something about
non-projectivity and mild-context sensitivity

from

Dependency Structures and Lexicalized Grammars
M. Kuhlmann 2010

JACOB LOUIS HOOVER
2019-02-06

Word-to-word dependency structures have a history in descriptive linguistics, based on the basic intuition that the structure of a sentence can be captured by the relationships between words.

Assumption: the structure of a sentence is captured by inferior-superior relation between individual pairs of words.

This is easy to formalize as a directed tree.
with a total order on its nodes
(note, not an ordered tree, which has order only on children of each node)

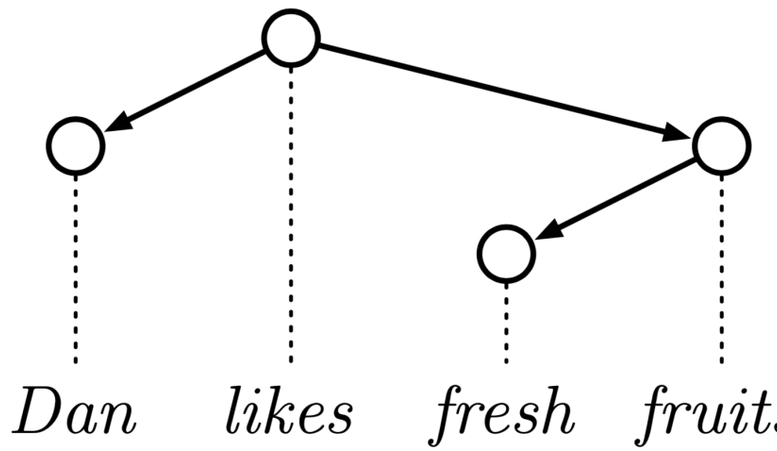
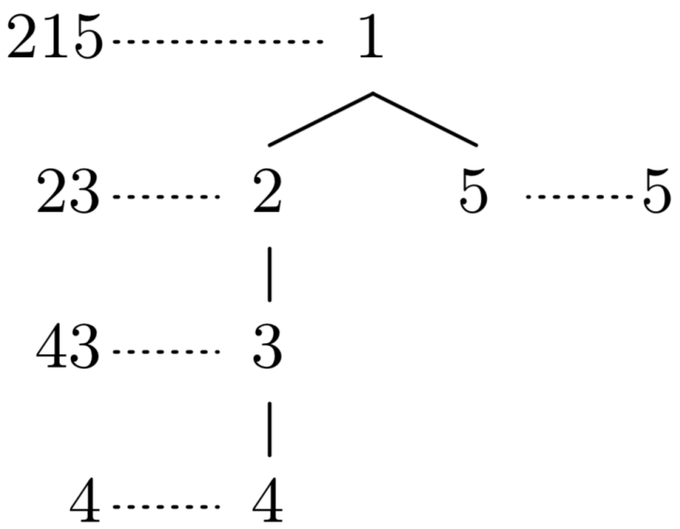
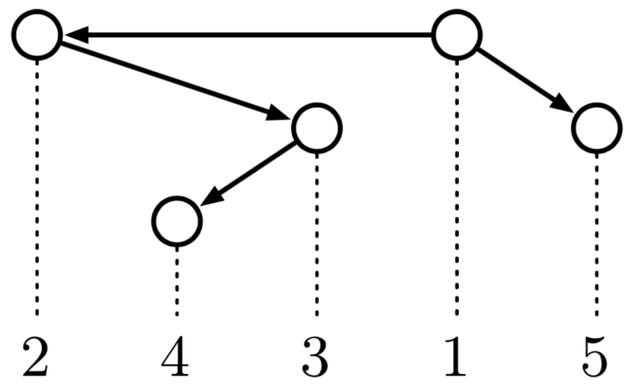


Fig. 1.1: A dependency structure

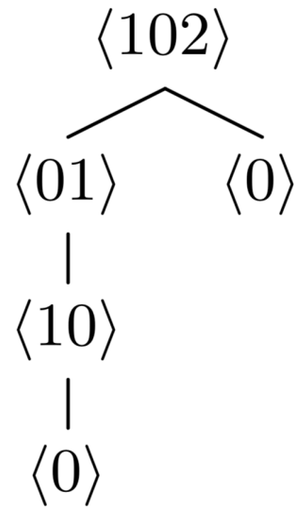
- **treelet**: the local tree formed by a node and its children
- **treelet-ordered tree:** a tree where each node is annotated with a total order on the nodes in the treelet rooted at that node



a treelet-ordered tree



corresponding
dependency structure



corresponding *term*

treelet-ordered trees (together with a procedure for their traversal)
is expressive enough to fully characterize the class of projective dependency structures

Dependency structures don’t look like phrase structure or categorial grammar approaches to syntactic structures…

how do they relate?

what are the linguistically relevant constraints on dependency structures?

A CFG

1. generates parse trees
2. generates strings

$$
\begin{aligned}
\text{S} &\rightarrow \text{SUBJ } \textit{likes} \text{ OBJ} \\
\text{SUBJ} &\rightarrow \textit{Dan} \\
\text{OBJ} &\rightarrow \text{MOD } \textit{fruits} \\
\text{MOD} &\rightarrow \textit{fresh}
\end{aligned}
$$

Fig. 1.2: A context-free grammar

This CFG is **lexicalized**

:= has exactly one terminal per production

A (**lexicalized**) CFG

1. generates parse trees
2. generates strings
3. generates **dependency structures**

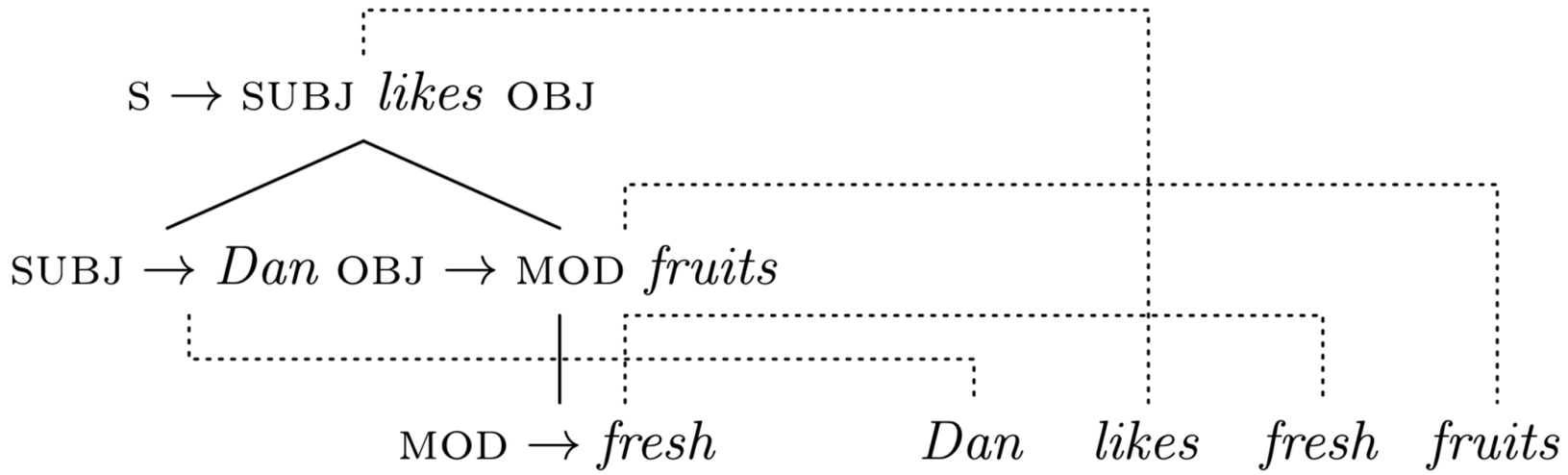
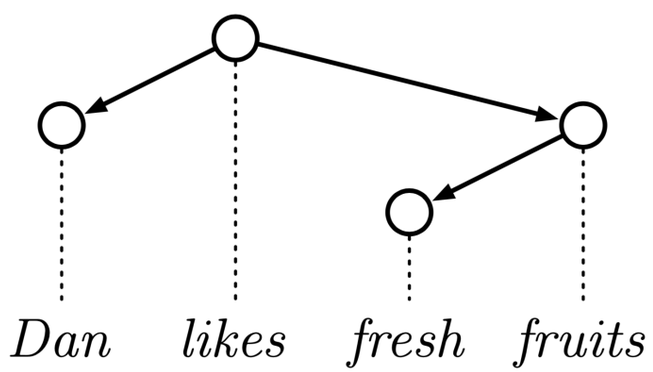
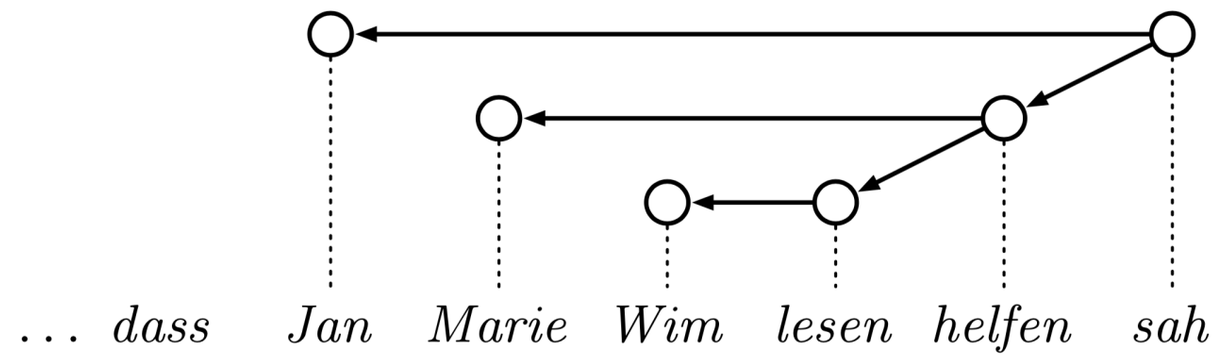


Fig. 1.3: Lexicalized derivations induce dependency structures

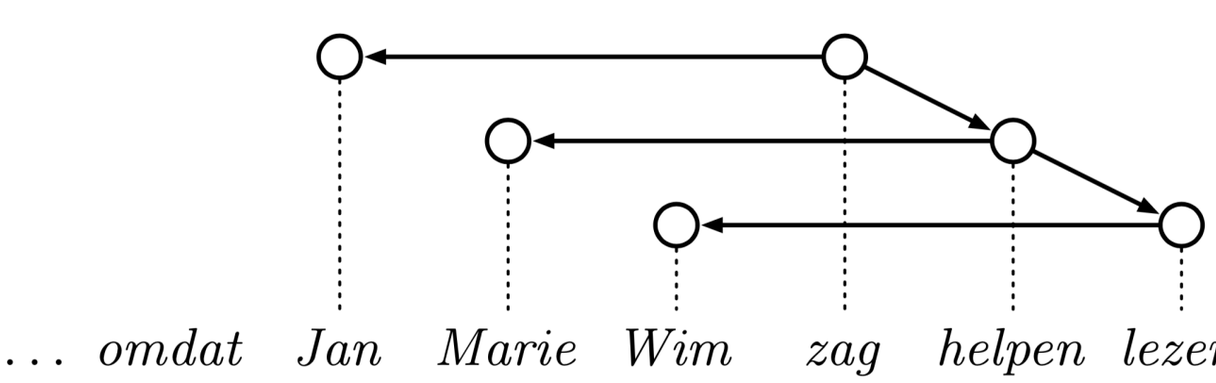


Not all linguistically relevant dependency structures can be induced from lexicalized CFG derivations.



(German) nested dependencies

- possible with **CFG**
- **projective** dependency graph



(Dutch) cross-serial dependencies

- possible with **TAG**, but not CFG
- **well-nested** dependency graph

Fig. 1.4: Nested and cross-serial dependencies

How to characterize the structural properties of these dependencies, and how do they relate to formalisms that induce them?

Outline

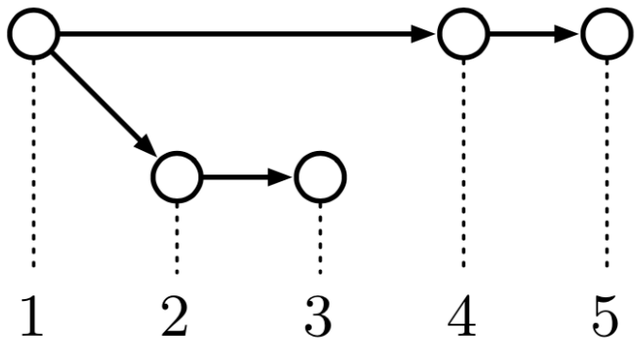
1. **projective**—each subtree is an interval in the string (lexicalized CFGs)

- nonprojective
  2. **block-restricted/bounded degree**—subtrees can be spread over a bounded number of intervals (LCFRSs)
  3. **weakly non-projective**—edges may not overlap
  4. **well-nested**—pairs of disjoint subtrees may not cross (related to parenthesis matching)

well-nested *and* bounded degree k (Coupled-CFGs)
(case $k = 2$: TAGs)

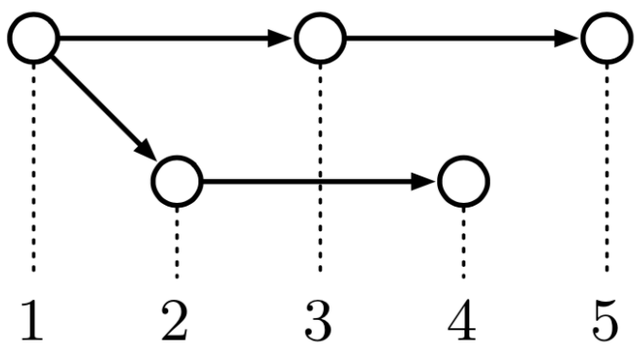
ex:

projective



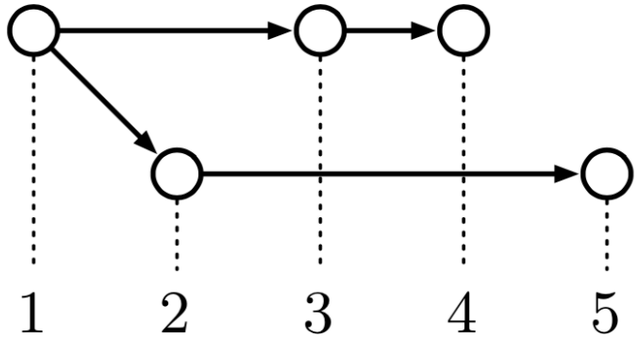
(a) D_1

~~projective~~, **~~well-nested~~**



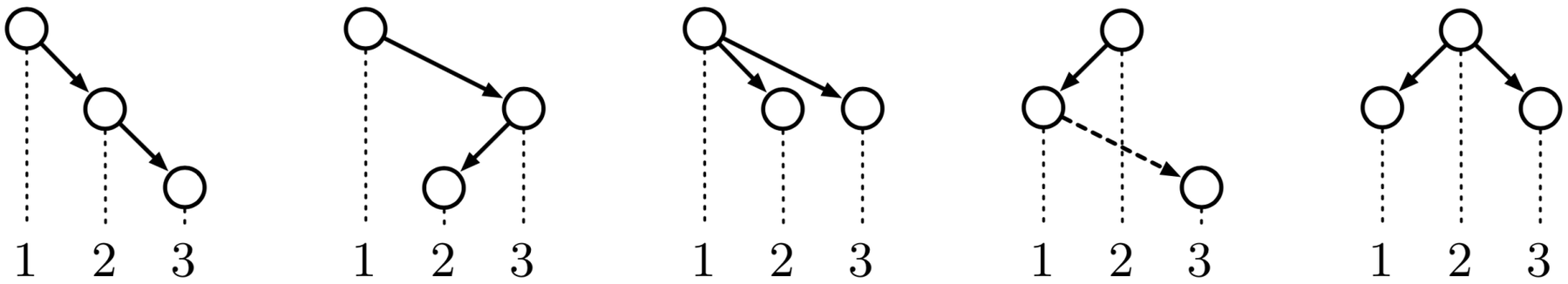
(c) D_2

~~projective~~, **well-nested**



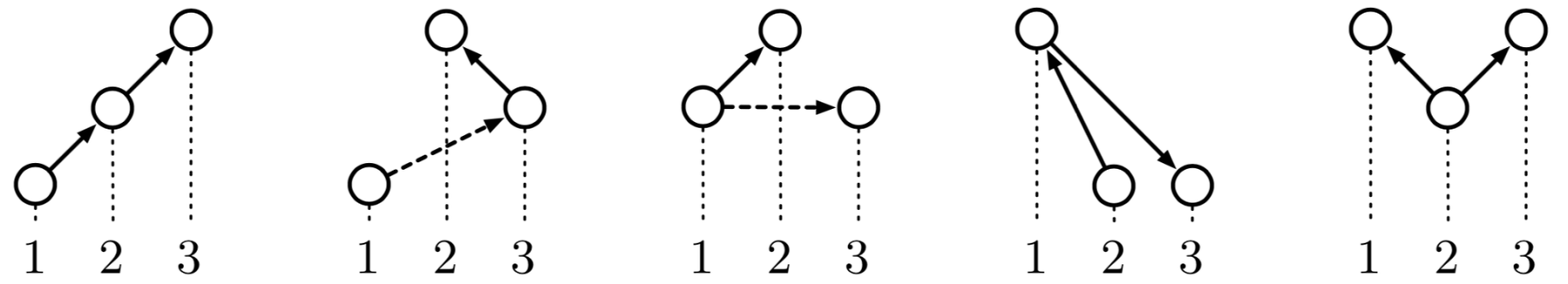
(e) D_3

Projective dependency structures



(a) D_1 (b) D_2 (c) D_3 (d) D_4 (e) D_5

Fig. 3.1: Five (of nine) dependency structures with three nodes



(a) D_1 (b) D_2 (c) D_3 (d) D_4 (e) D_5

‘no edges crossing projection lines’ is not a good definition.
Instead: the definition used is
a dependency is **projective** iff each subtree forms an interval

Danish Dependency Treebank

Prague Dependency Treebank

	DDT		PDT 1.0		PDT 2.0	
projective	3 730	84.95%	56 168	76.85%	52 805	77.02%
non-projective	661	15.05%	16 920	23.15%	15 757	22.98%
TOTAL	4 391	100.00%	73 088	100.00%	68 562	100.00%

Table 3.4: The number of projective dependency structures in three treebanks

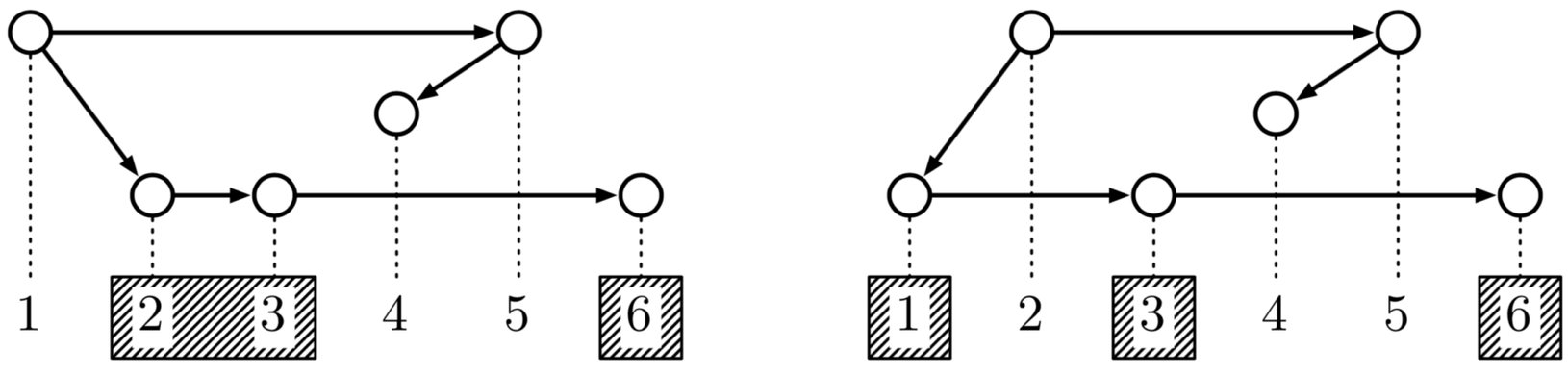
Under the assumption that the three treebanks constitute a representative sample of the set of practically relevant dependency structures, our experiments clearly show that non-projectivity cannot be ignored without also ignoring a significant portion of real-world data. For the DDT, we see that about 15% of all analyses are non-projective; for the PDT, the number is even higher, around 23% in both versions of the treebank. Neither theoretical frameworks nor practical applications that are confined to projective analyses can account for these analyses, and hence cannot achieve perfect recall even as an ideal goal. In a qualification of this interpretation, one should note that projectivity fares much better under an evaluation metric that is based on the set of individual edges, rather than on the set of complete analyses: less than 2% of the edges in the PDT data, and just around 1% of the edges in the DDT data are non-projective [76, 88].

Dependency structures of **bounded block-degree**

In projective dependency structures, each *yield* forms an interval.

In nonprojective, yields may have gaps wrt precedence. The number of intervals ("blocks") is the **block-degree.**

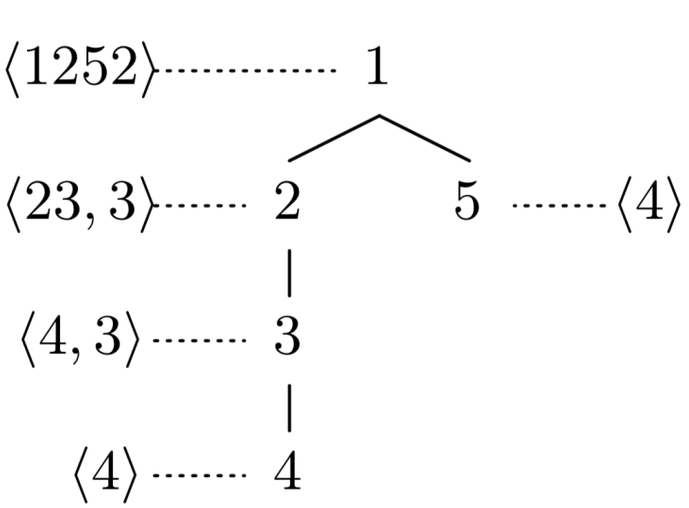
The most immediately obvious way to limit projectivity: put an upper-bound on the block-degree.



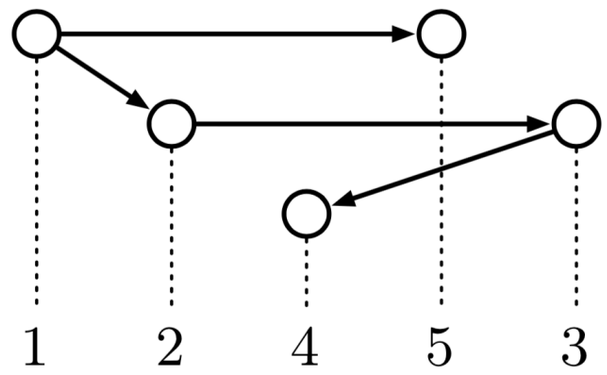
(a) D_1, block-degree 2 (b) D_2, block-degree 3

Fig. 4.2: Two non-projective dependency structures

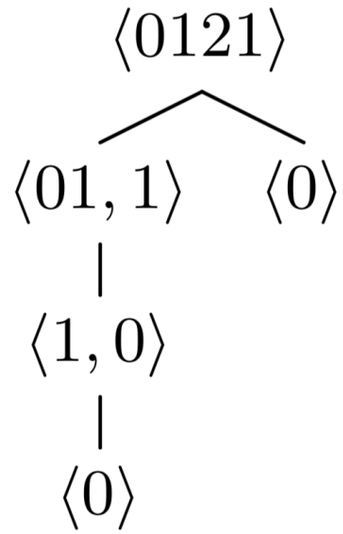
- **block-ordered tree:** (roughly) a tree where each node is annotated with lists of nodes in the treelet rooted at that node that each form an interval



a block-ordered tree



corresponding
dependency structure



corresponding *term*

block-ordered trees (together with a procedure for their traversal)
is expressive enough to fully characterize the class of finite block-degree dependency structures

block-degree	DDT		PDT 1.0		PDT 2.0	
1 (projective)	3 730	84.95%	56 168	76.85%	52 805	77.02%
2	654	14.89%	16 608	22.72%	15 467	22.56%
3	7	0.16%	307	0.42%	288	0.42%
4	–	–	4	0.01%	1	< 0.01%
5	–	–	1	< 0.01%	1	< 0.01%
TOTAL	4 391	100.00%	73 088	100.00%	68 562	100.00%

Table 4.3: Dependency structures of various block-degrees in three treebanks

The general impression that we get from the experiments is that even a small step beyond projectivity suffices to cover virtually all of the data in the three treebanks. Specifically, it is sufficient to go up to block-degree 2: the structures with block-degree greater than 2 account for less than half a percent of the data in any treebank. These findings confirm similar results for other measures of non-projectivity, such as Nivre's *degree* [85] and Havelka's *level types* [44]. Together, they clearly indicate that to contrast only projective and non-projective structures may be too coarse a distinction, and that it may be worthwhile to study classes of dependency structures with intermediate degrees of non-projectivity. The class of dependency structures with block-degree at most 2 appears to be a promising starting point.

Note that, from a linguistic point of view, block-degree is a measure of the discontinuity of a syntactic unit. While we have formulated it for dependency trees, recent work has also evaluated it as a measure for phrase-structure trees [71].

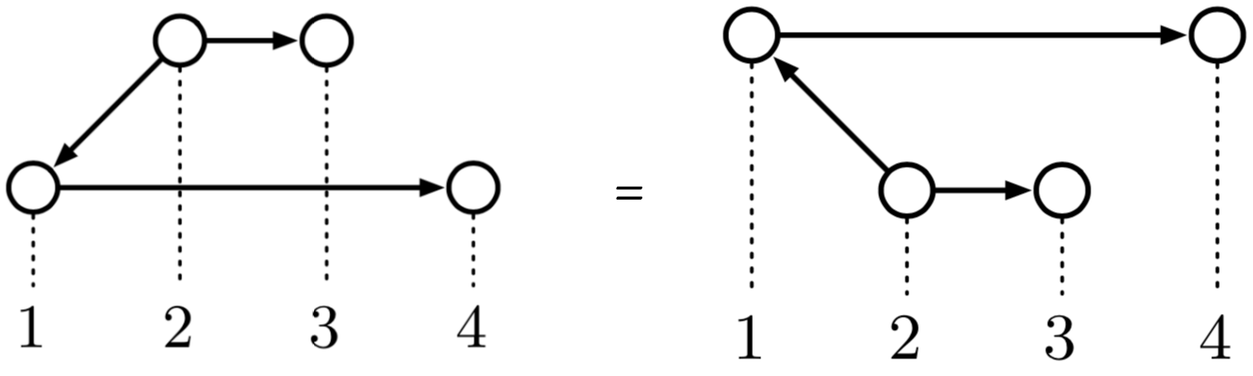
Dependency structures 'without crossings'

Block-degree is a quantitive measure of the nonprojectivity of a dependency structure. Wellnestedness on the other hand is a constraint related not to the degree, but the form of nonprojectivity.

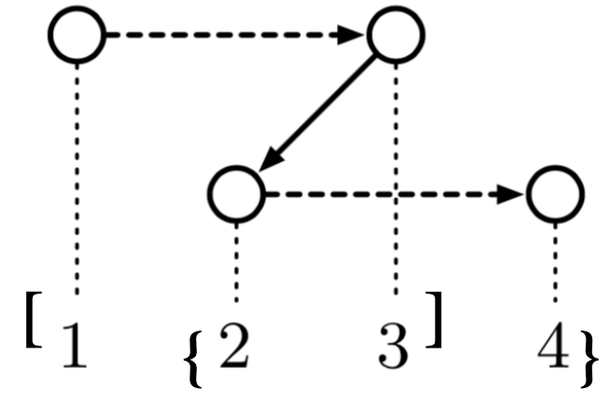
First, define simpler version:

Weakly non-projective = dependency structures that can be drawn without any edges that cross projection lines.

Equivalently: no edges *overlap:* [----------] { ---------- }



weakly non-projective



not weakly non-projective

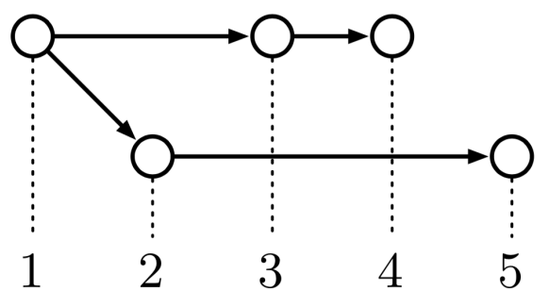
Dependency structures 'without crossings'

Weakly non-projective = dependency structures that can be drawn without any edges that cross projection lines.

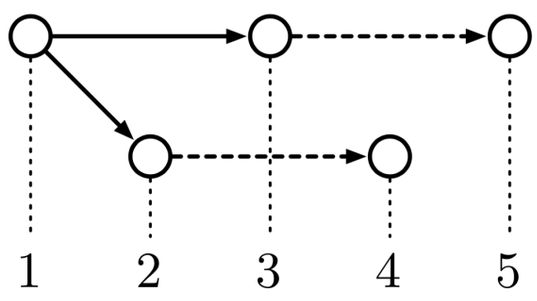
lemma: weakly non-proj. $\Longrightarrow$ block degree ≤ 2.

However, there are block degree 2 structures that are not w.n.p., so it is a weak relaxation of projectivity indeed. Slightly stronger:

Well-nested = no overlapping edges *in disjoint subtrees*
(edges may overlap, as long as a node of one edge governs the other edge)



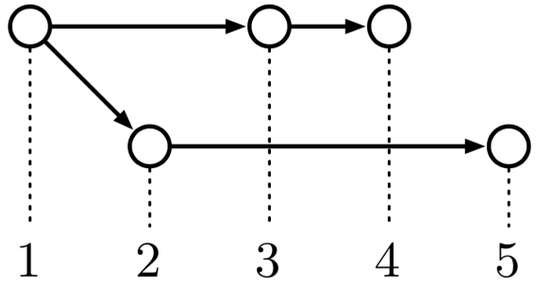
well-nested



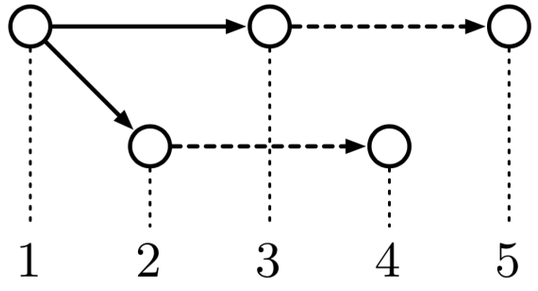
ill-nested

(← neither is w.n.p.)

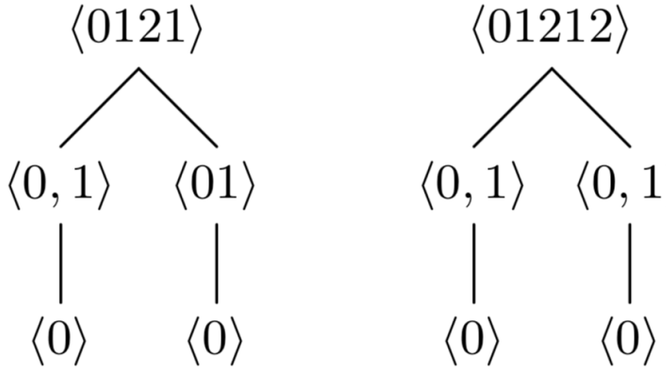
unlike w.n.p., wellnestedness is independent of block degree.



well-nested



ill-nested



Wellnestedness is algebraically transparent:
a structure is well nested iff its term does not
contain *ijij* as a scattered substring anywhere.

All dependency structures

	DDT		PDT 1.0		PDT 2.0	
projective	3 730	84.95%	56 168	76.85%	52 805	77.02%
weakly non-proj.	3 794	86.40%	60 048	82.16%	56 367	82.21%
well-nested	4 386	99.89%	73 010	99.89%	68 481	99.88%
TOTAL	4 391	100.00%	73 088	100.00%	68 562	100.00%

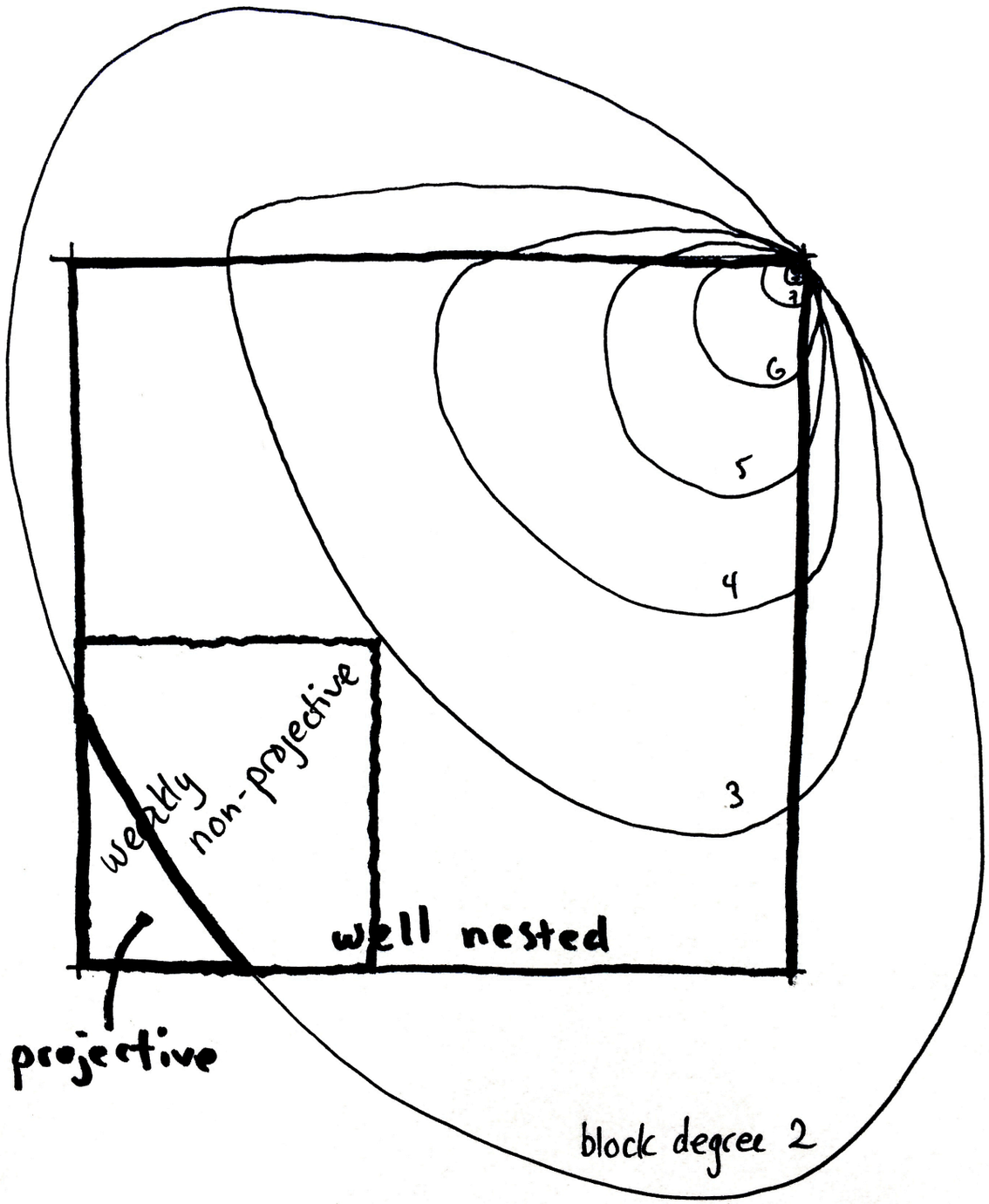
Non-projective dependency structures only

	DDT		PDT 1.0		PDT 2.0	
weakly non-proj.	64	9.68%	3 880	22.93%	3 562	22.61%
well-nested	597	90.32%	16 842	99.54%	15 676	99.49%
TOTAL	661	100.00%	16 920	100.00%	15 757	100.00%

Table 5.1: The number of weakly non-projective and well-nested dependency structures in three treebanks

projective $\subsetneq$ weakly non-projective $\subsetneq$ well-nested $\subsetneq$ unrestricted

block degree 1 block degree ≤ 2 independent of block degree

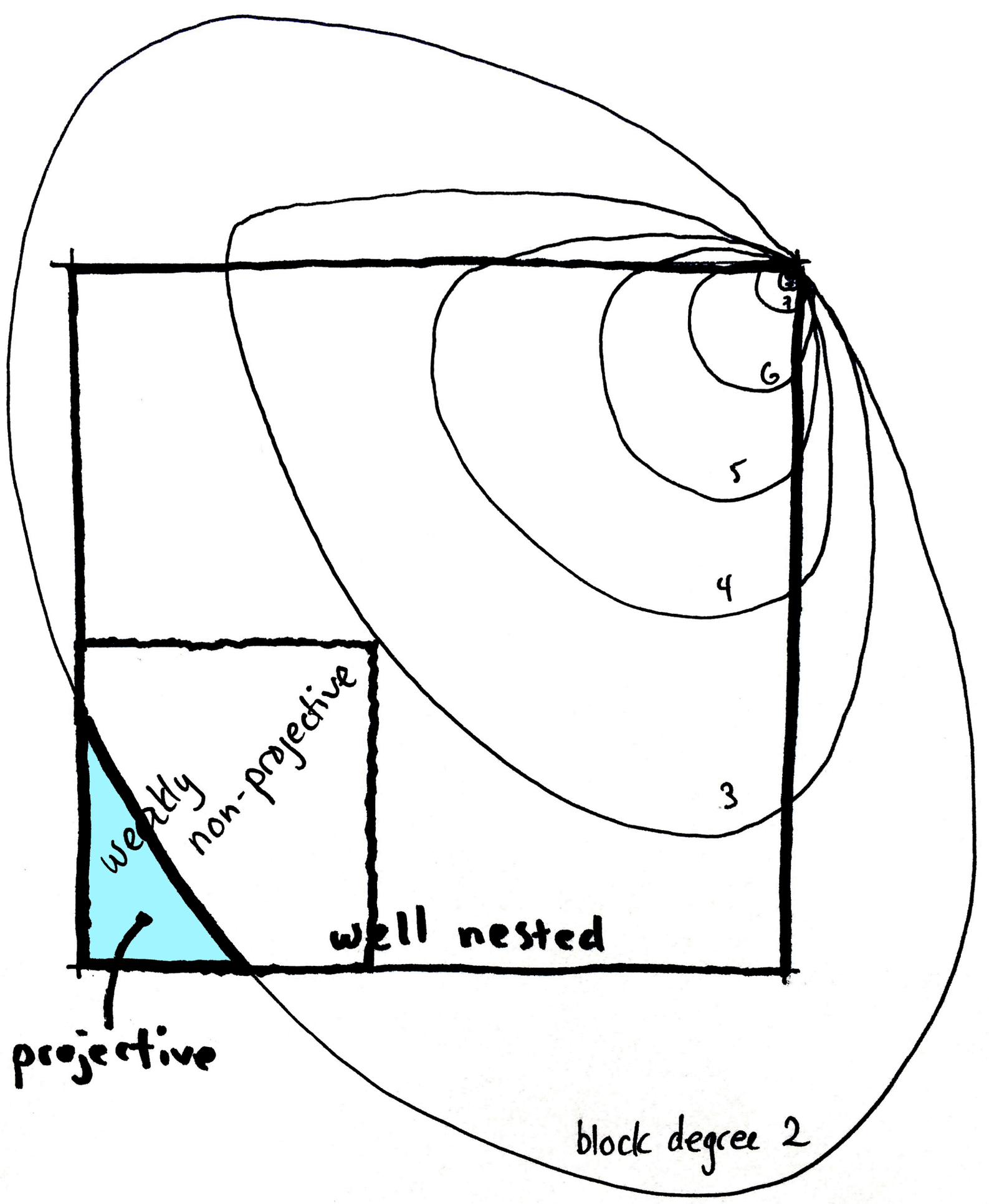


Any CFG.
put in Greibach Normal Form.

(lexicalized)

CFG

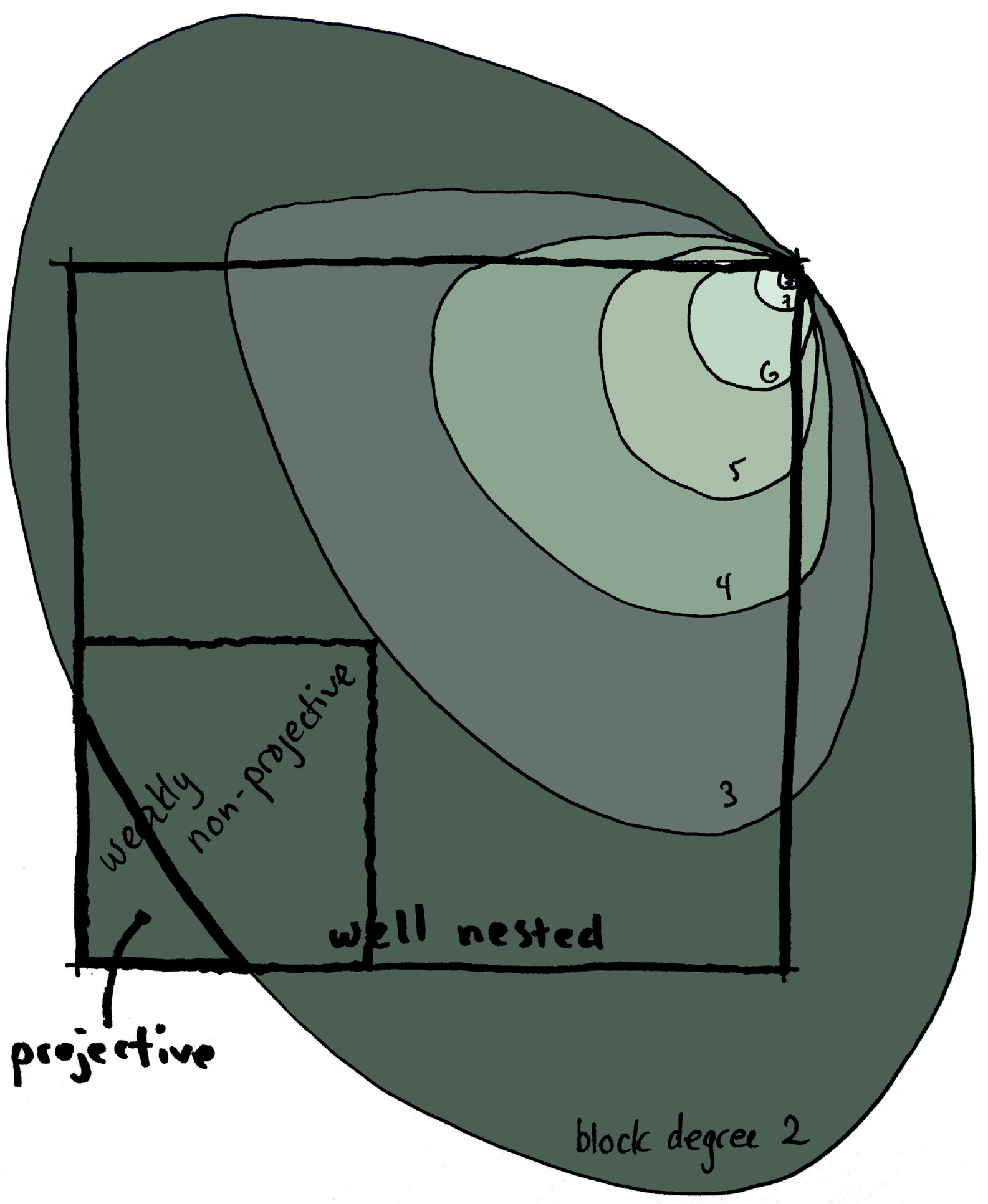
dependency class:
projective



lexicalized

LCFRS(k)

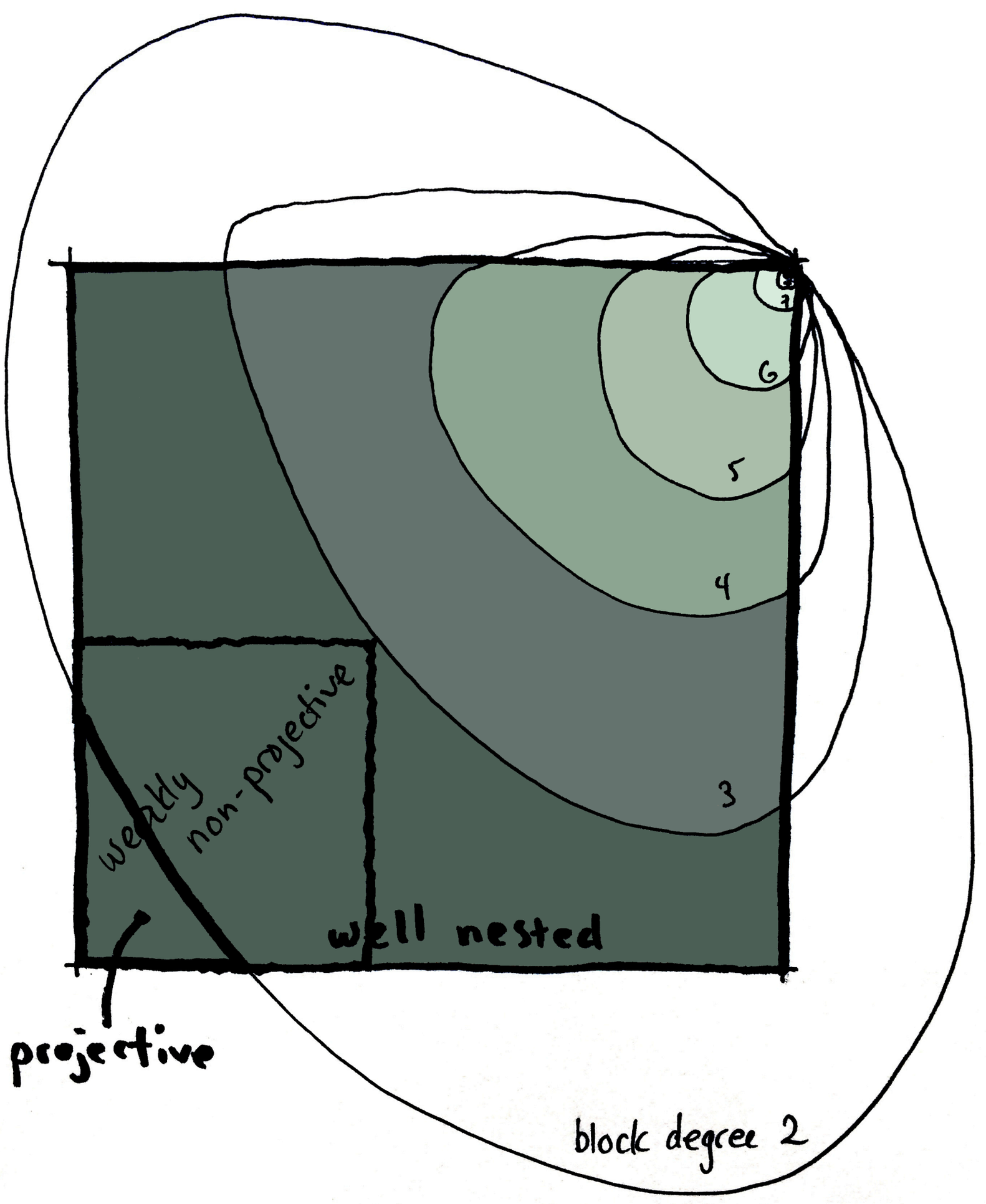
dependency class:
bounded degree k



lexicalized

CCFG(k)

dependency class:
bounded degree k
∩ **well-nested**

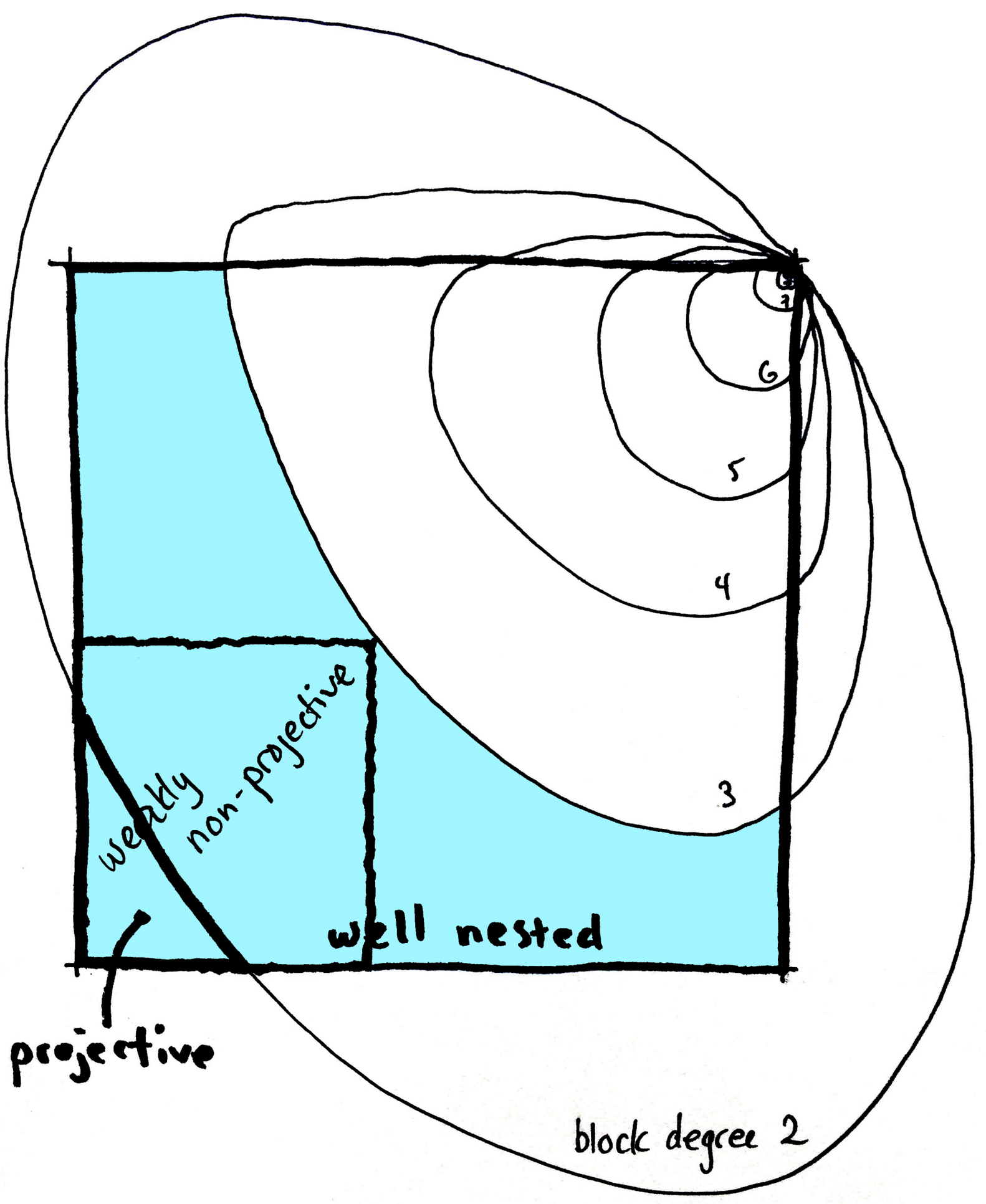


Any TAG can be put in a lexicalized normal form. (?)

(lexicalized)

TAG

dependency class:
bounded degree 2
∩ **well-nested**



The point so far:

- given a lexicalized grammar formalism, its derivations induce dependency structures.

- the extent/form of non-projectivity is a measure of the generative capacity of that formalism

- projective dependencies are not sufficient for Danish and Czech $\Longrightarrow$ lexicalized CFGs aren't either

- close to 99.5% of all the dependency analyses in the treebanks are well-nested and have a block-degree of at most 2

    - so, theoretically a lexicalized TAG could capture almost all these structures.

What's next: (what I don't know yet and can't tell you about)

- any projective dependency structure can be induced by some lexicalized CFG, and lexicalized CGFs only induce such structures

- but, it is not true that every *set* of projective dependency structures can be induced by a lexicalized CFG

- (so, we haven't made the link between grammars and dependency languages).

- The link: a set of dependency structures that can be recognized / induced from a grammar := *regular dependency language*

- regular dependency languages ~ regular term languages. regular term languages are *semilinear* (=> constant growth phenomenon)