Abstract

Progress toward distinguishing clearly between generative and model-theoretic syntactic frameworks has not been smooth or swift, and the obfuscatory term 'constraint-based' has not helped. This paper reviews some elementary subregular formal language theory relevant to comparing description languages for model-theoretic grammars, generalizes the results to trees, and points out that HPSG linguists have maintained an unacknowledged and perhaps unintended allegiance to the idea of strictly local description: unbounded dependencies, in particular, are still being conceptualized in terms of plugging together local tree parts annotated with the SLASH feature. Adopting a description language with quantifiers holds out the prospect of eliminating the need for the SLASH feature. We need to ask whether that would be a good idea. Binding domain phenomena might tell us. More work of both descriptive and mathematical sorts is needed before the answer is clear.

1 Introduction

What sort of system should we employ to give a formal description of a human language? Two sharply distinct views compete for linguists' attention. One emerged in the mid 1950s, when Chomsky persuaded most younger linguists that grammars should be formalized as nondeterministic constructive set generators composed of an initial symbol (traditionally S) and a set of expansion-oriented rewriting rules that build derivations ultimately yielding terminal strings, the whole system being interpreted as a constructive definition of the set of all and only those strings that it could in principle construct. Chomsky (1959) is generally taken to be the foundational work on this type of system.

The other emerged later and made much more hesitant progress. Its mathematical foundations go back to a proposition which was proved independently by several mathematicians (Medvedev 1956 [1964], Büchi 1960, Elgot 1961, Trakhtenbrot 1962), but is most often called Büchi's Theorem (Büchi, 1960). It says a set of strings is finite-state if and only if it is the set of all finite string-like models of a closed formula of weak monadic second-order logic. This offers a new way of characterizing a set of strings: instead of asking what sort of a rewriting system can generate all and only the members of a certain set,

[†]I am grateful to the organizers of the 2019 HPSG conference for inviting me to present a paper, and especially to Gabriela Bîlbîie for her brilliant local organization. I thank the attendees for their questions and discussion. I owe a major debt to James Rogers, whose work is the source of the observations in sections 4 and 5 of this paper. Conversations with Mark Steedman before the conference were extremely useful to me, and after the conference I benefited from careful successive critiques of several drafts by Bob Levine, Bob Borsley, and especially Stefan Müller. They all helped me to correct serious errors I had made. The remaining faults and blunders are solely mine.

ask what sort of logic can express a statement that is true of all and only the sorts of things that are members of that set. Within theoretical computer science it has led to significant results such as that existential second-order logic characterizes stringsets recognizable in nondeterministic polynomial time (Fagin, 1974), and has spawned new subdisciplines such as descriptive complexity theory (Immerman, 1999).

Introducing the second kind of thinking into linguistics created model-theoretic syntax (MTS), but progress toward accepting it has been anything but straight and smooth. McCawley (1968) is widely thought to have presaged it, but did not (see §3). Lakoff (1971) groped toward it but botched the job (Soames, 1974). Kac (1978) clearly adumbrates it but has been overlooked. Johnson & Postal (1980) makes the most serious attempt at it, but contradicts itself with its misguided ideas (in Chapter 14) about formalizing transderivational constraints.

HPSG perhaps comes closer than any other framework to developing in purely MTS mode, but even there the progress has been hesitant. HPSG grew out of GPSG (Gazdar et al., 1985), which was developed as a kind of hybrid theory, a generative grammar with filters. It was only in 1987 that Gerald Gazdar realized that GPSG should have been conceptualized model-theoretically. In unpublished lectures at the 1987 LSA Linguistic Institute he showed how this could be done. By then Pollard and Sag had developed the first version of HPSG (Pollard & Sag, 1987), very much within mainstream generative thinking. By 1994 Pollard and Sag were laying more stress on principles which stated facts about about well-formed structures, but still employed 'schemata' written in the form 'A $\rightarrow$ B C' to outline the gross properties of syntactic constructions, and those are visibly like context-free (CF) rules. A formula like 'Clause $\rightarrow$ NP VP' (and it makes no difference if NP and VP are replaced by complex AVMs) cannot be construed as a statement that is truth-evaluable within the structure of a sentence.

By the second half of the 1990s, both Pollard and Sag had commenced using the term 'constraint-based grammar' to characterize their approach, and some have equated this with MTS. I do not favor this term, and try to explain why in what follows. I then discuss some relevant results concerning subregular families of stringsets, which I then generalizes to trees. I conclude by attempting to bring all this to bear on HPSG.

2 The 'constraint-based' label

The term 'constraint-based grammar' (henceforth CBG) figures prominently in works like Pollard (1996) and the textbook by Ivan Sag et al. (Sag & Wasow 1999; 2nd edition Sag et al. 2003). Müller (2019) takes it to be a syn-

onym for MTS, but it seems to me to have a mainly sociological import: the crucial requirement for membership in the CBG community is not positing transformations (hence not following Chomsky). The CBG membership roll according to Sag & Wasow (1999) includes GPSG, HPSG, LFG, functional unification grammar, dependency grammar, categorial grammar, construction grammar, and the framework Sag was working on in his last years, sign-based construction grammar (SBCG). Sag et al. (2003) adds brief sections on three other syntactic theories which are claimed not to fit into the typology: relational grammar (RG), tree-adjoining grammar (TAG), and optimality theory (OT). But this claim of failure to fit suggests incoherence in the typology.

RG uses no transformations and should surely be classed as CBG — its more highly mathematicized descendant arc pair grammar (APG) is correctly recognized by Sag et al. as 'the first formalized constraint-based theory of grammar to be developed' (Sag et al. 2003: 539).

TAG, by contrast, is straightforward composition-oriented (bottom-up) generative grammar, analogous to categorial grammar but founded on trees rather than strings, hence should surely be excluded (especially if the adjunction operation is taken to be analogous to a generalized transformation, as seems to be suggested by Chomsky 1993: 21).

And if OT is not based in constraints, no framework is: OT posits a universal set of constraints, different grammars being distinguished solely by different orders of application priority, so why does it not fit the classification?

Further puzzlement arises when Culicover & Jackendoff (2005) classify their work as CBG. They class categorial grammar and tree-adjoining grammar with 'mainstream generative grammar' (which seems correct to me), but count their 'simpler syntax' (along with LFG, HPSG, and Construction Grammar) as CBG, despite indications that it employs generative components for each of phonology, syntax, and semantics. They claim that in CBG theories:

> Each constraint determines or licenses a small piece of linguistic structure or a relation between two small pieces. A linguistic structure is acceptable overall if it conforms to all applicable constraints. There is no logical ordering among constraints, so one can use constraints to license or construct linguistic structures starting at any point in the sentence: top-down, bottom-up, left-to-right, or any combination thereof. Thus a constraint-based grammar readily lends itself to interpretations in terms of performance...

Note the locutions 'determines or licenses' and 'license or construct'. Which is it? Constructing sentences? Licensing them as having been constructed correctly? Or stating conditions on the structure they are permitted to have?

In a footnote they suggest that CBG 'was suggested as early as [McCawley (1968)], who referred to "node admissibility conditions"; other terms for this

formulation are “declarative”, “representational”, and “model-theoretic”.’ This embodies a confusion that is worth discussing in detail.

3 Node admissibility conditions

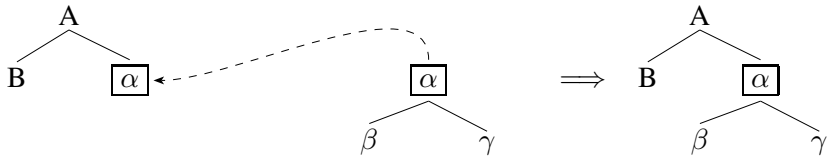
The novel interpretation of CF rules to which Culicover and Jackendoff allude was suggested to James McCawley by Richard Stanley in 1965. The idea was to reinterpret phrase structure rules as ‘node admissibility conditions’ (henceforth NACs). This meant treating a CF rule ‘A $\rightarrow$ B C’ not as meaning ‘if the current string has an A you may rewrite it with the A replaced by B C’, but rather as meaning ‘a subtree consisting of an A-labeled node with a first child labeled B and a second child labeled C is permissible’. Context-sensitive rules can also be thus reinterpreted: ‘A $\rightarrow$ B C / D $\_\_$ E’ would standardly be read as ‘if the current string has an A with a D preceding and an E following, you may rewrite it with the A replaced by B C’; the new interpretation would be: ‘a node labeled A with a first child labeled B and a second child labeled C is permissible if in the tree a D-node immediately precedes the replaced A and an E-node immediately follows it’.

McCawley observed that for context-sensitive rules there was a difference in expressive power between the two interpretations: he exhibited a tiny grammar which under one interpretation generated a single tree and under the other generated nothing. Clarifying this, a later mathematical result of Peters & Ritchie (1969) showing that the NAC interpretation yielded only context-free stringsets (CFLs). However, none of this has much to do with MTS. To start with, ‘node admissibility condition’ (henceforth NAC) was always a misnomer. NACs are not conditions on the admissibility of nodes or trees or anything else. An NAC saying ‘A $\rightarrow$ B C’ doesn’t place any condition on nodes, not even on nodes labeled A: it requires neither that a node labeled A should have the child sequence B C (there could be another NAC saying A $\rightarrow$ D E F) and it doesn’t require that a child sequence B C must have a parent node A (there could be another NAC saying ‘D $\rightarrow$ B C’).

The Stanley/McCawley interpretation makes trees directly answerable to the content of the grammar without the need for Chomsky’s procedure for constructing trees from the information in derivations, so the connection between rules and structures becomes far more transparent. A grammar becomes in effect a finite library of pictures of local regions of a tree, and a tree is well formed iff every region of appropriate size matches one of the pictures (see Rogers 1999, where CF grammars (CFGs) are introduced in this way). It was part of what motivated Gerald Gazdar to reconsider the descriptive value of CF rules.

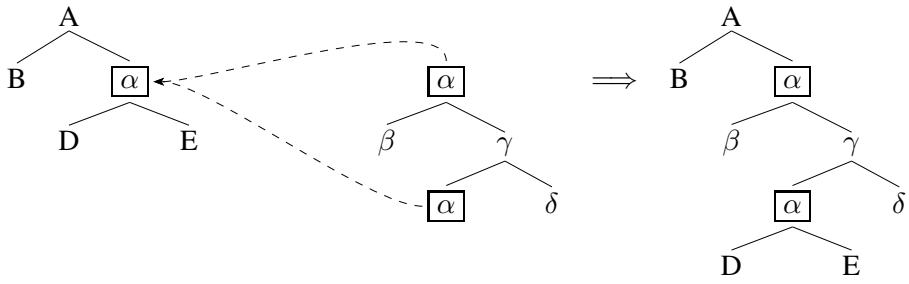
But grammars under the Stanley/McCawley NAC interpretation are purely

generative grammars, though of the composition-oriented type first exhibited in categorial grammar (Ajdukiewicz 1935). A grammar consists of some building blocks and a mode of composition for putting them together. The implicit composition operation for NAC grammars is what I will call *frontier nonterminal substitution*: wherever a frontier node in a developing tree is labeled with a nonterminal symbol α you can plug in some local tree in the grammar that has root node labeled α:



The set of trees generated is the set of all and only those trees with a root label S (or whatever root node label may be specified) and a frontier entirely labeled by terminal symbols (lexical items). The strings generated are all and only the frontiers of those trees.

Tree-adjoining grammar (TAG) also defines composition-oriented generative grammars, differing in that they feature a more complex kind of composition operation, based on *auxiliary trees*, which have a frontier nonterminal node label matching the root node label. These provide for operations of what I will call *internal nonterminal substitution*: a designated internal node labeled α is replaced by an auxiliary tree that has α both as root label and a frontier label, like this:



Neither NAC grammars nor TAGs are anything like MTS. Notice that MTS constraints express *necessary conditions on expression structure*, and well-formedness is determined by *satisfaction of all the constraints*. But no node can ever match more than one NAC, so there could never be a tree that satisfied two or more NACs.

McCawley himself makes an error on this point, saying that 'the admissibility of a tree is defined in terms of the admissibility of all of its nodes, i.e., in the form of a condition which has the form of a logical conjunction' (p. 248). It is in fact a *dis*junction. An NAC is a one-place predicate of nodes; for exam-

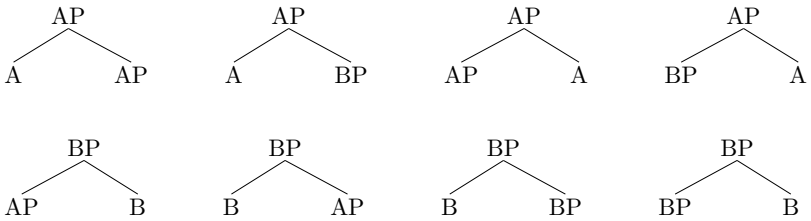
ple, the NAC corresponding to the rule 'A $\rightarrow$ B C', expressed as a property of a node x, says 'x is labeled A and has a left child labeled B and a right child labeled C'. For generality, it will be convenient to add a trivial one-node NAC of depth zero for each terminal symbol: the NAC '*eat*' will correspond to the statement 'x is a node labeled *eat*'. One might want to stipulate that that a node with no child (a node on the frontier, also known as a leaf) must be labeled with a member of the terminal vocabulary (though that rules out some theories of 'empty categories'), and perhaps also that a node that has no parent (the root) is labeled with some designated symbol such as S or Clause; but these are matters of detail. The main thing that has to be true in a tree to make it well-formed according to a set $\varphi_1, \ldots, \varphi_k$ of k NACs is the multiple disjunction shown in (1).

(1) $(\forall x)[\bigvee_{1 \leq i \leq k} \varphi_i(x)]$ 'Every node satisfies the disjunction of all the NACs.'

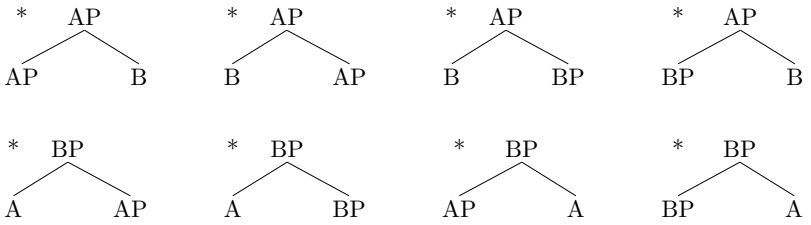
So NACs are in effect atomic propositions, each stating a sufficient condition for some specific local tree (of depth 0 or 1) to be a legitimate subpart of a well-formed tree, and written in a way that is isomorphic to such a subpart. The grammar is really just a finite list of local trees, generating all and only the trees that are entirely composed of the local trees on the list and could be constructed by combining them using frontier nonterminal substitution.

Stanley/McCawley CFG thus construed is a composition-oriented generative formalism employing a set of small building blocks and a set of operations for putting them together to make larger units. In that respect, it is just like categorial grammar, TAG, and Chomsky's minimalism. However, it will be relevant in the next section that there is a particularly easy way to turn a set of NACs into a model-theoretic description that is in many respects equivalent. It depends on the fact that there are only finitely many trees employing a given finite vocabulary V of node labels and given finite bounds on depth (d) and breadth (b). Hence for any finite subset of them interpreted as NACs we can take the complement of that set (relative to all trees of depth $\leq d$ and width $\leq w$ labeled from V), and interpret each as a subtree prohibition. Each local tree T_i in the complement set will be understood as the statement φ_i meaning 'the configuration T_i does not occur as a subtree'. Then a tree $\mathcal{T}$ is well-formed if and only if each φ_i is true in $\mathcal{T}$.

To illustrate, consider this set of NACs constituting a tree-generating grammar defining (in effect) a one-bar-level X-bar theory on binary trees with only two lexical categories, A and B:



Now consider the complement set (which happens in this case to be exactly the same size):



Making sure a tree does NOT contain any of the configurations in the second set yields exactly the same results as making sure it IS entirely composed of the local trees in the first set.

Now in the next section I want to begin to make clear the sense in which this yields an MTS description, though one stated in an extremely primitive description language. Various results were achieved after 1968 seem in retrospect highly relevant to clarifying this point. I will first review the results, and then, returning to my theme, relate them to HPSG.

4 Expressive power of description languages for strings

It was noted in Rogers (1997) that stating a CFG as a set of local trees is strongly analogous to a bigram description of a set of strings. A bigram description over an alphabet Σ is a finite list of 2-symbol sequences over Σ, and a string is grammatical according to it if every length-2 substring of the string (every *factor*, as the formal language theorists put it) is on the list. And as I have pointed out, using the complement of the set of bigrams instead permits the description to be construed in MTS terms.

But bigram descriptions define only a very small and primitive class of stringsets, the SL_2 stringsets. Using depth-1 local trees as building blocks for trees is analogous to using bigrams as building blocks for strings. When implemented in detail it employs a finite vocabulary Σ plus an additional symbol $\rtimes \notin \Sigma$ to mark of the beginning of a string and $\ltimes \notin \Sigma$ to mark the end. A string w is generated iff it begins with some symbol σ such that $\rtimes\sigma$ is one of the bigrams and it ends with some symbol σ such that $\ltimes\sigma$ is one of the bigrams,

and for every substring $\sigma_1\sigma_2$ the string $\sigma_1\sigma_2$ is one of the bigrams. Model-theoretically, it amounts to using a description language of atomic propositions interpreted as substring bans: a proposition ab means 'the substring ab does not occur'.

Letting $k = 3$ then yields the trigram stringsets, a proper superset; letting $k = 4$ yields the quadrigram stringsets, larger still; and so on upward: n-gram stringset for any for any positive integer n can be defined by letting $k = n$. So there is an infinite hierarchy of strictly local (SL) stringsets.

If we add in the results of taking unions, intersections, and complements of SL stringsets we get a strictly larger class known as the Locally Testable stringsets (LT). And again, there is a class LT_2 where the basis is SL_2 stringsets, a class LT_3 where the basis is SL_3 stringsets, and so on.

LT stringsets can be described model-theoretically by allowing not just atomic propositions like $\sigma_1\sigma_2$ (meaning 'the substring $\sigma_1\sigma_2$ does not occur') but also propositional calculus formulas which are conjunctions, disjunctions, or negations of such propositions. Now you can say things like 'either ab does not occur or bc and cd do not both occur', and so on. The class of stringsets describable is now larger, a proper superset of the SL stringsets known as the LT stringsets. For a simple example of a stringset that is LT but not SL, consider a^*ba^*. It contains all and only those strings over $\{a, b\}$ that contain just a single b, and it has no SL_k description for any k. So the apparently very simple notion 'contains a b' is not expressible in a language as primitive as the language of atomic propositions about n-gram presence or absence.

Allowing quantifiers adds considerably to expressive power. If we permit first-order quantification and assume a binary relation symbol $<_1$ intuitively meaning 'immediately precedes' (= 'left-adjacent to' = 'predecessor of') we can describe a larger class of stringsets known as the (locally) Threshold Testable (TT) sets (Thomas 1982: 372). This permits us to verify that a certain substring occurs at least a certain number of times in each string, up to a fixed threshold.

We can step up the expressive power yet more by choosing the binary relation '$<_1^*$', the reflexive transitive closure of $<_1$, interpretable as 'precedes (not necessarily immediately)'.[1] We get a larger family of stringsets, also obtainable by closing LT under union, intersection, complement, and concatenation, and thus known as the 'Locally Testable with Order' class in McNaughton & Papert (1971). But it is most commonly known under the name 'Star-Free' (SF), because the languages can be characterized by expressions very much like regular expressions except that they use the complement operator instead of asteration (Kleene star): every finite stringset is SF; every union of SF stringsets is SF; every concatenation of SF stringsets is SF; the complement of any SF stringset

[1] The $<_1$ relation is first-order definable from $<_1^*$, but not conversely.

is SF; and nothing else is.

An alternative characterization is as the class of *non-counting* stringsets, within which beyond a certain finite limit k there is no further possibility of counting whether there were k consecutive occurrences of some substring or more than k, so that if uv^kw is in a set $uv^{k+1}w$ is in as well.

An important result due to McNaughton & Papert (1971) shows that a language is SF iff it is the set of all finite stringlike structures satisfying a closed formula of first-order logic with '$<_1^*$'. This permits describing the set of all strings over $\{a, b, c\}$ satifying $\forall x[c(x) \Rightarrow \exists y[b(y) \wedge y <_1^* x]]$, in which any occurrence of c has to co-occur with a b somewhere earlier in the string.

As a final step up in expressive power, if we replace first-order logic by weak monadic second-order logic (wMSO), we have a theorem obtained independently by several researchers in the late 1950s (Medvedev 1956 [1964], Büchi 1960, Elgot 1961): using wMSO on string-like models, the describable stringsets are an even larger class, namely the regular (finite-state) stringsets.

Thus we have this tableau of progressively larger and larger families of stringsets:

(2)
a. Strictly Local $\boxed{\text{SL}}$ (finite n-gram lists); SL_2, SL_3, SL_4, ... SL_k
b. Locally Testable $\boxed{\text{LT}}$ (closure of SL under boolean connectives); LT_2, LT_3, LT_4, ... LT_k
c. Threshold Testable $\boxed{\text{TT}}$ (first-order logic with $<_1$); TT_2, TT_3, TT_4, ... TT_k
d. Star-Free $\boxed{\text{SF}}$ (star-free expressions; counter-free automata; FO with $<^*$)
e. Finite-State $\boxed{\text{FS}}$ (regular expressions or grammars; finite automata; wMSO)

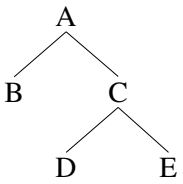
And they form a hierarchy (actually a hierarchy of hierarchies, because there are SL_3 stringsets that are not SL_2, LT_5 stringsets that are not LT_4, and so on).

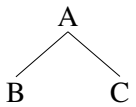
(3) $\boxed{\text{SL}} \subsetneq \boxed{\text{LT}} \subsetneq \boxed{\text{TT}} \subsetneq \boxed{\text{SF}} \subsetneq \boxed{\text{FS}}$

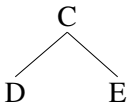
5 Expressive power of description languages for trees

The relevance of this material on the formal language theory of subregular stringsets (see Rogers & Pullum 2011 for more) will become clearer once we generalize to trees. The analog of n-grams are trees of depth $n + 1$ (where an isolated node has depth 0). Under the reformalization discussed above, a CFG can be given as simply a finite set of local trees, i.e. trees of depth 1. These correspond to bigrams: in the linear precedence dimension a bigram has a length

of two; a local tree has a corresponding measurement in the dominance dimension, from the root level to the child level. Just as you can decompose strings into the bigrams they contain, you can decompose a tree into the local trees it contains. There is an overlap of one symbol in each case. The string $abba$ is made up of the bigrams ab, bb, and ba. In an analogous way, he tree in (4a) is made up of the local trees (4b) and (4c).

(4) a. 

b. 

c. 

Adding a superscript τ to remind us that we are now talking about tree-sets rather than stringsets, the family of all tree-sets we can define using local trees as building blocks can be called the SL_2^τ tree-sets. But as with the stringset families SL_2, SL_3, and so on, we can use trees of greater and greater depth as building blocks to get an infinite hierarchy of larger and larger families SL_2^τ (the 2-local tree-sets), SL_3^τ (the 3-local tree-sets), and so on for all positive $n \geq 2$.

But there will be sets of trees we cannot describe with local trees of any finite maximum depth. Consider, for example, the set of binary trees in which most nodes are labeled α but there is *at least one* node somewhere that is labeled β, with α above and below it. This cannot be SL_k^τ for any k: no matter how large k is (i.e., no matter how deep the building-block trees are), a tree with no β will be indistinguishable from one containing a β more than k nodes above or below the bit you're currently checking.

I am simply using a tree analog of a simple theorem about SL string languages here. In an SL string language, a property that we could call Prefix-Blind Suffixing always holds, and the converse is also true. In a strictly k-local stringset (SL_k for some $k \geq 2$), given a string x of length $k-1$, if wxy and vxz are both in the set, then wxz has to be in the set. The occurrence of the suffix z cannot depend on anything before the x, because the x is too long.

The tree analog is that in a strictly k-local tree-set, once a tree contains a subtree T of depth $k-1$, the well-formedness of a tree formed by adding some further subtree below it (closer to the frontier) cannot depend on what occurs above it (closer to the root).

We can develop a more powerful description language by analogy with the LT string languages. Instead of just providing a list of depth-k trees as our grammar, we can close the defined sets under boolean operations by allowing grammars to say things like 'either T_1 does not occur or T_2 and T_3 do not both occur', and so on. This permits us to define for each k the tree-sets that are k-locally testable, which we can call LT_k^τ.

The family of LT_k^τ tree-sets is rich enough to contain the set of all and only those binary-branching trees in which there is at least one node labeled β, but every β has $alpha$ nodes both immediately above it and immediately below it. This is the intersection of three obviously SL_2^τ tree-sets: (i) the trees which have α as the root label and all the frontier labels, (ii) the trees which do not contain any cases of a β node with a β-labeled child, and (iii) the trees in which there is an α-labeled node with a β-labeled child. That proves it is LT_2^τ, because the LT_k^τ tree-sets can be described by propositional calculus formulas in which the primitive propositions say things like 'k-depth subtree T does not occur'. The set just mentioned is describable by the conjunction of three SL_2^τ descriptions couched in terms of primitive propositions.

We can make a yet more powerful description language by allowing ourselves first-order quantification over nodes in a language containing a binary relation symbol $<_2$ to denote 'immediately dominates'. I will call this description language $\text{FO}_{<_2}$. This gives us a still larger family, TT_k^τ, the tree analog of the k-locally threshold testable stringsets. With this as our description language, we finally have enough expressive power to describe the set of all trees in which all nodes are labeled α except for a unique β-labeled node which has α-labeled nodes both above it (its parent) and below it (all of its children).

We will use $x <_2 y)$ for 'x immediately dominates y'. Let '$\text{ROOT}(x)$' (meaning intuitively that x is the root of the tree) be defined by '$\neg(\exists y)[y <_2 x]$' (which says that nothing immediately dominates x). Define '$\text{LEAF}(x)$' (meaning that x is on the frontier of the tree) by '$\neg(\exists y)[x <_2 y]$' (which says that x does not immediately dominate anything). And define '$\text{BINARY}(x)$' (meaning that x is a node with exactly two children) by this formula:

(5) $(\exists y, z)[x <_2 y \wedge x <_2 z \wedge y \neq z \wedge (\forall u)[x <_2 u \Rightarrow (u = y \vee u = z)]]$

'The node x has exactly two distinct children.'

Then the set of all purely binary-branching trees is the set in which BINARY-ONLY is true:

(6) $\text{BINARYONLY} \equiv_{\text{def}} (\forall x)[\text{ROOT}(x) \vee \text{LEAF}(x) \vee \text{BINARY}(x)]$

'Every node is either the root, or a leaf, or binary-branching.'

Let LONELYBETA be the proposition that all nodes are labeled α except for a unique node labeled β like this:

(7) $\text{LONELYBETA} \equiv_{\text{def}} (\exists x)[\beta(x) \wedge (\forall y)[(\beta(y) \Rightarrow (y = x)) \wedge (\neg\beta(y) \Rightarrow \alpha(y))]]$

'There is an x that is labeled β, and x is the only node labeled β (i.e., any y labeled β is identical with x), and any other node (i.e., any y not labeled β) is labeled α.'

Now the set we want is the set satisfying the conjunction of BINARYONLY and LONELYBETA.

A further increase in expressive power can be obtained (though to save space I won't illustrate) if we move to a language in which the relation $<_2$ is replaced by its reflexive and transitive closure $<_2^*$, so that we can say 'dominates' as well as 'immediately dominates'.

And we have still not reached maximum expressive power for languages describing tree-sets. It would still not be possible to describe a set of trees in which, for some fixed k, the depth in terms of k-depth subtrees is always an even number. To achieve that, we could move from first-order logic to wMSO, which is capable of describing such sets. It was proved in the 1960s (Thatcher 1967, Thatcher & Wright 1968) that a set of trees is recognizable by a finite-state tree automaton if and only if its string yield is a CFL, and by a fundamental result of Doner (1970), wMSO on finite trees yields exactly the expressive power of finite-state tree automata.

McCawley did not know about Doner's result, and may not have known Thatcher and Wright's work, but he did recognize that the string yield of a tree-set defined by NACs (i.e., defined using SL_2^τ) is always a CFL. This insight influenced Gerald Gazdar in devising what came to be known as generalized phrase structure grammar in 1978–1979. I think it influenced the creators of its direct heir HPSG as well. But it is natural to ask what sorts of stringset you can define by using more powerful description languages. And considering the string yields of the larger and larger tree-sets describable with the analogs of the more and more powerful description languages just briefly reviewed reveals something rather amazing — though the proofs are straightforward, some covered in textbooks like Libkin (2004) and others just basically trusted mathematical folklore:

(8)

TYPE OF TREE-SET DEFINITION	STRINGSET YIELD
strictly 2-local (local trees $\equiv$ NACs)	context-free
strictly 3-local (depth-2 trees)	context-free
strictly k-local (depth-$k-1$ trees, $k > 3$)	context-free
context-sensitive 2-local NACs	context-free
locally k-testable, $k \geq 2$	context-free
first-order logic with $<_2$	context-free
first-order logic with $<_2^*$	context-free
weak monadic second-order logic (wMSO)	context-free

We seem to have reached a plateau: no matter what description language you choose, from strictly 2-local all the way up to wMSO, you just get the CFLs over and over again.

Notice that the third line in this table tells us that no matter what the size of your tree-like building blocks, if you close the set of building blocks under the

plugging-in that I earlier called frontier nonterminal substitution, you get a set of trees that has a CFL as its string yield. Thus the data-oriented parsing proposed by Remko Scha and others, and developed in Bod (1998), where in effect the grammar is simply a (statistically annotated) treebank — a set containing all of some set of trees plus all of their subtrees — can only yield CFLs.

I should make it clear that this does *not* mean that MTS is doomed to remain within the context-free realm. James Rogers (2003) realized that if you settle on wMSO as your description language, you can define a hierarchy of classes of structures of increasing complexity that has a hierarchy of classes of strings going along with it, the classes of structures being differentiated by their number of dimensions. A sentence considered as an unanalyzable unit has zero dimensions. A string has one dimension, hence only one way in which two nodes can be adjacent, the one called linear precedence, denoted by $\leq_1$. A tree has two. One is $\leq_1$. The other, denoted by $\leq_2$, allows a node to be adjacent to an entire 1-dimensional string (its children). Tree-like objects with three dimensions can be defined by adding a third relation, $\leq_3$, in terms of which a single node can be adjacent to an entire 2-dimensional tree. And so on upward. Rogers proved that the following holds:

(9) Stringset classes definable by wMSO on models fo various dimensions

NUMBER OF DIMENSIONS	TYPE OF MODELS	RESULTING STRINGSET YIELD CLASS	PROOF
0	points	finite stringsets	(obvious)
1	strings	regular stringsets	(Büchi 1960)
2	trees	context-free stringsets	(Doner 1970)
3	3-d trees	tree-adjoining stringsets	(Rogers 2003)
4	4-d trees	(no name for the class)	(Rogers 2004)
...	...	...	...

The hierarchy continues without bound — though there is currently no terminology for the stringset yields of wMSO-characterizable sets of singly-rooted tree-like models of 4 dimensions, 5 dimensions, etc. Furthermore, it has been proved by Jens Michaelis that the infinite union of all the stringset classes in the Rogers hierarchy is the one characterized by minimalist grammars as formalized by Stabler (see Stabler, Jr. 1997, Michaelis 2001). In other words, minimalist grammars in Stabler's sense are the stringsets that are the string yields of model classes wMSO-characterizable sets of singly-rooted tree-like graph models of arbitrary finite dimensionality (and thus, surely, far more expressive than will be needed for describing human languages).

6 Expressive power of HPSG

Bringing this back to the issue of HPSG is made more difficult by the curious state of the current literature. The standard works introducing the basics of HPSG contain no discussion of phrasal reduplication (as has been claimed to exist in some African languages) or the sort of cross-serial dependencies found in certain subordinate clause constructions of Dutch and Züritüütsch (Zurich Swiss German). Züritüütsch is particularly important because the varying case marking governed by different verbs yields an argument that its stringset cannot be a CFL (Shieber, 1985). Yet Swiss German does not figure in Pollard & Sag (1994), or in any of the basic pedagogical works such as Sag & Wasow (1999), Sag et al. (2003), or Levine (2017).

A number of more technical works — more than I have space to review or even list here — do cover ways of giving HPSG accounts of non-CF phenomena of the sort Swiss German exemplifies. Reape (1992) is perhaps the most influential, but see Müller (2019), Chapter 9, for pointers to the rest of the literature. A variety of different techniques are involved: re-entrancy (structure sharing) is one; relational constraints of arbitrary power are sometimes alluded to; and what is particularly important is argument merger, allowing the list-valued valence feature COMPS to gather up arguments of a subordinate constituent and then break up the list to permit checking off its members in some desired sequence. This looks as if it has the power to use the COMPS as a queue rather than a stack, which immediately provides for greater than CFG power.

I do not think there is any unitary answer to the question of what generative power results from the different uses of these mechanisms that various linguists make. Some versions of HPSG may be limited to the weak generative capacity of combinatory categorial grammar (Steedman, 2000); some probably have Turing-machine power. I confess to not having enough understanding of the voluminous literature to adjudicate on such matters; it seems to me that there is much scope for mathematical linguists to do some focused work on the weak generative capacity of HPSG in various forms, and the ways in which the various mechanisms contribute.

Such issues are important. Consider, for example, what was discovered after Richter (2000) developed a language named RSRL explicitly for stating constraints of the sort presupposed by Pollard & Sag (1994). RSRL turned out to be so expressive that even its finite model checking problem is undecidable (Kepser, 2004). In other words, full HPSG structures can be so complex, and queries expressed in RSRL can be so rich, that the task of determining what finite structures are compliant with a given RSRL constraint can lose its way as if it were being evaluated in an infinite structure, with the result that no algorithm can guarantee to determine in finite time whether a given arbitrary structure

is grammatical according to a given arbitrary RSRL-expressed grammar. This result alone tells us, of course, that RSRL on HPSG structures is vastly more complex than wMSO on trees (which guarantees decidability not just for finite model-checking but also for satisfiability).

The only point I want to contribute here has to do with unbounded dependencies. Given that description languages of significantly more expressive power than strictly local ones are available for HPSG structures and have been explored, it is a curious fact that Pollard & Sag (1994) develop their analysis of unbounded dependencies using the SLASH feature inherited from GPSG. The SLASH mechanism, we can now see (though this was not clear to the developers of GPSG in 1980), is a way of sticking to SL_2^τ descriptions, or equivalently, modifying the nonterminal vocabulary so that simple unmodified CFGs can describe unbounded dependencies.

This seems odd to me. It is as if Pollard and Sag were following Gazdar (1981) in assuming that their description had to be couched in the most primitive description language possible, namely SL_2^τ. Gazdar's breakthrough observation about unbounded dependencies was that it only takes adding a finite number of slashed categories to the inventory (upper-bounded by the square of the number of full phrasal categories, since they are the ones that can be 'extracted') to cope with unbounded dependencies and island constraints using strictly 2-local tree description (i.e., Stanley/McCawley NACs). Pollard & Sag (1994) follows Gazdar point for point on the general theory, developing different analyses where the syntactic phenomena call for it but always assuming the basic 2-local-equivalent technology that Gazdar developed.

Pollard & Sag (1994) employ the full redundancy of Gazdar's system. Take the very simple case of 'topicalization' (unbounded complement preposing) as treated in their Chapter 4. The tree in their (18) on page 165 has (when the Nonlocal Feature Principle on p. 400 is consulted to flesh it out) 'SYNSEM|NONLOCAL|INHER|SLASH $\{\,\}$' on the root of the entire sentence; 'SYNSEM|NONLOCAL|TO-BIND|SLASH $\{\boxed{1}\}$' on the root of the topicalization construction; the same thing on the trace; and 'SYNSEM|NONLOCAL|INHER|SLASH $\{\boxed{1}\}$' on each of the eight head nodes in between them.

In addition, trace nodes have to have full details of the INHER and TO-BIND values of SLASH (and QUE and REL); there is a complex specification of the internal feature structure of a trace; there is a 'Nonlocal Feature Principle', given in only the most casually informal terms on p. 164 (I quote the different version on p. 400) saying that 'For each nonlocal feature, the value of SYNSEM|NONLOCAL|INHERITED|SLASH on the mother is the set difference of the union of the values on all the daughters and the value of SYNSEM|NONLOCAL|TO-BIND|SLASH on the head daughter'; and (fn. 5, p. 164) all other ID schemata introducing phrasal heads have to be modified to include '[TO-BIND|SLASH $\{\,\}$]' on

the head daughter. All of this highly redundant feature annotation is employed simply to guarantee that there has to be a trace in the clause that accompanies a preposed complement.

If instead we do not (implicitly) restrict ourselves to SL_2^τ rules, but allow the power of (say) first-order logic in our description language for trees, we can easily guarantee the presence of a 'trace' in some subconstituent accompanying a dislocated element, *without* using GPSG-style paths of slashed categories.

For simplicity, let me assume with Huddleston et al. (2002) (henceforth *CGEL*) that preposed ('topicalized') complements are distinguished by bearing the grammatical relation 'Prenucleus' in the main clause, and every Prenucleus phrase is accompanied by a following phrasal head bearing the relation Nucleus (which is really just a special case of the head relation). We can give a simple direct statement of the fact that the head clause accompanying a Prenucleus NP must contain an NP trace. I represent grammatical relations top-down to match dominance, so '$x <_2^* y$)' means 'x dominates y' and 'Prenucleus(x, y)' means 'x has a child y bearing the Prenucleus relation to it'. Here is the statement we need:

(10) $(\forall x, y)[(\text{Prenucleus}(x, y) \wedge \text{NP}(y)) \Rightarrow$
$(\exists z)[\text{Nucleus}(x, z) \wedge (\exists t)[(z >_2^* t) \wedge \text{Trace}(t) \wedge \text{NP}(t)]]]$

'If x has an NP child y in Prenucleus function, then x also has a child z in Nucleus (= head) function and z contains an NP trace.'

This shows that we do not need SLASH to guarantee that a clause accompanying a Prenucleus ('extracted') constituent must contain a trace, even in an entirely CF-restricted descriptive system like using first-order logic on labeled trees.

Various constructions with non-subject gaps (such as the so-called '*tough*-movement' construction) can be described in a similar way, though they do not call for a Prenucleus constituent; instead they involve a complement specifically required to contain an NP gap. That is, the complement is required to be rooted at a node z such that $(\exists t)[(z >_2^* t) \wedge \text{Trace}(t) \wedge \text{NP}(t)]$ (see the discussion of 'hollow VP' complements in *CGEL*, 1245–1251, for an informal survey of the several constructions at issue).

The foregoing remarks should not be taken as an argument that we *should* describe unbounded dependencies without the now familiar GPSG-style chains of nodes bearing SLASH values. I am only pointing out that it could easily be done, using a description language that is not very rich, and has a decidable satisfiability problem.

It will take more work before we can decide whether SLASH as used in Pollard & Sag (1994) is a valuable idea in the HPSG context or or an unnecessary hold-over from GPSG. The most interesting phenomena to study in this con-

text might be the binding domain phenomena discussed by Zaenen (1983) — syntactic phenomena in various languages that are encountered only between a left-extracted constituent and the gap in subordinate structure associated with it. Zaenen posits a feature [bnd] present on every node along the spine between the two constituents, just where Gazdar's work had posited a category with a SLASH value. Could these phenomena be insightfully described in a way that involves no SLASH or BND features? We do not know, because the unacknowledged bias toward strictly local treatments of phrase structure has meant that linguists have not been asking that question during the last four decades. It might be interesting to reopen the questions raised by the data that Zaenen considered.

References

Ajdukiewicz, Kazimierz. 1935. Die syntaktische Konnexität. *Studia Philosophica* 1. 1–27. English translation published in Storrs McCall (ed.), *Polish Logic 1920–1939*, 207–231, Oxford University Press.

Bod, Rens. 1998. *Beyond grammar: An experience-based theory of language*. Stanford, CA: CSLI Publications.

Büchi, J. Richard. 1960. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 6. 66–92.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* 2(2). 137–167. Reprinted in *Readings in Mathematical Psychology*, Volume II, ed. by R. Duncan Luce, Robert R. Bush, and Eugene Galanter, 125–155, New York: John Wiley & Sons, 1965 (citation to the original on p. 125 of this reprinting is incorrect).

Chomsky, Noam. 1993. A minimalist program for linguistic theory. In Kenneth Hale & Samuel Jay Keyser (eds.), *The view from Building 20*, 1–52. Cambridge, Massachusetts: MIT Press.

Culicover, Peter W. & Ray S. Jackendoff. 2005. *Simpler syntax*. Oxford: Oxford University Press.

Doner, John. 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences* 4. 406–451.

Elgot, Calvin C. 1961. Decision problems of finite automata and related arithmetics. *Transactions of the American Mathematical Society* 98. 21–51.

Fagin, Ronald. 1974. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of computation: SIAM-AMS proceedings*, vol. 7, 43–73. American Mathematical Society.

Gazdar, Gerald. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12. 155–184.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum & Ivan A. Sag. 1985. *Generalized phrase structure grammar*. Oxford: Basil Blackwell.

Huddleston, Rodney, Geoffrey K. Pullum et al. 2002. *The Cambridge grammar of the English language*. Cambridge: Cambridge University Press.

Immerman, Neil. 1999. *Descriptive complexity*. New York: Springer.

Johnson, David E. & Paul M. Postal. 1980. *Arc pair grammar*. Princeton, NJ: Princeton University Press.

Kac, Michael B. 1978. *Corepresentation of grammatical structure*. London: Croom Helm.

Kepser, Stephan. 2004. On the complexity of RSRL. *Electronic Notes in Theoretical Computer Science (ENTCS)* 53. 146–162. In Proceedings of the Joint Meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language; online at http://dx.doi.org/10.1016/S1571-0661(05)82580-0.

Lakoff, George. 1971. On generative semantics. In Danny D. Steinberg & Leon A. Jakobovitz (eds.), *Semantics: An interdisciplinary reader in philosophy, linguistics and psychology*, 232–296. Cambridge: Cambridge University Press.

Levine, Robert D. 2017. *Syntactic analysis: An HPSG-based approach*. Cambridge: Cambridge University Press.

Libkin, Leonid. 2004. *Elements of finite model theory* Texts in Theoretical Computer Science. Springer.

McCawley, James D. 1968. Concerning the base component of a transformational grammar. *Foundations of Language* 4. 243–269. Reprinted in James D. McCawley, *Grammar and Meaning*, 35–58 (New York: Academic Press; Tokyo: Taishukan, 1973).

McNaughton, Robert & Seymour Papert. 1971. *Counter-free automata*. Cambridge, MA: MIT Press.

Medvedev, Yu. T. 1956 [1964]. On the class of events representable in a finite automaton. In Edward F. Moore (ed.), *Sequential machines: Selected papers*, vol. II, 215–227. Reading, MA: Addison-Wesley. Originally published in Russian in *Avtomaty* (1956), 385–401.

Michaelis, Jens. 2001. Transforming linear context-free rewriting systems into minimalist grammars. In Philippe de Groote, Glyn Morrill & Christian Retoré (eds.), *Logical Aspects of Computational Linguistics: 4th international conference* (Lecture Notes in Artificial Intelligence 2099), 228–244. Berlin and New York: Springer.

Müller, Stefan. 2019. *Grammatical theory: From transformational grammar to constraint-based approaches*. Berlin: Language Science Press 3rd edn.

Peters, P. Stanley & Robert W. Ritchie. 1969. Context-sensitive immediate constituent analysis — context-free languages revisited. In *Proceedings of the ACM conference on the theory of computing*, 1–8. Republished in *Mathematical Systems Theory* 6 (1973), 324-333.

Pollard, Carl J. 1996. The nature of constraint-based grammar. Presented at Pacific Asia Conference on Language, Information, and Computation, Kyung Hee University, Seoul, Korea, December 20. Plain text draft available online at http://lingo.stanford.edu/sag/L221a/pollard-96.txt.

Pollard, Carl J. & Ivan A. Sag. 1987. *Information-based syntax and semantics, volume 1: Fundamentals*. Stanford, CA: CSLI Publications.

Pollard, Carl J. & Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. Stanford, CA: CSLI Publications.

Reape, Mike. 1992. *A formal theory of word order: A case study in West Germanic*. Edinburgh, UK: University of Edinburgh dissertation.

Richter, Frank. 2000. *A mathematical formalism for linguistic theories with an application in Head-driven Phrase Structure Grammar*. Tübingen, Germany: Universität Tübingen dissertation.

Rogers, James. 1997. Strict LT_2 : Regular :: Local : Recognizable. In Christian Retoré (ed.), *Logical Aspects of Computational Linguistics: First international conference, LACL '96 (selected papers)* (Lecture Notes in Artificial Intelligence 1328), 366–385. Berlin and New York: Springer.

Rogers, James. 1999. The descriptive complexity of generalized local sets. In Hans-Peter Kolb & Uwe Mönnich (eds.), *The mathematics of syntactic structure: Trees and their logics* (Studies in Generative Grammar 44), 21–40. Berlin: Mouton de Gruyter.

Rogers, James. 2003. wMSO theories as grammar formalisms. *Theoretical Computer Science* 293. 291–320.

Rogers, James. 2004. Wrapping of trees. In Donia Scott (ed.), *Proceedings of the 42nd annual meeting of the association for computational linguistics*, Morristown, NJ: Association for Computational Linguistics. Article no. 558, doi 10.3115/1218955.1219026; online at http://portal.acm.org/citation.cfm?id=1219026#.

Rogers, James & Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20. 329–342.

Sag, Ivan A. & Thomas Wasow. 1999. *Syntactic theory: A formal introduction*. Stanford, CA: CSLI Publications 1st edn.

Sag, Ivan A., Thomas Wasow & Emily M. Bender. 2003. *Syntactic theory: A formal introduction*. Stanford, CA: CSLI Publications 2nd edn.

Shieber, Stuart. 1985. Evidence against the context-freeness of human language. *Linguistics and Philosophy* 8. 333–343.

Soames, Scott. 1974. Rule orderings, obligatory transformations, and derivational constraints. *Theoretical Linguistics* 1. 116–138.

Stabler, Jr., Edward P. 1997. Derivational minimalism. In Christian Retoré (ed.), *Logical Aspects of Computational Linguistics, LACL '96* (Lecture Notes in Artificial Intelligence 1328), 68–95. Berlin: Springer Verlag.

Steedman, Mark. 2000. *The syntactic process*. Cambridge, MA: MIT Press.

Thatcher, James W. 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences* 1. 317–322.

Thatcher, James W. & J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory* 2(1). 57–81.

Thomas, Wolfgang. 1982. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences* 25. 360–376.

Trakhtenbrot, Boris A. 1962. Finite automata and monadic second order logic. *Sibirskii Matematicheskii Zhurnal* 3. 101–131. In Russian; English translation in AMS Translations 59:23–55, 1966.

Zaenen, Annie. 1983. On syntactic binding. *Linguistic Inquiry* 14. 469–504.