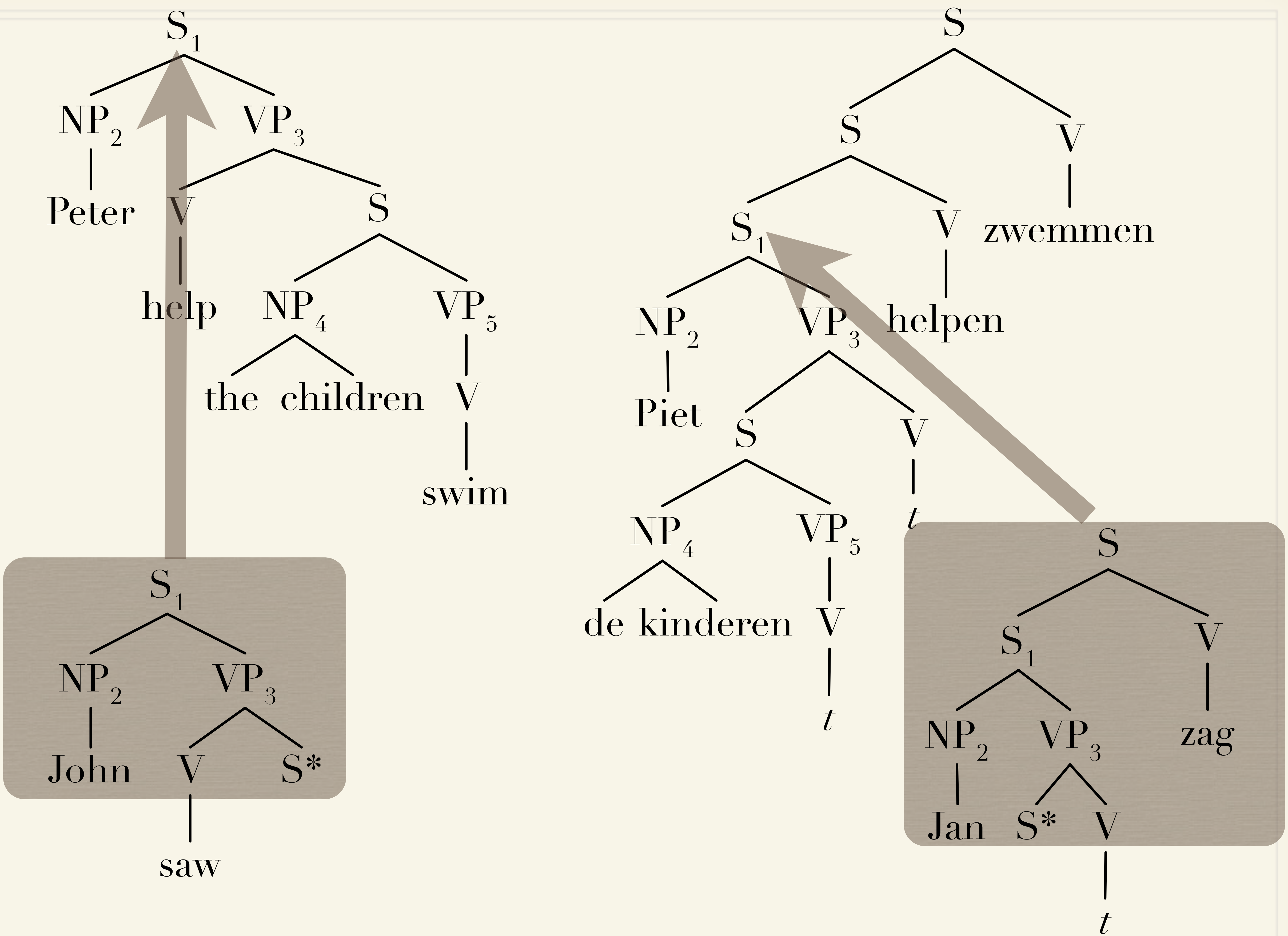


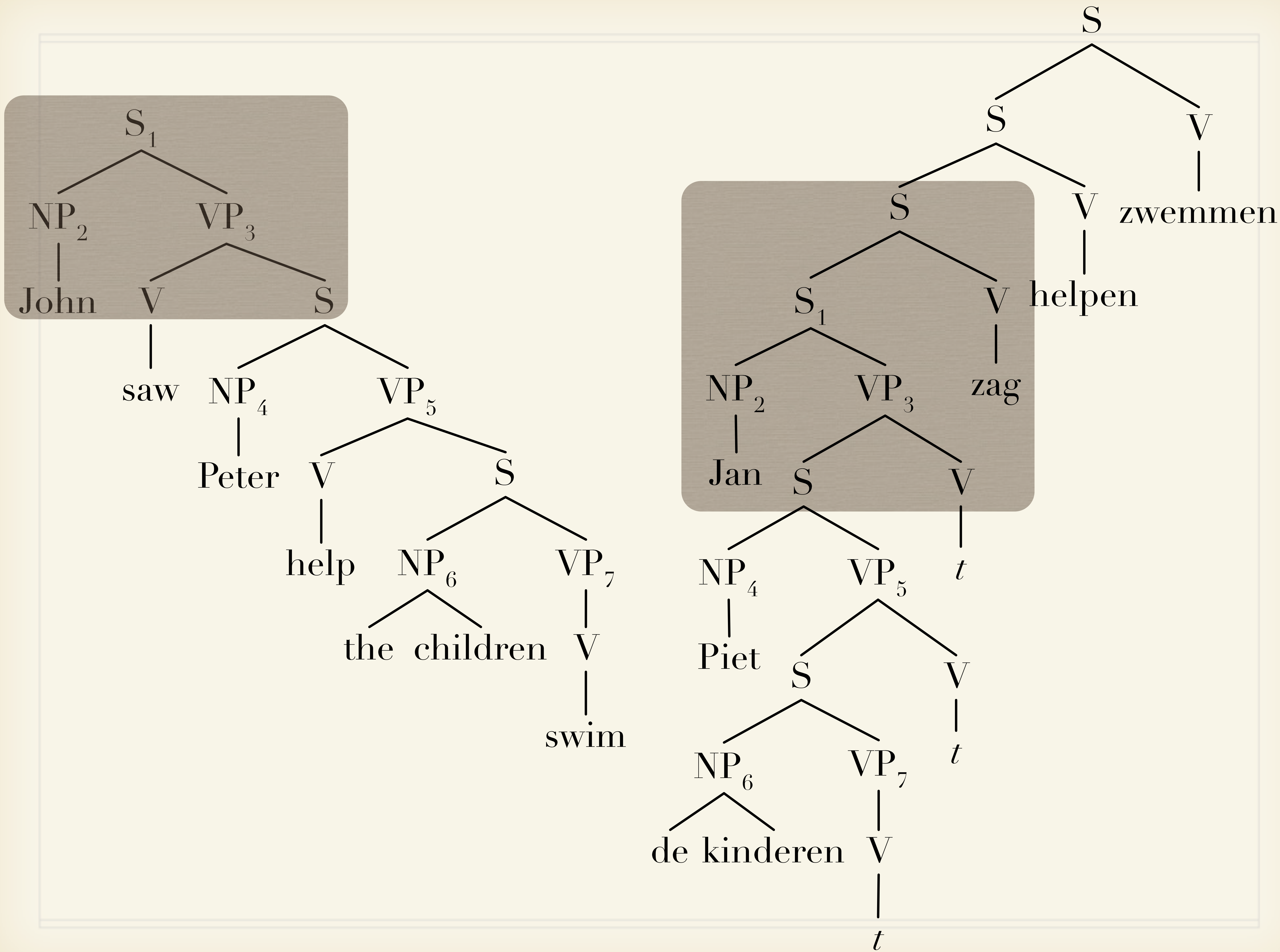
Synchronous TAG




S
S1
V
NP2
VP3
zwemmen
de kinderen
V
t
S
S1
V
NP2
VP3
helpen
Piet
S*
V
t
S1
NP2
VP3
the children
V
swim
S1
NP2
VP3
Peter
V
S*
help


S1
NP2
VP3
Peter
V
S
help
NP4
VP5
the children
V
swim
S
S
V
zwemmen
S1
V
helpen
NP2
VP3
Piet
S
V
t
NP4
VP5
de kinderen
V
t

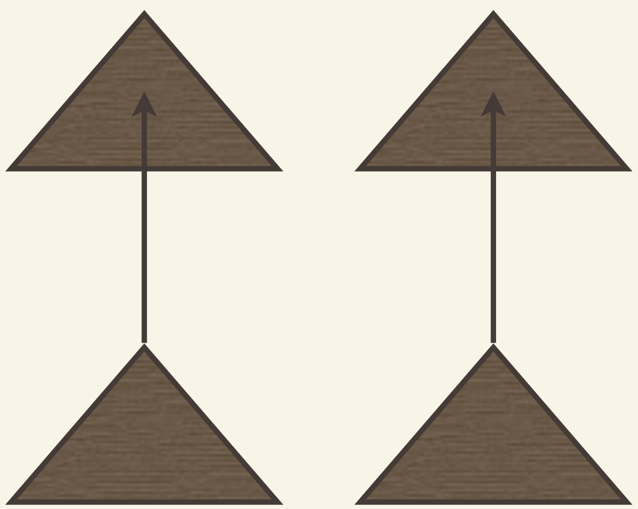

S1
NP2 VP3
Peter V S
help NP4 VP5
the children V
swim
S1
NP2 VP3
John V S*
saw
S
S V
S1 V zwemmen
NP2 VP3 helpen
Piet S V
NP4 VP5 t
de kinderen V
t
S
S1 V
NP2 VP3 zag
Jan S* V
t



S1
NP2
VP3
John
V
S
saw
NP4
VP5
Peter
V
S
help
NP6
VP7
the children
V
swim
S
S
V
S
V
zwemmen
S1
V
helpen
NP2
VP3
zag
Jan
S
V
NP4
VP5
t
Piet
S
V
NP6
VP7
t
de kinderen
V
t

Synchronous TAGs & multicomponent TAGs

- Synchronous TAG (Shieber, 1994) ≈ set-local 2-component TAG
- Synchronous TAG (Shieber & Schabes, 1990) ≈ non-local 2-component TAG



Properties

Chomsky normal form

$$X \rightarrow Y\, Z$$

$$X \rightarrow a$$

Chomsky normal form

$A \rightarrow \quad B \;\; C \;\; D \;\; E \;\; F$ rank 5

Chomsky normal form

A → [[[B C] D] E] F   rank 5

A → V1 F

V1 → V2 E

V2 → V3 D

V3 → B C

rank 2

A hierarchy of synchronous CFGs

$$1\text{-CFG} \subsetneq 2\text{-CFG} = 3\text{-CFG} = 4\text{-CFG} = \ldots$$

$$1\text{-SCFG} \subsetneq 2\text{-SCFG} = 3\text{-SCFG} \subsetneq 4\text{-SCFG} \subsetneq \ldots$$

ITG
(Wu, 1997)

Synchronous CNF?

$$A \rightarrow (B_1 \;\; C_2 \;\; D_3 \;,\;\; C_2 \;\; D_3 \;\; B_1)$$ rank 3

Synchronous CNF?

$$A \rightarrow (B_1\ [C_2\ \ D_3],\ [C_2\ \ D_3]\ B_1)$$

rank 3

$$A \rightarrow (B_1\, V1_2\ \ ,\, V1_2\, B_1)$$

$$V1 \rightarrow (C_1\, D_2\ \ ,\, C_1\, D_2)$$

rank 2

Synchronous CNF?

$$A \rightarrow (B_1\, C_2\, D_3\, E_4,\; C_2\, E_4\, B_1\, D_3)$$

rank 4

$$A \rightarrow ([B_1\, C_2]\, D_3\, E_4,\; [C_2\, E_4\, B_1]\, D_3)$$

$$A \rightarrow (B_1\, [C_2\, D_3]\, E_4,\; [C_2\, E_4\, B_1\, D_3])$$

$$A \rightarrow (B_1\, C_2\, [D_3\, E_4],\; C_2\, [E_4\, B_1\, D_3])$$

Synchronous CNF?

$$A \rightarrow (B_1\ C_2\ \ D_3,\ C_2\ D_3\ B_1)$$

	1	2	3
1			B
2	C		
3		D	

$$A \rightarrow (B_1\ C_2\ D_3\ E_4,\ C_2\ E_4\ B_1\ D_3)$$

	1	2	3	4
1			B	
2	C			
3				D
4		E		

Inversion Transduction Grammar

$$A \rightarrow (B_1\ [C_2\ [D_3\ E_4]],\ [[E_4\ D_3]\ C_2]\ B_1)$$

	1	2	3	4
1				B
2			C	
3		D		
4	E			

A hierarchy of synchronous CFGs

$$1\text{-CFG} \subsetneq 2\text{-CFG} = 3\text{-CFG} = 4\text{-CFG} = \ldots$$

$$1\text{-SCFG} \subsetneq 2\text{-SCFG} = 3\text{-SCFG} \subsetneq 4\text{-SCFG} \subsetneq \ldots$$

ITG
(Wu, 1997)

A hierarchy of synchronous TAGs

$$\text{1-TAG} \subsetneq \text{2-TAG} = \text{3-TAG} = \text{4-TAG} = \ldots$$

$$\text{1-STAG} \subsetneq \text{2-STAG} = \text{3-STAG} \subsetneq \text{4-STAG} \subsetneq \ldots$$

weakly

?

Algorithms

Overview

- Translation
- Bitext parsing

Review: CKY

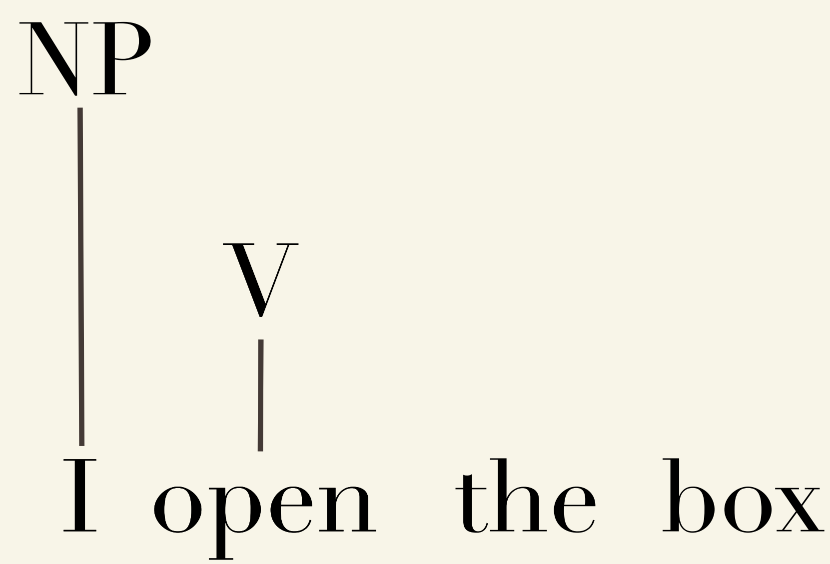
S → NP VP

NP → I

NP → the box

VP → V NP

V → open



Review: CKY

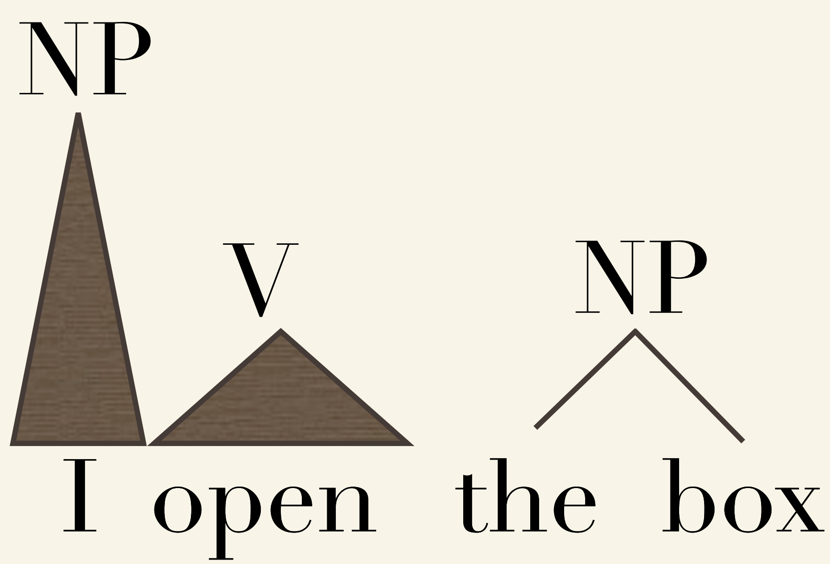
$S \rightarrow NP\ VP$

$NP \rightarrow I$

$NP \rightarrow \text{the box}$

$VP \rightarrow V\ NP$

$V \rightarrow \text{open}$



Review: CKY

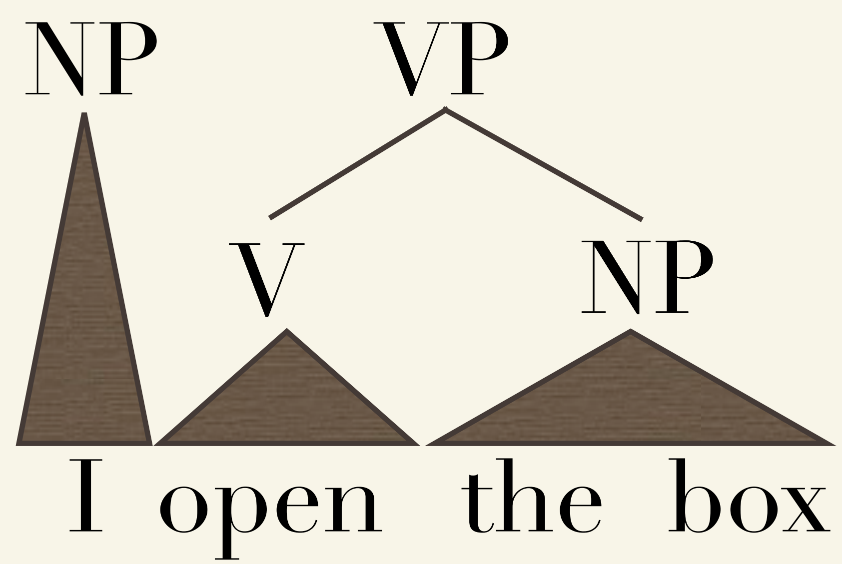
$S \rightarrow NP\ VP$

$NP \rightarrow I$

$NP \rightarrow$ the box

$VP \rightarrow V\ NP$

$V \rightarrow$ open



Review: CKY

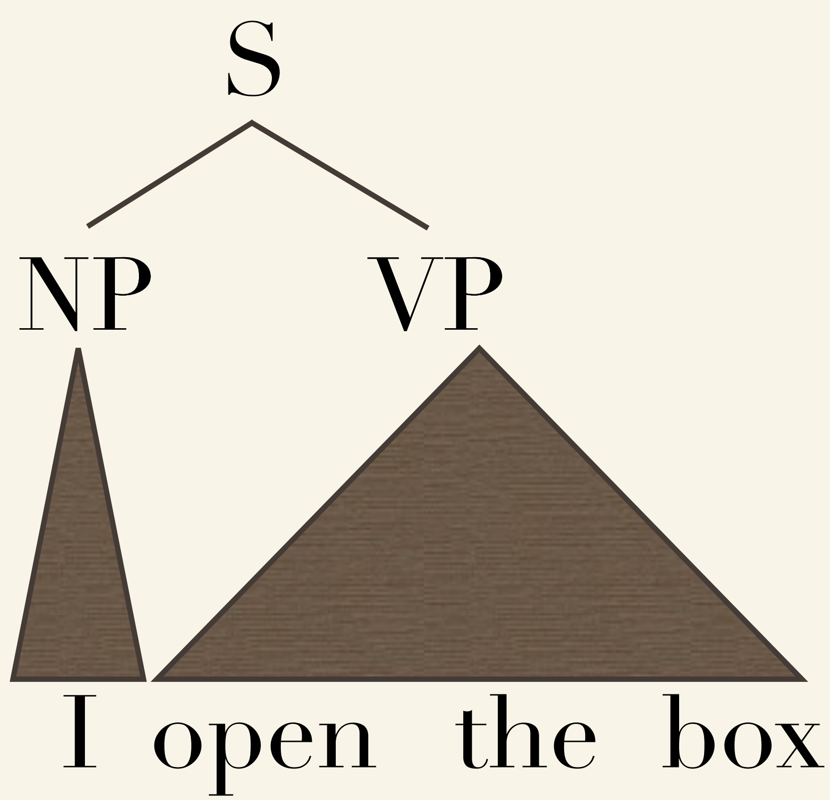
$S \rightarrow NP\ VP$

$NP \rightarrow I$

$NP \rightarrow \text{the box}$

$VP \rightarrow V\ NP$

$V \rightarrow \text{open}$



Review: CKY

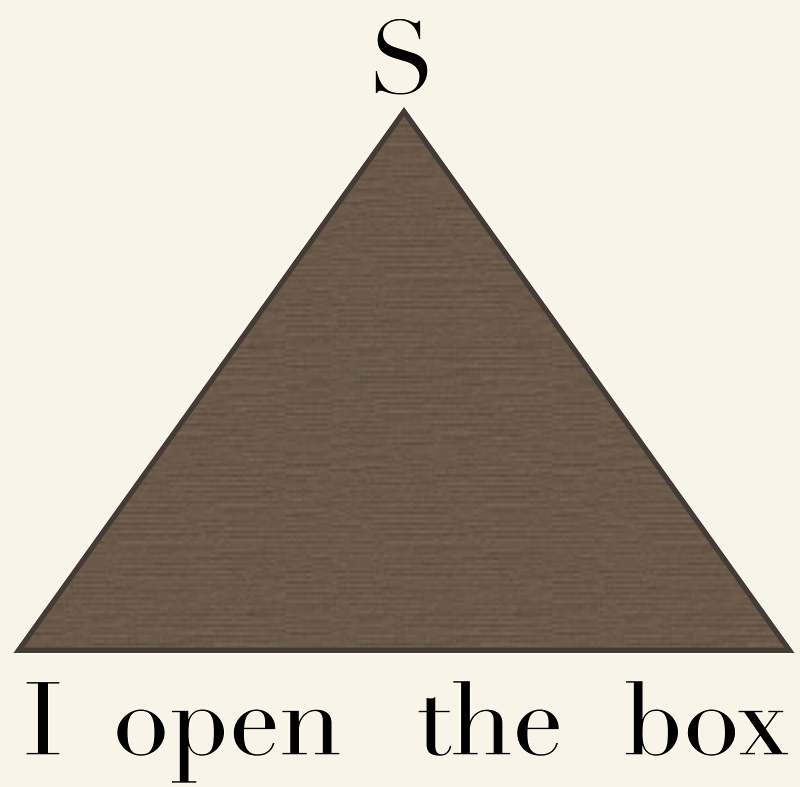
S → NP VP

NP → I

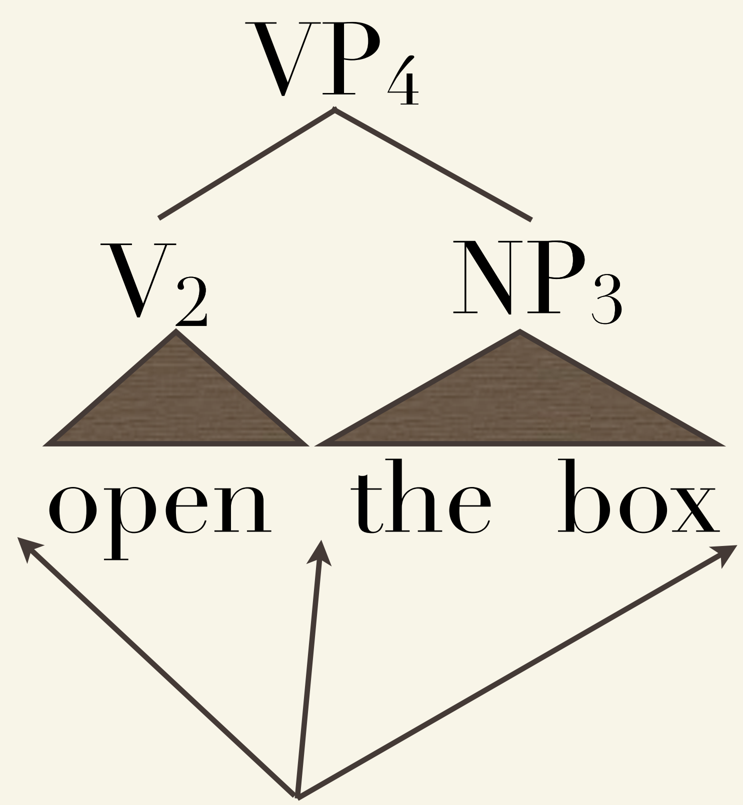
NP → the box

VP → V NP

V → open

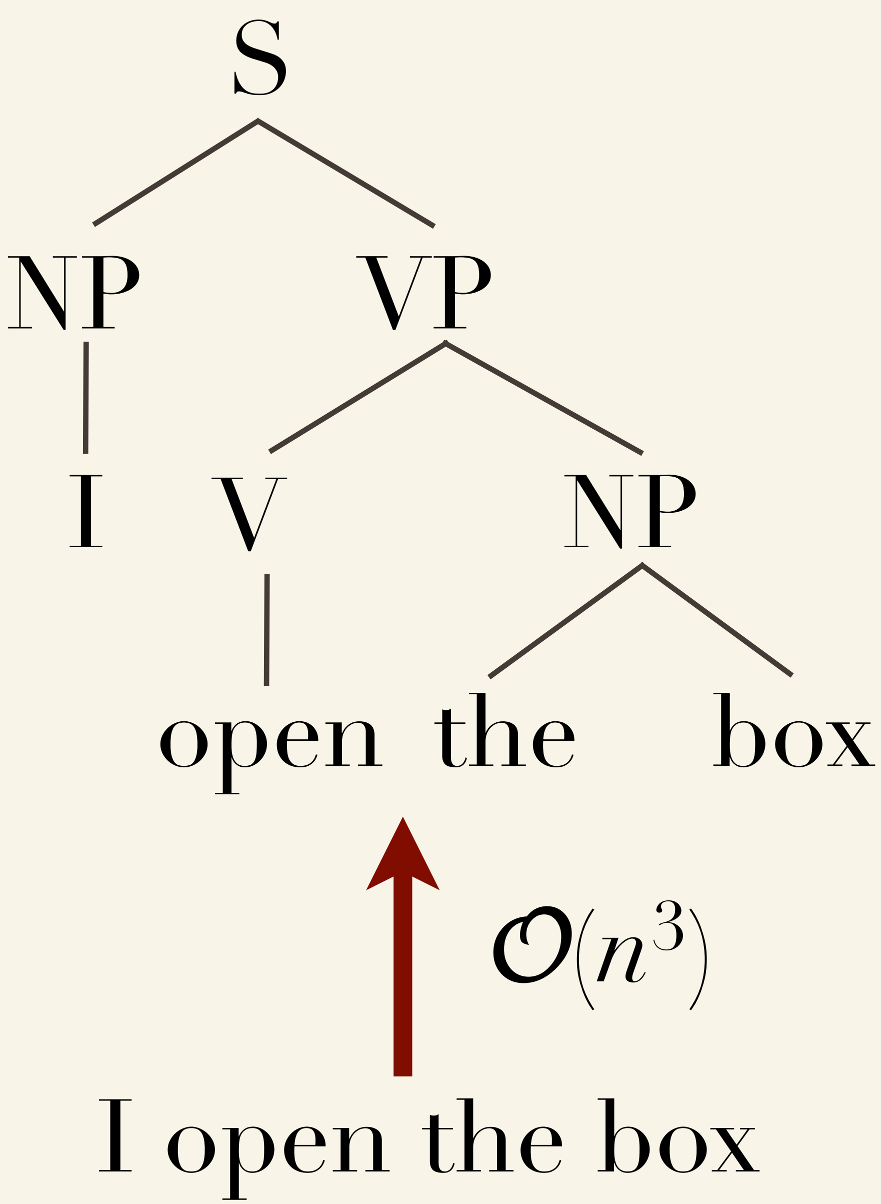


Review: CKY

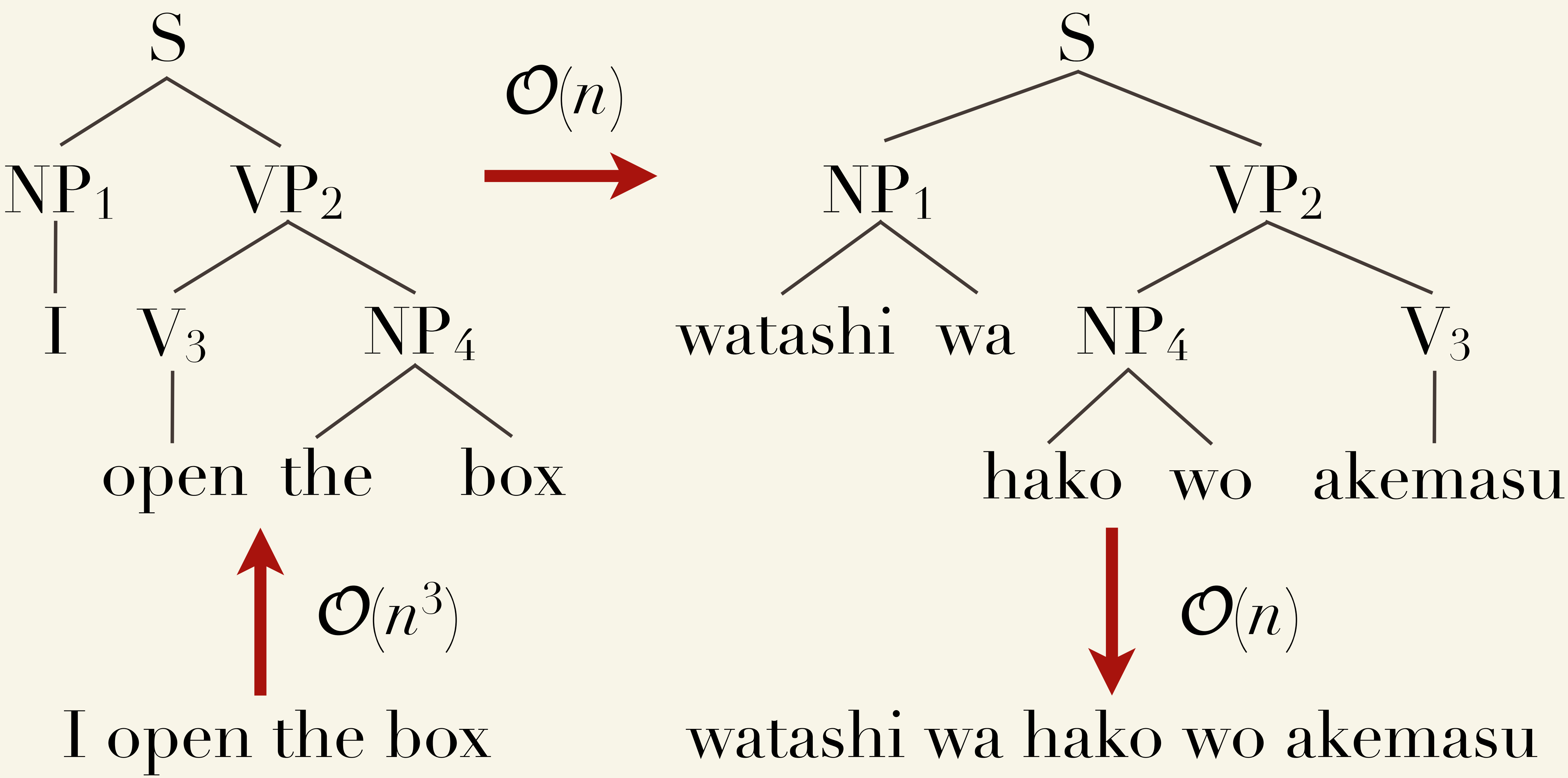


$\mathcal{O}(n^3)$ ways of matching

Translation

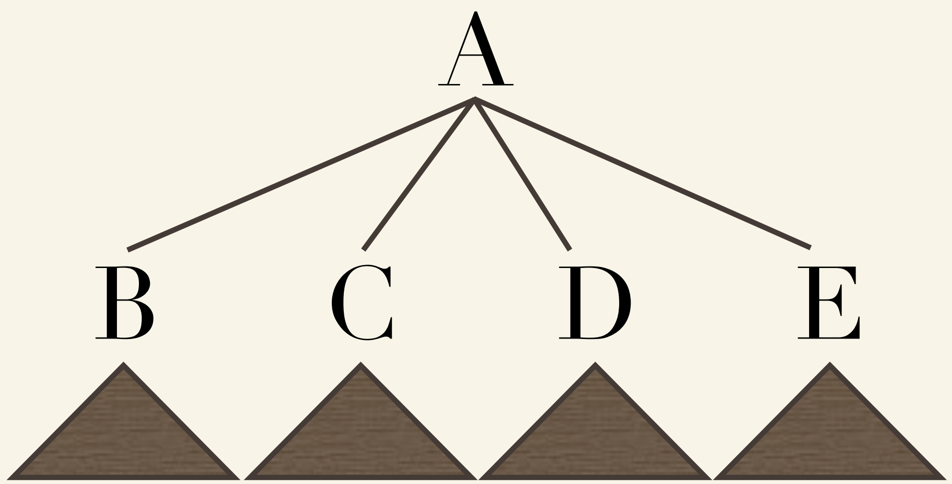


Translation



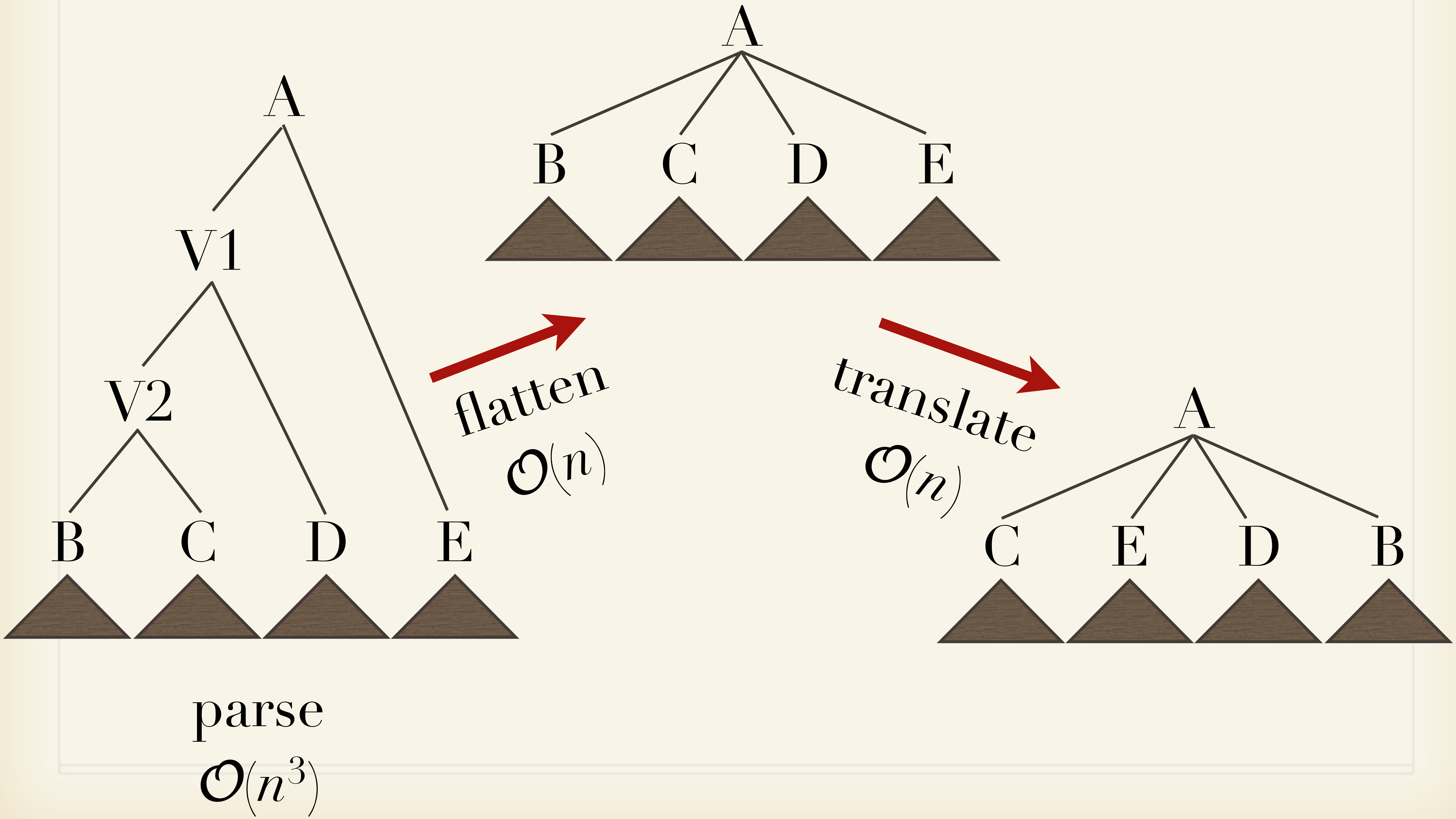
Translation

What about…

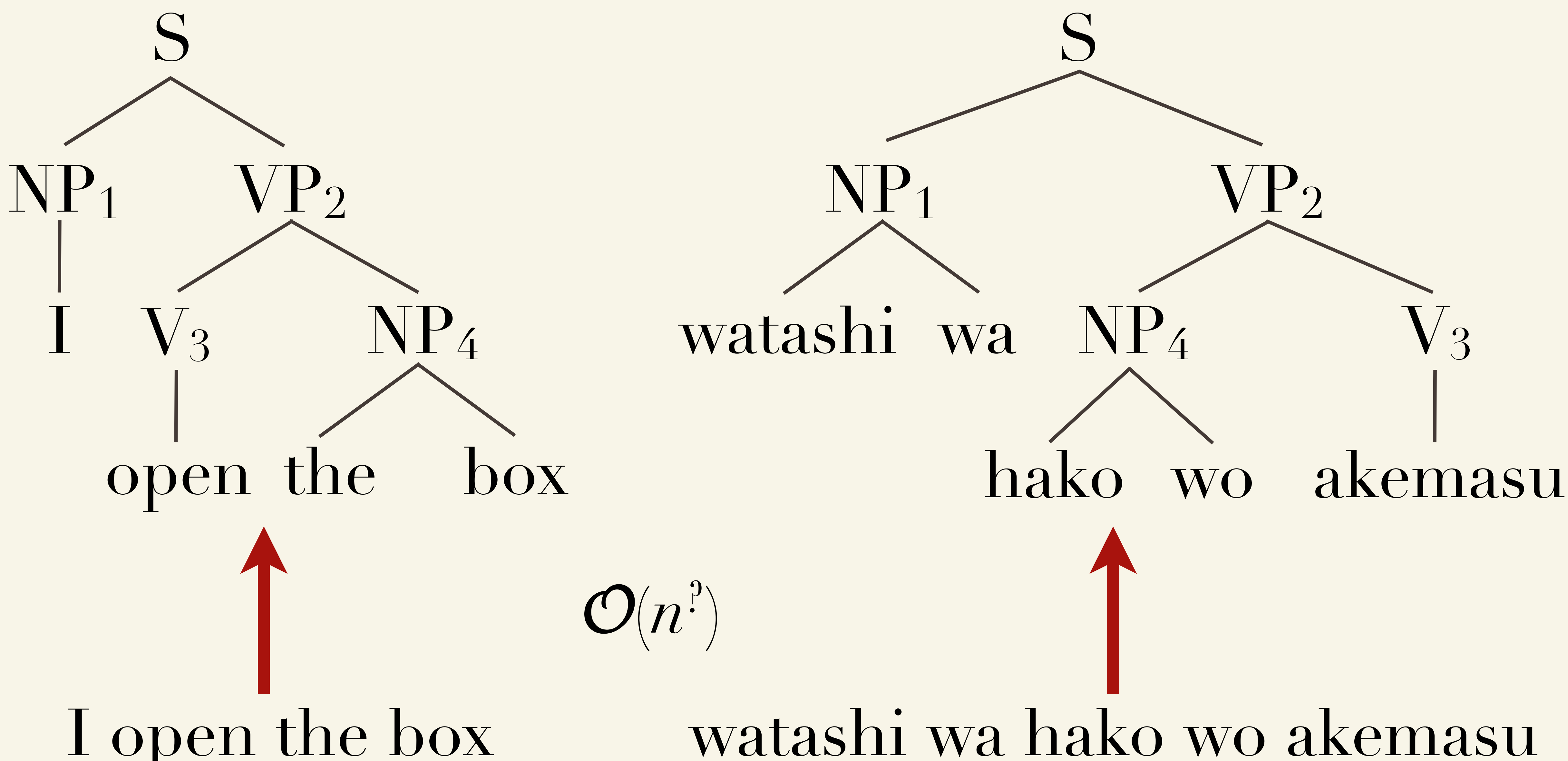


$\mathcal{O}(n^5)$ ways of combining?

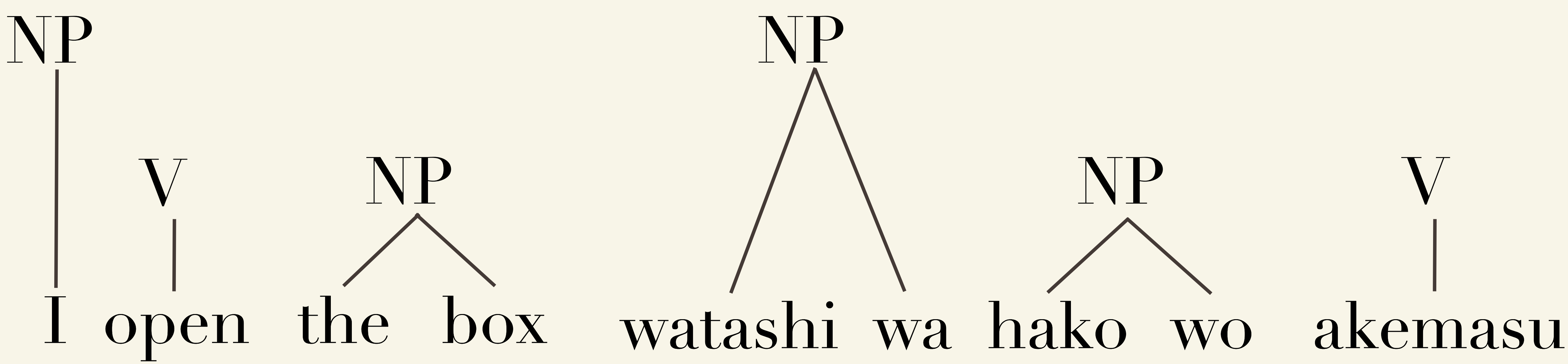
Translation



Bitext parsing

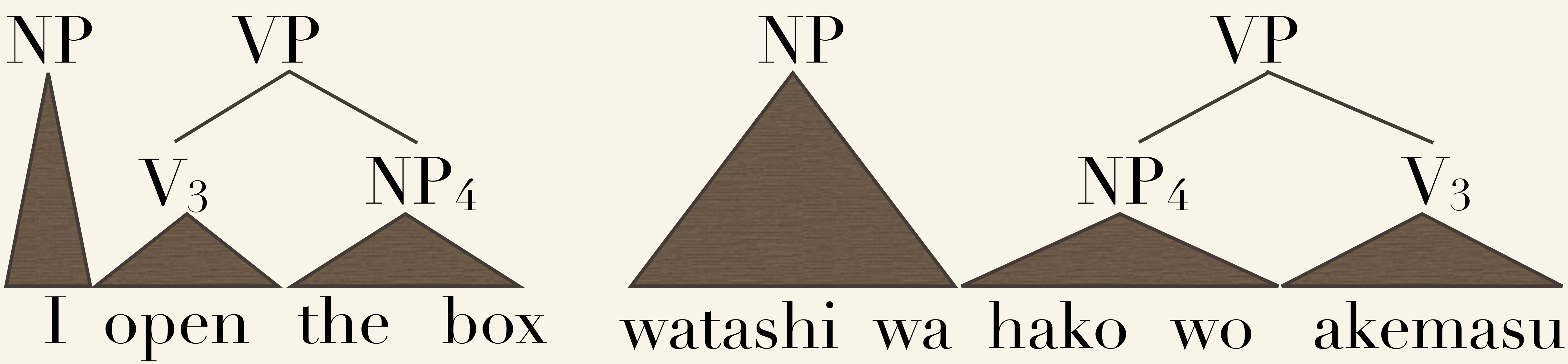


Bitext parsing

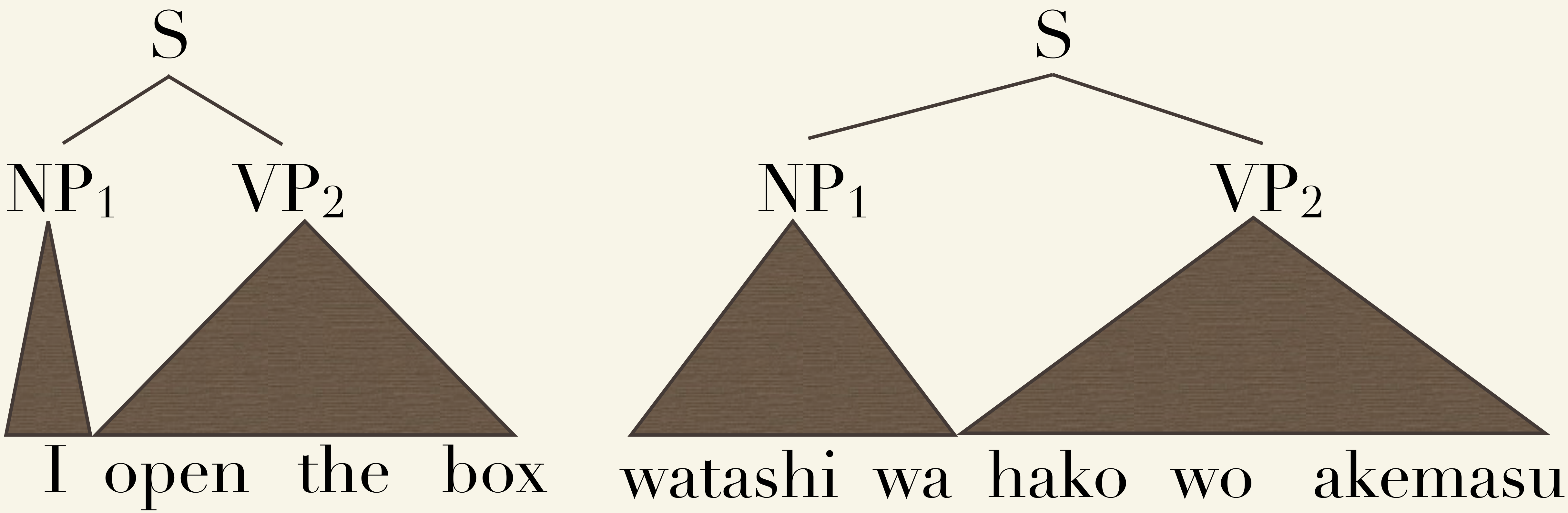


Consider rank-2 synchronous CFGs for now

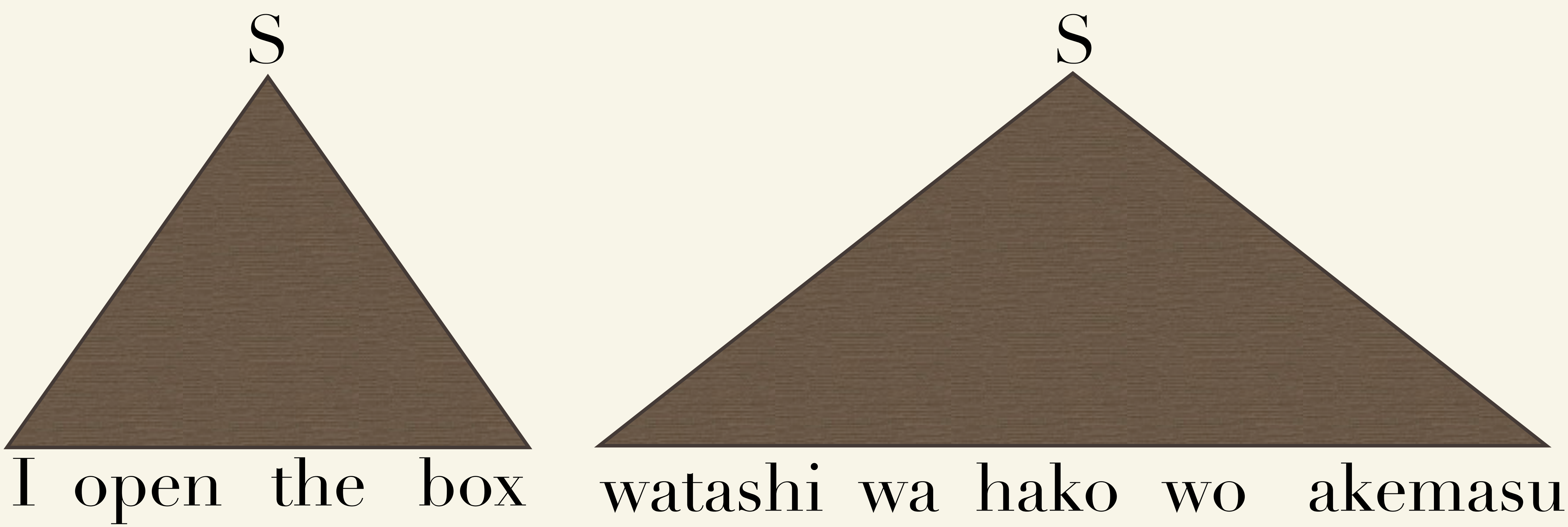
Bitext parsing



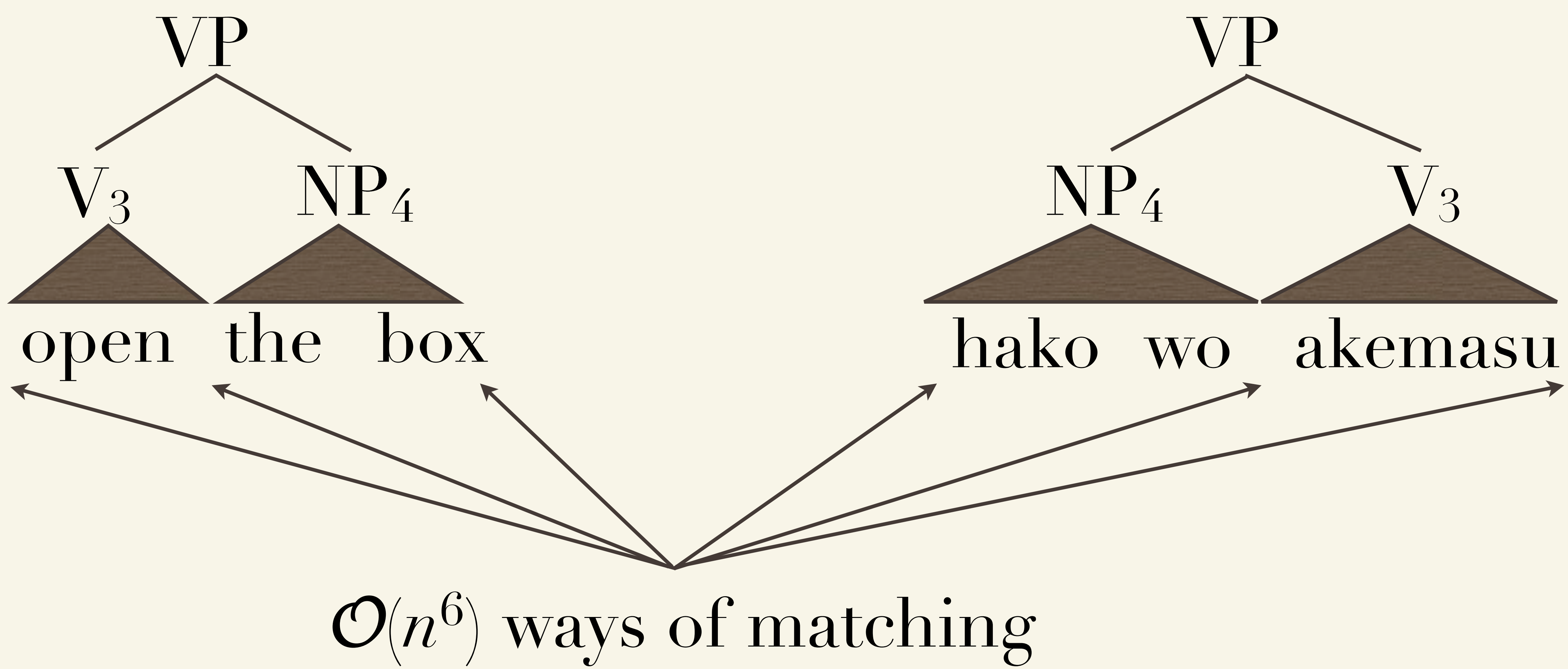
Bitext parsing



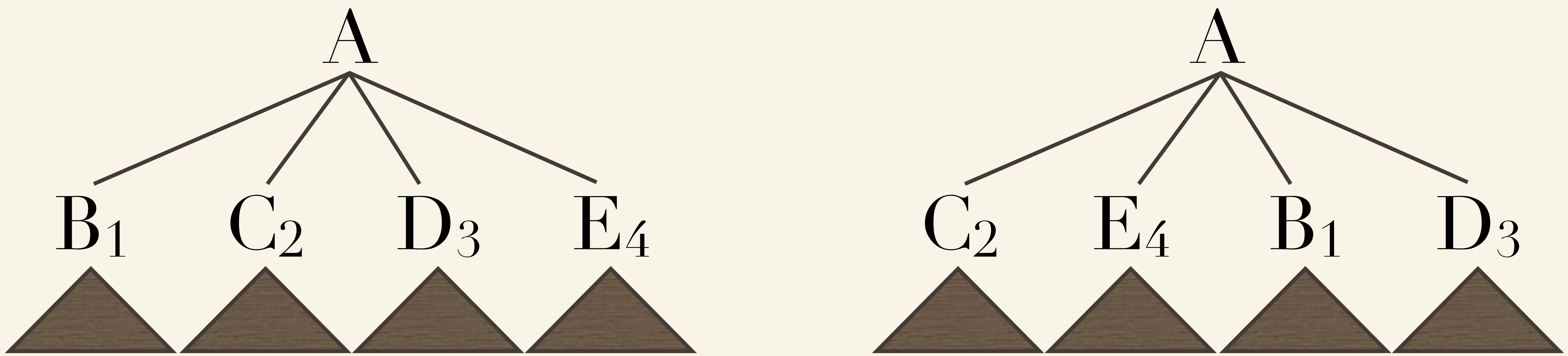
Bitext parsing



Bitext parsing



Bitext parsing



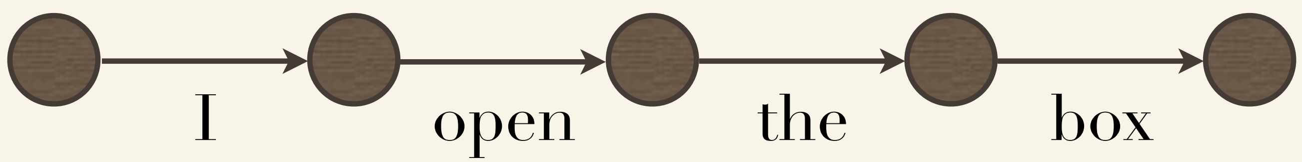
$\mathcal{O}(n^{10})$ ways of combining!

Summary so far

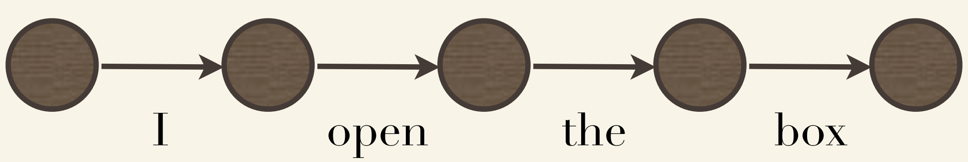
- Translation: essentially parsing on the source side, $\mathcal{O}(n^3)$
- Bitext parsing: polynomial in n but worst-case exponential in rank, $\mathcal{O}(n^{2(r+1)})$

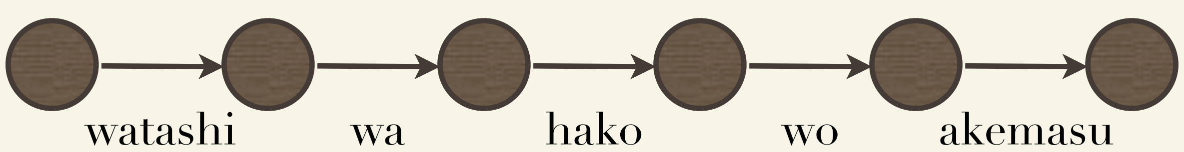
Parsing as intersection

- Translation is like intersecting with a finite-state automaton on source side



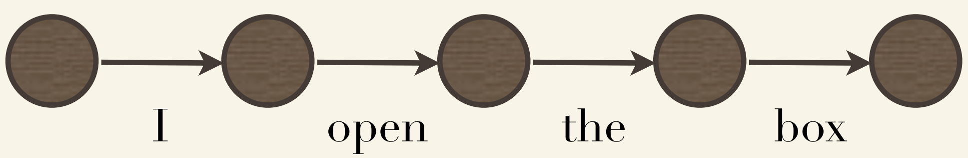
- Bitext parsing is like intersecting with FSAs on both sides



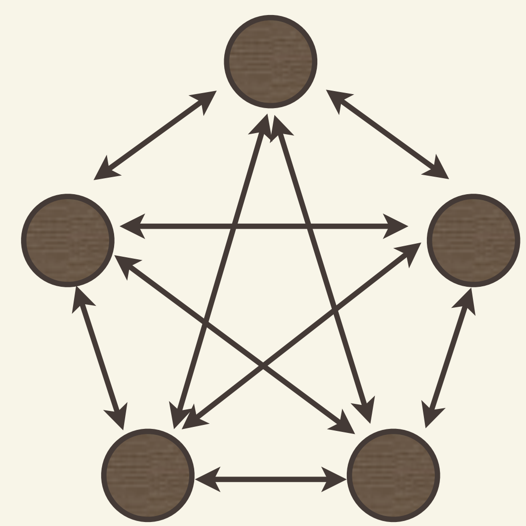


Translation with a language model

is also like intersecting with FSAs on both sides



input string

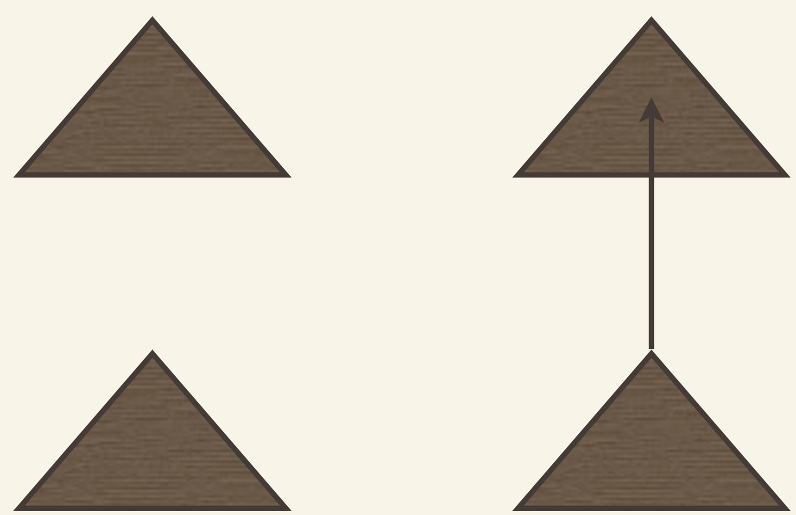


n-gram language model

Extensions

Synchronous TAGs & multicomponent TAGs

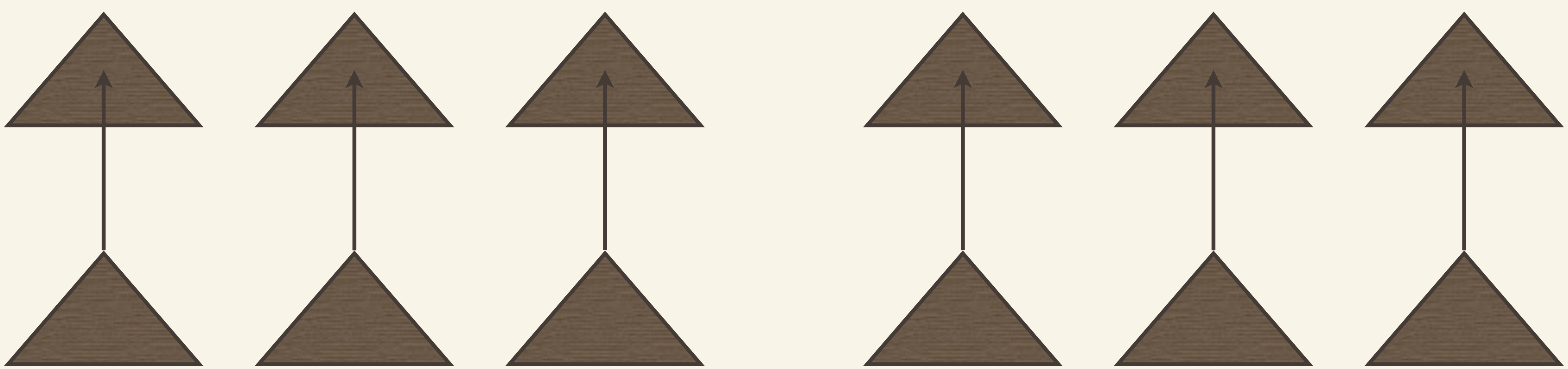
- Synchronous TAG (Shieber, 1994) ≈ set-local 2-component TAG



Synchronous TAGs & multicomponent TAGs

- Synchronous set-local k-component TAG

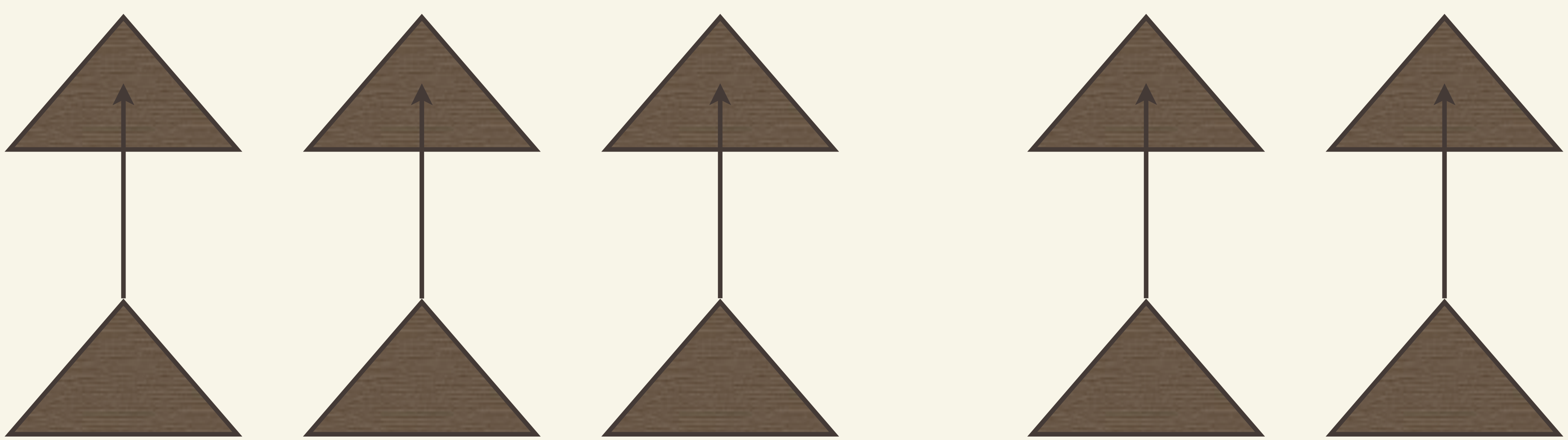
  $\approx$ set-local $2k$-component TAG



Synchronous TAGs & multicomponent TAGs

- Set-local k-component TAG : set-local k'-component TAG

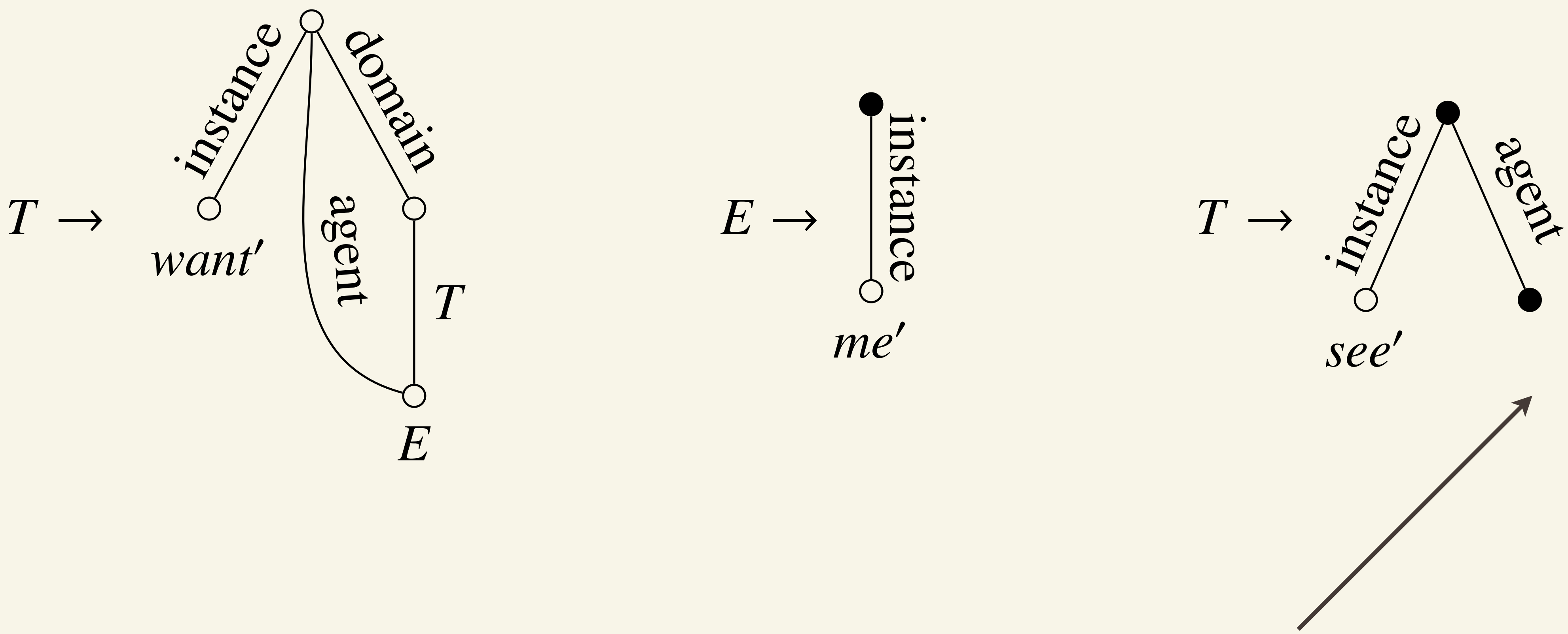
  $\approx$ set-local $(k+k')$-component TAG



Synchronous LCFRS

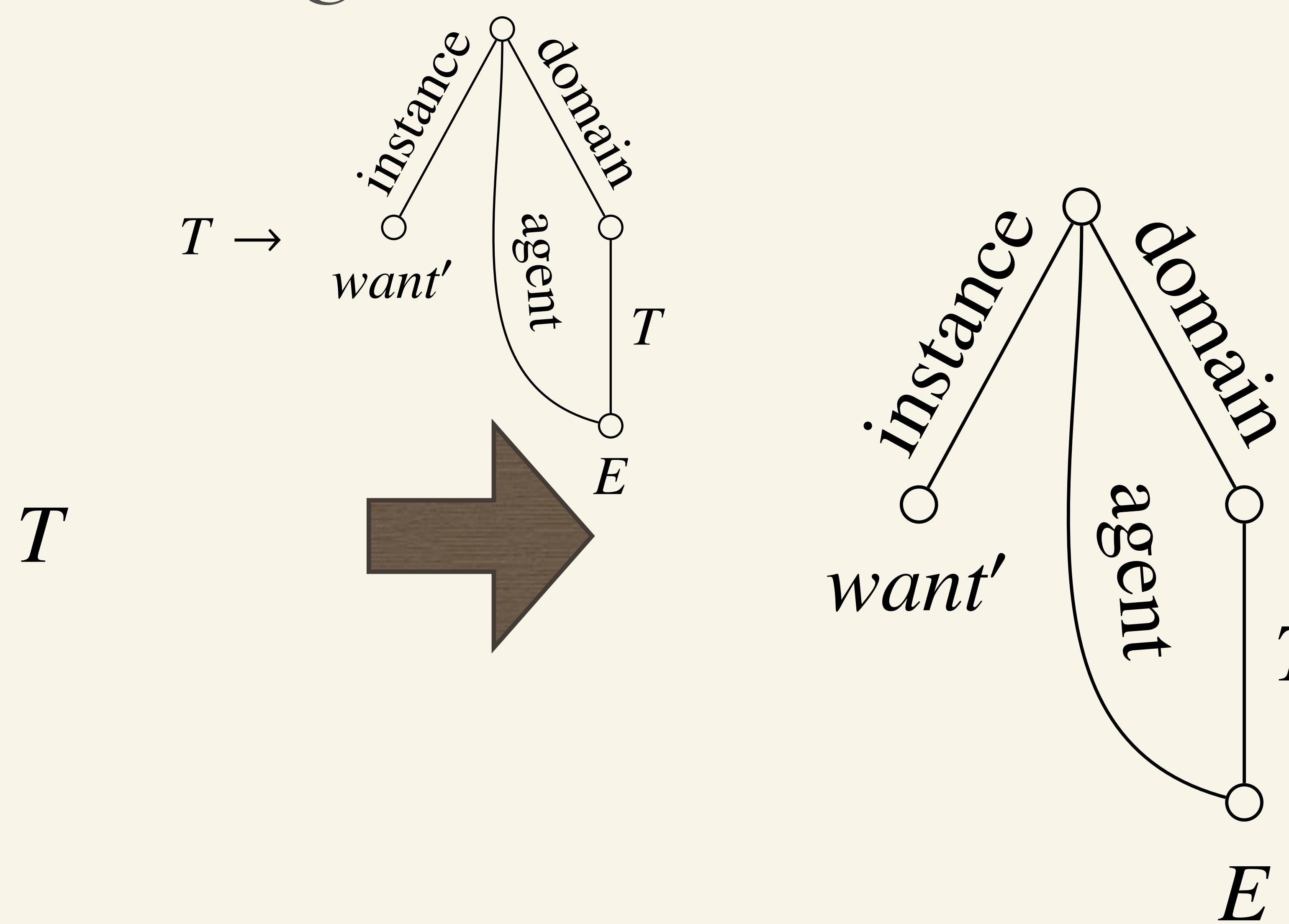
- rank = "how many things a rule combines"
- fanout = "how many pieces does each thing have" $(\mathrm{CFG} = 1, \mathrm{TAG} = 2)$
- synchronize any (r, f) and (r, f') LCFRSs. Bitext parsing: $\mathcal{O}(n^{(r+1)(f+f')})$

Hyperedge replacement grammars

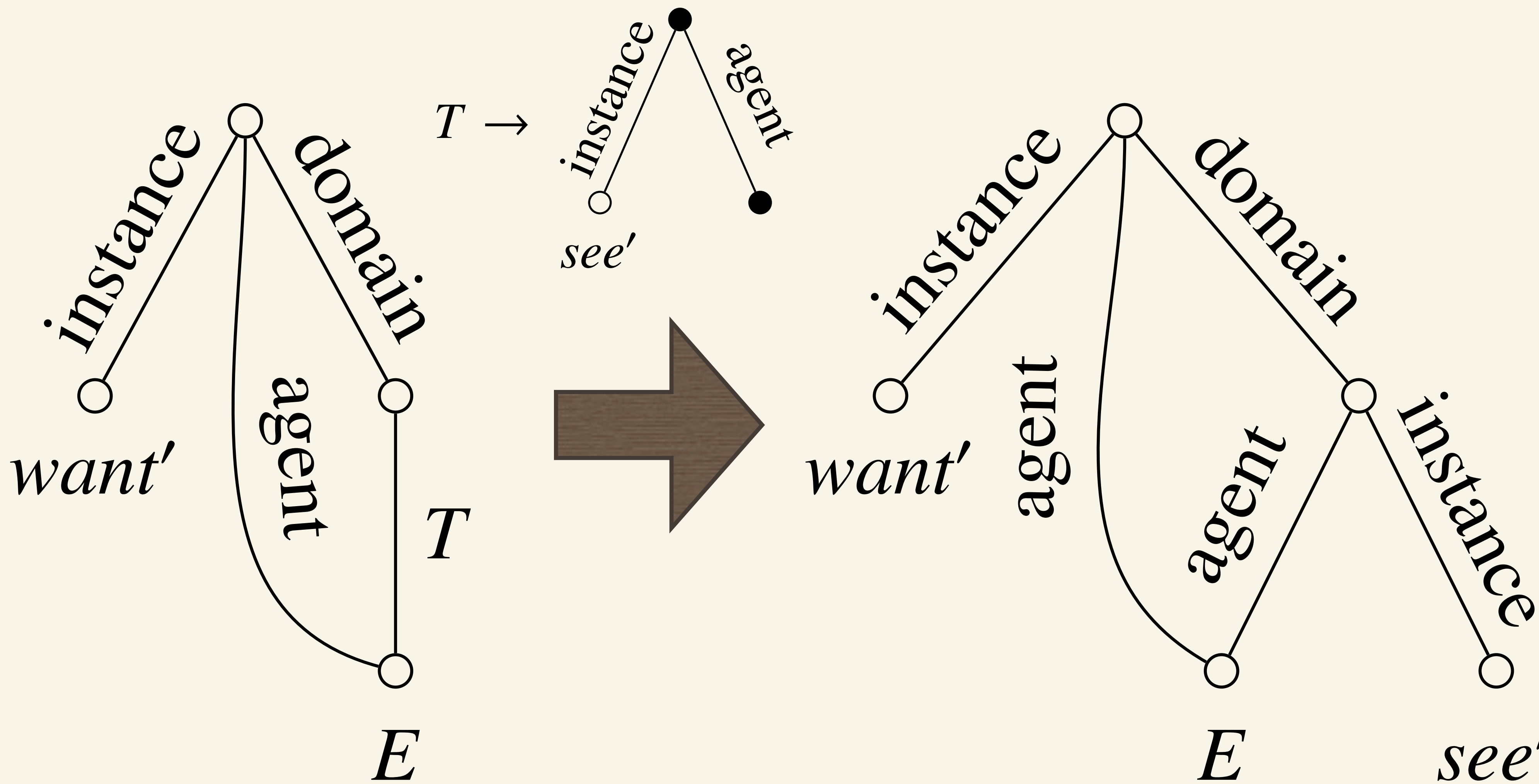


external node
(like a foot node)

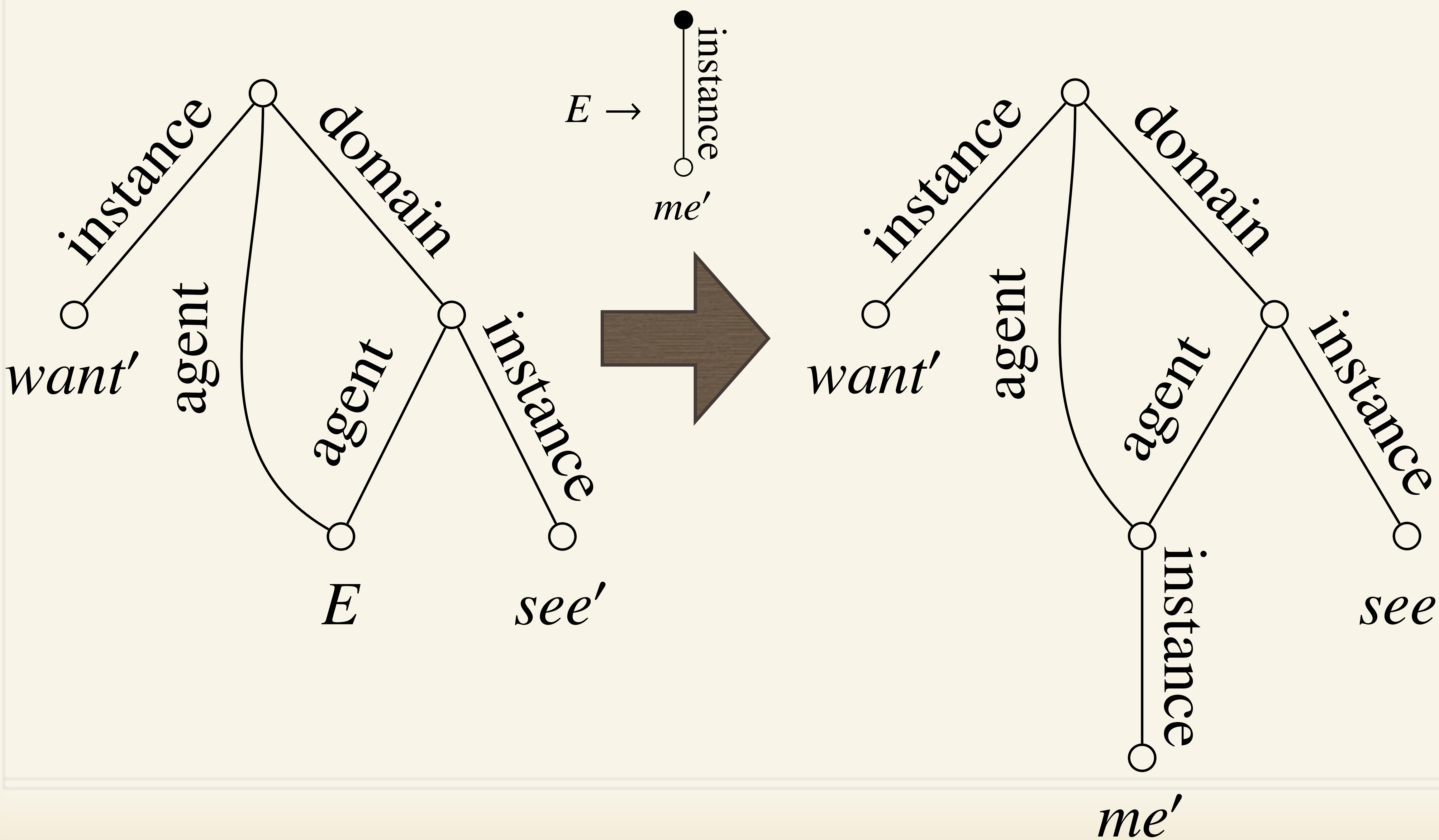
Hyperedge replacement grammars



Hyperedge replacement grammars

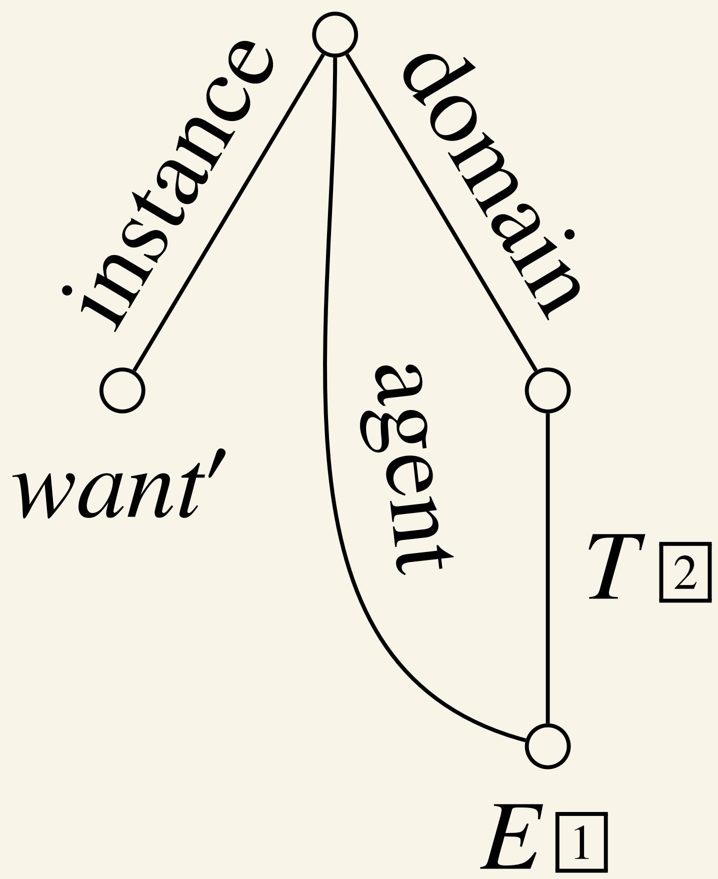


Hyperedge replacement grammars

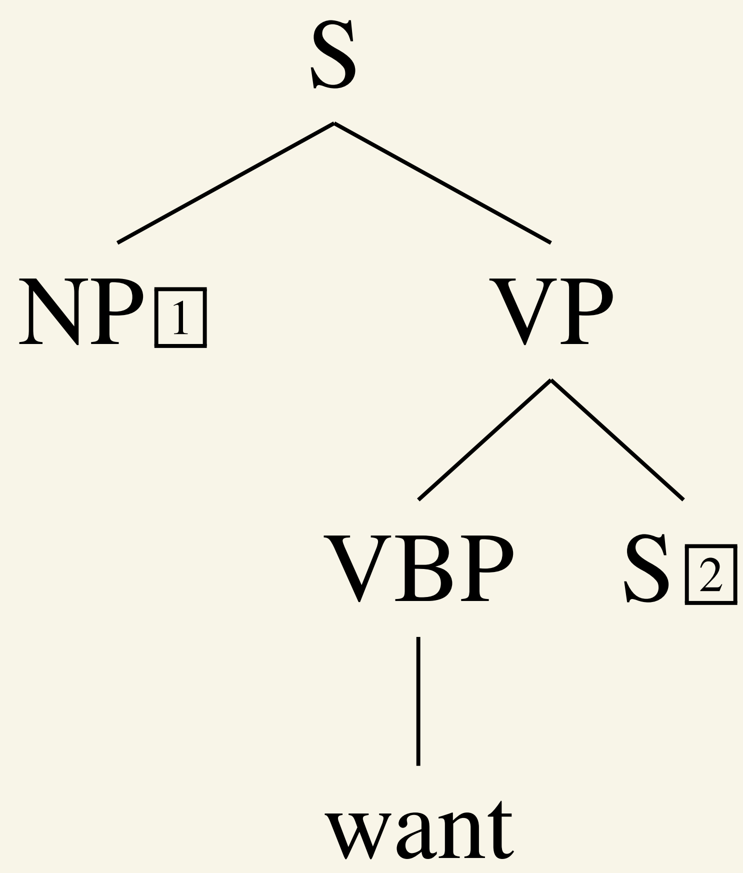


Synchronous HERGs

$T \rightarrow$



$S \rightarrow$



hyperedge replacement grammar

tree substitution grammar

Summary

- Synchronous grammars are useful for various tasks: translation, understanding/generation
- In some ways, they are straightforward (even trivial) extensions of standard formalisms
- But they can add significant complexity