Transfer

Martin Kay

Stanford University and
The University of the Saarland

Spelling Conventions

flap+s → flaps
flash+s → flashes

When *s* has a sibilant and a morpheme boundary to the left and a word boundary to the right, replace it with *es*.

Compile:
s → es / (s | z | c h | s h) + __ #

	after	c	s	z	+	#	h	?
0	*start*	1	2	3	0	0	0	0
1	c	1	2	3	0	0	3	0
2	s	1	2	3	4	0	3	0
3	c h, s h, z	1	2	3	4	0	0	0
4	sib +	1	6	3	0	0	0	0
5	sib + s	1	2	3	0	0	0	0
6	?	1	2	3	0	0	0	0

```
  f   l   a   p   +   s   #
0   0   0   0   0   0   1   0
```

	after	c	s	z	+	#	h	?
0	*start*	1	2	3	0	0	0	0
1	c	1	2	3	0	0	3	0
2	s	1	2	3	4	0	3	0
3	c h, s h, z	1	2	3	4	0	0	0
4	sib +	1	5	3	0	0	0	0
5	sib + s	1	2	3	0	0	0	0
6	?	1	2	3	0	0	0	0

```
  f   l   a   s   h   +   s   #
0   0   0   0   2   3   4   5↔0
```

	after	c	s	z	+:ε	+:e	#:ε	h	?
0	*start*	1	2	3	0		0	0	0
1	c	1	2	3	0		0	3	0
2	s	1	2	3	4	5	0	3	0
3	c h, s h, z	1	2	3	4	5	0	0	0
4	sib +:ε							0	0
5	sib +:e								
6	sib +:ε s	1	2	3	4			3	0
7	sib +:e s						0		
8	?	1	2	3	0		0	0	

Finite-state Transducers

Two languages in parallel

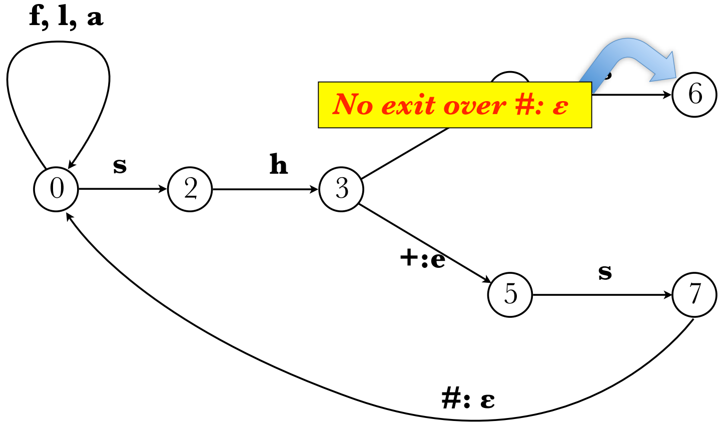
	f	l	a	p	+:ε	s	#:ε	
0	0	0	0	0	0	0	0	

	after	c	s	z	+:ε	+:e	#:ε	h	?
0	*start*	1	2	3	0		0	0	0
1	c	1	2	3	0		0	3	0
2	s	1	2	3	4	5	0	3	0
3	c h, s h, z	1	2	3	4	5	0	0	0
4	sib +:ε	1	6	3	0		0	0	0
5	sib +:e		7						
6	sib +:ε s	1	2	3	4		✗	3	0
7	sib +:e s						0		
8	?	1	2	3	0		0	0	

	f	l	a	s	h	+:ε	s	#
0	0	0	0	2	3	4	6	✗

	after	c	s	z	+:ε	+:e	#:ε	h	?
0	*start*	1	2	3	0		0	0	0
1	c	1	2	3	0		0	3	0
2	s	1	2	3	4	5	0	3	0
3	c h, s h, z	1	2	3	4	5	0	0	0
4	sib +:ε	1	6	3	0		0	0	0
5	sib +:e		7						
6	sib +:ε s	1	2	3	4			3	0
7	sib +:e s						0		
8	?	1	2	3	0		0	0	

	f	l	a	s	h	+:e	s	#
0	0	0	0	2	3	5	7	0



No exit over #: ε

A Context-free Relation

S → <s, ε> <(, ε > NP VP <), ε >
NP → <np, ε> <(, ε > DET N <), ε >
VP → <vp, ε ><(, ε > V NP <), ε >
DET → <det, th
N → <n, dog
N → <n, cat
V → <v, chased>

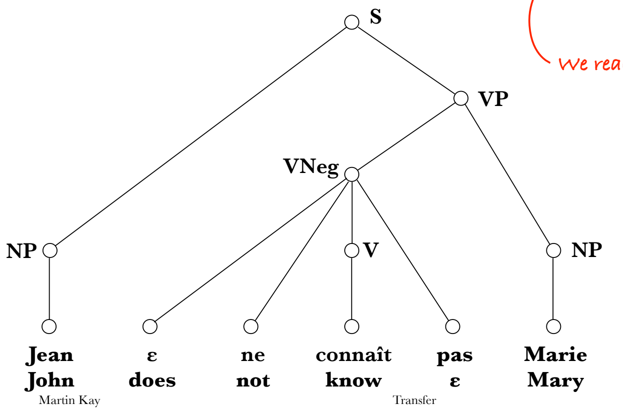
Context-free Transducers

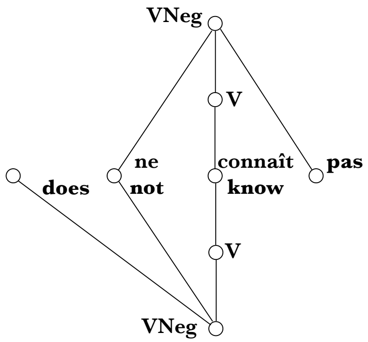
Two languages in parallel

```
s ( np ( det   n ) vp (   v     np ( det   n   )))
         the dog         chased      the cat
```

$S \rightarrow NP_{subj}\ VP_{pred},\ NP_{subj}\ VP_{pred}$
NP → Jean, John
NP → Marie, Mary
$VP \rightarrow VNeg_n,\ VNeg_n$
$VNeg_n \rightarrow \varepsilon$ ne V_v pas, does not V_v ε
V → connaît, know

We really don't need these





$VNeg_n \rightarrow$ ne V_v pas, does not V_v

Suppose I always want this one!

der Mann the man	gab gave	dem Jungen the boy	den Hund the dog
der Mann the man	gab gave	den Hund the dog	dem Jungen the boy
den Hund the dog	gab gave	der Mann the man	dem Jungen the boy
dem Jungen the boy	gab gave	der Mann the man	den Hund the dog

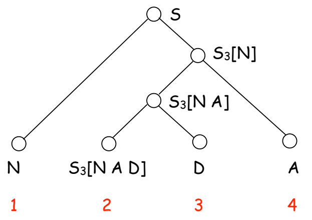
$$\begin{array}{l}
S \rightarrow N_n\ V_v\ D_d\ A_a,\ \overbrace{N_n\ V_v\ N_d\ N_a}^{\textit{All the same}} \\
S \rightarrow N_n\ V_v\ A_a\ D_g,\ N_n\ V_v\ N_d\ N_a \\
S \rightarrow A_a\ V_v\ N_n\ D_g,\ N_n\ V_v\ N_d\ N_a \\
S \rightarrow D_g\ V_v\ N_n\ A_a,\ N_n\ V_v\ N_d\ N_a
\end{array}$$

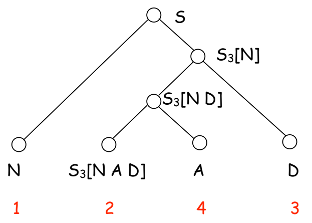
Suppose I also want

"The man gave a dog to the boy"

$$\begin{array}{l}
S \rightarrow N_n\ V_v\ D_d\ A_a,\ N_n\ V_v\ N_d\ N_a \\
S \rightarrow N_n\ V_v\ A_a\ D_g,\ N_n\ V_v\ N_d\ N_a \\
S \rightarrow A_a\ V_v\ N_n\ D_g,\ N_n\ V_v\ N_d\ N_a \\
S \rightarrow D_g\ V_v\ N_n\ A_a,\ N_n\ V_v\ N_d\ N_a
\end{array}$$

$$\begin{array}{l}
S \rightarrow N_n\ V_v\ D_d\ A_a,\ N_n\ V_v\ N_a \text{ to } N_d \\
S \rightarrow N_n\ V_v\ A_a\ D_g,\ N_n\ V_v\ N_a \text{ to } N_d \\
S \rightarrow A_a\ V_v\ N_n\ D_g,\ N_n\ V_v\ N_a \text{ to } N_d \\
S \rightarrow D_g\ V_v\ N_n\ A_a,\ N_n\ V_v\ N_a \text{ to } N_d
\end{array}$$





$S_3[N\ D] \rightarrow S_3[N\ A\ D]\ A$

$S_3[N\ A] \rightarrow S_3[N\ A\ D]\ D$

$S_3[A\ D] \rightarrow S_3[N\ A\ D]\ N$

$S_3[N] \rightarrow S_3[N\ A]\ A$

$S_3[D] \rightarrow S_3[A\ D]\ A$

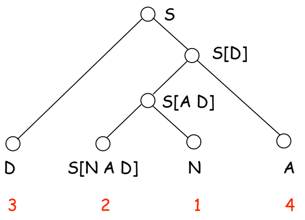
$S_3[N] \rightarrow S_3[N\ D]\ D$

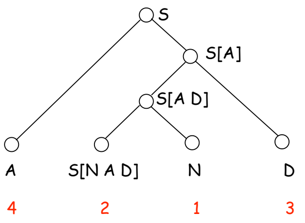
$S_3[A] \rightarrow S_3[A\ D]\ D$

$S \rightarrow A\ S_3[A]$

$S \rightarrow D\ S_3[D]$

$S \rightarrow N\ S_3[N]$





$S_3[N\ D] \rightarrow S_3[N\ A\ D]\ A$

$S_3[N\ A] \rightarrow S_3[N\ A\ D]\ D$

$S_3[A\ D] \rightarrow S_3[N\ A\ D]\ N$

$S_3[N] \rightarrow S_3[N\ A]\ A$

$S_3[D] \rightarrow S_3[A\ D]\ A$

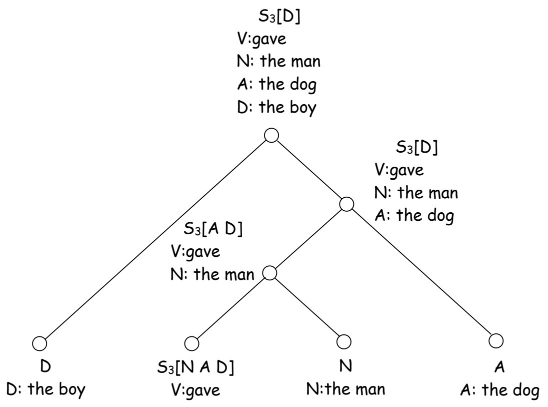
$S_3[N] \rightarrow S_3[N\ D]\ D$

$S_3[A] \rightarrow S_3[A\ D]\ D$

$S \rightarrow A\ S_3[A]$

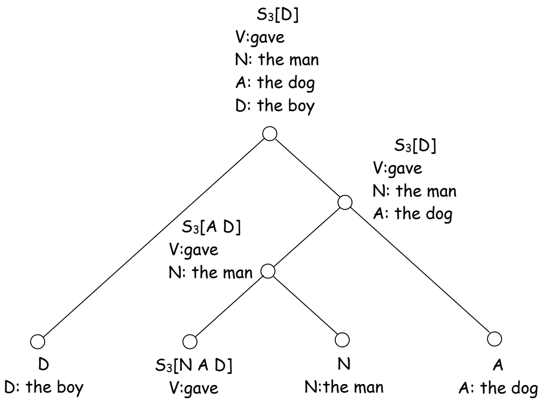
$S \rightarrow D\ S_3[D]$

$S \rightarrow N\ S_3[N]$



Abstract away from word order

S_3[N D] → S_3[N A D] A
S_3[N A] → S_3[N A D] D
S_3[A D] → S_3[N A D] N
S_3[N] → S_3[N A] A
S_3[D] → S_3[A D] A
S_3[N] → S_3[N D] D
S_3[A] → S_3[A D] D
S → A S_3[A]
S → D S_3[D]
S → N S_3[N]

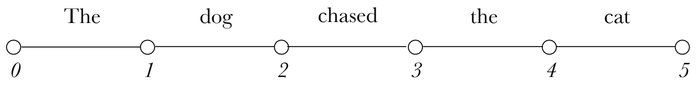


Abstract away from word order ⇒ semantics!

to

1	*2*	*3*	*4*	*5*	
The					*0*
	dog				*1*
		chased			*2*
			the		*3*
				cat	*4*

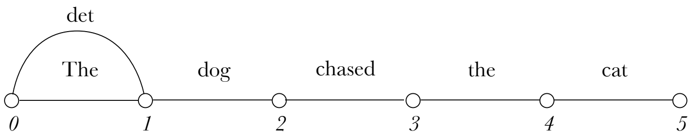
from



to

1	*2*	*3*	*4*	*5*	
det The					*0*
	dog				*1*
		chased			*2*
			the		*3*
				cat	*4*

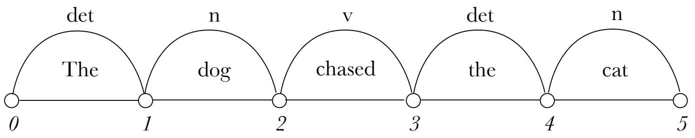
from



to

1	*2*	*3*	*4*	*5*	
det The					*0*
	n dog				*1*
		v chased			*2*
			det the		*3*
				n cat	*4*

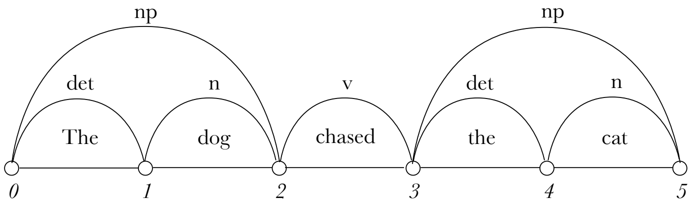
from



to

	1	*2*	*3*	*4*	*5*
0	det The	np			
1		n dog			
2			v chased		
3				det the	np
4					n cat

from



to

	0	*1*	*2*	*3*	*4*	*5*
0	s/[np vp]	det				
1			n			
2				v		
3					det	
4						n
5						

from

Zero-length items!

Active Edges

to

	0	*1*	*2*	*3*	*4*	*5*
0	s/[np vp] np/[det n]	det				
1			n			
2				v		
3					det	
4						n
5						

from

to

	0	*1*	*2*	*3*	*4*	*5*
0	s/[np vp] np/[det n]	det np/[n]				
1			n			
2				v		
3					det	
4						n
5						

from

to

0	1	2	3	4	5	*from*
s/[np vp] np/[det n]	det np/[n]	np				0
		n				1
			v			2
				det		3
					n	4
						5

to

0	1	2	3	4	5	*from*
s/[np vp] np/[det n]	det np/[n]	np s/[vp]				0
		n				1
			v			2
				det		3
					n	4
						5

to

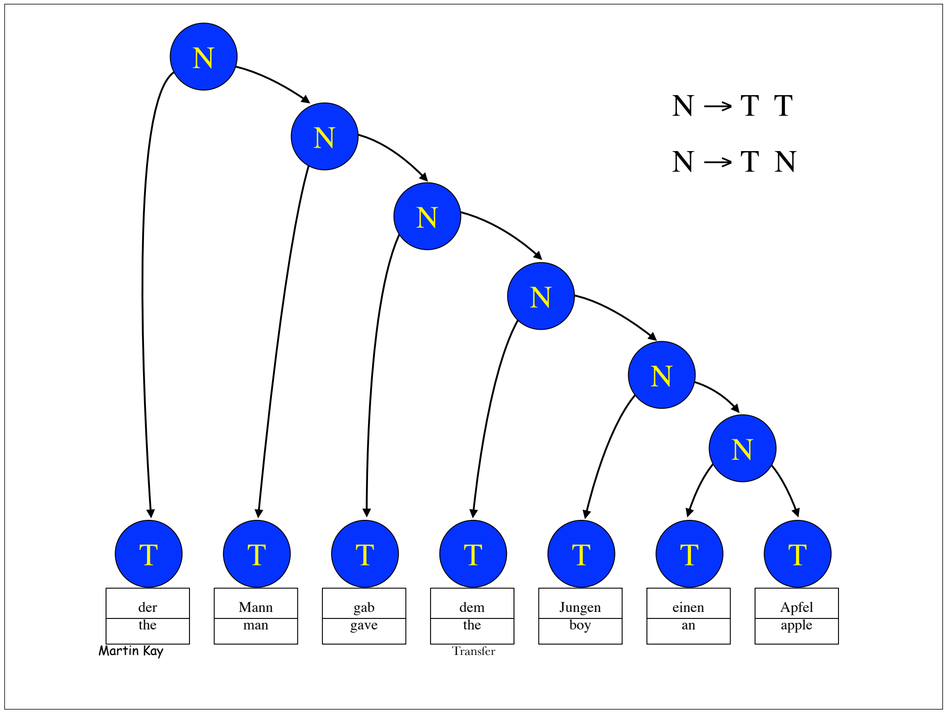
0	1	2	3	4	5	*from*
s/[np vp] np/[det n]	det np/[n]	np s/[vp]				0
		n				1
		vp/[v np]	v			2
				det		3
					n	4
						5

Advantages

- Rules with right sides of arbitrary length
- Top-down guidance

to

0	1	2	3	4	5	*from*
s/[np vp] np/[det n]	det np/[n]	np s/[vp]			s	0
		n				1
		vp/[v np]	v vp			2
			vp/[np] np/[det n]	det np/[n]	np	3
					n	4
						5



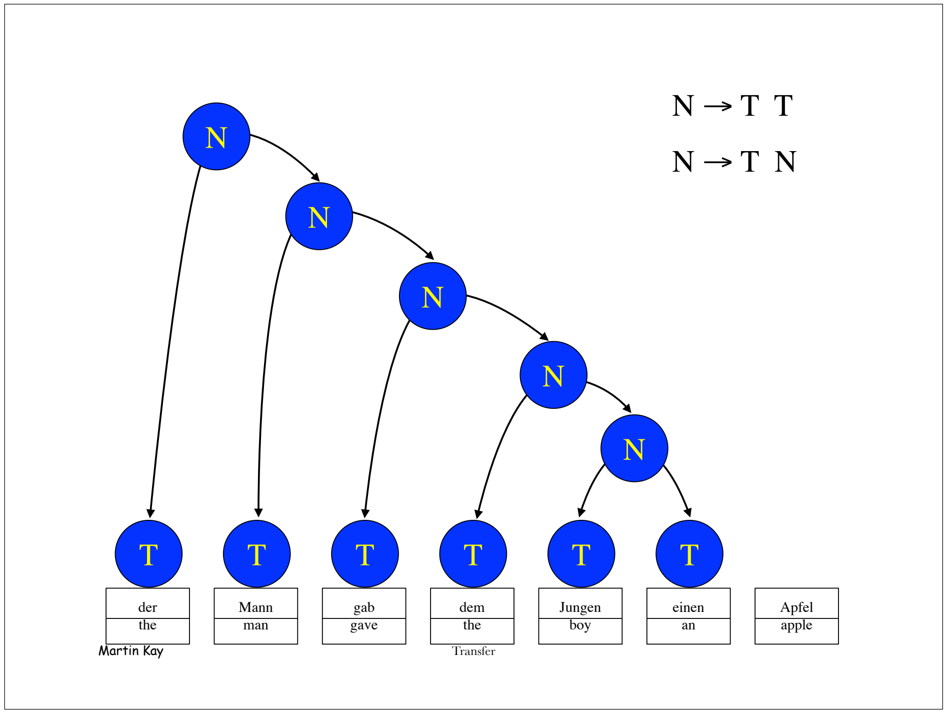
N → T T

N → T N

der	Mann	gab	dem	Jungen	einen	Apfel
the	man	gave	the	boy	an	apple

Martin Kay

Transfer



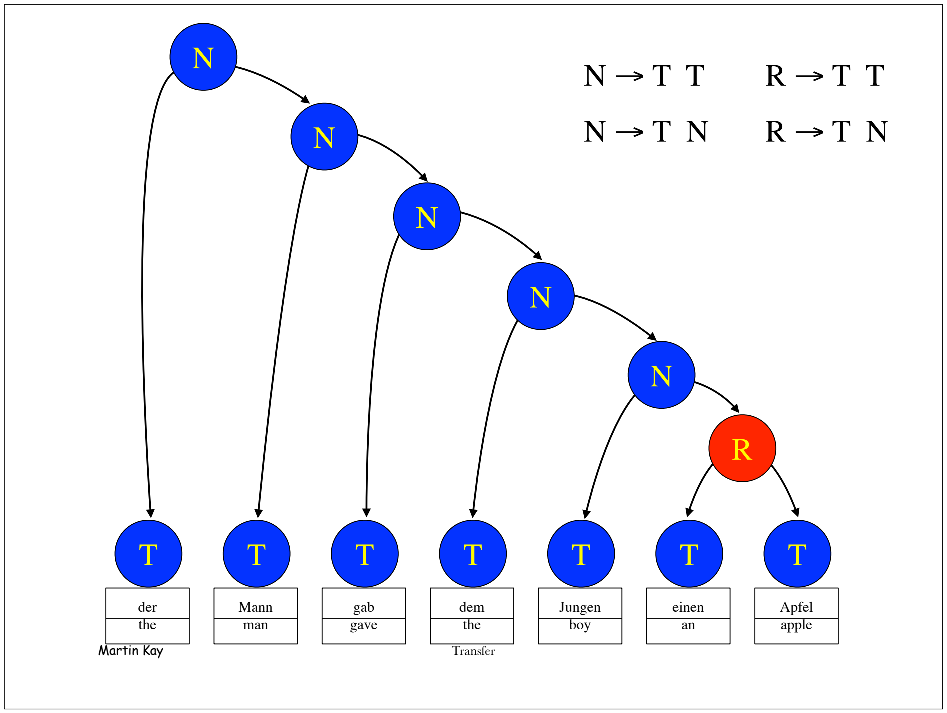
N → T T

N → T N

der	Mann	gab	dem	Jungen	einen	Apfel
the	man	gave	the	boy	an	apple

Martin Kay

Transfer



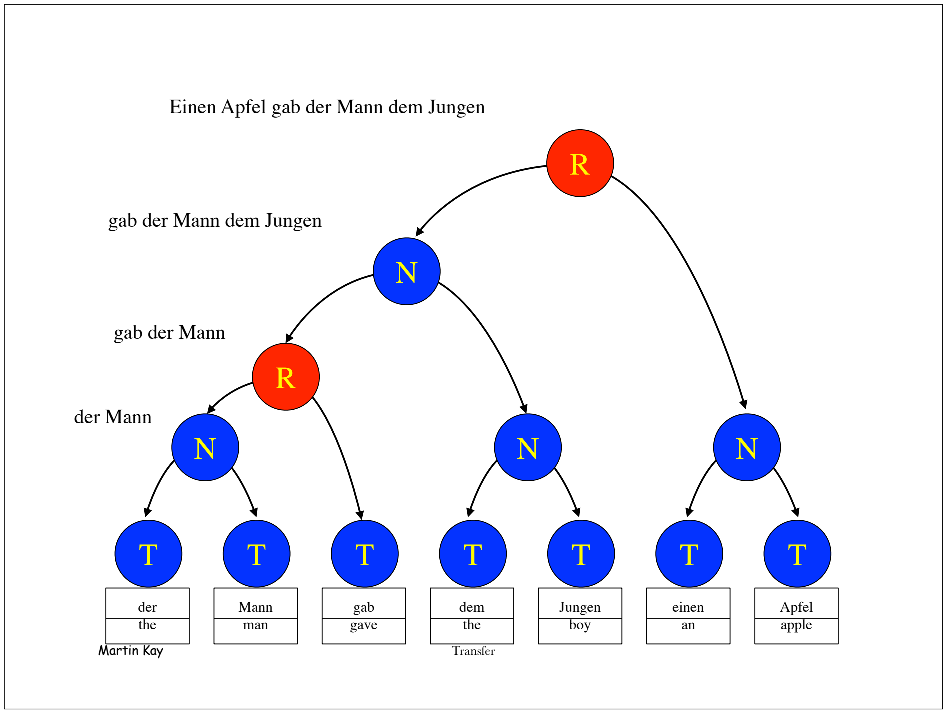
N → T T R → T T

N → T N R → T N

der	Mann	gab	dem	Jungen	einen	Apfel
the	man	gave	the	boy	an	apple

Martin Kay

Transfer



Einen Apfel gab der Mann dem Jungen

gab der Mann dem Jungen

gab der Mann

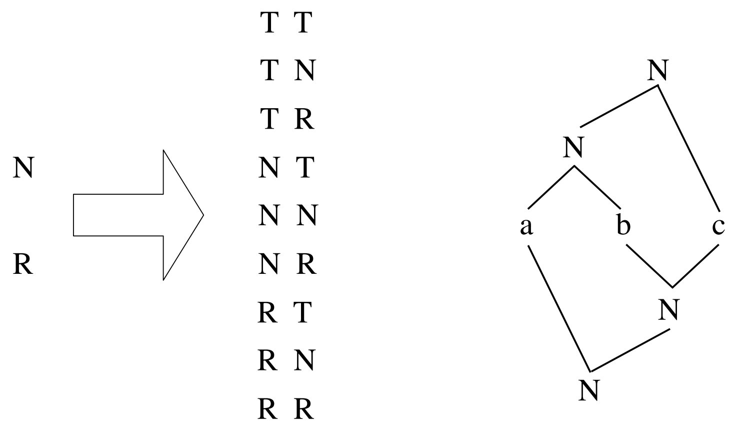
der Mann

der	Mann	gab	dem	Jungen	einen	Apfel
the	man	gave	the	boy	an	apple

Martin Kay

Transfer

Redundant Structures

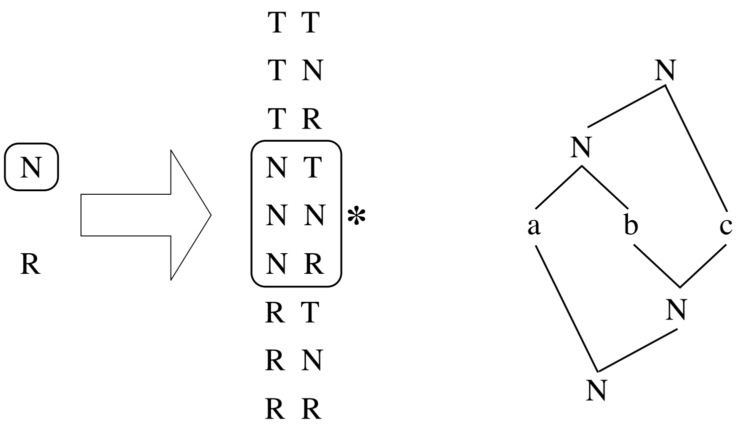


N

R

T T
T N
T R
N T
N N
N R
R T
R N
R R

Redundant Structures

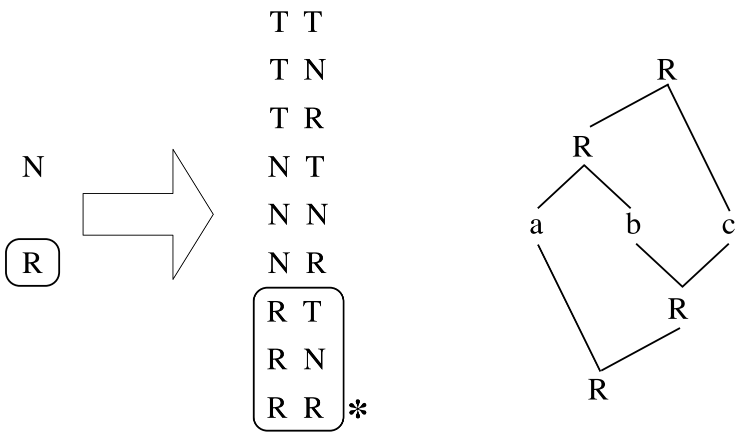


N

R

T T
T N
T R
N T
N N *
N R
R T
R N
R R

Redundant Structures

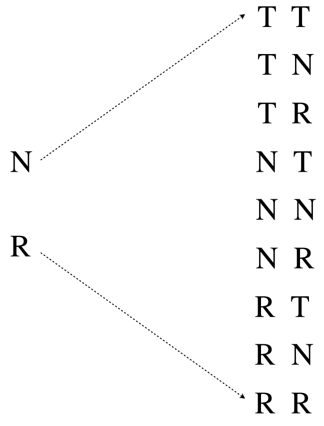


N

R

T T
T N
T R
N T
N N
N R
R T
R N
R R *

Redundant Structures



N

R

T T
T N
T R
N T
N N
N R
R T
R N
R R

18 → 12 rules

Dekai Wu: ***Inversion Transduction Grammars***

Not all reorderings are possible

Den Hund hat der Mann dem Jungen gegeben
1 2 3 4
The man gave the dog to the boy
2 4 1 3

1 2 3 4
Den Hund hat der Mann dem Jungen gegeben
3 1 4 2

Center Embedding

This is the house that Jack built.

The house the malt lay in
built.

This is the rat that ate the malt

That lay in the house that Jack built.

This is the cat that killed the rat

That ate the malt that lay in the house that Jack built.

Center Embedding

This is the house that Jack built.

This is the malt that lay in the house that Jack built.

The house the malt the rat ate lay in

That lay in the house that Jack built.

This is the cat that killed the rat

That ate the malt that lay in the house that Jack built.

Center Embedding

This is the house that Jack built.

This is the malt that lay in the house that Jack built.

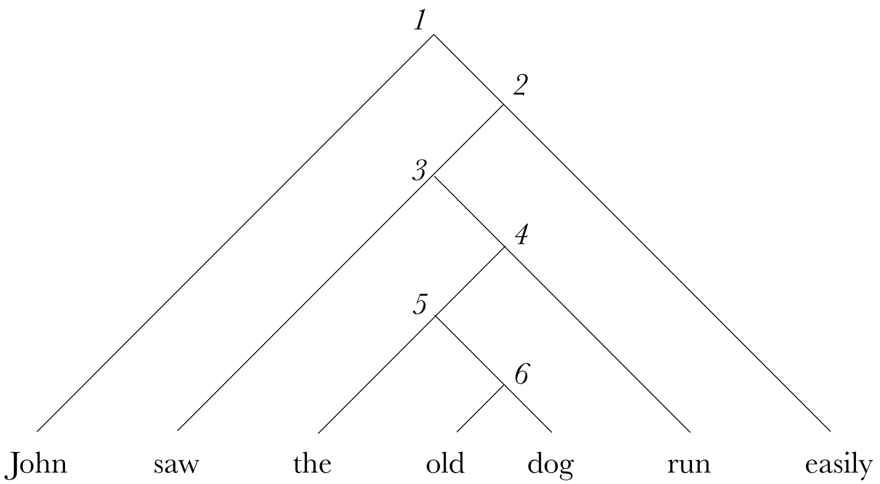
This is the rat that ate the malt

That lay in the house that Jack built.

The house the malt the rat the cat killed ate lay in ck

built.

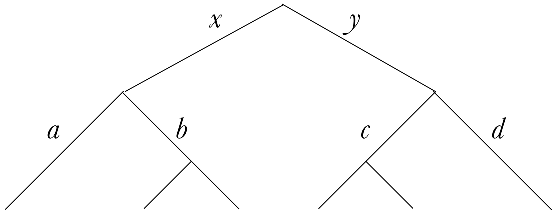
Center Embedding



Computing Embeddings

$$X_{x,y} \rightarrow Y_{a,b}\ Z_{c,d}$$

$$x = \max(a, b+1)$$
$$y = \max(c+1, d)$$



A Touch more Syntax

- **Introduce real grammar rules opportunistically.**
- **With a higher priority than simple ordering rules.**
- **Robustness is not affected.**

The dog saw the cat

dog(d), def(d), saw(s), past(s), cat(c), def(c), argl (s, d), arg2(s, c)

<u>Grammar:</u>

s(x) → np(y) vp(x, y)

vp(x,y) → v(x,y, z) np(z)

np(x) → det(x) n(x)

Lexicon

Word	Cat	Semantics
cat	n(*x*)	*x*: cat(*x*)
saw	v(*x*, *y*, *z*)	see(*x*), past(*x*), arg1(*x*, *y*), arg2(*x*, *z*)
dog	n(*x*)	*x*: dog(*x*)
the	det(*x*)	*x*: def(*x*)

Initial Agenda

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	dog	n(d)	d: dog(d)
s	saw	v(s, d, c)	s: see(s), past(s), arg1(s, d), arg2(s, c)
c	the	det(c)	d: def(d)
	cat	n(c)	d: cat(d)

Initial Agenda

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	the	np(d)/n(d)	d: def(d)
	dog	n(d)	d: dog(d)
s	saw	v(s, d, c)	s: see(s), past(s), arg1(s, d), arg2(s, c)
c	saw	v(s, d)/np(c)	s: see(s), past(s), arg1(s, d)
	the	det(c)	d: def(d)
	the	np(c)/n(c)	d: def(d)
	cat	n(c)	d: cat(d)

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	the	np(d)/n(d)	d: def(d)
	dog	n(d)	d: dog(d)
s	s		arg1(s, d), arg2(s, c)
c	saw	v(s, d)/np(c)	s: see(s), past(s), arg1(s, d)
	the	det(c)	d: def(d)
	the	np(c)/(n(c)	d: def(d)
	cat	n(c)	d: cat(d)

the dog	np(d)	d: dog(d), def(d)

the cat	np(c)	d: cat(c), def(c)

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	the	np(d)/n(d)	d: def(d)
			g(d)
			s: see(s), past(s), arg1(s, d), arg2(s, c)
c	saw	v(s, d)/np(c)	s: see(s), past(s), arg1(s, d)
	the	det(c)	d: def(d)
	the	np(c)/(n(c)	d: def(d)
	cat	n(c)	d: cat(d)
	the cat	np(c)	c: def(c), cat(c)

saw the cat	v(s, d)/np(d)	s: see(s), past(s), arg1(s, d), arg2(s, c), cat(c), def(c)

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	the	np(d)/n(d)	d: def(d)
	dog	n(d)	d: dog(d)
	the dog	np(d)	d: def(d), dog(d)
	saw the cat	v(s, d)/np(d)	s: see(s), past(s), arg1(s, d), arg2(s, c), cat(c), def(c)
s	saw	v(s, d, c)	s: see(s), past(s), arg1(s, d), arg2(s, c)
c	saw	v(s, d)/np(c)	s: see(s), past(s), arg1(s, d)
	the	det(c)	d: def(d)
	the	np(c)/(n(c)	d: def(d)
	cat	n(c)	d: cat(d)
	the cat	np(c)	c: def(c), cat(c)

s: see(s), past(s), arg1(s, d), arg2(s, c)

Vertex	Word	Cat	Semantics
d	the	det(d)	d: def(d)
	the	np(d)/n(d)	d: def(d)
	dog	n(d)	d: dog(d)
	the dog	np(d)	d: def(d), dog(d)
s	saw	v(s, d, c)	s: see(s), past(s), arg1(s, d), arg2(s, c)
	saw the cat	v(s, d)/np(d)	s: see(s), past(s), arg1(s, d), arg2(s, c), cat(c), def(c)
c	saw	v(s, d)/np(c)	s: see(s), past(s), arg1(s, d)
	the	det(c)	d: def(d)
	the	np(c)/(n(c)	d: def(d)
	cat	n(c)	d: cat(d)
	the cat	np(c)	c: def(c), cat(c)