PART 7. TREE ADJOINING GRAMMARS

7.1 TAGS

I have argued that Dutch cross serial dependencies are outside the scope of context free grammars. But how far outside are they? A very useful framework for comparing grammar formalisms for the analysis of natural language that go beyond context free grammars, and for studying these questions, has been developed by Aravind Joshi. It is based on the idea of taking the I-sets and A-sets, that we developed in the course of Parikh's theorem, as grammars in their own right. Such grammars are called tree adjoining grammars.

Before, the notions of I-trees and A-trees were defined as parse trees in a given context free grammar. Since we don't have such a grammar now, we need to define these notions independently.

> A **vocabulary** $V = V_N \cup V_T$ is the union of a finite set V_N of non-terminal symbols, containing S, and a finite set V_T of terminal symbols, where V_N and V_T are disjoint.
>
> An **I-tree in V** is any tree with topnode labeled S, yield in V_T*, non-terminals from V_N on the non-leaves.
>
> An **A-tree in V** is any tree with topnode some non-terminal X in V_N, yield of the form $\alpha X \beta$, with α, β in V_T*, and non-terminals for V_N on the non-leaves.
>
> A **tree adjoining grammar**, **TAG**, is a set of **elementary trees** $E_V = I_V \cup A_V$, where I_V, the set of **initial trees**, is a **finite** set of I-trees in V, and
> A_V, the set of **auxiliary trees**, is a **finite** set of A-trees in V.
>
> **Tree adjunction**:
> Let T be an I-tree or an A-tree in V and let T_X be an A-tree in V with toplabel X, let n be a node in T with label X:
>
> ADJ[T,A,n] is the result of adjoining A in T at node n, where adjunction is as we have defined before.

Note that this definition is a bit wider than what we have defined before: it allows adjunction not only in I-trees, but also in A-trees.

FACT: If T is an I-tree, then ADJ[T,T_X,n] is an I-tree, if defined.
If T is an A-tree, then ADJ[T,T_X,n] is an A-tree, if defined.

We will define for tree adjunction grammars **two** notions of derivation. The first is the notion we have used before, the second is new. The notions will be equivalent for tree adjoining grammars, but not for the later extension of the theory to multiple adjunction.

I. DER: ADJOIN AUXILIARY TREES IN DERIVED I-TREES

$DER_E[I_V]$, the set of **derived I-trees in E_V,** is the smallest set of I-trees in V such that:

1. $I_V \subseteq DER_E[I_V]$
2. If $T \in DER_E[I_V]$ and $T_X \in A_V$, n a node in T,
   then $ADJ[T,T_X,n] \in DER_E[I_V]$, if defined.

On this notion, you always adjoin **into an initial or derived I-tree**, but what you adjoin with is not just an A-tree, but an **auxiliary** tree in A_V.

II. DER*: ADJOIN DERIVED A-TREES IN ELEMENTARY TREES

$DER_E^*[A_V]$, the set of **derived A-trees in E_V**, is the smallest set of A-trees in V such that:

1. $A_V \subseteq DER_E^*[A_V]$
2. If $T_X \in A_V$ and $T_Y \in DER_E^*[A_V]$, and n a node in T_X,
   then $ADJ[T_X,T_Y,n] \in DER_E^*[A_V]$, if defined.

$DER_E^*[I_V]$, the set of **derived I-trees in E_V,** is the smallest set of I-trees in V such that:

1. $I_V \subseteq DER_E^*[I_V]$
2. If $I \in I_V$ and $T_X \in DER_E^*[A_V]$, and n a node in I,
   then $ADJ[I,T_X,n] \in DER_E^*[I_V]$, if defined.

On this notion, you build up, from an elementary auxiliary tree a derived A-tree by adjoining a derived A-tree into it. All the tree building takes place in the A-tree. Finally, you get an I-tree by adjoining the complex A-tree into an initial tree.
So on this notion of derivation, what you adjoin can be a **derived A-tree, but you never adjoin into derived trees, only into elementary trees**.

FACT: For any tree adjunction grammar $E = I_V \cup A_V$:
$DER_E[I_V] = DER_E^*[I_V]$

Thus, for TAGs indeed the two notions of derivation are equivalent.

It can be proved that TAGs, as defined here, are a bit richer than context free grammars, but not a lot, and in fact, not enough to be interesting from a grammatical point of view. That is, while TAGs generate all context free languages and some non-context free languages, none of the non-context free languages that we have discussed are generated (like $a^nb^mc^nd^m$). The reason is, that you cannot **control** where you can adjoin and where not. On the first derivation notion, if T contains label X, it is not just that you can adjoin an auxiary tree with toplabel X in T, but you can adjoin that auxilary tree **at any node** in T where X occurs. This means typically that you cannot prevent unwanted adjunctions.

TREE ADJOINING GRAMMARS WITH CONSTRAINTS

For this reason, tree adjoining grammars have been extended with **constraints**.
Let E_V be a tree adjoining grammar as above.

> A **tree with constraints in A_V** is an I-tree or an A-tree in V with on each non-terminal node n **two constraint labels**: C_n and O_n, where:
> $C_n \subseteq A_V$ and $O_n \subseteq A_V$.
> C_n is called the **constraint set.**
> **O_n** is called the **obligatory adjunction set.**

Thus, the constraint labels that occur at a node are (finite) sets of auxiliary trees from A_V.

C_n is going to mean:
Only trees derived from auxiliary trees in C_n can be adjoined here.
If $C_n = A_V$ there are no constraints on adjunction at node n. In this case I will write * instead of C_n.
If $C_n = \emptyset$ then nothing can be adjoined at node n. In this case I will leave out C_n.

O_n is going to mean:
Some tree derived from one of the trees in O_n must be adjoined at this node.
If $O_n = \emptyset$, there is no obligatory adjunction at this node. In this case I will leave out O_n.

We incorporate these notions as **constraints on adjunction**:

I. WITH DER.

Let T be an I-tree in V with constraints in A_V, n a node with label X, C_n,O_n, and T_X an auxiliary tree with constraints in A_V with with toplabel X. Then:

> ADJ[T,T_X,n] is only defined if $T_X \in C_n$ and if $O_n \neq \emptyset$ then $T_X \in O_n$

II. WITH DER*.

Let $Z \in A_V$
$DER_{E^*}[Z]$, **the set of A-trees derived from Z**, is the smallest set of A-trees such that:
1. $Z \in DER_E^*[Z]$
2. If $T_X \in DER_E^*[A_V]$ and n a node in Z,
then ADJ[Z,T_X,n] $\in DER_E^*[Z]$, if defined.

Let E be an elementary tree with constraints in A_V, n a node with label X, C_n, O_n, and T_X an A-tree with constraints in A_V with toplabel X. Then:

> ADJ[E,T_X,n] is only defined if
> 1. for some $Z \in C_n$: $T_X \in DER_E^*[Z]$
> 2. if $O_n \neq \emptyset$, for some $Z \in O_n$: $T_X \in DER_E^*[Z]$

So, labels C_n and O_n on node n provide **input constraints** on the adjunction at that node. In ADJ[E,T_X,n], the labels C_n and O_n on node n have disappeared and are replaced by the constraint labels of the topnode and bottom leaf node of T_X.

With obligatory constraints, we need to define the set of generated trees separately.

$T(E_V)$, the **set of I-trees generated by** E_V is:
$T(E_v) = \{T \in DER_E[I_V]$: for every node n in T; $O_n = \emptyset\}$
$L(E_V) = \{\alpha$: for some $T \in T(E_V)$; α = yield(T)$\}$

$T^*(E_V)$, the **set of I-trees *-generated by** E_V is:
$T^*(E_V) = \{T \in DER_E^*[I_V]$: for every node n in T; $O_n = \emptyset\}$
$L^*(E_V) = \{\alpha$: for some $T \in T^*(E_V)$; α = yield(T)$\}$

FACT: for every tree adjoining grammar with constraints $E_V = I_V \cup A_V$:
$T(E_V) = T^*(E_V)$ (and hence $L(E_V) = L^*(E_V)$).

So, also for tree adjoining grammars with constraints the two notions of derivation are equivalent.

From now on, when we say tree adjoining grammars, TAGS, we mean tree adjoining grammars with constraints.
The languages generated by tree adjoining grammars are called **tree adjoining languages, TALs.**

TAGS with constraints are more powerful than TAGS without constraints. The reason is that you can impose in an A-tree a different constraint on the top label from the constraint on the non-terminal bottom label: other constraints can be imitated in TAGS without constraints by introducing new category labels, but a TAG without constraints must, by definition, have the same top label and bottom label in an A-tree.

FACT 1: For every context free grammar G there is a TAG E such that T(G) = T(E) (and hence L(G) = L(E)).

FACT 2: There are TAGs E such that L(E) is context free, but for no context free grammar G: T(G) = T(E).

FACT 3: The context free languages are properly contained in the TALs.
The TALs are properly contained in the Index Languages (in fact, the TALs are the Linear Index Languages).

FACT 4: $a^nb^nc^n$, $a^nb^mc^nd^m$, copy language $\alpha\alpha$ are TALs.

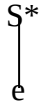
FACT 5.1. There is a pumping lemma for TALs.
$a^nb^nc^nd^ne^n$, $n \geq 1$ is not a TAL, nor is $\alpha\alpha\alpha$ (but see multi adjunction below).
FACT 5.2: Every TAL is semi-linear.
FACT 5.3: Every TAL can be parsed in polynomial time.
(at worst in n^6 steps, where n is the length of the input. For context free languages, as we have mentioned, this is n^3 steps)

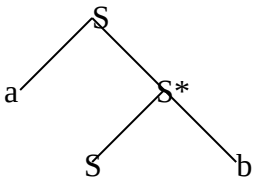
JOSHI: A language is **mildly context sensitive** if it contains limited cross-serial dependencies, is semi-linear, and allows polynomial parsing.

So TALS are examples of mildly context sensitive languages.

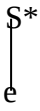
EXAMPLE 1 (showing FACT 2). USING DER

$E = I \cup A$, $I = \{i\}$, $A = \{a\}$

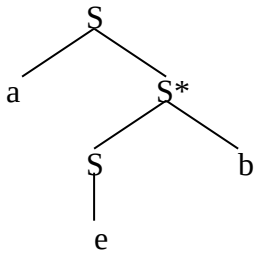
i = 

a = 

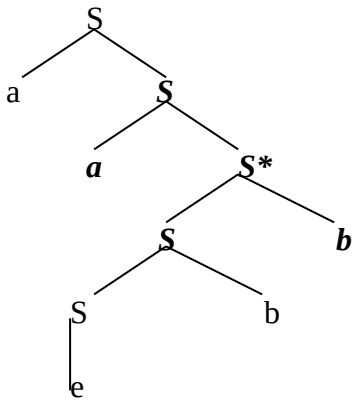
1. $e \in L(E)$:
   $i \in DER_E[I]$

i: 

2. $ab \in L(E)$
   $t_1 = ADJ[i,a,*] \in DER_E[I]$

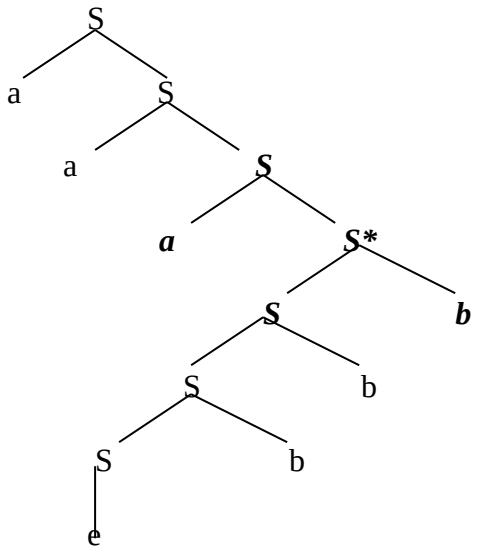
t_1 

3. $aabb \in L(E)$
   $t_2 = ADJ[t_1,a,*] \in DER_E[I]$

t_2 

4. aaabbb ∈ L(E)
   $t_3 = ADJ[t_2,a,*] \in DER_E[I]$

t_3

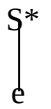


$L(E) = a^n b^n \; n \geq 0$
T(E) is not contextfree.

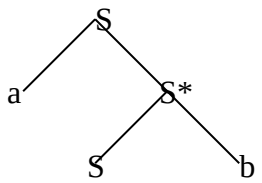
EXAMPLE 1. USING DER*

E = I ∪ A, I = {i}, A = {a}

i =

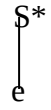


a =



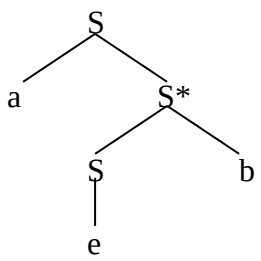
We take the same grammar, but use the other notion of derivation. We derive the same set of trees.
1. e ∈ L*(E):
   $i \in DER_E^*[I]$

i:



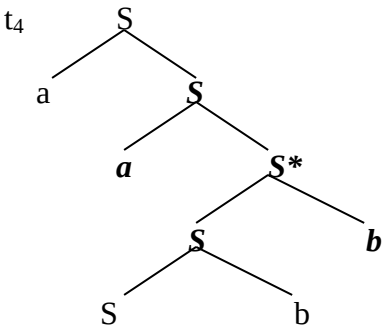
2. ab ∈ L*(E)
   $t_1 = ADJ[i,a,*] \in DER_E^*[I]$

t_1

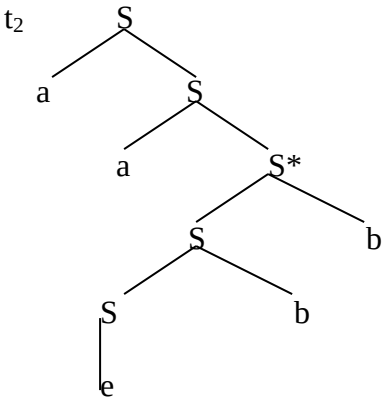


3. aabb ∈ L*(E)

t_4 = ADJ[a,a,*] ∈ DER_E*[A]

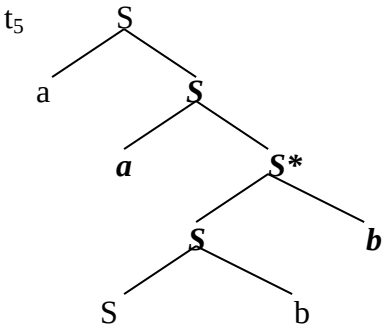


t_2 = ADJ[i,t_4,*] ∈ DER_E*[I]

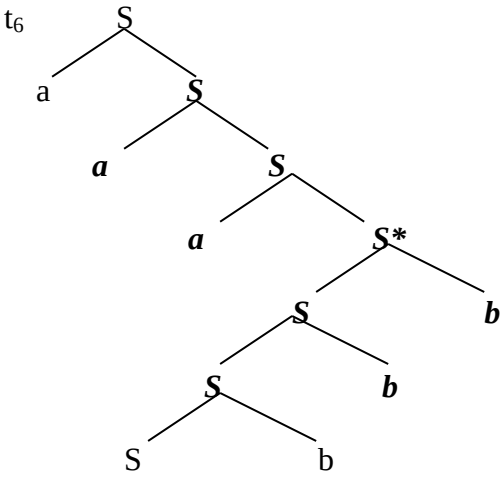


4. aaabbb ∈ L(E)

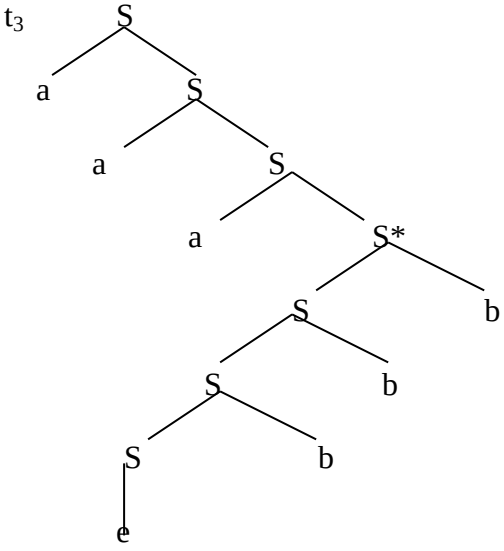
t_5 = ADJ[a,a] ∈ DER_E*[A]



$t_6 = ADJ[a,t_5] \in DER_E^*[A]$

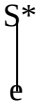


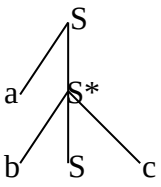
$t_3 = ADJ[i,t_6,*] \in DER_E^*[I]$

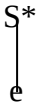


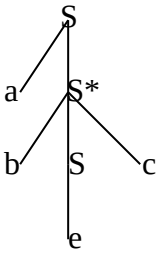
EXAMPLE 2 (USING DER)

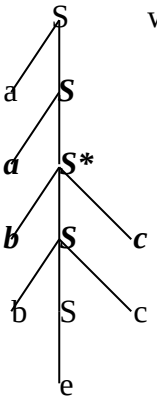
E = I ∪ A, I = {i} A = {a}

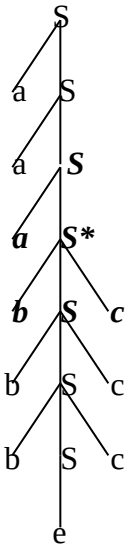
i: 

a_1: 

i: 

with a: 

with a: 

with a: 

$L(E) = a^n b^n c^n$

With DER*: delete e from the example and adjoin into S*.

Without constraints you cannot enforce the alternation of adjunction of a_1 and a_2, and you cannot generate $a^n b^n c^n$.
Without constraints, the above grammar generates the following language L:

$L = \{\alpha c^n: |a|_\alpha = |b|_\alpha = n$ and if $n>o$ α ends in b$\}$

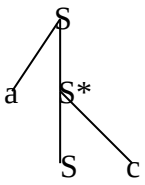
This is not $a^n b^n c^n$: aababbccc $\in$ L.
But L is not context free!
$L \cap a^n b^m c^k, n,m,k \geq 0 = a^n b^n c^n$!

Thus indeed tree adjoining grammars without constraints can generate non-context free languages, but you need constraints to generate $a^n b^n c^n$.

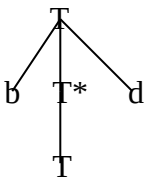
EXAMPLE 3 (USING DER)

$E = I \cup A$, $I = \{i\}$, $A = \{a_1, a_2, a_3\}$

i: 

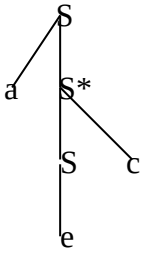
a_1: 

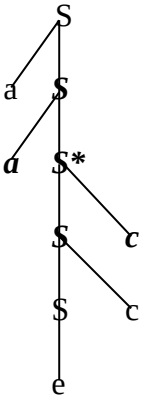
a_2: 

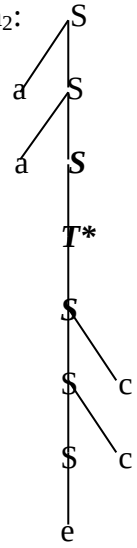
a_3: 

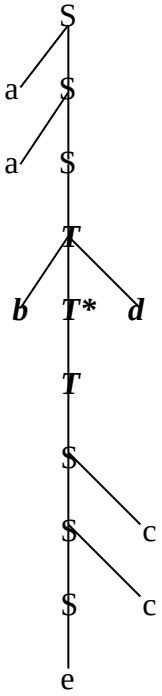
Sample derivation:

i: 

$+a_1$: 

$+a_1$: 

$+a_2$: 

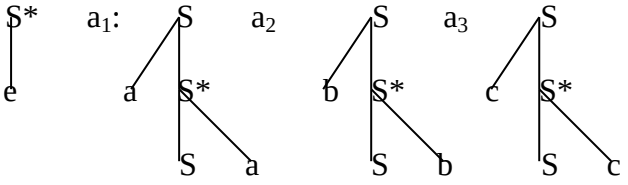
$+a_3$ 

$L(E) = a^n b^m c^n c^m$

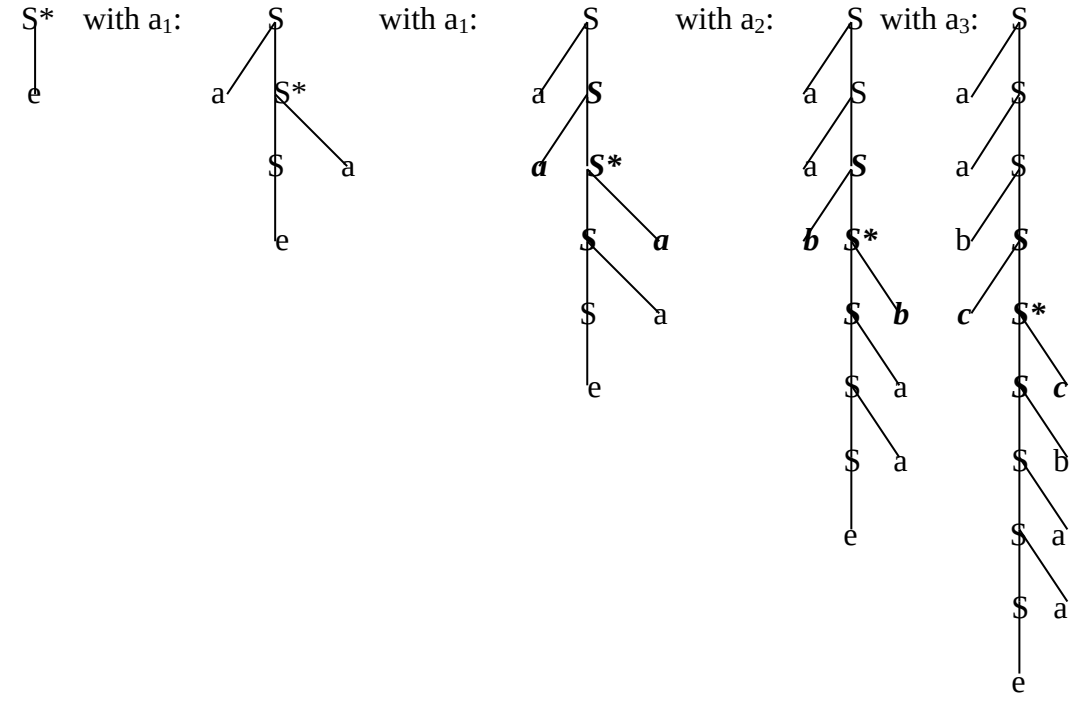
EXAMPLE 4 (USING DER)

$E = I \cup A$, $I = \{i\}$, $A = \{a_1, a_2, a_3\}$

i: S* a_1: S a_2 S a_3 S



i: S* with a_1: S with a_1: S with a_2: S with a_3: S



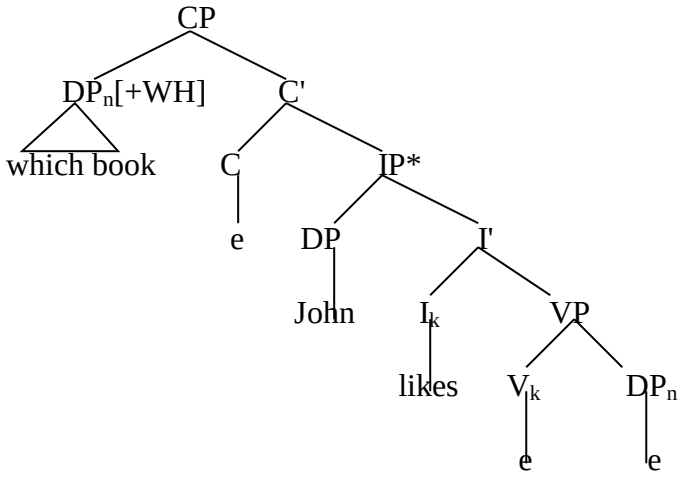
$L(E) = \alpha\alpha$

EXAMPLE 5. JOSHI/KROCH

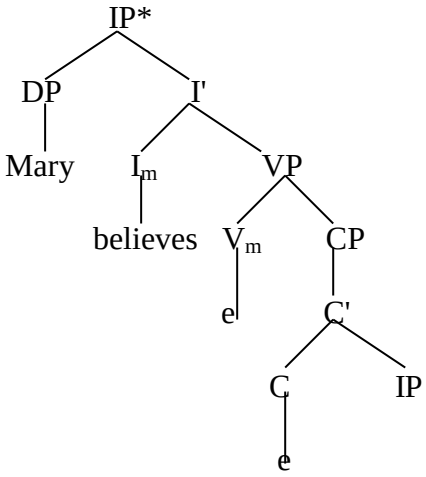
Using TAGs for long distance movement.

Only a simple example here (again,to avoid subject aux inversion, think of the CP as the complement of *Bill wonders*).

i =

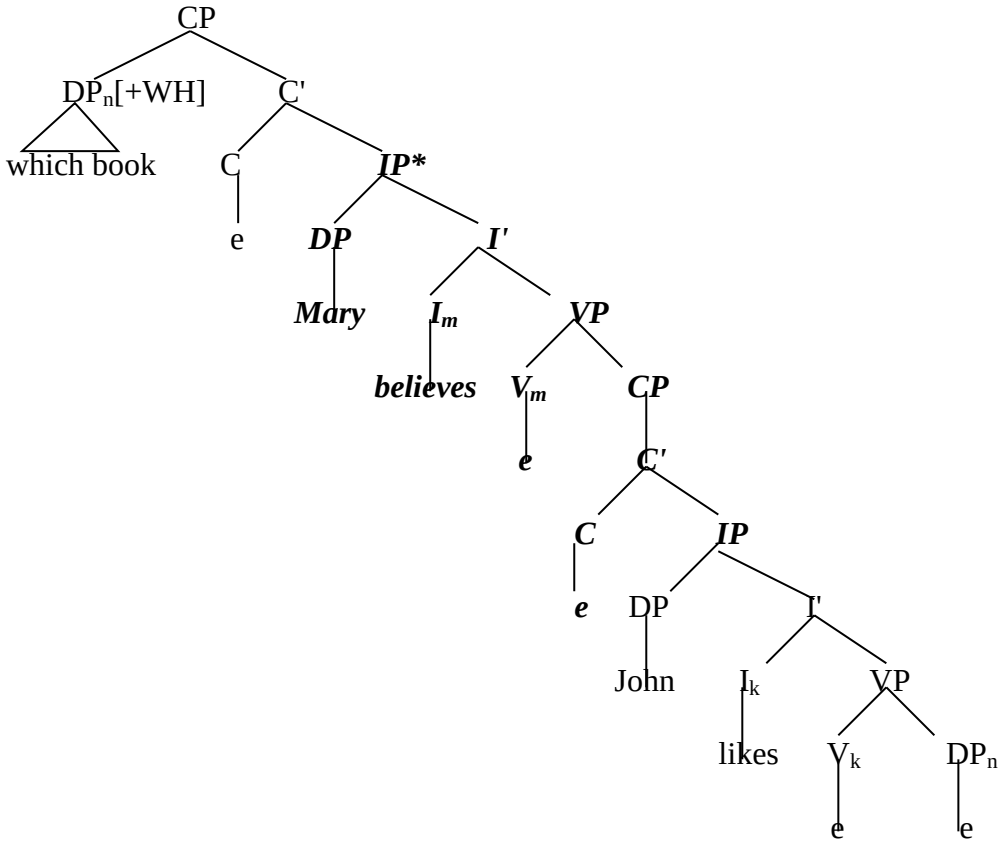


a:



Adjoining a in i gives the effect of long distance movement.

i =

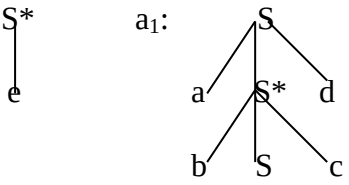


Limitations of TAGs:

We saw can generate $a^n b^n c^n$. With a simple modification of the grammar given above, we can generate $a^n b^n c^n d^n$:

E = I ∪ A, I = {i} A = {a}

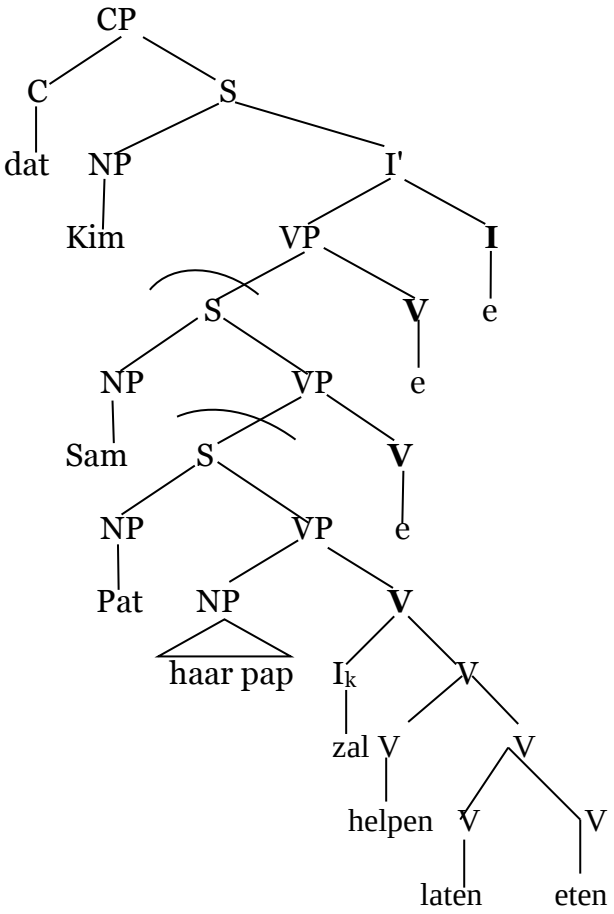
i: S* a$_1$:



This builds up four homogenous stacks simultaneously from the middle: two above the middle, two below the middle. But, obviously, there are only four such places available, hence, this method does not extend to more than four stacks:
$a^n b^n c^n d^n f^n \notin$ TAL.
As David Weir puts it: context free grammars can count up to 2, while TAGs can count up to 4.
Similarly, while we can generate ωω, the method given does not allow you to generate ωωω, and indeed ωωω ∉ TAL.

There are more problems with strong generative capacity.

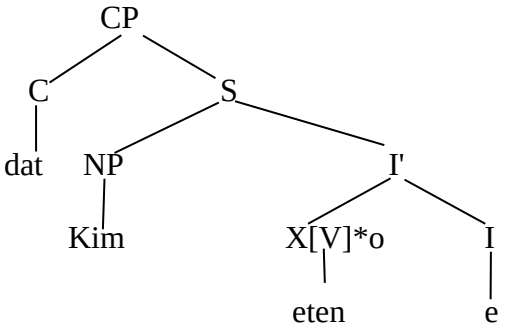
We have seen that in TAGs we can generate cross serial dependencies. But we are more ambitious than that. We really want to generate them with the structures suggested (I spell out the complex verb here as a tree):



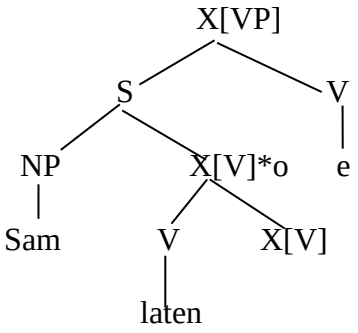
Now, we can do this in a TAG, but only in a non-sensible way.
We need to first change the labels VP and V to labels X[VP] and X[V], where VP and V are features on the *same* label X, so that we can have X[VP] at the top of an A-tree and X[V] at the bottom. (This is not necessarily a problem, you need to do womething similar for long distance extraction from a DP into a CP).

For instance, a TAG with the following I-tree:

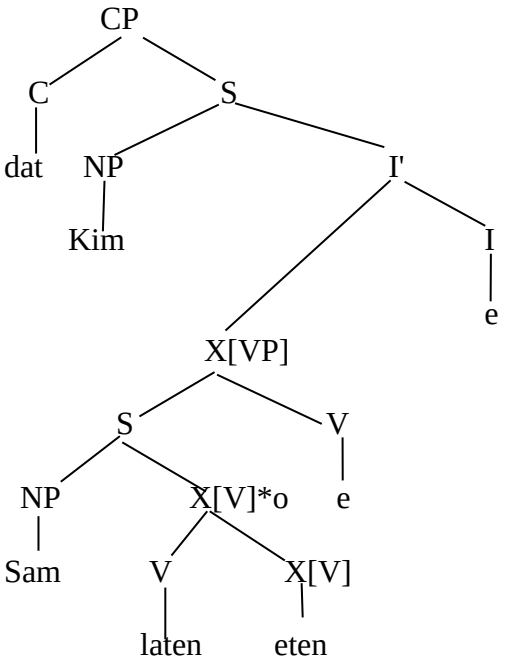
I:



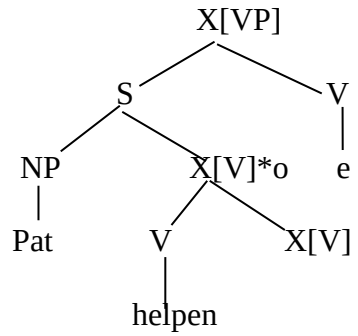
And as A-trees, first:



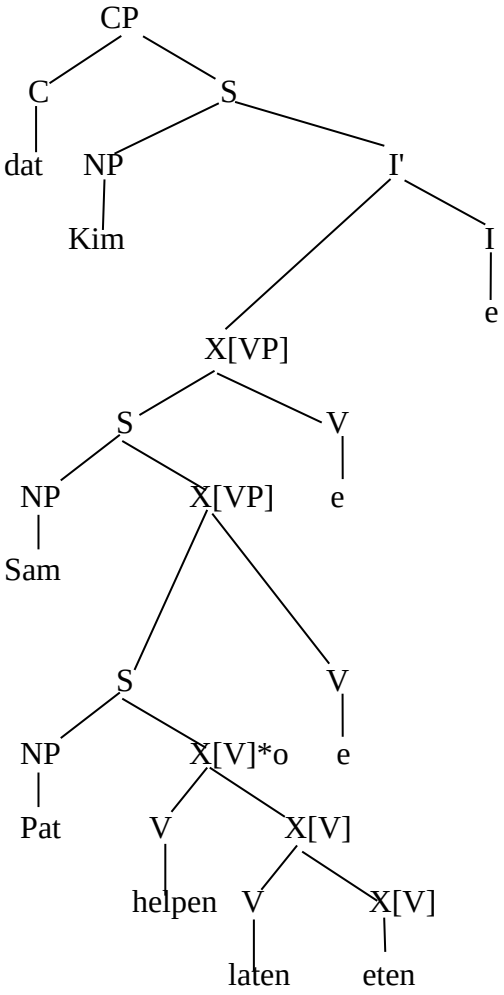
(Here o indicates obligatory adjunction.)
which forms the derived tree:



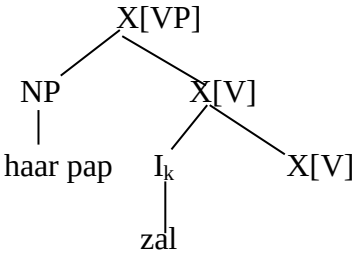
A second A-tree is:



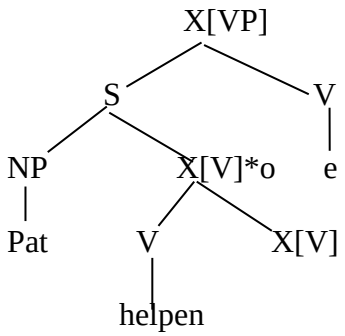
which gives the derived tree:



Finally, we assume the following A-tree:



And this derived the correct structure.
This gramar derives the correct structure, but obviously in a non-sensible way, i.e. it derives it in the wrong way. For instance, the A-tree



is ridiculous, because Pat is in no way the argument of *helpen*.

So, we are not just interested in deriving the cross serial dependencies sentences with the right structures, but also with reasonable derivations, meaning, for instance, that they have to be derived with A-trees that have a sensible semantic interpretation.

We will not formulate the constraints here, but assume again some minimal consensus notion: if you *can* derive a structure with grammar formalisms of a certain power, but only with a derivation that no sensible linguist would entertain for independent linguistic reasons (i.e. more than that the linguist happens not to like say TAGS), the structure falls, for all linguistically relevant purposes outside the scope of the grammars of that power. In this sense, Dutch and German crossserial dependencies fall outside the capacity of TAGS: you cannot derive the correct structures with TAGS in a sensible way. This is where multi adjunction comes in.

7.2. MULTI ADJUNCTION

Multi adjunction is simultaneous adjunction. The multi adjunction that we will consider **increases** the generative power with respect to TAGs. But the resulting multi adjunction TAGs are mildly contextsensitive.

Let $E_V = I_V \cup A_V$ be a TAG.

A **multi component TAG based on E_V** is a set of **elementary tree sets** $\mathcal{E}_V = \mathcal{I}_V \cup \mathcal{A}_V$
where: $\mathcal{A}_V$ is a (finite) set of subsets of A .
The set of **auxiliary tree sets**.
$\mathcal{I}_V$ is a (finite) set of singleton subsets of I.
The set of **initial tree sets.**

Constraints on multi adjunction:
MCADJ[$\mathcal{E}$,$\mathcal{A}$,π] is only defined if:

1. $\mathcal{A}$ is a **derived auxilary tree set** (defined below).
2. $\mathcal{E}$ is an **elementary tree set** (i.e. $\mathcal{E}$ is in $\mathcal{I}_V$ or in $\mathcal{A}_V$)
3. π is a function that maps every tree T_A in $\mathcal{A}$ onto a pair $<T_E,n>$, where $T_E \in \mathcal{E}$ and n is a node in T_E such that ADJ[T_E,T_A,n] is defined and such that for every T_1,T_2 in $\mathcal{A}$: $\pi(T_1) \neq \pi(T_2)$.
(This means that no two trees in $\mathcal{A}$ can be adjoined in the **same** tree at the **same** node. Yes, the same tree, but then at different nodes.)

When defined, MCADJ[$\mathcal{E}$,$\mathcal{A}$,π] is again a tree set:

$$MCADJ[\mathcal{E},\mathcal{A},\pi] = \{ ADJ[T_{\pi 1(T)},T, \pi 2(T)]: T \in \mathcal{A} \}$$

where π1(T) is the first element of the pair π(T) (a tree in $\mathcal{E}$)
and π2(T) is the second element of the pair π(T) (a node in that tree in $\mathcal{E}$).

We need to similarly generalize the notions of constraints and obligatory adjunction. I will not specify that here.
(Note: the above conditions are Weir's original conditions, for discussion and improvement, see Kallmeyer 2005.)

ADJOIN DERIVED AUXILIARY TREE SETS INTO ELEMENTARY TREE SETS (DER $\mathcal{E}$*)

DER $\mathcal{E}$*($\mathcal{A}_V$) is the smallest set of tree sets such that:
1. $\mathcal{A}_V \subseteq$ DER $\mathcal{E}$*($\mathcal{A}_V$)
2. If $\mathcal{A} \in \mathcal{A}_V$ and $\mathcal{T} \in$ DER $\mathcal{E}$*($\mathcal{A}_V$) then MCADJ[$\mathcal{A}$,$\mathcal{T}$,π] $\in$ DER $\mathcal{E}$*($\mathcal{A}_V$), if defined.

DER $\mathcal{E}$*($\mathcal{I}_V$) is the smallest set of tree sets such that:
1. $\mathcal{I}_V \subseteq$ DER $\mathcal{E}$*($\mathcal{I}_V$)
2. If $\mathcal{I} \in \mathcal{I}_V$ and $\mathcal{T} \in$ DER $\mathcal{E}$*($\mathcal{A}_V$) then MCADJ[$\mathcal{I}$,$\mathcal{T}$,π] $\in$ DER $\mathcal{E}$*($\mathcal{I}_V$), if defined.

FACT: The members of DER $\mathcal{E}$*($\mathcal{I}_V$) are singleton tree sets.

T($\mathcal{E}_V$) = {T: {T} $\in$ DER $\mathcal{E}$*($\mathcal{I}_V$) and T has no obligatory constraints}
L($\mathcal{E}_V$) = {α: for some tree T in T($\mathcal{E}_V$): α = yield(T)}

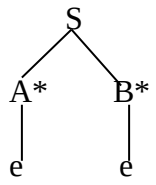
FACT 1: MCTAGs are stronger than TAGs in weak and strong generative capacity.
FACT 2: Every MCTAL is semi linear.
FACT 3: $a_1^n \ldots a_m^n$, $n \geq 0 \in$ MCTAL
$copy_n[\alpha] \in$ MCTAL (n copies of α)
FACT 4: $a_1^n \ldots a_m^n$, $n \geq 0$, $m \geq 0 \notin$ MCTAL
$copy[\alpha] \notin$ MCTAL (unbounded many copies of α)

where:
$copy_1[\alpha] = \alpha$ = $\{\omega: \omega \in A^*\}$
$copy_2[\alpha] = \alpha\alpha$ = $\{\omega\omega: \omega \in A^*\}$
$copy_n[\alpha] = copy_{n-1}[\alpha]\alpha$ = $\{\beta\omega: \beta \in copy_{n-1}[\alpha]$ and $\omega \in \alpha\}$
$copy[\alpha] = \cup_{n\geq 1}[\alpha \ldots \alpha]_n$

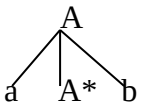
EXAMPLE 1:

$\mathcal{E} = \mathcal{I} \cup \mathcal{A}$, $\mathcal{I} = \{\{i\}\}$ $\mathcal{A} = \{\{a_1,a_2\},\{a_3,a_4\}\}$

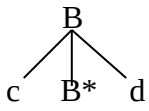
i:



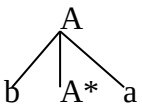
a_1:



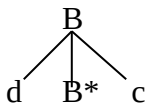
a_2:



a_3:



a_4:

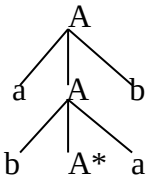


Let:

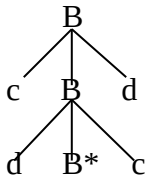
$\pi(a_3) = \langle a_1, A^* \rangle$
$\pi(a_4) = \langle a_2, B^* \rangle$

MCADJ[$\{a_1,a_2\},\{a_3,a_4\}, \pi$] = $\{t_1,t_2\}$

t_1:



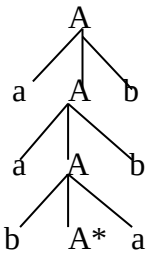
t_2:



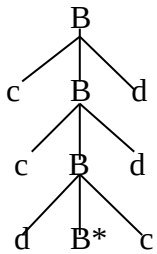
$\pi(t_1) = \langle a_1, A^* \rangle$
$\pi(t_2) = \langle a_2, B^* \rangle$

MCADJ[$\{a_1,a_2\}$ $\{t_1,t_2\}, \pi$] = $\{t_3,t_4\}$

t_3:



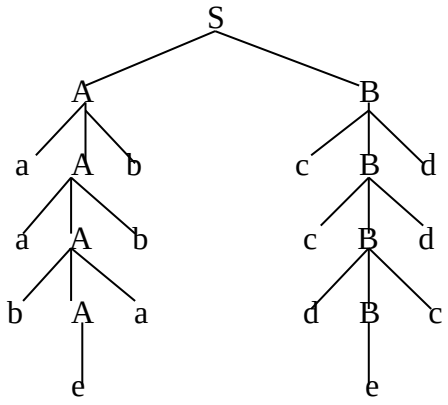
t_4



$\pi(t_3) = \langle i, A^* \rangle$
$\pi(t_4) = \langle i, B^* \rangle$

MCADJ[{{i}}, $\{t_3,t_4\}$, π] = $\{\{t_5\}\}$

:

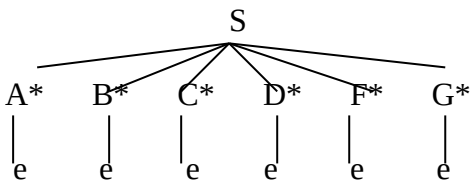


Generated string:
aababbccdcdd

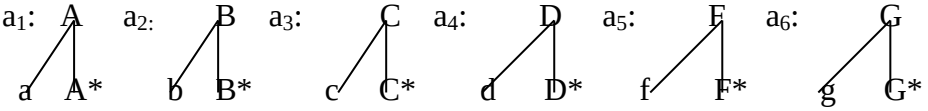
What more MCTAGs can generate depends on the size of the auxiliary tree sets. Unlike TAGs, you can in principle generate: $a_1^n ... a_k^n$, with k some finite bound.

EXAMPLE 2.
$\mathcal{E} = \mathcal{I} \cup \mathcal{A}$, $\mathcal{I} = \{\{i\}\}$, $\mathcal{A} = \{\{a_1,a_2,a_3,a_4,a_5,a_6\}\}$

i:

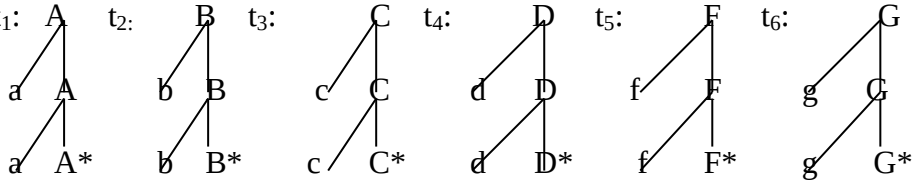


a_1: A a_2: B a_3: C a_4: D a_5: F a_6: G

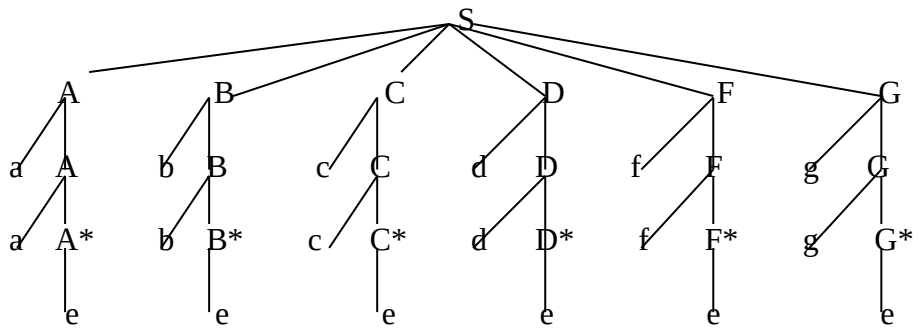


1. Adjoin $\{a_1,a_2,a_3,a_4,a_5,a_6\}$ into $\{a_1,a_2,a_3,a_4,a_5,a_6\}$. This gives $\{t_1,t_2,t_3,t_4,t_5,t_6\}$:

t_1: A t_2: B t_3: C t_4: D t_5: F t_6: G



2. Adjoin $\{t_1,t_2,t_3,t_4,t_5,t_6\}$ into $\{\{i\}\}$. This gives $\{\{t_7\}\}$:



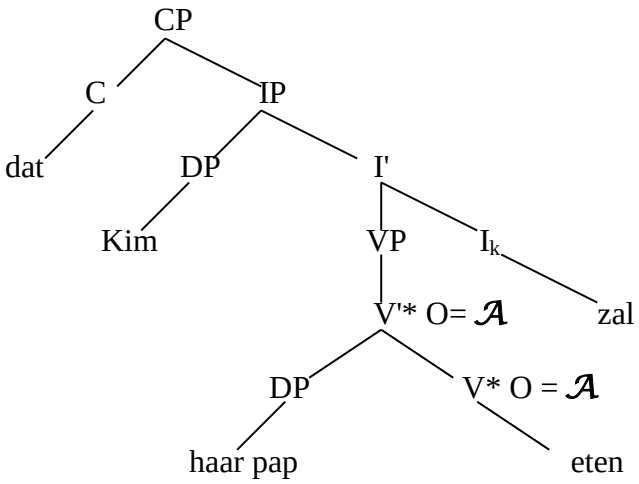
Generated language: $a^n,b^nc^nd^nf^ng^n$, $n\geq 0$.

We can reformulate this in terms of copy languages as well.
Clearly, a similar argument shows that you can generate for any n, the copy language α…α(n times), the set of all strings consisting of n-copies of strings in Σ^*.
But what you cannot, for instance, generate with MCTAGS is a language of **unbounded copies,** like $\cup_{n>0}$ **α…α(n times).**

EXAMPLE 3: DUTCH CROSS SERIAL DEPENDENCIES
Finally, we come back to the Duth cross serial dependencies.
In the following grammar, I will only deal with the incorporation of the infinitives, not the tensed auxiliary. That is, I will take the tensed auxiliary as part of the I-trees.

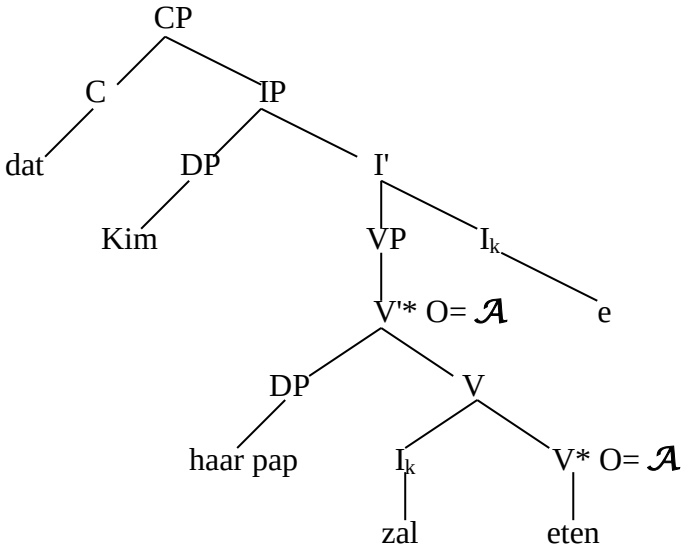
$\mathcal{E} = \mathcal{I} \cup \mathcal{A}$, $\mathcal{I} = \{\{i_1\},\{i_2\}\}$, $\mathcal{A} = \{\{a_1,a_2\}\}$

i_1:



(Semantics: λ*R*.WILL(*R*(k,EAT)))

i_2:

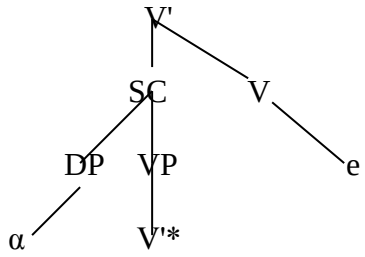


(Semantics: λ*R*.WILL(*R*(k,EAT)))

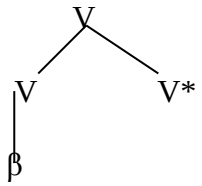
Th obligatory constraint says that some multi-adjunction must take place at the V'* and V* node.

{a_1,a_2}:

a_1:



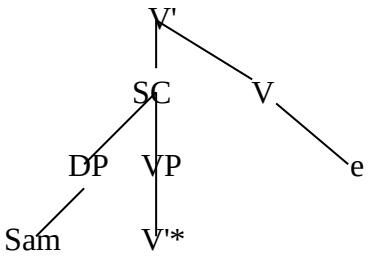
a_2:



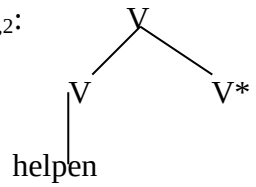
where α ∈ {Sam, Pat} and β ∈ {helpen, laten}

This is a schema, so among others we have:

$a_{1,1}$:

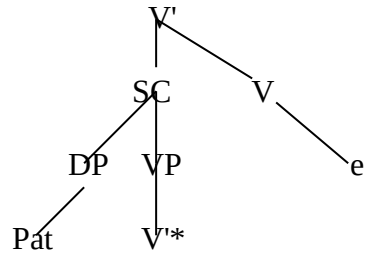


$a_{1,2}$:

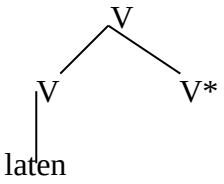


(Semantics: λPλx.HELP(x,P(s)))

$a_{2,1}$:



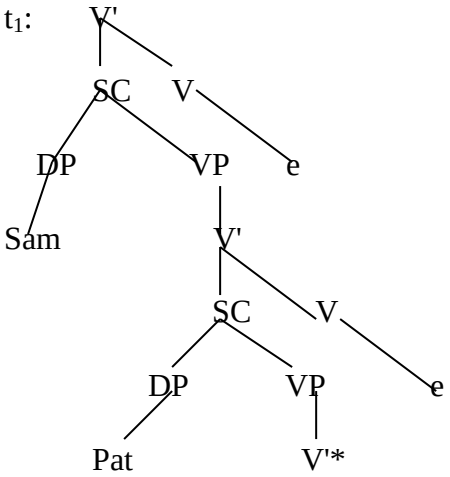
$a_{2,2}$:



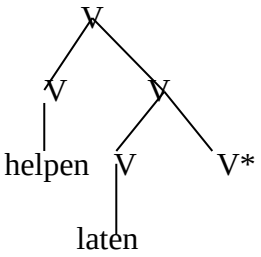
(Semantics: λPλx.LET(x,P(p)))

1. Adjoin {a_1(Pat),a_2(laten)} into {a_1(Sam),a_2(helpen)}. This gives {t_1,t_2}:

t_1:

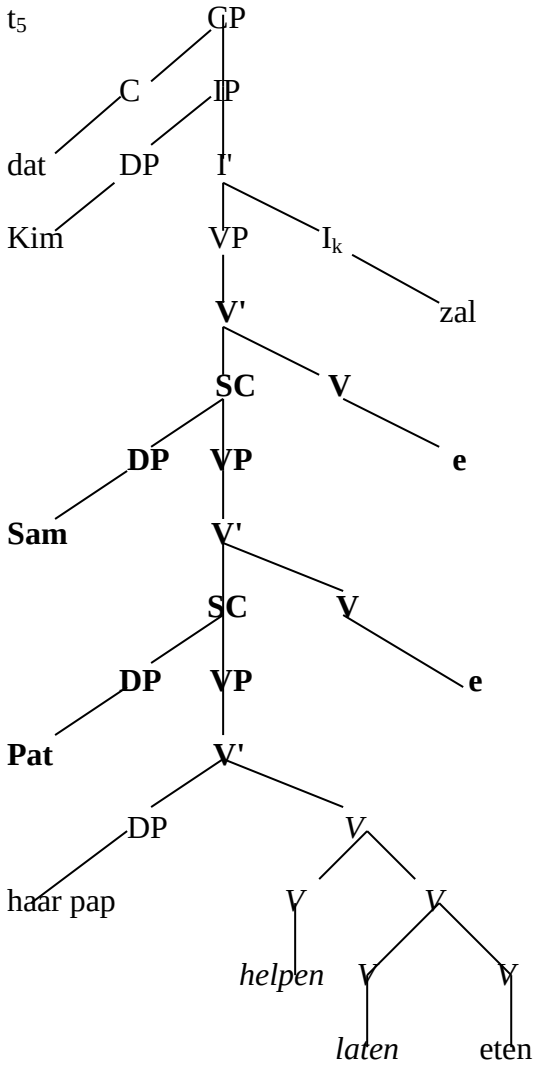


t_2:

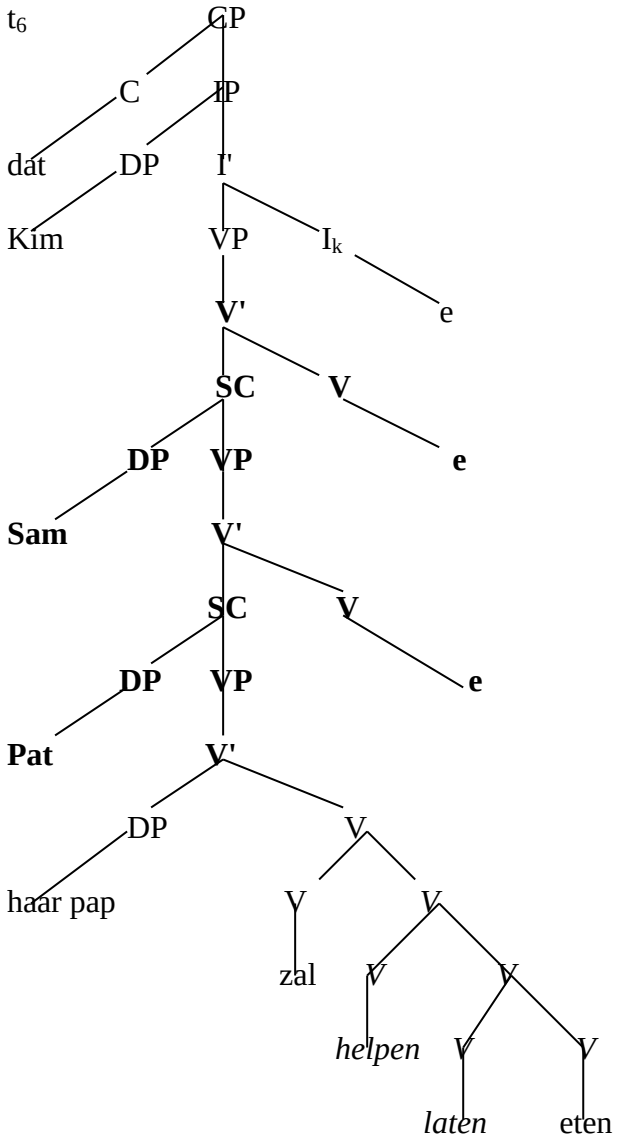


(Semantics: λPλx.HELP(x,LET(s,P(p)))

Adjoining $\{t_1,t_2\}$ into $\{i_1\}$ gives t_5, into $\{i_2\}$ gives t_6.



(Semantics: WILL(HELP(k,LET(s,EAT(p,pap))))



(Semantics: WILL(HELP(k,LET(s,EAT([p,pap))))

What we see then is that Dutch cross serial dependencies are both weakly and strongly within the capacity of multi component TAGs, and hence within the class of phenomena that can be analyzed with mildly context sensitive tools (but see below).

This **does** tell us something about the power of incorporation. Even if we represent incorporation through a restructuring operation, we have now analyzed the 'computational content' of that operation: we need an operation that goes beyond simple context free tree building: already the weak generative capacity argument for surface structure strings shows that. While simple adjunction is enough to get the strings, TAGs do not give you the right trees from the right parts.

The point about cross serial constructions is that the verb taking a small clause and the small clause subject must match. In terms of adjunction, they ought to be introduced simultaneously. In TAGs this means that they should be part of the same auxiliary tree. But they can't be in these structures.

But we can build up the tree by building up simultaneously the small clause structure and the verb structure, each with simple adjunction, and use multi adjunction to build the whole structure. This is what MCTAGs allow us to do.

7.3. LINEAR CONTEXTFREE REWRITE SYSTEMS

Linear Contextfree Rewrite Systems were introduced in Vijay-Shanker, Weir and Joshi 1987. They are really a generalization of head grammars (in the format of Vijay-Shanker 1987) and proved to be weakly equivalent to MCTAGs in Weir 1988. (For much more about LCFRSs see Kallmeyer 2013).

Since I think that the standard format of rewriting rules is obscuring somewhat what is going on, I will use a format that brings that out most clearly.

In linear contextfree rewrite systems non-terminal symbols play a double role. They are, as usual, category labels, but they are also sets of grammatical objects and the rules are easier to understand if we think of them not as rewrite rules but as rules determining the contents of these sets. For non-terminal A, I will use A inside the rules and **A** for its interpretation as a set.

A *Linear Contextfree Rewrite System* is a structure $G = <V_N, V_T, VAR, dim, S, R>$ where:

1. V_N is a finite set of non-terminal symbols
2. V_T is a finite set of terminal symbols
3. VAR is a finite set of variables
4. $V_N \cap V_T = \emptyset$ $V_N \cap VAR = \emptyset$ $V_T \cap VAR = \emptyset$
5. $S \in V_N$
6. dim, *the fan-out function* is a function from V_N to $\mathbb{N}$
   $dim(S) = 1$
7. R is a set of rules specified below.

We associate with every non-terminal A a set **A** of dim(A)-tuples
$<\alpha_1,\ldots,\alpha_{dim(A)}>$ with $\alpha_1,\ldots,\alpha_{dim(A)} \in V_T^*$

The intuition is that instead of generating string $\alpha_1\ldots\alpha_{dim(A)}$, we generate substrings $\alpha_1,\ldots,\alpha_{dim(A)}$ and allow two things that we wouldn't allow for a string in a context free grammar:

-the positions in between the substrings are still open for grammatical manipulation: $\alpha_1\uparrow\alpha_2\uparrow\ldots.\uparrow\alpha_{dim(A)}$ (which is what happens in a limited form in head grammars)

- the linear order of the substring is still open for grammatical manipulation.

dim(A), the fan-out of category A, determines how many positions open for grammatical manipulation category A fixes.

The general form of the rules is:

$x \in A$ ψ
$[\textit{if}\ x_1 \in A_1\ \textit{and}\ \ldots\ \textit{and}\ x_m \in A_m\ \textit{then}\ x \in A]$ (if φ then ψ)

More specifically, rules are of the form:

$[<\omega_1,\ldots,\omega_{dim(A)}> \in A]$ where $\omega_i \in V_T^*$ (for every index i)

or

$[\textit{if}\ <X^1_1,\ldots,X^1_{dim(A_1)}> \in A_1\ \textit{and}\ \ldots\textit{and}\ <X^m_1,\ldots,X^m_{dim(A_m)}> \in A_m\ \textit{then}\ <\omega_1,\ldots,\omega_{dim(A)}> \in A]$
where $\omega_i \in (V_T \cup VAR)^*$, $X^j_i \in VAR$, $A_i \in V_N$ (for every index i,j)

Constraint: Let r ∈ R and r = [*if* φ *then* ψ]
Every variable X that occurs in r occurs *exactly once* in φ and *exactly once* in ψ

It is this constraint that keeps the formalism mildly contextsensitive.

Let $A \in V_{N.}$
A, the *yield* of A, is the smallest set such that:
1. If r = [$<\omega_1,\ldots,\omega_{dim(A)}> \in A$] then $<\omega_1,\ldots,\omega_{dim(A)}> \in \mathbf{A}$
2. If r = [*if* $<X^1_1,\ldots,X^1_{dim(A^1)}> \in A_1$ *and* …*and* $<X^m_1,\ldots,X^m_{dim(A^m)}> \in A_m$ *then* $<\omega_1,\ldots,\omega_{dim(A)}> \in A$]
and $<\beta^1_1,\ldots,\beta^1_{dim(A^1)}> \in \mathbf{A_1}$ and… and $<\beta^m_1,\ldots,\beta^m_{dim(A^m)}> \in \mathbf{A_m}$
then $<\alpha_1,\ldots,\alpha_{dim(A)}> \in \mathbf{A}$, where $\alpha_k = \omega_k[\beta^j_i / X^j_i]$, the result of replacing every variable X^j_i in ω_k by phrase β^j_i

$L(G) = \{\alpha: <\alpha> \in \mathbf{S}\}$

An n-LCFRS grammar is an LCFRS grammar with fan-out at most n.
An n-LCFRS language is a language that has an n-LCFRS grammar.

This looks more complex than it is.

Example 1: $a^n b^n c^n$, $n > 0$
1. [<a,b,c> ∈ A]
2. [*if* <X,Y,Z> ∈ A *then* <aX, bY, cZ> ∈ A]
3. [*if* <X,Y,Z> ∈ A *then* <XYZ> ∈ S]

1. <a,b,c> ∈ **A**	1. <a,b,c> ∈ **A**	1. <a,b,c> ∈ **A**
2. <abc> ∈ **S**	2. <aa,bb,cc> ∈ **A**	2. <aa,bb,cc> ∈ **A**
3. abc ∈ L(G)	3. <aabbcc> ∈ **S**	3. <aaa,bbb,ccc> ∈ **A**
	4. aabbcc ∈ L(G)	4. <aaabbbccc> ∈ **S**
		5. aaabbbccc ∈ L(G)

Example 2: $a^n b^n c^n d^n f^n$, $n \geq 0$
1. [<e,e,e,e,e> ∈ A]
2. [*if* $<X_1, X_2, X_3, X_4, X_5> \in A$ *then* $<aX_1, bX_3, cX_3, dX_4, fX_5> \in A$]
3. [*if* $<X_1, X_2, X_3, X_4, X_5> \in A$ *then* $<X_1X_2X_3X_4X_5> \in S$]

Example 3: ααα

1. [<e,e,e> ∈ A]
2. [*if* <X,Y,Z > ∈ A *then* <Xa, Ya, Yc> ∈ A]
3. [*if* <X,Y,Z > ∈ A *then* <Xb, Yb, Yb> ∈ A]
4. [*if* <X,Y,Z > ∈ A *then* <Xc, Yc, Yc> ∈ A]
5. [*if* <X,Y,Z > ∈ A *then* <XYZ> ∈ S]

1. <e,e,e> ∈ **A** [by 1]
2. <a,a,a> ∈ **A** [by 2]
3. <ab,ab,ab> ∈ **A** [by 3]
4. <abb,abb,abb> ∈ **A** [by 3]
5. <abbc,abbc,abbc> ∈ **A** [by 4]
6. < abbcabbcabbc > ∈ **S** [by 5]
7. abbcabbcabbc ∈ L(G)

Example 4:

1. [<haar pap,eten> ∈ A]
2. [*if* <X,Y> ∈ A *then* <Pat X, Y> ∈ B]
3. [*if* <X,Y> ∈ B *then* <Sam X, laten Y> ∈ B]
4. [*if* <X,Y> ∈ B *then* <Kim X, helpen Y> ∈ B]
5. [*if* <X,Y> ∈ B *then* <X, heeft Y> ∈ C]
6. [*if* <X,Y> ∈ B *then* <X, Y heeft> ∈ C]
7. [*if* <X,Y> ∈ C *then* <XY> ∈ D]
8. [<X> ∈ D then <dat X> ∈ S]

1. < haar pap, eten> ∈ **A**
2. < *Pat* haar pap, eten> ∈ **B**
3. < *Sam* Pat haar pap, *laten* eten> ∈ **B**
4. < *Kim* Sam Pat haar pap, *helpen* laten eten> ∈ **B**

5a. < Kim Sam Pat haar pap, *heeft* helpen laten eten> ∈ **C**
5b. < Kim Sam Pat haar pap, helpen laten eten *heeft*> ∈ **C**
6a. < Kim Sam Pat haar pap heeft helpen laten eten> ∈ **D**
6b. < Kim Sam Pat haar pap helpen laten eten heeft> ∈ **D**
7a. <dat Kim Sam Pat haar pap heeft helpen laten eten> ∈ **S**
7b. <dat Kim Sam Pat haar pap helpen laten eten heeft> ∈ **S**
8a. dat Kim Sam Pat haar pap heeft helpen laten eten ∈ L(G)
8b. dat Kim Sam Pat haar pap helpen laten eten heeft ∈ L(G)

Example 5

Haar pap helpen laten eten heeft Kim Sam Pat
1. [<haar pap, eten> $\in A_1$]
2. <e, Pat> $\in A_2$]
3. [*if* $\langle X_1,X_2\rangle \in A_1$ *and* $\langle X_3,X_4\rangle \in A_2$ *then* $\langle X_1,X_2,X_3,X_4\rangle \in A$]
4. [*if* $\langle X_1,X_2,X_3,X_4\rangle \in A$ then <e X_1,helpen X_2,X_3,Sam X_4> $\in$ A]
5. [*if* $\langle X_1,X_2,X_3,X_4\rangle \in A$ then <e X_1,laten X_2,X_3,Kim X_4> $\in$ A]
6. [*if* $\langle X_1,X_2,X_3,X_4\rangle \in A$ then $\langle X_1,X_2$,heeft X_3, $X_4\rangle \in B$]
7. [*if* $\langle X_1,X_2,X_3,X_4\rangle \in B$ then $\langle X_1X_2X_3X_4\rangle \in S$]

1. <haar pap, eten> $\in \mathbf{A_1}$
2. <e, Pat> $\in \mathbf{A_2}$
3. <haar pap, eten, e, Pat> $\in$ **A**
4. <haar pap, helpen eten, e, Sam Pat> $\in$ **A**
5. <haar pap, laten helpen eten, e, Kim Sam Pat> $\in$ **A**
6. <haar pap, laten helpen eten, heeft, Kim Sam Pat> $\in$ **B**
7. <haar pap laten helpen eten heeft Kim Sam Pat> $\in$ **S**
8. <haar pap laten helpen eten heeft Kim Sam Pat> $\in$ L(G)

The inspiration of LCRSs comes from string manipulation, but if you look at the derivations given under example 3, you see that the derivation in the grammars given puts the strings together much in the way that we did in the MCTAG above (and example 4 generalizes that). This means that rewriting this grammar as a tree grammar that builds the same trees as in the MCTAG should be manageable.

PROPERTIES OF n-LCFRS languages:

LCFRSL = the class of languages generated by LCFRS grammars

FACT 1: LCFRSL is closed under union, concatenation, Kleene closure, intersection with a regular language, substitution.

FACT 2: all languages in LCFRS are semi-linear, allow polynomial parsing: there is pumping lemma for LCFRSL

FACT 3: LCFRSL = MCTAL

7.4. BEYOND CONTEXT FREE

JOSHI: A language is **mildly context sensitive** if it contains limited cross-serial dependencies, is semi-linear, and allows polynomial parsing.

Constant Growth
Joshi actually uses a weaker notion that semi-linearity, the Constant Growth Property:

> a language has the **constant growth property** if its length of its strings grows with increments smaller than a constant.

The discussion in Bhatt and Joshi 2004 makes it clear that Joshi is really thinking of semilinearity, rather than constant growth.

While semilinearity is a highly interesting property when it comes to natural languages, the constant growth property, when taken at face-value, is trivially satisfied by natural languages: take a number n such that there exists an English sentence of length n. What is the next number up where there exists an English sentence? Well, obviously n+1 (stick in an adjective or an adverb).

So I think we are better off defining mild contextsensitivity in terms of semi-linearity.

Stable language classes
What we find is that in going beyond context free, we reach a first stable class of grammar formalisms and languages in tree adjoining languages.

> TAL: the class of languages generated by:
> -Tree adjoining grammars
> -Combinatory categorial grammars
> -Head grammars
> -Indexed grammars with linear expansion (indices are copied onto one non-termimial daughter only).
>
> TAL $\subseteq$ IL

A wide variety of grammar formalisms turned out to be weakly equivalent and generate the mildly contextsensitive class of tree adjoining languages TAL.
By the equivalence with indexed languages with linear expansion, we know that TALs are indexed languages.

Arguments concerning strong generative capacity: the preferred structures of cross serial dependencies made us look for grammars that go beyond TAGs.
We formulated multi component tags, MCTAGs and their generated languages MCTALs.

MCTAL: the class of languages generated by:
-Multi Component Tree Adjunction Grammars
-Linear Contextfree Rewriting Systems
-Multiple Contextfree Grammars
-Simple Range Concatenation Grammars (see Kallmeyer 2013)
-Stabler's formalization of Minimalist Grammar (Stabler 2010) [with a caveat given below].

TAL $\subseteq$ MCTAL
MCTAL is not a subset of IL

MCTALs are mildly contextsensitive, and form a class of languages MCTAL that properly includes TAL. Arguably, MCTAL forms a second stable class of mildly contextsensitive grammar formalisms and languages.

MCTAL and IL do not stand in an inclusion relation with respect to each other.

-Obviously there are indexed languages that are not in MCTAL: a^{2^n} is not-semilinear.
-But there are also languages in MCTAL that are not indexed languages.
This was proved in Engelfriet and Skyum 1976, I have not been able to find easy examples to illustrate this.

Recent development (rather surprisingly) Sylvain Salvati, Salvati 2015:
MIX = $\{\alpha \in \{a,b,c\}^*: |a|_\alpha = |b|_\alpha = |c|_\alpha\}$, the permutation closure of $a^nb^nc^n$.
Salvati proved that MIX is in MCTAL. In fact, MIX can be generated by a LCFRS with a fan-out of only 2!

This is surprising, because this means that the notion of 'locality' imposed on MCTAGs does apparently not exclude languages like MIX in which the dependencies do not seem very local, at least, at first sight. In other words, MCTAGs are more powerful than originally thought. (And possibly strain the notion of 'limited cross-serial dependencies' more than originally thought.)

But remember that Bill Marsh in the 1980ies conjectured that MIX is not an indexed language. That conjecture is still open. So, at this point MIX is an example of a language known to be an MCTAL but of which it is not known whether or not it is an indexed language.

Constraints on LCFRS grammars excluding MIX restrict LCFRS to **well nested LCFRSs, introduced by** Uwe Mönnich (Mönnich 2010). See Kallmeyer for definitions and discussion.

We will not discuss this further, rather we discuss challenges to the hypothesis that natural languages are within the class MCTAL.

7.5. BEYOND MILDLY CONTEXT SENSITIVE

7.5.1. CHINESE NUMBERNAMES (Radzinsky 1991)

5 = *wu*
10^4 = *wan*
10^8 = *wan wan* or *yi*
10^{12} = for some speakers: *wan wan wan* but many prefer: *wan yi*
10^{16} = for some speakers: *wan wan wan wan* but many prefer: *yi yi*
10^{12} = *zhao* the highest lexical item
10^{24} = *zhao zhao*
10^{36} = *zhao zhao zhao*

So with *zhao* we get rid of the interference of higher number names.
Now look at the sets of strings:

R = {*wu* (*zhao*)^{n1} *wu* (*zhao*)^{n2} *wu* (*zhao*)^{n3}… *wu* (*zhao*)$^{nk-1}$ *wu* (*zhao*)^{nk}: for every n_i: if $i \leq k$: $n_i \in \mathbf{N}$}
R is, of course, a regular set.

N = {*wu* (*zhao*)^{n1} *wu* (*zhao*)^{n2} *wu* (*zhao*)^{n3}… *wu* (*zhao*)$^{nk-1}$ *wu* (*zhao*)^{nk}:
for every n: if $i \leq k$: $n_i \in \mathbf{N}$ and $n_{i-1} > n_i$ }

N is an indexed language, but not in MCTAL.

Radzinsky looks at the category of Chinese number names, call it NN, and claims: $\mathbf{NN} \cap R = N$. And concludes: the set of wellformed strings in the category of Chinese number names is not in MCTAL.

Discussion I
Radzinsky is careful not to make a claim about the category of numerical phrases in general. This is good, because the intersection argument is unlikely to hold for numerical phrases in general.

Thus, the descending condition on Chinese number names is, of course, similar to what we find in English:

(1) a. ✓*five* ***thousand*** *five* ***hundred***
b. #*five* ***hundred*** *five* ***thousand*** .
c. ✓*five hundred* ***and*** *five thousand*

(1a) is an English numbername denoting a single number, (1b) is not, (1c) is.

Furthermore adjectival position usually allows only **one** numerical adjective phrase:

(2) a. ✓There were *thirty three* students at the party.
b ??There were *thirty three* **ninety six** students at the party.
c. ??There were *between thirty and fourty* **between thirty five and fourty five** students at the party.

(The infelicity is not simply mathematical inconsistency as (2c) shows.)

If the constraint in (2) were absolute, we could use Radzinky's argument as an argument about numerical adjectives. But the constraint is not absolute.
-In many languages (including Chinese, as argued by Li and Rothstein), ascending sequences of numbers can be used as approximatives, and what numbers can occur in the sequence is a question of setting up the context. (3a) is from the famous Dutch translation *Waar blijft de tijd* of Jean Ferrat's chanson *On ne voix passer le temps* (note that the list can consist of more items).

(3) a. Met *drie, vier* kinderen, ach dat went, je hebt geen tijd meer voor geluk.
With three, four children, well, you get used to that, you no longer have time for bliss.
b. There are *five thousand* ***fifty thousand*** *maybe five hundred thousand* molecules in the tube, but *not more*. (with pooh pooh intonation).

Thus, we see lists of **ascending** numbers in the adjectival prenominal position.

-Other kinds of contexts where you find sequences of ascending numbers in adjectival position are contexts giving online process descriptions:

(4) [In a game show] Ok, the tap is open and out stream *five hundred* **five thousand** *fifty thousand* **five hundred thousand** *f i v e m i l l i o n* dollarcents! And the tap stops. Well, you will have a heavy time taking all that home. Please give them a warm applaus.!!!

This means that you cannot argue that the category of numerical adjectives requires sequences of numbers to be descending (as they are in the numbernames (4b)).

(5) a. *five hundred five thousand fifty thousand five hundred thousand five million*
b. *five million five hundred thousand fifty thousand five thousand five hundred*

For Chinese, this would mean that the intersection of R with the set of Chinese numerical adjectives is not NN, and the argument does not go through for numerical adjectives.

Discussion II
So this is not what Radzinsky is claiming.
He is only making a claim about **the category of Chinese number names**:

(5a) may be felicitous as an English numerical adjective *phrase***,** but it is not an English number name.

And it is part of knowing English that (5b) *is* an English number name and (5a) is not, and part of knowing Chinese that strings in N are Chinese numbernames, but strings in R – N are not.

I think we can accept this description of the facts without accepting Radzinsky's conclusions.

What do we know when we know that (4b) is an English number name, and why do we know it?
The answer is that we know that (4b) *denotes* a number, a single number, and we know that, because for these expressions we have been given an algorithm for determining *which* number the expression denotes, an algorithm that yields a succesful result in (5a) and in (5c), but (in English) not in (5b):

(5) a. *five million five thousand five hundred*
$(5 \times 10^6) + (5 \times 10^3) + (5 \times 10^2)$
b. ??*five million five hundred five thousand*
c. ✓*five million five hundred* ***and*** *five thousand*
$(5 \times 10^6) + ((5 \times 10^2) + 5) \times 10^3)$

There are two issues here.

1. If we assume a linguistic category of number names NN, do we assume that the expressions that are of that category are **exactly** the expressions that denote numbers, or do we assume, say, that the category NN is generally sequences of number expressions (with and without *and*) and among those expressions there are felicitous number names and infelicitous number names, where the first ones are the ones that are defined (by one of the algorithms)?

In the latter case, the syntactic category of number names can be simply taken to be regular or contextfree, and felicity is a semantic question: felicitous number names are names that are known to denote numbers.

To take a not so linguistic example for comparison: take **the category of Names of British Royalty.**
There are some generalizations within this class
like: *Henry the Fifth* is a Name of English Royaly,
and: *the Fifth Henry* is not (really) a name
and nor is: *Buck the Third*,
or: *Henry the Seventeenth*
or: *Bonny Princess Ann*

You could easily write a grammar generating all and only the Names of British Royalty (even capturing the generalization tendencies).
However, there doesn't seem to be any particurly good reason to **prevent** generating syntactically the names:
Henry the Seventeenth
William the Unready
Bloody Charles
just for the reason that they happen to be **non-denoting at the present time,** and are, hence, **infelicitous as Names of British Royalty**.

2. If we assume a linguistic category of number names NN **and** we assume that the expressions that are of that category are **exactly** the expressions that denote numbers, **we still do not have to assume** that that category is syntactically defined.

We can assume the class of **denoting** number names as a well-defined **semantic** category.

Thus, we can assume that there **is no** syntactic category of number names, there are just number phrases.
Within the category of number phrases, we define a **semantic subcategory** of felicitous number names, defined semantically.

This is in fact my own preferred strategy.
Note that when you apply this strategy to the category of Names of British Royalty, you get the following:

> there is no syntactic category of Names of British Royalty,
> there is a syntactic category of noun phrases that Names of British Royalty are part of (with a bit more syntax in them than other names, i.e. allowing adjectivally modified noun phrases, noun phrases with epithets, etc.).

If you want a subcategy of noun phrases that are Names of British Royalty you will have define this class in terms of denotation, and you will have to accept the fact that this class is not a semantically regular class, in that the argorithm for determining denotation is not semantically illuminating (and in fact is by and large not or not much semantic at all).

This is, of course, different for number names: there the denotation determining algorithms encode the "meaning" of the number names:
If you know how to apply the algorithm you know something very basic about how to use number names.

Discussion III
Radzinsky discusses this strategy and claims that basically that strategy pushes the same problem one level up: from the syntax to the syntax-semantics interface.

And he claims that if you look at the linguistic system as comprising both a syntax and a syntax-semantic interface, and regarding this as one single grammatical system, the combined system will have the same problem all over again.

This raises a very fair point. If we are concerned with semantically interpreted grammar, asking just ***what is the complexity of syntax*** is only part of the real problem: ***what is the complexity of the combined system*** is a reasonable question.

On the other hand, complexity issues of inference systems (logics) has been very well studied; as we know, decidability issues come up much more easily here, so that efficiency of semantic inference and semantic parsing is clearly not really on a par with the efficiency of syntactic parsing.

In other words, we know that the problem of turning our nice linguistically insightful semantic interpretation systems into semantically interpreted grammars for humans requires solving complex problems of a totally different order than keeping our combination rules semilinear.

So, at the interfact of syntax and semantics we need other techniques anyway, it is an illusion to think that we can just extend the syntactic generation mechanism to a syntax-semantics generation system of which we only need to require that it (the combined object) be mildly contextsensitive.

In this sense, I think that if we push the interpretation algorithm for number names to the semantics or the syntax-semantics interface, the fact that the combined syntax-interpretation system cannot be mildly contextsensitive **is unlikely to be our worst problem there.**

This makes it rather a reasonable strategy to move the licensing of numbernames to this level: push the problem to a level where a solution to more difficult problems that we need to solve *there* anyway is likely to solve this one on the side.

In sum, even if we agree that the knowledge that (1a) and (1c) are felicitous number names in English and (1b) is not, and that the intersection of R and NN, the class of felicitous nunber names in Chinese is NN, we do not have to accept Radzinsky's conclusion about Chinese number names being not mildly contextsensitive.

Native speakers have judgements about felicity, linguists interpret these and make proposals as to what accounts for these judgements. Native speakers and linguistics do not have judgements about whether the account for the judgements should be syntactic or semantic.

Thus, even **postulating** the category NN **as a linguistic category** is a **theoretical** decision, and so is deciding that it is a syntactic category.
(This in no way means, that I think that we linguists are unjustified in describing number systems in the world's languages by using a category of Number Names. I think that is perfectly fruitful. One just needs to be careful about what that implies for the organization of the system that Native speakers use in interpreting and using their language).

Discussion IV
Classical Chinese grammarians list a whole series of number names bigger than *zhao*. Radzinsky feels free to ignore these because they are, as he claims, not part of the active volcabulary of contemporary native speakers of Chinese. This is rather an important decision, because, as we have seen, native speakers don't all accept *wan wan wan*, some prefer *wan yi*.

So what *do* native speakers prefer when it comes to *zhao zhao* or *zhao zhao zhao*?

My native informant shrugged his shoulders, *did* notice the possibility of saying *wan wan*, but his attitude was clearly something like: I'll decide when I get there.

And speakers don't seem to get there. Googling things like *wu zhao zhao* (by me and by my informant) gave exactly zero finds.

Since we *are* dealing here with names, maybe the best strategy is the same as for potential English Royalty names like Charles the 3rd or William the 5th : we'll accept them when we get there.

7.5.2. CROSS SERIAL DEPENDENCIES AGAIN (Manaster Ramer 1987)

Some more data.
Above I assumed an analysis for Dutch where the auxilary incorporates into the verb cluster, and ends up rightmost, or stays in I position, and ends up left most. This is too simple an assumption, though.

We are now concerned with conjunctions. We have V conjunctions of the cluster like (1), and the auxiliary can be on either side (as expected).

(1) a. dat Kim Sam Pat Ronya **zal** *laten helpen kammen* ***en*** *helpen laten voeden*
that Kim Sam Pat Ronya **will** let help groom **and** help let feed
that Kim will let Sam help Pat groom Ronya and let Sam help Pat feed Ronya.

b. dat Sam Pat Ronya *laten helpen kammen* ***en*** *helpen laten voeden* **zal**

But we also have I-conjunctions, with different auxiliaries, and here the auxiliaries in the conjuncts can be on either side:

(2)
a. dat Kim Sam Pat Ronya **moet** *laten helpen kammen* ***en*** **zal** *helpen laten voeden*
that Kim Sam Pat Ronya **must** let help groom **and will** help let feed

b. dat Kim Sam Pat Ronya **moet** *laten helpen kammen* ***en*** *helpen laten voeden* **zal**
c. dat Kim Sam Pat Ronya *laten helpen kammen* **moet** ***en*** **zal** *helpen laten voeden*
c. dat Kim Sam Pat Ronya *laten helpen kammen* **moet** ***en*** *helpen laten voeden* **zal**

What this shows is that we must be able to make the sequence with the auxiliary at the end into something that can be conjoined with the sequence with the auxiliary at the beginning. There are various ways in which this can be achieved.

For instance the equivalent of:

adjoin the cluster [$_V$ *helpen laten voeden*]
to [$_I$ *zal*]: [$_I$[$_V$ *helpen laten voeden*] [$_I$ *zal*]]
but allow this to be analyzed as [$_V$ [$_V$ *helpen laten voeden*] [$_V$ *zal*]],
allowing it to incorporate into the lower V-position.

Or allow in Dutch incorporation of the I-element on the either side.

So the facts about I-conjunction show that the analysis given so far needs more work.

We are not concerned with I conjunction here but with V conjunction.
Dutch allows conjunction inside lexical categories quite freely. The most extreme example is (3):

(3) a. Deze bus rijdt niet op *zon en feestdagen*
This bus rides not on [[Sun and holi] days]
This bus doesn't go on Sundays and holydays

And conjunction is possible inside the verb cluster as well:

(4) a. dat Kim Sam Pat Ronya **zal** laten helpen [kammen **en** voeden]
b. dat Kim Sam Pat Ronya **zal** laten [helpen kammen **en** zien voeden]
c. dat Kim Sam Pat Ronya **zal** [laten helpen kammen **en** helpen laten voeden]

As we have seen, the requirement in the verb cluster is that there cannot be less verbs than noun phrases, although there can be less noun phases than verbs, because of implicit arguments. This holds in the conjunction as well: we could have:

(5) a. dat Kim Sam Pat Ronya **zal** laten helpen [kammen **en** zien eten]
that Kim will let Sam help Pat [groom Ronya and see Ronya eat]
b. #dat Kim Sam Pat Ronya **zal** [voeden **en** laten helpen kammen]

In (5b) there are not enough verbs in the the first conjunct.

So the requirement that there have to be at least as many verbs in the verb cluster as there are nouns in the NP sequence holds in conjunctions ***per*** conjunct.

But conjunction is unbounded: you can have as many conjuncts as you like.
This means that we have, in principle, a structure of the form:

$NP^n [V^{m1}$ and V^{m2} and … and $V^{mk}]$, with n, $m_1,\ldots,m_k \in N$ and $n \leq m_1,\ldots,n \leq m_k$.

With this we can make an argument that Dutch is not an MCTAL. Manaster Ramer gives the intersection with a regular set argument.

Manaster Ramer himself brings up an argument against accepting the arguments from cross-serial dependencies. The crucial point of the data is that the number of verbs in the sequence cannot be smaller than the number of noun phrase arguments. Manaster Ramer argues that you cannot in fact control this. Namely, take a sequence of NPs and Verbs, with more NPs than Verbs:
Kim Sam Pat Ronya – helpen voeren.

Manaster Ramer argues: you cannot exclude the possibility that *Kim Sam* is **one** name, nameing one individual, and that *Pat Ronya* is **one** name, naming another individual. In that case the same **string** counts as a grammatical string of Dutch, and the intersection argument fails. Since there are no real linguistic constraints on what counts as a name in Dutch, any argument concerning weak generative capacity is doomed to fail.

I find this argument frivolous. The same argument shows that no string of English words, with a transitive verb like *kissed* non-initial and non-final, like:

*in lately **kissed** will the for*

can be excluded from the set of sentences that are grammatical in English, because of the novel that I will write in which the beautiful *In Lately* indeed kissed shy *Wil the For*.

Manaster Ramer's own proposal is to move from weak generative capacity to what he calls **classificatory capacity**, which basically means the capacity of the grammar to characterize a construction or a subcategory.

My scepsis concerning this notion has been expressed more than once in this chapter: it presupposes that there *are* constructions that the grammar isolates (like relative clauses or possessive constructions) and it presupposes that we know that the relevant subcategories are syntactic.

The better strategy is, I think, to accept that weak generative capacity is always a help for diagnosing what ultimately should interest us more: strong generative capacity. It is not hard to eliminate Manaster Ramer's counterexamples from the discussion via a surface structure generative capacity argument (for instance, again, via case features).

It seems to me that for our purposes that is enough: the orginal argument shows that sensible grammars for cross serial dependencies in Dutch, German and Swiss German must go beyond contextfree, and Manaster Ramers argument shows similarly, I think, that sensible grammars for cross serial in these languages must go a bit beyond mildly contextsensitive, *if* we define that in terms of equivalence to MCTAGs.

What do we need to add? Do we need to liberate MCTAGs from the requirement that only a bounded number of adjunctions can take place simultaneously? That would get the required result in one go, but it gives up the idea of dependencies that are in some sense bounded.

My suggestion is to think more about co-ordination. We have seen that co-ordination is special in that we find 'across-the-board' phenomena. Co-ordination is special too in that it is cross-categorial, there are no or hardly restrictions on which categories can be co-ordinated, and in fact, co-ordination of non-constituent categories have been discussed widely in the literature (under various names, things like

[*Kim kissed and Sam hugged*] *Pam*.
Kim simultaneously [*introduced Pam and gave the book*] *to Chomsky.*

Semantically, co-ordination requires the co-ordinates to be of the same semantic type (like: conjunction maps two two-place relations onto a two-place relation, etc.)

We need to ask: how does co-ordination extend to MCTAGs?

Thus, with MCTAGs, we have derived auxiliary tree sets $\{\alpha, \beta\}$, and we observe that here too, α or β or both could be a co-ordinate structure. But in this case, co-ordination in β is constrained by α. Let us use **and** for the relevant coordination

structure. Then the obvious way in which co-ordination would generalize to MCTAGs is:

AND_1: $<\alpha_1, \beta> + <\alpha_2, \beta> \rightarrow <\alpha_1$ **and** $\alpha_2, \beta>$
AND_2: $<\alpha, \beta_1> + <\alpha, \beta_2> \rightarrow <\alpha, \beta_1$ **and** $\beta_2>$

These operations easily generate Manaster Ramer's co-ordinations, and generate them correctly: β_1 and β_2 can only be co-ordinated if each of them matches α. And we can add as many copies as we want, with the same matching:

AND_2: $<\alpha, \beta_1$ **and** $\beta_2> + <\alpha, \beta_3> \rightarrow <\alpha, \beta_1$ **and** β_2 **and** $\beta_3>$

This operation obviously *does* extend the generative capacity beyond MCTAGs: it is an operation that takes two *derived* auxilary tree sets as input and gives a *derived* auxiliary tree set as output. Such an operation we didn't have before in MCTAGs.

Of course, the *diagnosis* of what happens in co-ordination here is that you *want* to co-ordinate derived verb sequences, and you can only do that, in the tree set where they are derived, and that means, relative to the derived NP-sequences. This analysis is actually, I think, within the spirit of Joshi's mildly context-senstive grammars: I assume that polynomial parsing is not affected; The operations AND_1 and AND_2 do not affect semi-linearity; and, in fact, the number of simultaneous adjunctions stays bounded.

The effect of unbounded adjunctions is created by liberalizing, *for co-ordination* the requirement of involving elementary trees somewhat.

I suggest to **name** grammars that are semi-linear, allow polynomial parsing, allow at most the cross-serial dependencies of mildly-context sensitive grammars, but allow conjuntion generalized as above: **grammars that are within the context free corona**.

Thus, Dutch may not be an MCTAL language, but we may hope that it is **a language within the context free corona.**
Whether it is I leave that for you to investigate.

7.5.3. A NOTE ON COPY MOVEMENT

Above I mentioned that Stabler's formalization of Minimalist Grammar is weakly equivalent to MCTAGs. There is a caveat here: Stabler's movement operation is not the operation whiuch is popular in some areas of the Minimalist program: **copy movement**.

Copy movement is a movement transformation whereby at each landing site, the transformation doesn't leave a trace, but a **full copy** of the moved tree.

(1) Which boy does Mary think that Bill knows that Henry regrets that Mary likes.

(2) **Which boy** does Mary think **which boy** that Bill knows **which boy** that Henry regrets **which boy** that Mary likes **which boy**.

Since the length of the wh-phrase is **unbounded** (*which boy, which boy that Mary likes, which boy that Mary likes that Bill hates,...*) and the number of landing sites is **unbounded** (as we see in (1)**,** it is easy to see that copy movement is **not** within the capacity of TAGs, and **not** within the capacity of MCTAGs, since we minimally get a language of unbounded many unbounded copies.

(3) **Which boy that every girl thinks Jan is in love with** does Mary think **Which boy that every girl thinks Jan is in love with** that Bill knows **Which boy that every girl thinks Jan is in love with** that Henry regrets **Which boy that every girl thinks Jan is in love with** that Mary likes **Which boy that every girl thinks Jan is in love with**.

Whether copy movement is in the context free corona or not depends on what you decide to want the operation to do (if you want it in the first place). This means that the question can hardly be formulated in a non-theory internal way (which makes the situation similar to the status of general transformations in earlier syntactic theory). Thus, whether copy-movement will put natural language grammars outside the CF corona, whether it will stay within the CF corona, or whether it will itself actually go the way of earlier transformations (as it may well) lies in the future of theory development.