PART 4. SEMI LINEAR LANGUAGES AND PARIKH'S THEOREM

Context free languages have two important computational properties that make them atrractive and that have played an important role in the development of grammar formalisms for the analysis of natural language that take computational concerns seriously. These properties are **polynomial parsing** and **semi-linearity**.

As we will see, there are phenomena in natural languages that require our grammar formalisms to go beyond context free grammars. But how far should we deviate from context free grammars? Aravind Joshi, possibly the foremost researcher in developing such grammar formalisms, has proposed to take polynomial parsing and semi-linearity as **methodological constraints** on developing 'computationally aware grammars for natural languages': it is computationally (and linguistically) fruitful to assume that natural languages allow polynomial parsing and are semi-linear. Whether ultimately true or not, fact is that Yoshi's assumption has played an important role in the development of such grammar formalisms over the last twenty years.

In this text I will not discuss parsing extensively. But I will mention here a few aspects of polynomial parsing.

Let G be a grammar.

A **parsing algorithm** for grammar G is an algorithm which determines for every string of V_T* whether or not it is in L(G).

The **time that** a parsing algorithm for G **requires** to parse **string** α of V_T* is the minimal number of steps required by the algorithm to decide whether α is in L(G) or not.

Take for each string α of V_T* of lenght n the time that the parsing algorithm for G requires to parse that string. This is a set of numbers. Take the **maximal number.** This maximal number is the **time that** a parsing algorithm for G **requires** for parsing **strings of length n** of V_T*.

The **speed** with which a parsing algorithm for G parses **language** L(G) is the **function which maps every number n onto the time that the parsing algorithm requires for parsing strings of length n of V_T*.**

A language L can be **parsed in polynomial time** if there is an algorithm for a grammar G for L such that the speed with which that algorithm parses L(G) is a function of the form:

for every n: **speed**(n) = $a_k(n^k) + a_{k-1}(n^{k-1}) + ... + a_{k2}(n^2) + a_1(n) + a_0$
where k, $a_0,...,a_k$ are numbers.

The highest exponent k is called the **degree** of the time functions. It can be shown that for polynomial time functions only the degree is relevant (meaning that algorithms in a polynomial of degree k can be speeded up to run in time n^k).

Note that parsability in polynomial time expresses a **worst case boundary**. If a language is parsable in cubic time n^3, this means that **at worst** you will find the answer to whether a string of length n is in the language or not in n^3 steps.

This may well mean that for many strings of length n **less steps** are required (with a minimum of n steps, since reading the string takes n steps).

Of course, an algorithm that runs in time, say, n^{64} will be very slow on its worst cases. But, the intuition in computer science is: that is still better than nothing:
In practice, computer scientists are not afread of algorithms that run in, say, cubic time (or even worse). In particular, because experience tells you that if a parsing algorithm works in cubic time, **in practice** many cases (typically including the ones you will come across most) require much less time (linear time: n+k, sometimes n^2, rarely n^3).

If the speed function is of the form **speed**(n)=m^n (with n the length of the input), the algorithm runs in **exponential time**. .

The boundary between polynomial time and exponential time is a natural one: we can define how fast functions increase, and with that definition, it can be shown that every exponential function increases faster than any polynomial function.

Algorithms that operate in exponential time are **ineffective.** Languages that require exponential time for parsing may be theoretically parsable, but are **in practice unparsable**.

Within the class of languages parsable by, or more general problems decidable by, algorithms that run in polynomial time we distinguish between class **P** of problems decidable by **deterministic algorithms** that run in polynomial time, and the class **NP** of problems decidable by **non-deterministic algorithms** that run in polynomial time.
The class P of problems decidable by deterministic algorithms is generally regarded as the class of problems that have **effective** solutions.
Non-deterministic algorithms that run in polynomial time are not regarded as effective.

While it is the case that the class of all problems decidable by deterministic algorithms (turing machines) coincides with the class of problems decidable by non-deterministic algorithms, this doesn't give effective solutions to all polynomially decidable problems: if we have a non-deterministic polynomial algorithm deciding a problem, we know that a deterministic algorithm exists as well, but not that the latter is also polynomial: and a deterministic exponential algorithm is ineffective.

It has so far not been shown that there is any problem for which a non-deterministic polynomial algorithm exists, but no deterministic polynomial algorithm.
On the other hand, there are many problems for which a non-deterministic polynomial algorithm exists, (i.e. the problem is in NP), but for which no deterministic polynomial algorithm is **known**.

The question: P = NP? is a famous open problem (meaning that you can make money by solving it).
If NP−P is not empty, the problems in that set will be regarded as ineffective as well.

We now come to grammars.

We first mention two facts:

Fact: For every context sensitive grammar G a parsing algorithm for G exists.
But there is no guarantee that context sensitive languages can be parsed in polynomial time.
The same is true for indexed grammars.

Fact: Every context free language can be parsed in (deterministic) polynomial time.

The standard general algorithm for parsing context free grammars, called, the Cocke-Younger-Kasami (CYK) algorithm runs in cubic time n^3. Another well known algorithm, Earley's algorithm, also runs in cubic time in general, but in n^2 for unambiguous grammars, and in linear time for many grammars. Faster algorithms exist, for instance, Valiant's algorithm from 1975 runs in the time it takes you to multiply two $n \times n$ matrices (which in 1981 was $n^{2.52\ldots}$).

With Joshi, it seems reasonable to assume that **the least** you want to require of your grammar formalism for natural languages is that it allows (deterministic) polynomial parsing.

I will not discuss parsing further. So we move to Joshi's other methodological constraint: semi-linearity. That brings us to technically the most difficult part of this class: a discussion of Parikh's theorems for context free grammars and context free languages.

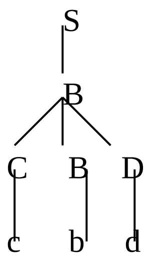
1. LINEAR AND SEMI LINEAR TREE SETS

Let G be a context free grammar with non-terminal vocabulary V_N and terminal vocabulary V_T. I repeat:

> An **I-tree** in G is a parse tree in G with top label S and only terminal symbols (or e) on the leaves.

Thus, for context free grammar G, **T(G)** is the set of all I-trees for G.

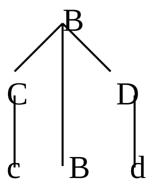
Example:
In a grammar with rules $S \rightarrow B$, $B \rightarrow C\ B\ D$, $C \rightarrow c$, $B \rightarrow b$, $D \rightarrow d$, the following tree is an I-tree:



Next, we define:

> An **A-tree** in G is a parse tree in G with some non-terminal label B on the top node and terminal symbols (or e) on all leaves, **except for one**, which has the **same label** B as the top node.

In the same grammar, the following tree is an A-tree:



Remember we discussed the length of a path in an I-tree, and we assumed that it is the number of nodes from top to leaf, minus the leaf-node which is labeled by the terminal symbol. We extend this notion to A-trees, and assume that also for the one path that ends in the non-terminal, the length of the path is the number of nodes from top to leaf, minus the leaf-node which is labeled with the non-terminal.
Thus, the I-tree above has three paths of length 3, the A-tree above has two paths of length 2, and one of length 1.

> The **height of** tree T is the length of the maximal path in T.

Let T be a parse tree in G.

> The **signature** of T, **s**(T), is the set of non-terminal labels occurring on one or more nodes in T.

So the signature of the above I-tree is: {S,B,C,D}
the signature of the above A-tree is: {B,C,D}

The next notions require careful attention:

Let Z be a set of non-terminal labels of G. Z is, of course, a finite set of non-terminals. Assume |Z|=n.

> $\mathbf{T_Z}$ is the set of all I-trees in G with signature Z.

For T_Z we define:

> $\mathbf{I_Z}$, the set of **initial** trees for T_Z, is the set of all I-trees in T_Z of height at most n^2+1.
>
> $\mathbf{A_Z}$, the set of **auxiliary** trees for T_Z, is the set of all A-trees T in G of height at most n^2+1 such that **s**(T) $\subseteq$ Z.

This means the following:
T_Z is the set of all constituent structure trees of G in which the non-terminal labels occurring on the nodes **are exactly** the non-terminals in Z.
This means that on I-trees in T_Z the non-terminals in Z may occur more than once.
But an I-tree in which a non-terminal occurs which is not in Z is not in T_Z. And, importantly, an I-tree in which a non-terminal from Z does **not** occur is also not in T_Z.

I_Z is a **subset** of T_Z. It is the set of all I-trees with signature Z in which the length of the maximal path is restricted to not more than the square of the size of Z plus 1 itself. (the rationale for this restriction will become clear in the proofs later).

A_Z is **not a subset** of T_Z. A_Z is a set of A-trees in a signature which is part of Z. This means that all the non-terminal labels occurring on nodes in a tree in A_Z are labels from Z.
As before, a label from Z may occur more than once in a tree in A_Z.
As before, if a label occurs on a A-tree which is not in Z, that A-tree is **not** in A_Z.
Similar to what we required for I-trees, if the length of the maximal path in an A-tree exceeds n^2+1 that A-tree is **not** in A_Z.

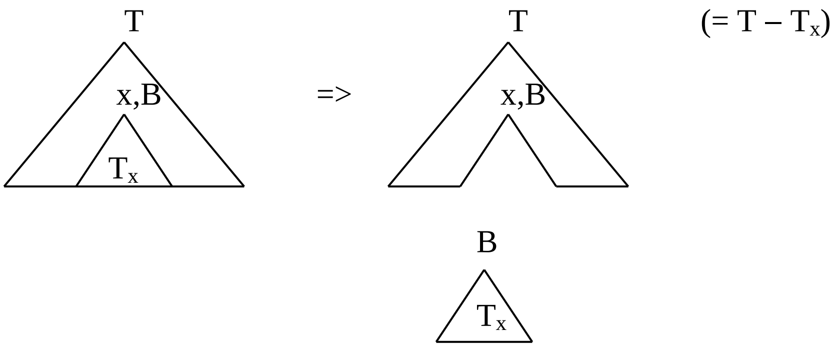
But this time, if an A-tree satisfies the above constraints, but doesn't contain all the non-terminal labels of Z on its nodes, we **still** include it in A_Z.

So the trees in T_Z and I_Z are required to contain all the non-terminal labels in Z, the trees in A_Z are required to contain a subset of the non-terminal labels in Z.
And the trees in T_Z and A_Z are restricted in height.
Fact: Since Z is a finite set of non-terminals, and the height of the trees in $\mathbf{I_Z}$ and $\mathbf{A_Z}$ are restricted to n^2+1, both $\mathbf{I_Z}$ and $\mathbf{A_Z}$ are **finite** sets of trees, for each set $Z \subseteq V_N$.

Tree Adjunction.

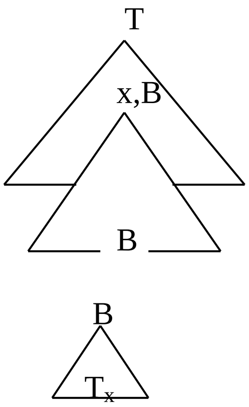
Let T be an I-tree in G with node x labeled B.
Let T_B be an A-tree in G with topnode labeled with B.
Tree adjunction forms the following tree: ADJ[T,T_B,x]

1. Let T_x be the subtree of T dominated by node x.
   Remove T_x from T, leaving node x (i.e. replace tree T_x in T by x).



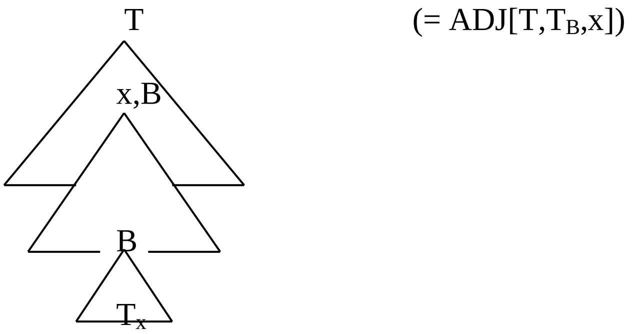
We will call this tree T – T_x

2. Attach tree T_B at node x in T – T_x (identifying their toplabels B):



We will call this tree (T – T_x) + T_B

3. Attach tree T_x to the bottom node with label B (identifying their labels):



So this is: ((T – T_x) + T_B) + T_x
If we call this tree T' then extend the use of – to: T' – B = T

ADJ[T,T_B,x] is itself an I-tree in G, since T is an I-tree in G, T_B is a parse tree in G, and after adjunction, the resulting tree has only terminal strings at the leaves.

Moreover, if T $\in$ $\mathbf{T_Z}$ and T_B $\in$ $\mathbf{A_Z}$, then ADJ[T,T_B,x] $\in$ $\mathbf{T_Z}$, because all the labels in T_B already occur in T, i.e. s(ADJ[T,T_B,x]) = s(T) = Z.

We define **DER[I,A]**, the set of I-trees **derived** from set of I-trees **I** and set of A-trees **A** by tree adjunction:

> **DER[I,A]** is the smallest set of I-trees such that:
> 1. **I** $\subseteq$ **DER[I,A]**.
> 2. If T $\in$ **DER[I,A]**, x a node in T with label B and T_B $\in$ **A** with toplabel B, then: ADJ[T,T_B,x] $\in$ **DER[I,A]**.

Thus, the result of tree adjoining any A-tree at a node with the right label in an I-tree is a derived tree, and adjoining an A-tree in a derived tree is a derived tree as well.

> A set of I-trees **X** is **linear** iff for some **finite** set of I-trees **I** and some **finite** set of A-trees **A**, **such that each A-tree can be adjoined in each I-tree**:
> **X** = **DER[I,A]**.
>
> A set of I-trees **X** is **semi linear** iff it is the union of a **finite number** of linear sets of I-trees.

FACT 1: **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]** is a linear set of I-trees.

PROOF:
By the definition of $\mathbf{I_Z}$, any initial tree in $\mathbf{I_Z}$ (and hence any derived tree) **will** have a node with the right label for adjunction, for each auxiliary tree in $\mathbf{A_Z}$ (since all the toplabels of the auxiliary trees occur in each initial (and hence in each derived) tree. So each auxiliary tree can be adjoined in each initial tree.
Moreover, we have already noted above that $\mathbf{I_Z}$ and $\mathbf{A_Z}$ are **finite** sets of trees, and that the result of adjoining an auxiliary tree in an I-tree is an I-tree, hence **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]** is indeed a linear set of I-trees.

THEOREM (Parikh): For every set of nonterminals Z: $\mathbf{T_Z}$ = **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]**.

PROOF:
A. **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]** $\subseteq$ $\mathbf{T_Z}$
We have already noticed this above: all the trees in $\mathbf{I_Z}$ are in $\mathbf{T_Z}$, and the result of adjoining auxiliary trees into trees in $\mathbf{T_Z}$ stays in $\mathbf{T_Z}$.

B. $\mathbf{T_Z}$ $\subseteq$ **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]**
1. If T $\in$ $\mathbf{I_Z}$, T $\in$ **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]** by definition.
2. If T $\in$ $\mathbf{T_Z}$, but T $\notin$ $\mathbf{I_Z}$, then the height of T is bigger than n^2+1.
What you show is that you can always reduce such a tree T to a smaller tree in T_Z by cutting out of it an auxiliary tree in $\mathbf{A_Z}$. The resulting smaller tree will be in **DER[**$\mathbf{I_Z}$,$\mathbf{A_Z}$**]** by induction hypothesis, and hence, T is in that set by adjunction with the auxiliary tree you cut out.

Assume: every tree $T \in \mathbf{T_Z}$ with at most k nodes is in $\mathbf{DER[I_Z,A_Z]}$.
Let $T \in \mathbf{T_Z}$ with k+1 nodes and height more than n^2+1. We prove that $T \in \mathbf{DER[I_Z,A_Z]}$.
The height of T is more than n^2+1. This means that T contains a path of length more than n^2. This means that at least one label of Z occurs at least n+1 times on this path.

Claim: There is a node $x \in T$ such that:

1. T_x contains a path with exactly n+1 occurrences of the same label.
2. There is no node y in T_x (except for x) such that T_y contains a path with more than n occurrences of any label.

Start out on a path of lenght more than n^2+1, walk up from the leaf, until you find for the first time a label B for the n+1-th time, say at node x. Stop there. If the tree T_x satisfies condition 2, you are done. If it doesn't, go to the daughter nodes. At least one of them will satisfy the first condition, but be of smaller height. Repeat for that daughternode the same procedure,.etc. Since you get to smaller and smaller trees, you **will** ultimately find a tree that **does** satisfy both conditions. That is the tree we are interested in.

So let us assume that we have found our tree T_x which satisfies both properties in the claim, say, with label B on x the only label occurring n+1 times in T_x on any path. This means that each path in T_x has maximal length of n^2+1, namely label B maximally n+1 times, any other label maximally n times. This means that any subtree of B, including any A-tree has height at most n^2+1.

There are two cases:
- Label B occurs n+1 times in T_x on exactly one path.
- Label B occurs n+1 times in T_x on more than one path.

In the first case, follow the path down, in the second case, choose on of the paths to follow down. Number the nodes on the relevant path 1,…n+1 down from T_x.

Now come the crucial observations:

1. For every $i \leq n$: $s(T_{i+1}) \subseteq s(T_i)$
   This is obvious, because T_{i+1} is a subtree of T_i (since i and i+1 are on the same path).
2. For every $i \leq n+1$: $s(T_i) \neq \emptyset$.
   This is also obvious, because each of them contain label B.
3. $|Z|=n$ and there are n+1 nodes with label B on the path.
   This means that **it cannot be the case** that for every $i \leq n$: $s(T_i) \neq s(T_{i+1})$.

This is because we have an increasing chain of subsets of Z, and there aren't enough elements in Z to make all subsets in the chain distinct.

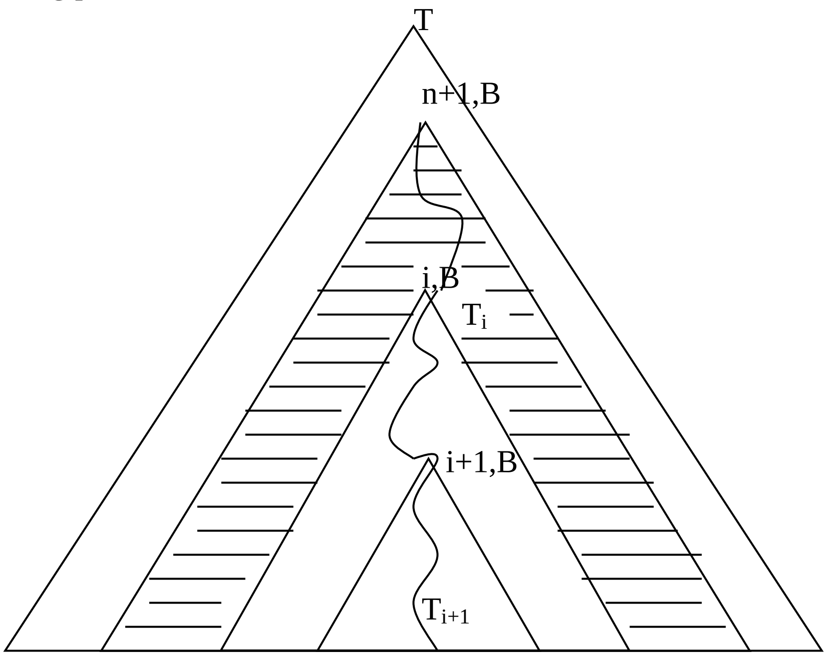
Hence we have shown that for some $i \leq n$: $s(T_i) = s(T_{i+1})$

These nodes i and i+1 are the ones we need. What we can prove now is: if we cut out the bit between nodes i and i+1, the result is an I-tree with the same signature, hence in $\mathbf{T_Z}$, and the bit between i and i+1 is an A-tree in $\mathbf{A_Z}$. More precisely:

We look at I-tree: $(T - T_i) + T_{i+1}$ and A-tree $T_i - T_{i+1}$, and we observe:
1. $(T - T_i) + T_{i+1} \in \mathbf{T_Z}$ (because T_i and T_{i+1} have the same signature).
2. The number of nodes in $(T - T_i) + T_{i+1}$ is at most k (Minimally one node with label B is removed). This means, by induction hypothesis, that $(T - T_i) + T_{i+1} \in \text{DER}[\mathbf{I_Z},\mathbf{A_Z}]$
3. $T_i - T_{i+1}$ is an A-tree. $s(T_i - T_{i+1}) \subseteq Z$. As we observed above, the height of $T_i - T_{i+1}$ is at most n^2+1, hence $T_i - T_{i+1} \in \mathbf{A_Z}$.

Now $T = \mathrm{ADJ}[(T - T_i) + T_{i+1}, T_i - T_{i+1}, i]$, Hence we have proved that $T \in \mathrm{DER}[\mathbf{I_Z},\mathbf{A_Z}]$.

The following picture shows what we have done:



CORROLLARY 1: For every set of non-terminals Z: $\mathbf{T_Z}$ is a linear set of I-trees.

FACT 2: There are only finitely many subsets Z of V_N, since V_N is finite.
i.e. $\mathrm{pow}(V_N) = \{Z_1,...,Z_n\}$.

FACT 3: $\mathbf{T(G)} = \mathbf{T}_{Z_1} \cup ... \cup \mathbf{T}_{Z_n}$

PROOF: Obvious:

-for any $\mathbf{T_Z}$: $\mathbf{T_Z} \subseteq \mathbf{T(G)}$, by definition, so $\mathbf{T}_{Z_1} \cup ... \cup \mathbf{T}_{Z_n} \subseteq \mathbf{T(G)}$

-If $T \in \mathbf{T(G)}$, its non-terminal labels are in some subset Z_i of V_N, hence $T \in \mathbf{T}_{Z_i}$, hence $T \in \mathbf{T}_{Z_1} \cup ... \cup \mathbf{T}_{Z_n}$.

CORROLLARY 2: For any context free grammar G: **T(G)** is semi linear.

2. LINEAR AND SEMI LINEAR LANGUAGES.

The notions of 'linear' and 'semi linear' defined here depend on the grammar. Parikh defines notions of 'linear' and 'semi linear' that apply to **languages** independent of the grammar.

Let Σ be an alphabet. $\Sigma = \{\sigma_1,...,\sigma_n\}$

> $\psi_\Sigma : \Sigma^* \rightarrow N^n$, where:
> for every $\alpha \in \Sigma^*$: $\psi_\Sigma(\alpha) = <|\sigma_1|_\alpha,...,|\sigma_n|_\alpha>$

ψ_Σ defines the **length characteristic** for each string α.

Example:
If $\Sigma = \{\sigma_1,\sigma_2,\sigma_3\}$ where $\sigma_1 = a$ and $\sigma_2 = b$ and $\sigma_3 = c$, then
$\psi_\Sigma(aabba) = <3,2,0>$ (3 a's, 2 b's, 0 c's)

Let $L \subseteq \Sigma^*$
> $\psi_\Sigma(L) = \{ \psi_\Sigma(\alpha): \alpha \in L\}$

The **length profile** of a language is the set of length characteristics of its strings.

For instance: Let $\Sigma = \{\sigma_1,\sigma_2\}$ where σ_1=a and σ_2=b.

$\psi_\Sigma(a^nb^m(n,m\geq0)) = \{<n,m>: n,m \in N\}$

$\psi_\Sigma(a^nb^n(n,m\geq0)) = \{<0,0>,<1,1>,<2,2>,\dots \}$

We define **addition** of length characteristics pointwise:

> $<x_1,...,x_n> + <y_1,...,y_n> = <x_1+y_1,...,x_n+y_n>$

So $<3,4> + <7,8> = <3+7,4+8> = <10,12>$.

Let $I = \{i_1,...,i_k\}$ and $A = \{a_1,...,a_m\}$ be **finite** subsets of N^n.
We will define the length profile **derived from initial** profile I and **auxiliary** profile A.

> DER[I,A] the profile derived from I and A, is the smallest profile such that:
> 1. $I \subseteq$ DER[I,A].
> 2. If $d \in$ DER[I,A] and $a \in A$, then $d + a \in$ DER[I,A].

We can write this in a bit different way:

Fact: Let again $I = \{i_1,\dots,i_k\}$ and $A = \{a_1,\dots,a_m\}$
> DER[I,A] =
> $\{x \in N^n$: for some $i \in I$ and some numbers $n_1,\dots,n_m \geq 0$:
> $x = i + n_1 \times a_1 + \dots + n_m \times a_m \}$

This means that DER(I,A) is the set of all characteristics that are a **linear function** of some $i \in$ I and $a_1 \ldots a_m$.
So: you can take any $i \in$ I and choose any numbers $n_1 \ldots n_m \geq 0$:
then $i + n_1 \times a_1 + \ldots + n_m \times a_m$ is in DER(I,A).
This motivates the terminology **linear**.
Now we define linear and semilinear profiles:

Let $X \subseteq N^n$.
Profile X is **linear** iff for some **finite** profiles I,A $\subseteq N^n$: X = DER[I,A].

Profile X is **semi linear** iff for some **linear** $X_1 ... X_n$: $X = X_1 \cup ... \cup X_n$.

Examples:
Both profiles $\psi_\Sigma(a^n b^m (n,m \geq 0))$ and $\psi_\Sigma(a^n b^n (n,m \geq 0))$ are linear profiles.

$\psi_\Sigma(a^n b^m (n,m \geq 0))$ = DER({<0,0>}, {<0,1>,<1,0>}}

$\psi_\Sigma(a^n b^n (n,m \geq 0))$ = DER({<0,0>},{<1,1>})

We extend function ψ_Σ to apply to strings of terminals and non-terminals in the following way:
-if $\alpha \in (V_N \cup V_T)^*$, then $\alpha \lceil V_T$ is the result of deleting every non-terminal symbol from α.
- if $\alpha \in (V_N \cup V_T)^*$, then $\psi_A(\alpha) = \psi_A(\alpha \lceil V_T)$.

Thus, the length characteristic of a string of terminals and non-terminals is the length characteristic of the result of deleting the non-terminals (i.e. we ignore non-terminals in a length characteristic).

We finally come to the theorem that is known as Parikh's theorem:

PARIKH'S THEOREM: Every context free language is semi linear.

PROOF:
In this proof I will use boldface to indicate the notion of **derived tree set** and non-boldface to indicate the notion of **derived profile**.
Remember: yield(**T**) = {yield(T): T $\in$ **T}**

We show that if L is a context free language in alphabet Σ and G is a context free grammar for L, then for each $Z \subseteq V_N$: yield($\mathbf{T_Z}$) is linear.

Since $\mathbf{T_Z} = \mathbf{DER[I_Z,A_Z]}$, $\psi_\Sigma(\text{yield}(\mathbf{T_Z})) = \psi_\Sigma(\text{yield}(\mathbf{DER[I_Z,A_Z]}))$

This means that it suffices to show that:

ψ_Σ(yield(**DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]**)) = DER[ψ_Σ(yield($\mathbf{I_Z}$)), ψ_Σ(yield($\mathbf{A_Z}$))].

Step A. ψ_Σ(yield(**DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]**)) $\subseteq$ DER[ψ_Σ(yield($\mathbf{I_Z}$)), ψ_Σ(yield($\mathbf{A_Z}$))].

Assume that $x \in \psi_\Sigma$(yield(**DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]**)).
Then there is some tree $T \in$ **DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]** such that ψ_Σ(yield(T)) = x.

$T \in$ **DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]** means that there is some I-tree $T_i \in \mathbf{I_Z}$ and some A-trees $T_{a1},\dots,T_{am} \in \mathbf{A_Z}$ such that T is the result of a finite sequence of adjunctions of trees $T_{a1}\dots T_{am}$ starting with T_i.
These adjunctions will take place in a certain order, and it may well be that a particular tree T_{ai} will need to be adjoined at more than one place in the sequence.
But, since $T \in$ **DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$],** in the end you will get T, by definition of **DER[$\mathbf{I_Z}$,$\mathbf{A_Z}$]**.

This means that for each tree T_{ai} there is a certain number of times n_i that that tree is adjoined in this sequence starting with T_i and ending with T.

Now, there is an obvious fact about adjunction:

Fact: ψ_Σ(yield(ADJ[T,T_{ai},k]) = ψ_Σ(yield(T)) + ψ_Σ(yield(T_{ai}))

Say that the yield of T is $\alpha_1\alpha_2\alpha_3$ with characteristic $\psi_\Sigma(\alpha_1\alpha_2\alpha_3)$.
And the yield of T_{ai} is $\beta_1 B\beta_2$ with characteristic $\psi_\Sigma(\beta_1 B\beta_2) = \psi_\Sigma(\beta_1\beta_2)$
(since for characteristic we only count terminal symbols)
And the adjunction is at node k, and the yield of T_n is α_2.
Then the yield of ADJ(T,T_{ai},k) = $\alpha_1\beta_1\alpha_2\beta_2\alpha_3$
and the characteristic of the yield of this tree is $\psi_\Sigma(\alpha_1\beta_1\alpha_2\beta_2\alpha_3)$ =
$\psi_\Sigma(\alpha_1\alpha_2\alpha_3) + \psi_\Sigma(\beta_1,\beta_2)$.

But this means that:
ψ_Σ(yield(T)) = ψ_Σ(yield(T_i)) + $n_1 \times \psi_\Sigma$(yield(T_{a1})) + … + $n_m \times \psi_\Sigma$(yield(T_{am}))

Now, obviously ψ_Σ(yield(T_i)) $\in \psi_\Sigma$(yield($\mathbf{I_Z}$**))** and
for each i: ψ_Σ(yield(T_{ai})) $\in \psi_\Sigma$(yield($\mathbf{A_Z}$)).

Hence, by definition of derived profile:

ψ_Σ(yield(T)) $\in$ DER[ψ_Σ(yield($\mathbf{I_Z}$)), ψ_Σ(yield($\mathbf{A_Z}$))

Since ψ_Σ(yield(T)) = x, we have proved that:
$x \in$ DER[ψ_Σ(yield($\mathbf{I_Z}$)), ψ_Σ(yield($\mathbf{A_Z}$))

This proves step A.

Step B. $\mathrm{DER}[\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z})), \psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))] \subseteq \psi_\Sigma(\mathrm{yield}(\mathbf{DER[I_Z,A_Z]}))$.

Assume $x \in \mathrm{DER}[\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z})), \psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))]$.

$\mathbf{I_Z}$ is a finite set of trees, so $\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z}))$ is a finite profile $\{i_1,\dots,i_k\}$.
$\mathbf{A_Z}$ is a finite set of trees, so $\psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))$ is a finite profile $\{a_1,\dots,a_m\}$.

$x \in \mathrm{DER}[\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z})), \psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))]$ means that for some $i \in \psi_\Sigma(\mathrm{yield}(\mathbf{I_Z}))$ and some numbers $n_1\dots n_m$:

$$x = i + n_1 \times a_1 + \dots + n_m \times a_m$$

Choose a tree $T_i \in \mathbf{I_Z}$ with $\psi_\Sigma(\mathrm{yield}(T_i)) = i$ and choose for each j:
a tree $T_{aj} \in \mathbf{A_Z}$ with $\psi_\Sigma(\mathrm{yield}(T_{aj})) = a_j$.

Let T be the result of adjoining each tree T_{aj} n_j times in T_i.
So you start with T_i, and you choose some sequence of adjunctions (it doesn't really matter which sequence) such that in that sequence each tree T_{aj} gets adjoined n_j times.

Since every auxiliary tree in $\mathbf{A_Z}$ is adjoinable in any initial tree in $\mathbf{I_Z}$, every auxiliary tree in $\mathbf{A_Z}$ is adjoinable in every tree in $\mathbf{DER(I_Z,A_Z)}$, and the result of adjunction is again in $\mathbf{DER(I_Z,A_Z)}$.
This means that tree $T \in \mathbf{DER(I_Z,A_Z)}$.
And this means that $\psi_\Sigma(\mathrm{yield}(T)) \in \psi_\Sigma(\mathrm{yield}(\mathbf{DER(I_Z,A_Z)}))$

But, the same argument about the characteristic of adjunction tells us that:

$$\psi_\Sigma(\mathrm{yield}(T)) = \psi_\Sigma(\mathrm{yield}(T_i)) + n_1 \times \psi_\Sigma(\mathrm{yield}(T_{a1})) + \dots + n_m \times \psi_\Sigma(\mathrm{yield}(T_{am})) = i + n_1 \times a_1 + \dots + n_m \times a_m .$$

So $\psi_\Sigma(\mathrm{yield}(T)) = x$, and hence:

$x \in \psi_\Sigma(\mathrm{yield}(\mathbf{DER(I_Z,A_Z)}))$

This proves step B.

Putting steps A and B together, we have shown:

$\psi_\Sigma(\mathrm{yield}(\mathbf{DER[I_Z,A_Z]})) = \mathrm{DER}[\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z})), \psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))]$.

And, with Parikh's theorem about linear tree sets, this means:

$\psi_\Sigma(\mathrm{yield}(\mathbf{T_Z}])) = \mathrm{DER}[\psi_\Sigma(\mathrm{yield}(\mathbf{I_Z})), \psi_\Sigma(\mathrm{yield}(\mathbf{A_Z}))]$

Thus, for each, $Z \subseteq V_N$, $\mathrm{yield}(\mathbf{T_Z})$ is linear.

Since $L(G) = \mathrm{yield}(\mathbf{T}_{Z_1}) \cup \dots \cup \mathrm{yield}(\mathbf{T}_{Z_n})$, L(G) is semi linear, and we have proved Parikh's theorem: every context free language is semi linear.

3. EXAMPLES OF LINEAR LANGUAGES THAT ARE NOT CONTEXT FREE.

Example 1: $a^nb^nc^n(n\geq 0)$ is linear.

Let $\Sigma = \{\sigma_1,\sigma_2,\sigma_3\}$ where σ_1=a and σ_2=b and σ_3=c.

$\psi_\Sigma(a^nb^nc^n(n\geq 0)) = \{<n,n,n>: n \geq 0\}$

I = {<0,0,0>} A = {<1,1,1>}
DER[I,A] = {<0,0,0>,<1,1,1>,<2,2,2>,...}

Example 2: $a^nb^mc^nd^m$ $(n,m\geq 0)$ is linear.

Let $\Sigma = \{\sigma_1,\sigma_2,\sigma_3,\sigma_4\}$ where σ_1=a and σ_2=b and σ_3=c and σ_4=d.

$\psi_\Sigma(a^nb^mc^nd^m(n,m\geq 0)) = \{<n,m,n,m>: n,m \geq 0\}$

I = {<0,0,0,0>} A = {<1,0,1,0>, <0,1,0,1>}
DER[I,A] =
{<0,0,0,0>,<1,0,1,0>,<0,1,0,1>,<1,1,1,1>,<2,1,2,1>,<1,2,1,2>,<2,2,2,2>,...
<2,0,2,0>,<3,0,3,0>,...}

Example 3: αα is linear.

The fact that αα is semi linear follows from the fact that $\alpha\alpha^R$ is context free, hence semi linear, and the fact that, obviously, αα and $\alpha\alpha^R$ have the **same** length profile. But you can also show easily directly that αα is linear.

If the alphabet is $\Sigma = \{\sigma_1,...,\sigma_n\}$, then:
$\psi_\Sigma(\alpha\alpha) = \{<2k_1,...,2k_n>: k_1,...,k_n \geq 0\}$

I = {<0,...,0>} A = {<2,0,...,0>, <0,2,...,0>,...,<0,...,2,0>,<0,...,0,2>}

Not semi linear are, for instance, exponential languages: a^{2^n}, a^{n^2}.

4. SEMI LINEAR LANGUAGES THAT ARE NOT LINEAR.

The class of semi linear languages is the closure of the class of linear languages under union. The fact that we introduce the notion of semi linearlity at all means that the class of linear languages is itself not closed under union, i.,e. that there are semi linear languages which are not themselves linear:

Example: $a^nb^n(n\geq 0) \cup ab^m(m\geq 0)$ is semilinear, but not linear.

The profile of $a^nb^n(n\geq 0)$ is:
F = {<0,0>,<1,1>,<2,2>,<3,3>,...}
This is a linear profile: F = DER({<0,0>},{<1,1>})

The profile of $ab^m(m\geq 0)$ is:
G = {<1,0>, <1,1>, <1,2>, <1,3>,...}
This is a linear profile: G = DER({<1,0>},{<0,1>})

The profile of $a^nb^n(n\geq 0) \cup ab^m(m\geq 0)$ is:
$F \cup G$ = {<0,0>,<1,1>,<2,2>,<3,3>,...<1,0>, <1,1>, <1,2>, <1,3>,...}
This is, by definition, a semilinear profile.
The claim is: $F \cup G$ is not itself a linear profile.

Proof:
DER = {<0,0>,<1,1>,…} $\cup$ {<1,0>,<1,1>,<1,2>,…}
We show: the only pair of numbers that can be in A is<0,0>. But, of course, you cannot derive DER from a finite set I with A = {<0,0>}

Case 1: if $n \neq 0$, then <n,n> $\notin$ A
Namely: if <n,n> $\in$ A, then <1,0> + <n,n> $\in$ DER, hence <n+1,n> $\in$ DER.
But <n+1,n> $\notin$ DER.

Case 2: if $n \neq 0$ and $n \neq m$, then <n,m> $\notin$ A.
Namely: if <n,m> $\in$ A, then <1,1> + <n,m> $\in$ DER, hence <n+1,m+1> $\in$ DER, where $n+1 > 1$ and $m+1 \neq n+1$.
But <n+1,m+1> $\notin$ DER.

These two cases show that if <n,m> $\in$ A then n = 0.

Case 3: if $m \neq 0$, then <0,m> $\notin$ A.
Namely, if <0,m> $\in$ A, then <2,2> + <0,m> $\in$ DER, hence <2,m+2> $\in$ DER.
But <2,m+2> $\notin$ DER.

With the above, this shows that if <n,m> $\in$ A, then n=0 and m = 0.

5. CHARACTERIZATION OF SEMI LINEAR LANGUAGES

Which languages are semi linear?

Theorem: For every linear profile, there is a regular language with that profile.

Proof:
Let X be a linear profile (of n-tuples),
X = DER[I,A] with I = $\{i_1,\dots,i_k\}$ and A = $\{a_1,\dots,a_m\}$
Choose an alphabet $\Sigma = \{\sigma_1,\dots,\sigma_n\}$, and choose for each i in A a string α_i of Σ^* of characteristic i and for each $a \in A$ a string α_a of Σ^* of characteristic a.

We define:

$L = \{\alpha_{i1},\dots,\alpha_{ik}\} \times \{\alpha_{a1}\}^* \times \dots \times \{\alpha_{am}\}^*$

Claim: $\psi_\Sigma(L) = X$.

A. Any string in L is of the form: $\alpha_i \wedge \alpha_{a1}^{n1} \wedge \dots \wedge \alpha_{ak}^{nk}$.
$\psi_\Sigma(\alpha_i \wedge \alpha_{a1}^{n1} \wedge \dots \wedge \alpha_{ak}^{nk}) = \psi_\Sigma(\alpha_i) + n_1 \times \psi_\Sigma(\alpha_{a1}) + \dots + n_k \times \psi_\Sigma(\alpha_{ak}) =$
$i + n_1 \times a_1 + \dots + n_k \times a_k \in X$
Hence $\psi_\Sigma(L) \subseteq X$.

B. Let $x \in X$. Then for some $i \in I$ and $n_1,\dots,n_k \geq 0$: $x = i + n_1 \times a_1 + \dots n_k \times a_k$.
As we have just seen, this is the characteristic of a string in L, namely
$\alpha_i \wedge \alpha_{a1}^{n1} \wedge \dots \wedge \alpha_{ak}^{nk}$. So $x \in \psi_\Sigma(L)$.
Hence $X \subseteq \psi_\Sigma(L)$.

So indeed $\psi_\Sigma(L) = X$.
But L is obviously a regular language (formed with $\times$ and * from finite languages.
This proves the theorem.

Corrollary: For any semi linear profile, there is a regular language with that profile.

Proof:
Let D be a semi linear profile. That means that for some linear profiles $P_1,\dots,P_n$:
$D = P_1 \cup \dots \cup P_n$.
Take for each P_i a regular language L_i that has profile P_i.
Then $L = L_1 \cup \dots \cup L_n$ has profile D.
Since the union of regular languages is a regular language, it follows indeed that for every semi linear profile D, some regular language has that profile.

Since every regular language is semi linear (this follows, of course, from Parikh's theorem) we have proved the following:

THEOREM: A language L in alphabet Σ is semi linear iff there is a regular language R in alphabet Σ such that $\psi_\Sigma(L) = \psi_\Sigma(R)$

Thus the semi linear languages are precesely the languages that have the same profile as a some regular language.

Example:
We know that $a^nb^nc^n$ is not contextfree. its profile is $\{<n,n,n>, n \geq 0\}$. Which regular language has that profile? Well, for instance $(abc)^n$.

6. CLOSURE PROPERTIES OF SEMI LINEAR LANGUAGES

Theorem: For any two languages L_1 and L_2 in alphabet Σ:

$$\psi_\Sigma(L_1 \cap L_2) = \psi_\Sigma(L_1) \cap \psi_\Sigma(L_2)$$
$$\psi_\Sigma(L_1 \cup L_2) = \psi_\Sigma(L_1) \cup \psi_\Sigma(L_2)$$

Proof:
- $x \in \psi_\Sigma(L_1 \cap L_2)$ iff
some string α in $L_1 \cap L_2$ has profile x iff
some string α such that $\alpha \in L_1$ and $\alpha \in L_2$ has profile x iff $x \in \psi_\Sigma(L_1)$ and $x \in \psi_\Sigma(L_2)$ iff $\psi_\Sigma(L_1) \cap \psi_\Sigma(L_2)$.

- $x \in \psi_\Sigma(L_1 \cup L_2)$ iff
some string α in $L_1 \cup L_2$ has profile x iff
some string α such that $\alpha \in L_1$ or $\alpha \in L_2$ has profile x iff $x \in \psi_\Sigma(L_1)$ or $x \in \psi_\Sigma(L_2)$ iff $\psi_\Sigma(L_1) \cup \psi_\Sigma(L_2)$.

Corrolary: If X and Y are semi linear profiles in N^n then $X \cup Y$ and $X \cap Y$ are semi linear profiles.

Proof:
Let X and Y be semi linear profiles. Let R_X be a regular language of profile X and R_Y a regular language of profile Y (they exist by the theorem proved above).
By the theorem just proved, $\psi_\Sigma(R_X \cap R_Y) = X \cap Y$ and $\psi_\Sigma(R_X \cup R_Y) = X \cup Y$.
Since R_X and R_Y are regular languages, so are $R_X \cap R_Y$ and $R_X \cup R_Y$. This means that both for $X \cap Y$ and $X \cup Y$ there is a regular language with that profile, and by the theorem earlier, this means that $X \cap Y$ and $X \cup Y$ are semi linear profiles.

Corrollary: If L_1 and L_2 are semi linear languages, then $L_1 \cap L_2$ and $L_1 \cup L_2$ are semi linear languages.

Proof:
Let L_1 and L_2 be semi linear languages. This means that their profiles $\psi_\Sigma(L_1)$ and $\psi_\Sigma(L_2)$ are semi linear. By the theorem proved above $\psi_\Sigma(L_1 \cap L_2) = \psi_\Sigma(L_1) \cap \psi_\Sigma(L_2)$ and
$\psi_\Sigma(L_1 \cup L_2) = \psi_\Sigma(L_1) \cup \psi_\Sigma(L_2)$, and by the corrollary, $\psi_\Sigma(L_1) \cap \psi_\Sigma(L_2)$ and $\psi_\Sigma(L_1) \cup \psi_\Sigma(L_2)$ are semi linear. Hence, $L_1 \cap L_2$ and $L_1 \cup L_2$ are semi linear.

Theorem: The class of semi linear languages is **not** closed under complementation.

Proof:
Let $\Sigma = \{a,b\}$. Look at the language a^{2^n} b $(n\geq 0)$.
We notice a simple fact:
for each n: $\psi_\Sigma(a^{2^n}b) = \psi_\Sigma(ba^{2^n})$ and $ba^{2^n} \in \Sigma^* - a^{2^n}$ b $(n\geq 0)$.
This means that: $\psi_\Sigma(\Sigma^* - a^{2^n}b\ (n\geq 0)) = \psi_\Sigma(\Sigma^*) = N^n$,
This is because eliminating all strings in a^{2^n} b $(n\geq 0)$ from Σ^* leaves for each characteristic in the profile of Σ^* (which is N^n) another string in, hence no characteristic gets eliminated. So indeed the profile of $\Sigma^* - a^{2^n}$ b $(n\geq 0)$ is N^n.

Since N^n is a linear profile, the language $\Sigma^* - a^{2^n}$ b $(n\geq 0)$ is a linear language, hence a semi linear language. But **its** complement is $\Sigma^* - (\Sigma^* - a^{2^n}b\ (n\geq 0)) = a^{2^n}b\ (n\geq 0)$.
But the latter is, of course, not a semi linear language. So we have found a semi linear language $\Sigma^* - a^{2^n}$ b $(n\geq 0)$ whose complement is not a semi linear language, and hence the class of semi linear languages is not closed under complementation.

Theorem: The class of semi linear languages is closed under homomorphisms.

Proof:
Let $\Sigma_1 = \{\sigma_1,...,\sigma_n\}$ and let $h:\Sigma_1^* \to \Sigma_2^*$ be a homomorphism.
For $x \in N^n$, let: α be a string of Σ_1 such that $\psi_{\Sigma 1}(\alpha)=x$
$h(x) = \psi_{\Sigma 2}(h(\alpha))$
$h(X) = \{h(x): x \in X\}$

The following facts are easy to prove:

Fact 1: $h(x + y) = h(x) + h(y)$

Corrollary: h(DER(I,A))= DER(h(I),h(A))

Fact 2: $\psi_{\Sigma 2}(h(L)) = h(\psi_{\Sigma 1}(L))$

and the theorem follows from this,

7. SOME DISCUSSION.

The theorem which said that semi linear languages are exactly the languages that have the same profile as a regular languages tells us that the notion of semi linearity for languages **by itself** doesn't put any restrictions of the complexity of the language. You can see that easily as follows.

Let L be any language in alphabet $\Sigma = \{\sigma_1,...,\sigma_n\}$. Let c be a symbol not in Σ and look at the language $L' = \Sigma^* \times \{c\} \times L$: any string in Σ^* followed by c followed by any string in L.
Since the profile of L is a subset of the profile of Σ^*, the profile of L' is $N^n \times \{1\}$ (1 for the one c). Since N^n is a linear profile, $N^n \times \{1\}$is obviously also a linear profile. Hence L' is a linear language.
Since L can be any language, it can even be an intractable language, a language that doesn't even have an type 0 grammar.
If so, obviously, L' is going to be intractable as well (if L doesn't have a grammar, L' doesn't either). And this means that there are linear languages that are intractable.

Does this mean that the notion of semi linearity is useless?
Not at all. It should only be recognized that the heart of Parikh's theorem is **not** that context free languages are semi linear, **but** that they are semi linear **for the right reason**, because their **tree set is semi linear**. That is, we're **not** simply interested in (semi) linear languages. We're interested in (semi) linear languages **that can be generated in a (semi) linear way**.

There is a widely shared intuition, that natural languages work this way.
This intuition is based partly on computational concerns, partly on linguistic concerns.

The computational concerns are based I think partly on the fact that semi linearity seems to impose such a reasonable constraint on the operations that you can allow yourself in grammar building, partly on the hope that it might help in explaining why we process what we process fast as fast as we process what we process fast.

The linguistic concerns are based on the fact that the phenomena that have been discovered that pose reasonable challenges to the assumption that natural languages are context free do not bring natural languages outside the class of semi linear languages.

This means that semi linearity of the generated language **is** interesting as a **necessary** constraint on grammars for natural languages. If you look at the grammar formalisms that have been developed over the last twenty years for natural language research, in particular, HPSG grammars (headed phrase structure grammars), unification grammars, categorial grammars, tree adjunction grammars, and many related formalisms, then it can be argued that the Working Hypothesis (formulated by Joshi) that natural languages are semi linear, has played an important role in their formulation. (And even that other prominent grammar formalisms, in particular Lexical Functional Grammar, but also Government and Binding theory, and some versions of the Minimalist program have felt the semi linear wind.