A revised version is to appear in: M. Kanazawa, A. Kornai, M. Kracht and H. Seki (eds.), The Mathematics of Language (MOL 12), LNCS/LNAI Vol. 6878, pp. 126-142, Springer, Berlin, Heidelberg.

Disentangling Notions of Specifier Impenetrability: Late Adjunction, Islands, and Expressive Power

Gregory M. Kobele[1] and Jens Michaelis[2]

[1] University of Chicago, Chicago, Illinois, USA
[2] Bielefeld University, Bielefeld, Germany

Abstract. In this paper we investigate the weak generative capacity of *minimalist grammars with late adjunction.* We show that by viewing the **Sp**ecifier **I**sland **C**ondition as the union of three separate constraints, we obtain a more nuanced perspective on previous results on constraint interaction in minimalist grammars, as well as the beginning of a map of the interaction between late adjunction and movement constraints. Our main result is that minimalist grammars with the **SpIC** on movement generated specifiers only and with the **S**hortest **M**ove **C**onstraint, in conjunction with late adjunction, can define languages whose intersection with an appropriate regular language is not semilinear.

1 Introduction

In deviance from the type introduced by Stabler in [26], in [27] a revised type of *minimalist grammar* was defined, which incorporated a general ban on movement out from within a 'specifier' position. The effects of this restriction on movement, called the **Sp**ecifier **I**sland **C**ondition, have been studied in conjunction with other constraints on movement [3, 4, 14, 20, 21]. These studies have all treated the **SpIC** as a single constraint, barring extraction from certain geometrical configurations. In so doing, they have failed to observe the fact that different kinds of specifiers—base generated, movement generated, or adjoined—play different roles in the context of different additional constraints on movement. For example, the proof in [14], that minimalist grammars with no constraints but the **SpIC** are turing complete makes use only of specifiers generated by movement, and thus, a **SpIC**-variant applying only to movement generated specifiers were sufficient in this connections. On the other hand, a closer look at the proof in [21], that minimalist grammars with the **SpIC** and the **S**hortest **M**ovement **C**onstraint are strictly weaker than those with only the **SMC**, reveals that the essential restriction here is the **SpIC** applied to base generated specifiers.

Disentangling these notions of specifier impenetrability further, in Sec. 4 we investigate the formal properties of *minimalist grammars with late adjunction* as introduced in [5]. Our main result is that the interaction of all three the **SMC**, the **SpIC** only applied to moved specifiers and the operation of late adjunction allows the generation of a language L not derivable by any *multiple context-free grammar* in the sense of [25], where the intersection of L with an appropriate regular language is not semilinear.

2 Formal Preliminaries

Given a set A, 2^A is its power set. We will think of relations over A and B as functions $f : A \to 2^B$, and write $f(a) \to b$ in case $b \in f(a)$. The set of numbers $\{0, 1, 2, \ldots\}$ is denoted $\mathbb{N}$. For $n \in \mathbb{N}$, the set of numbers from 1 to n is denoted $[n]$, and so we have $[0] = \emptyset$. For non-empty, finite $N \subseteq \mathbb{N}$, $\mathsf{max}(N)$ denotes the greatest element of N. Given a finite set A, a sequence over A of length n is a function $f : [n] \to A$. Given a sequence $f : [n] \to A$, with $f(i) = a_i$ for $i \in [n]$, we write $a_1 \cdots a_n$, $(a_i)_{i\in[n]}$, or even $\boldsymbol{a}$ for f. The symbol ϵ denotes the empty sequence, i.e. the sequence of length 0. A^* denotes the set of all finite sequences of elements over A. Given $a \in A$, and $w \in A^*$, $|w|_a$ denotes the number of occurrences of the symbol a in w, in symbols $|w|_a := \left|w^{-1}(a)\right|$. A ranked alphabet is a finite set F together with a function $\mathbf{rank} : F \to \mathbb{N}$ mapping each symbol in F to a natural number indicating its arity. Given $f \in F$ with arity $n = \mathbf{rank}(f)$, we will sometimes write $f^{(n)}$ to denote f while indicating that it has arity n. The set T_F of terms over a ranked alphabet F is the smallest subset of F^* containing each $f^{(0)} \in F$, and such that whenever it contains $t_1, \ldots, t_n$, it contains $f^{(n)}t_1 \cdots t_n$ for each $f^{(n)} \in F$. We will insert parentheses and commas for readability, writing $f(t_1, \ldots, t_n)$ instead of $ft_1 \cdots t_n$.

Let $X = \{x_1, x_2, \ldots\}$ be an enumerable set of variables. Then $X_n := \{x_i : i \in [n]\}$ is the set of the first n elements of X. Treating X as a ranked alphabet with only nullary symbols, $T_F(X) := T_{F\cup X}$ is the set of contexts over F. For $C \in T_F(X_n)$, and $t_i \in T_F(X)$ for $i \in [n]$, $C[t_1, \ldots, t_n]$ denotes the result of simultaneously substituting each x_i in C with t_i, $i \in [n]$. To save space, we will sometimes write $C[\boldsymbol{t}]$ instead of $C[t_1, \ldots, t_n]$ when the intended assignment is clear.

$$x_j[\boldsymbol{t}] := t_j \qquad f^{(0)}[\boldsymbol{t}] := f$$

$$f^{(k)}(s_1, \ldots, s_k)[\boldsymbol{t}] := f(s_1[\boldsymbol{t}], \ldots, s_k[\boldsymbol{t}])$$

A context $C \in T_F(X)$ is *linear* just in case each $x \in X$ occurs in C at most once, i.e., $\mathsf{max}(\{|C|_x : x \in X\}) = 1$. Given a term $t \in T_F(X)$, an occurrence of $t' \in T_F(X)$ in t is a linear context $C \in T_{F\cup X}(\{x_0\})$, where $x_0 \notin X$ is a new variable, such that $C[t'] = t$. The address of a node in a term $t \in T_F(X)$ is a string over $[n]^*$, where $n = \mathsf{max}(\{\mathbf{rank}(f) : f \in F\})$. The set of addresses in $t \in T_F(X)$ is defined inductively as $\mathsf{addr}(f^{(0)}) = \{\epsilon\}$, and $\mathsf{addr}(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i\in[n]}\{iu : u \in \mathsf{addr}(t_i)\}$. Given a term $t = f(t_1, \ldots, t_n) \in T_F(X)$, and an address $iw \in \mathsf{addr}(t)$, the occurrence at iw in t is the occurrence at w in t_i, and t is the occurrence at ϵ in itself.

2.1 Minimalist Grammars

A *minimalist grammar (MG)* is given by a 5-tuple $G = \langle \Sigma, \mathbf{sel}, \mathbf{lic}, \mathit{Lex}, \mathsf{s}\rangle$ where Σ is a finite set, $\mathbf{sel}$ and $\mathbf{lic}$ are finite sets of *selection* and *licensing* features, respectively, which in their turn determine a set $\mathbb{F} := \{\mathsf{=x}, \mathsf{\approx x}, \mathsf{x} : x \in \mathbf{sel}\} \cup \{\mathsf{+y}, \mathsf{-y} : y \in \mathbf{lic}\}$ of *features*, $\mathit{Lex} \subseteq \Sigma \times \mathbb{F}^+$ is a finite set of lexical items, and $\mathsf{s} \in \mathbb{F}$ is the *start symbol*. Features of the form $\mathsf{=x}$ are *selector* features, those of the form $\approx\mathsf{x}$ are *adjunction* features, those of the form $\mathsf{+y}$ are *licensor* features, while those of the form x are *selectee* features, and those of the form $\mathsf{-y}$ are *licensee* features. Treating elements of Σ as nullary symbols, we define a ranked alphabet $S := \Sigma \cup \{\mathsf{t}^{(0)}, \mathsf{<}^{(2)}, \circ^{(2)}, \bullet^{(2)}, \mathsf{>}^{(2)}\}$. We are interested rather in the ranked alphabet $O := S \times \mathbb{F}^*$ (here O stands for *output*) such that $\mathbf{rank}(\langle s, \delta\rangle) = \mathbf{rank}(s)$ for $s \in S$ and $\delta \in \mathbb{F}^*$, and we usually write s^δ instead of $\langle s, \delta\rangle$.[3]

Given an element $t \in T_O$, we write t^δ only if the root of t is labeled with a symbol s^δ. In that case, t^γ stands for the result of replacing the label at the root of t with the label s^γ. We write t^ϵ as t.

The set of expressions $E(G)$ of an MG G is the smallest subset of T_O containing *Lex* and closed under the operations presented below.[4,5]

$$merge(t_1^{\mathsf{=x}\delta}, t_2^{\mathsf{x}\gamma}) = \begin{cases} \mathsf{<}^\delta(t_1, t_2^{\mathsf{x}\gamma}), & \text{if } t_1 \in \Sigma \times \mathbb{F}^+ \\ \\ \mathsf{>}^\delta(t_2^{\mathsf{x}\gamma}, t_1), & \text{otherwise} \end{cases}$$

$$move(C[t^{\gamma\mathsf{-y}}]^{\mathsf{+y}\delta}) \to \circ^\delta(t^\gamma, C[\mathsf{t}])$$

$$adjoin(t_1^{\approx\mathsf{x}\delta}, t_2^{\mathsf{x}\gamma}) = \bullet^{\mathsf{x}\gamma}(t_1^\delta, t_2)$$

An element $t^{\mathsf{c}} \in E(G)$ is a *complete expression of category* c iff every node in it is of the form s^ϵ or s^{f_s} for some selectee feature f_s. The derived (or *surface*) tree language of G at selectee feature c is defined to be the set of complete expressions of category c, $L_{\mathsf{c}}(G) := \{t^{\mathsf{c}} \in E(G) : t^{\mathsf{c}} \textit{ is complete}\}$. We write $Str(G) := \{\mathsf{yield}(t^{\mathsf{s}}) : t^{\mathsf{s}} \in L_{\mathsf{s}}(G)\}$ to denote the string language of G at the start category s.[6]

2.2 Late Adjunction

Following [5], we generalize the adjunction operation by relaxing the requirement that the adjunct has to adjoin at the root of the tree it adjoins to. Instead, an adjunct may adjoin *late*, in that the expression to which it adjoins is a proper subtree of its coargument. This extension renders the adjunction operation relational.

[3] A note is in order: while O as defined is infinite, we will not be interested in function symbols paired with feature strings longer than a fixed finite length k, where k is the maximal number of feature instances had by some lexical item.

[4] In contrast to other presentations of MGs, here licensee features are checked from *right to left*, and selectee features are not deleted by *merge*.

[5] The operator *adjoin* was introduced in [2] among the list of MG-operations.

[6] For $t^\delta \in E(G)$, $\mathsf{yield}(t^\delta)$ is inductively defined by $\mathsf{yield}(t^\delta) = t$ if $t \in \Sigma$, $\mathsf{yield}(t^\delta) = \epsilon$ if $t = \mathsf{t}$, and $\mathsf{yield}(t^\delta) = \mathsf{yield}(t_1^{\delta_1}) \cdot \mathsf{yield}(t_2^{\delta_2})$ if $t^\delta = \langle s, \delta\rangle(t_1^{\delta_1}, t_2^{\delta_2})$ for some $s^{(2)} \in S$, and $t_1^{\delta_1}, t_2^{\delta_2} \in E(G)$.

$$\mathit{lateAdjoin}(t_1^{\approx \mathsf{x}\delta}, C[t_2^{\mathsf{x}\gamma}]) \to C[\bullet^{\mathsf{x}\gamma}(t_1^{\delta}, t_2)]$$

The expressions derivable by a grammar G *with* late adjunction build the set $E^{+\mathbf{LA}}(G)$, and accordingly $L_{\mathsf{c}}^{+\mathbf{LA}}(G)$ denotes the set of complete expressions of category c from $E^{+\mathbf{LA}}(G)$, and $\mathit{Str}^{+\mathbf{LA}}(G)$ denotes the set of yields of complete expressions of start category s from $E^{+\mathbf{LA}}(G)$.

2.3 Conditions on Rules

Salvati [24] shows that minimalist grammars as defined above have a membership problem which is as hard as the reachability problem for vector addition tree automata in the sense of [7]. This problem is equivalent to provability in multiplicative exponential linear logic [6], the decidability of which is currently open, and which has a lower bound of ExpSpace, cf. [16].

Shortest Move The canonical condition on the operations above is the **S**hortest **M**ove **C**onstraint. The **SMC** is a restriction on the domain of the *move* operation, requiring that, for $\mathit{move}(t^{+\mathsf{y}\gamma})$ to be defined, there be exactly one node s^{δ} in t where the last feature of δ is $-\mathsf{y}$. This restriction on the domain of *move* makes it a function. We write $E^{+\mathbf{SMC}}(G)$ to denote the expressions of an MG, G, derivable using the **SMC**-restricted *move* operation, instead of the general *move* operation presented above. $L_{\mathsf{c}}^{+\mathbf{SMC}}(G)$ denotes the subset of $E^{+\mathbf{SMC}}(G)$ consisting of all complete expressions of category c, and $\mathit{Str}^{+\mathbf{SMC}}(G)$ is the set $\{\mathsf{yield}(t^{\mathsf{s}}) : t^{\mathsf{s}} \in L_{\mathsf{s}}^{+\mathbf{SMC}}(G)\}$. Michaelis [17, 19] and Harkema [9] prove $\mathcal{ML}^{+\mathbf{SMC}} := \{\mathit{Str}^{+\mathbf{SMC}}(G) : G \textit{ is an MG}\}$, the class of minimalist string languages, to be identical to $\mathcal{MCFL}$, the class of languages derivable by *multiple context-free grammars (MCFGs)* in the sense of [25].

Specifier Impenetrability Stabler [27] restricts movement further, requiring that the address of the moving subtree be either from $\mathbf{2}^*$ (so that the path from the root to the moving subtree has only right branches) or from $\mathbf{2}^*\mathbf{1}$ (so that the path from the root to the moving subtree has exactly one left branch and this at the end). This restriction is called the **Sp**ecifier **I**sland **C**ondition in [3, 4].

By $\mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}}$ we denote the class of string languages of MGs with both the **SMC** and the **SpIC** imposed as constraints on the *move*-operation. Michaelis [18, 20] shows that $\mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}}$ consists of exactly those languages generated by a subtype of MCFGs, called *monadic branching MCFGs* in [11]. These languages are shown to be properly included within the class of languages generated by MCFGs in [21], and even within the class of languages generated by *well-nested MCFGs* in [11].

Without the **SMC**, MGs with the **SpIC** generate all recursively enumerable languages [14].

Adjunct Islandhood Finally, in an investigation of the operation of late adjunction, Gärtner and Michaelis [4, p. 187] explicitly formulate the **A**djunct **I**sland **C**ondition, requiring that the first argument $t_1^{\approx \mathsf{x}\delta}$ of either *adjoin* or *lateAdjoin* has no nodes the second component of the label of which contains a licensee feature (with the possible exception of the root).[7] The **AIC** acting in conjunction with the **SMC** ensures that $\mathcal{ML}^{\mathbf{+SMC,+AIC,+LA}} = \mathcal{ML}^{\mathbf{+SMC}}$.

3 A Closer Look at the SpIC

The **SpIC** forbids movement from targeting subtrees in certain geometric positions, namely, those subtrees whose roots have an address which is not in the set $\mathbf{2}^*(\mathbf{1}+\boldsymbol{\epsilon})$. These forbidden positions are occurrences of the form $D[\otimes(C[x_0],t)]$, where D and C are linear contexts, C is non-trivial (i.e. $C \neq x_0$), and $\otimes \in \{\mathsf{>}, \circ, \bullet\}$. In the context $D[\otimes(t_1,t)]$ (where again D is linear, and $\otimes \in \{\mathsf{>}, \circ, \bullet\}$), we say that t_1 occurs in a specifier position (whence the name of the constraint). As this presentation of the **SpIC** makes salient, we can restrict it to particular instantiations of the connective $\otimes$ above, as per whether they originate from *merge*, *move*, or *lateAdjoin*. We define accordingly $\mathbf{SpIC}_{\boldsymbol{mrg}}$, $\mathbf{SpIC}_{\boldsymbol{mv}}$, and $\mathbf{SpIC}_{\boldsymbol{adj}}$ in the following manner:

$\mathbf{SpIC}_{\boldsymbol{mrg/mv/adj}}$
move$(C[t^{\gamma\mathsf{-y}}]^{\mathsf{+y}\delta}) \to \circ^{\delta}(t^{\gamma}, C[\mathtt{t}])$ only if there are no linear contexts D, E with E non-trivial and term t' such that $C[x_0] = D[\otimes(E[x_0],t')]$, where $\otimes = \mathsf{>} / \circ / \bullet$.

In prose, the $\mathbf{SpIC}_{\boldsymbol{mrg/mv/adj}}$ says that movement cannot take place out from inside a specifier generated by a(n) *merge* / *move* / *adjoin* operation. The restriction that E be non-trivial allows for movement of a specifier (of a particular sort), as long as it is not properly contained within another (of the same type).

The rationale for splitting the monolithic **SpIC** into three independent conditions comes from the observations

1. that the $\mathbf{SpIC}_{\boldsymbol{adj}}$ is an equivalent re-implementation of the **AIC**,
2. that in the context of the **SMC**, the $\mathbf{SpIC}_{\boldsymbol{mv}}$ has no effect, i.e.
$$\mathcal{ML}^{\mathbf{+SMC,+SpIC}_{\boldsymbol{mv}}} = \mathcal{ML}^{\mathbf{+SMC}} \text{ , and}$$
3. that without the **SMC** only the $\mathbf{SpIC}_{\boldsymbol{mv}}$ plays a role in the proof of the Turing completeness in [14], i.e.
$$\mathcal{ML}^{\mathbf{+SpIC}_{\boldsymbol{mv}}} = \mathcal{ML}^{\mathbf{+SpIC}} = \mathcal{R}.\mathcal{E}.$$

We will discuss each of these observations in turn, and will then turn our attention to the main novel contribution of this paper, which is the investigation of the interaction of late adjunction and various forms of **SpIC** in Sec. 4.

[7] Doing so, Gärtner and Michaelis generalize the **AIC**-definition proposed in [2] for the purposes of controlling the effects of just the 'simple' *adjoin*-operation.

3.1 The relation between the $\mathbf{SpIC}_{adj}$ and the AIC

The $\mathbf{SpIC}_{adj}$ states that it is not possible to extract from an adjoined specifier one of its proper subtrees even if the label of the root of such a subtree provides a licensee feature which would allow movement of the subtree otherwise. The focus of the **AIC** as formally defined in [4] is on *a priori* avoiding the creation of expressions showing such a configuration, while under the $\mathbf{SpIC}_{adj}$ these expressions can 'simply' not take part in a convergent derivation. In other words, we have $E^{+\mathbf{AIC}}(G) \subseteq E^{+\mathbf{SpIC}_{adj}}(G)$ for each MG $G = \langle \Sigma, \mathbf{sel}, \mathbf{lic}, Lex, \mathsf{s} \rangle$, but, in particular, we have $L_{\mathsf{s}}^{+\mathbf{AIC}}(G) = L_{\mathsf{s}}^{+\mathbf{SpIC}_{adj}}(G)$.

3.2 The $\mathbf{SpIC}_{mv}$ and the SMC

In the presence of the **SMC**, the $\mathbf{SpIC}_{mv}$ has no effect as to the class of derivable string languages, more formally, $\mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}_{mv}} = \mathcal{ML}^{+\mathbf{SMC}}$ and $\mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}} = \mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}_{mrg}}$ hold.

The first identity follows from the fact that for each MG G with the **SMC**, but without the **SpIC**, we can define a weakly equivalent MG G' such that (i) and (ii) hold:

(i) For each $v \in E^{+\mathbf{SMC}}(G')$ and $x \in \mathbf{lic}$, if there is some subtree $t^{\gamma\text{-}\mathsf{x}}$ then the address of $t^{\gamma\text{-}\mathsf{x}}$ is from $\mathbf{2}^*$ or $\mathbf{2}^*\mathbf{12}^*$.

(ii) Moreover, if for some $v \in E^{+\mathbf{SMC}}(G')$ and $x \in \mathbf{lic}$, there is a subtree $t^{\gamma\text{-}\mathsf{x}}$ such that for some $m, n \in \mathbb{N}$, the address of $t^{\gamma\text{-}\mathsf{x}}$ is 2^m12^n then the label of the node with address 2^m is of the form $>^{\delta}$ for some $\delta \in \mathbb{F}^*$.[8]

The line of argument here is the following: for each MG with the **SMC** there is a non-deleting MCFG of rank 2 deriving the same string language [18]. For each non-deleting MCFG of rank 2 there is a non-deleting and non-permuting MCFG of rank 2 deriving the same string language [18, Corollary 2.4.4(a)]. Starting from a non-deleting MCFG of rank 2, [19] presents a constructive method how to convert this MCFG into an MG with the **SMC** deriving the same string language. The fulfillment of (i) and (ii) by the resulting MG is an additional consequence, when starting from a non-deleting and non-permuting MCFG of rank 2.[9] From (i) and (ii) it follows that $E^{+\mathbf{SMC}}(G') = E^{+\mathbf{SMC},+\mathbf{SpIC}_{mv}}(G')$ holds.

The second identity, $\mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}} = \mathcal{ML}^{+\mathbf{SMC},+\mathbf{SpIC}_{mrg}}$, follows from the fact that for each MG G with the **SMC** and with the **SpIC**, we can define a weakly equivalent MG G'' such that (iii) holds:

(iii) Whenever, for some $u \in E^{+\mathbf{SMC},+\mathbf{SpIC}}(G'')$ and $x \in \mathbf{lic}$, there is some subtree $t^{\gamma\text{-}\mathsf{x}}$ of u then the address of the root of $t^{\gamma\text{-}\mathsf{x}}$ is from $\mathbf{2}^*$.

[8] That is, the subtree of v with address 2^m1, the specifier which $t^{\gamma\text{-}\mathsf{x}}$ is a subtree of, has been built by an application of *merge*.

[9] For concrete definitions of a non-permuting MCFG and a non-deleting MCFG see, e.g., [10]. A non-permuting MCFG is an *MCFG in monotone function form* in the sense of [18]; and a non-deleting and non-permuting MCFG is an *ordered simple RCG* in the sense of [29, 30] as well as a *monotone LCFRS* in the sense of [15].

For each MG with both the **SMC** and the **SpIC** there is a non-deleting monadic branching MCFG deriving the same string language [18, Corollary 4.1.14]. For each non-deleting monadic branching MCFG there is a non-deleting and non-permuting monadic branching MCFG deriving the same string language [18, Corollary 2.4.4(b)]. Each non-deleting and non-permuting monadic branching MCFG can be transformed into an MG with both the **SMC** and the **SpIC** fulfilling (iii), and deriving the same string language [20]. (iii), in fact, implies that $E^{\textbf{+SMC},\textbf{+SpIC}}(G') = E^{\textbf{+SMC},\textbf{+SpIC}_{mrg}}(G')$.

3.3 The SpIC without the SMC

Gärtner and Michaelis [3] show that MGs without the **SMC** but with the **SpIC** are able to derive languages which do not have the constant growth property. Kobele and Michaelis [14] show that, in fact, every language of type 0 can be derived by some MG without the **SMC** but with the **SpIC** for essentially two reasons: a) because of the **SpIC**, movement of a tree into a specifier position freezes every proper subtree within the moved tree, and b) without the **SMC**, therefore, the nodes of the rightmost branch of a tree, i.e., the nodes with addresses from $\mathbf{2}^*$, can technically be employed as a queue.

4 Late Adjunction and Specifier Impenetrability

It is easy to see that, as long as the adjunct island condition is not in effect, late adjunction allows the description of the commutation closure of any regular language.[10] We begin by presenting a construction associating each regular language with an MG generating it.

Let $L \subseteq \Sigma^*$ be regular, and let $P_L : \Sigma^* \to 2^{\Sigma^*}$ be the map associating with each $w \in \Sigma^*$ the set of strings which can be prefixed to w to obtain a string in L, i.e. $P_L(u) := \{v \in \Sigma^* : vu \in L\}$ for $u \in \Sigma^*$. We define $G_L = \langle \Sigma, \mathbf{sel}_L, \emptyset, Lex_L, \mathsf{s}\rangle$, the MG allowing the generation of L, by giving the lexicon, Lex_L, as the union of sets $\textsc{Trans}_L$, $\textsc{Finals}_L$, and $\textsc{Start}_L$.[11] The set $\mathbf{sel}_L$ is the disjoint union of the range of P_L and the singleton set containing the new symbol $\boldsymbol{s}$, i.e., $\mathbf{sel}_L := \{q : \exists w \in \Sigma^*.\ P_L(w) = q\} \cup \{\boldsymbol{s}\}$. By the Myhill-Nerode theorem, $\mathbf{sel}_L$ is finite.

$$\begin{aligned}
\textsc{Start}_L &:= \{\, \langle \epsilon, \mathsf{q}\rangle : q = P_L(\epsilon)\}\\
\textsc{Finals}_L &:= \{\, \langle \epsilon, \texttt{=r s}\rangle : r \in \mathbf{sel}_L - \{\boldsymbol{s}\} \ \&\ \epsilon \in r \,\}\\
\textsc{Trans}_L &:= \{\, \langle a, \texttt{=r q}\rangle : a \in \Sigma \ \&\ r \in \mathbf{sel}_L - \{\boldsymbol{s}\} \ \&\ \exists w \in r.\ q = P_L(aw) \,\}
\end{aligned}$$

[10] Given a string $s : [n] \to \Sigma$, the set $\textsc{c}(s) := \{s \circ \pi : \pi \text{ a bijection over } [n]\}$ is the commutation closure of s. Given a language $L \subseteq \Sigma^*$, $\textsc{c}(L) := \{w \in \textsc{c}(s) : s \in L\}$ is the commutation closure of L. As every language in $\mathcal{SL}$, the class of semilinear languages, is letter equivalent to some language in $\mathcal{REG}$, the class of regular languages, $\textsc{c}(\mathcal{L}) = \textsc{c}(\mathcal{REG})$ for any $\mathcal{REG} \subseteq \mathcal{L} \subseteq \mathcal{SL}$.

[11] G_L is the MG-representation of a canonical finite state automaton recognizing L.

We now take x to denote a new symbol, and in order to define the MG $\mathrm{c}G_L = \langle \Sigma, \mathbf{sel}_L \cup \{x\} \cup \Sigma, \Sigma, \mathrm{c}Lex_L, \mathsf{s}\rangle$ allowing the generation of $\mathrm{c}(L)$, the commutation closure of L, we modify Lex_L in two ways. First, we define $\mathrm{cT{\small RANS}}_L$ to contain a lexical item $\langle \epsilon, \texttt{=r +a q}\rangle$ iff $\mathrm{T{\small RANS}}_L$ contains $\langle a, \texttt{=r q}\rangle$. Next, we define $\mathrm{cS{\small TART}}_L$ to contain the lexical item $\langle \epsilon, \texttt{=x q}\rangle$, instead of $\langle \epsilon, \mathsf{q}\rangle$. We define $\mathrm{c}Lex_L := \mathrm{F{\small INALS}}_L \cup \mathrm{cS{\small TART}}_L \cup \mathrm{cT{\small RANS}}_L \cup \mathrm{A{\small DJOIN}}_\Sigma$, where the first three sets are as defined above, and $\mathrm{A{\small DJOIN}}_\Sigma$ is defined as the disjoint union of the two sets $\mathrm{{\small ADJ}}_\Sigma$ and $\mathrm{{\small BASE}}$:

$$\mathrm{{\small ADJ}}_\Sigma := \{\langle a, \texttt{=a} \approx\!\mathsf{x}\rangle, \langle \epsilon, \texttt{a -a}\rangle : a \in \Sigma\}$$
$$\mathrm{{\small BASE}} := \{\langle \epsilon, \mathsf{x}\rangle, \langle \epsilon, \texttt{=x x}\rangle\}$$

Theorem 1. *For $L \in \mathcal{REG}$ let $\mathrm{c}G_L$ be the MG as constructed above depending on L. Then we have $\mathrm{c}(L) = Str^{\textbf{+LA},\alpha}(\mathrm{c}G_L)$ for α not containing **+SpIC**$_{adj}$.*

Proof. Repeated merger of lexical items in $\mathrm{{\small BASE}}$ generates the expressions $b_n = <^{\mathsf{x}}(\epsilon, <^{\mathsf{x}}(\epsilon, \cdots <^{\mathsf{x}}(\epsilon, \epsilon^{\mathsf{x}}) \cdots))$, where $|b_n|_\epsilon = n$. This expression is the 'sandbox' wherein late merger of the expressions $\widehat{a} = merge(\langle a, \texttt{=a} \approx\!\mathsf{x}\rangle, \langle \epsilon, \texttt{a -a}\rangle) = <^{\approx \mathsf{x}}(a, \epsilon^{\mathsf{a}\ \mathsf{-a}})$ for $a \in \Sigma$ will occur. Given such a b_n, in order to obtain a complete expression of category s, we must merge it with the lexical item $\langle \epsilon, \texttt{=x q}\rangle \in \mathrm{cS{\small TART}}_L$, which results in the expression $s_n = <^{\mathsf{q}}(\epsilon, b_n)$ of category q.

If $L \neq \emptyset$, then there is a sequence of lexical items $t_1, \ldots, t_k \in \mathrm{cT{\small RANS}}_L$ which describe an accepting path through the canonical automaton of L, where $t_i = \langle \epsilon, \texttt{=q}_{i-1}\ \texttt{+a}_i\ \mathsf{q}_i\rangle$, where $q_0 = P_L(\epsilon)$, and $\epsilon \in q_k$. Letting $t_1, \ldots, t_k$ be such a path, we describe a set of convergent derivations of $\mathrm{c}G_L$ which generates $\mathrm{c}(a_k \cdots a_1) \subseteq \mathrm{c}(L)$. We define $d_0 = \{s_k\}$, and $d_i = \{move(merge(t_i, lateAdjoin(\widehat{a}_i, d))) : d \in d_{i-1}\}$ for $1 \leq i \leq k$. Note that for $0 \leq i \leq k$, every $d \in d_i$ is a complete expression of category q_i, and that $\{\mathsf{yield}(d) : d \in d_i\} = \mathrm{c}(a_i \cdots a_1)$. (This is due to the fact that there are k positions which late adjunction can target in b_k.)

Now let $d \in d_k$, then there is a final lexical item $\langle \epsilon, \texttt{=q}_k\ \mathsf{s}\rangle \in \mathrm{F{\small INALS}}_L$ which can be merged with d to form a complete expression of category s. As the derivation of d was consistent with the **SMC**, the **SpIC**$_{mv}$, and the **SpIC**$_{mrg}$, and as $\mathsf{yield}(merge(\langle \epsilon, \texttt{=q}_k\ \mathsf{s}\rangle, d)) = \mathsf{yield}(d)$, $\mathrm{c}(a_k \cdots a_1) \subseteq Str^{\textbf{+LA},\alpha}(\mathrm{c}G_L)$, for α not containing **+SpIC**$_{adj}$.

To see that every $w \in Str^{\textbf{+LA},\alpha}(\mathrm{c}G_L)$ is in $\mathrm{c}(L)$, note that only (late) adjunctions introduce overt material, and that these must be 'licensed' by some transition. $\square$

As an immediate corollary of the above theorem we note

Corollary 1. $\mathrm{c}(\mathcal{REG}) \subseteq \mathcal{ML}^{\textbf{+LA},\alpha}$ *for α not containing **+SpIC**$_{adj}$.*

4.1 Controlling Late Adjunction by the SpIC$_{mv}$

In this section we consider MGs with late adjunction, the **SMC**, and the **SpIC**$_{mv}$. While adding the **SpIC**$_{mv}$ to MGs with the **SMC** *without* late adjunction does not change the generable string languages (cf. 3.2), in the presence of late adjunction the $\mathbf{SpIC}_{mv}$ provides a way to control when and where late adjunction may occur. This can be used to derive a language $L \in \mathcal{ML}^{\mathbf{+LA},\mathbf{+SMC},\mathbf{+SpIC}_{mv}}$ such that $L \cap \boldsymbol{e}^*\boldsymbol{d}(\boldsymbol{b}(\boldsymbol{ac})^*)^*$ is a non-semilinear language.[12] This proves that $\mathcal{ML}^{\mathbf{+LA},\mathbf{+SMC},\mathbf{+SpIC}_{mv}}$ properly contains $\mathcal{ML}^{\mathbf{+SMC},\mathbf{+SpIC}_{mv}} = \mathcal{MCFL}$. It is not known whether $\mathcal{ML}^{\mathbf{+LA},\mathbf{+SMC},\mathbf{+SpIC}_{mv}}$ is closed under intersection with regular languages, or whether it is contained in the family of semilinear languages. We will see however that it cannot be both. On the other hand, it is straightforward to see that $\mathcal{ML}^{\mathbf{+LA},\mathbf{+SMC}}$ only contains semilinear languages. Whether it is also closed under intersection with regular languages is an open question, as is whether it properly contains $\mathcal{MCFL}$.[13]

Proofs showing that certain MG variants can derive non-semilinear languages [12, 13, 3, 14, 24] all have in common that they treat licensee features as resources which can grow without bound.[14] Imposing the **SMC** limits the amount of licensee features an expression may have to just one of each type, and thus blocks this approach to non-semilinearity. Late adjunction allows one to side-step the **SMC** in the sense that any number of moving items might be late adjoined into a particular subtree, giving rise to the 'illusion' that they were there all along. This sword cuts both ways, however, as it seems to remove the possibility of 'testing for zero' – of determining, explicitly in the cases of [13] and [14], or implicitly in the case of [24], whether a particular expression has any features of a certain type. The difficulty is simple: without limits on when something may be late adjoined, an expression which *can* be late adjoined into, can be late adjoined into *at any time*, including right after it had been determined not to contain any features of a certain type! The $\mathbf{SpIC}_{mv}$ provides a way to limit the use of adjunction in a way that allows an implicit test for zero – to block later adjunction inside of an expression, all that need to be done is to move that expression to a higher specifier position, at which point the $\mathbf{SpIC}_{mv}$ will—at least with regard to convergent derivations—ensure that no future late adjunctions may 'add' to the number of licensee features that expression contains.

We define here a language L which, when intersected with the regular language $\boldsymbol{e}^*\boldsymbol{d}(\boldsymbol{b}(\boldsymbol{ac})^*)^*$ contains only sentences where the number of instances of e is a power of two. We first provide an intuition regarding the generation of L. The

[12] Recall that the family of semilinear *string* languages is not closed under intersection with regular string languages; $\boldsymbol{a}^{2^n}\boldsymbol{b} + \boldsymbol{ba}^*$ is semilinear, but its intersection with $\boldsymbol{a}^*\boldsymbol{b}$ is not.

[13] To the best of our knowledge, whether $\mathrm{C}(\mathcal{MCFL}) \subseteq \mathcal{MCFL}$ is also open. If it could be proven that $\mathcal{MCFL}$ were not commutation closed, this would establish that $\mathcal{ML}^{\mathbf{+LA},\mathbf{+SMC}}$ properly contains $\mathcal{MCFL}$, by corollary 1.

[14] While Kobele [12, 13] claims to impose 'the **SMC**', one way of looking at what is done there (that is compatible with the generalizations expressed in this section) is that what is expressed there as 'feature percolation' is better viewed as 'pied piping', with a relaxed version of the **SMC**, which simply requires that the **move** function be deterministic, along with the requirement that the features of expressions standing in a certain geometrical relationship with the head of a subtree 'count' as belonging to that head for the purposes of determining the target of movement.

MG G_L generating L contains four lexical items which have adjunction features. These lexical items are the sole source of the symbol a in the sentences of L, i.e., anywhere an a occurs, it was introduced via the (late) adjunction of an expression derived by incorporating one of the four lexical items. Another important property of the adjuncts is that each of them introduces *two* distinct movable subtrees. Thus, because of the **SMC**, each late adjunction step must be followed by two *move*-steps before another adjunction step with the same adjunct may occur. The four lexical items with adjunction features are

$$\langle a, \mathsf{=1\ =3\ \approx e}\rangle \qquad \langle a, \mathsf{=1'\ =3'\ \approx e}\rangle$$

$$\langle a, \mathsf{=2\ =4\ \approx o}\rangle \qquad \langle a, \mathsf{=2'\ =4'\ \approx o}\rangle$$

and they select their arguments from the following list of lexical items:

$$\langle c, \mathsf{1\ -1}\rangle \qquad \langle c, \mathsf{3\ -3}\rangle \qquad \langle e, \mathsf{1'\ -0}\rangle \qquad \langle e, \mathsf{3'\ -5}\rangle$$

$$\langle c, \mathsf{2\ -2}\rangle \qquad \langle c, \mathsf{4\ -4}\rangle \qquad \langle e, \mathsf{2'\ -0}\rangle \qquad \langle e, \mathsf{4'\ -5}\rangle$$

In all expressions which can be constructed from the above eight lexical items, i.e., in all expressions providing a possible adjunct, instances of c and e do not co-occur:

$$>^{\approx \mathsf{e}}(c^{\mathsf{-3}}, <(a, c^{\mathsf{-1}})) \qquad >^{\approx \mathsf{e}}(e^{\mathsf{-5}}, <(a, e^{\mathsf{-0}}))$$

$$>^{\approx \mathsf{o}}(c^{\mathsf{-4}}, <(a, c^{\mathsf{-2}})) \qquad >^{\approx \mathsf{o}}(e^{\mathsf{-5}}, <(a, e^{\mathsf{-0}}))$$

Derivations proceed in cycles which will be referred to as ‘even’ and ‘odd’ (whence the names of the adjunction features, $\approx\mathsf{e}$ and $\approx\mathsf{o}$). An odd cycle begins with the successive merger of the two lexical items on the left, an even one with the the successive merger of the two lexical items on the right:

$$\langle \epsilon, \mathsf{=n\ x\ -5\ -0\ -4\ -2\ -x}\rangle \qquad \langle \epsilon, \mathsf{=d\ y\ -5\ -0\ -3\ -1\ -y}\rangle$$

$$\langle \epsilon, \mathsf{=x\ o}\rangle \qquad \langle \epsilon, \mathsf{=y\ e}\rangle$$

During the cycle adjunction will only take place inside the complement of the cycle-starting item. The **SMC** will ensure that only one of the four generally possible adjunct expressions listed above provides a cycle-appropriate adjunct. More precisely, the **SMC** will do so in collaboration with the somewhat unwieldy looking sequence of licensee features appearing in the corresponding cycle-starting item. Having started a cycle, we legitimate the repeated adjunction of the appropriate (even or odd) adjunct expression by repeated merger of one of the following items:

$$\langle \epsilon, \mathsf{=o\ +1\ o}\rangle \qquad \langle \epsilon, \mathsf{=e\ +2\ e}\rangle$$

$$\langle \epsilon, \mathsf{=o\ +3\ o}\rangle \qquad \langle \epsilon, \mathsf{=e\ +4\ e}\rangle$$

A cycle ends with the merger of one of the following two expressions, and the subsequent movement of the cycle-starting constituent, thereby barring later adjunctions inside of the moved constituent due to the $\mathbf{SpIC}_{mv}$. An even cycle ends in category n (eve**n**), and an odd one in category d (od**d**):

$$\langle b, \mathsf{=o\ +x\ +2\ +4\ +0\ +5\ d}\rangle \qquad \langle b, \mathsf{=e\ +y\ +1\ +3\ +0\ +5\ n}\rangle$$

The cycles are initialized with the following two lexical items (the cycles begin at 'odd').

$$\langle c, \mathsf{=i\ o}\rangle \qquad \langle \epsilon, \mathsf{i\ -5\ -0\ -4\ -2\ -x}\rangle$$

The final cycle begins with the merger of a d. This derives an expression of the start category, s.

$$\langle d, \mathsf{=n\ s}\rangle \qquad \langle d, \mathsf{=d\ s}\rangle$$

We continue by allowing the adjunction of an expression containing an a and two instances of e.

$$\langle \epsilon, \mathsf{=s\ +0\ +5\ s}\rangle$$

Theorem 2. *The language derived by G_L is the set $L \subseteq \boldsymbol{e^*d(ba^*c)(b(a|c)^*)^*}$ such that within any sentence belonging to L, the number of instances of a between any two subsequent instances of b is half the number of instances of c between the following pair of subsequent instances of b (or, if there is no such pair, after the last instance of b), and the number of instances of e is twice the number of instances of a after the last instance of b.*

Intersecting L with the regular language $\boldsymbol{e^*d(b(ac)^*)^*}$ requires that within any sentence of the resulting language, within any sentence of the resulting language, the number of instances of a and instances of c between two neighboring instances of b be the same, which means that the number of instances of c between two neighboring instances of b is twice the number of instances of c between the previous pair of neighboring instances of b. Then for any sentence $w \in L \cap \boldsymbol{e^*d(b(ac)^*)^*}$, $|w|_e = 2^{|w|_b}$.

Theorem 3. *$\mathcal{ML}^{\textbf{+LA},\textbf{+SMC},\textbf{+SpIC}_{mv}}$ properly contains $\mathcal{MCFL}$.*

Proof. To see the containment, note that every grammar $G \in \mathcal{MG}^{\textbf{+SMC},\textbf{+SpIC}_{mv}}$ is also a grammar in $\mathcal{MG}^{\textbf{+LA},\textbf{+SMC},\textbf{+SpIC}_{mv}}$.

To see that the containment is proper, note that $\mathcal{MCFL}$ constitutes a class of semilinear languages, and is closed under intersection with regular languages. Language L in Theorem 2 results in a non-semilinear language when intersected with a particular regular language, and thus $L \notin \mathcal{MCFL}$. $\square$

The remainder of this section is devoted to a sketch of the proof of Theorem 2. For convenience, the lexical items of G_L are numbered in Fig. 1, and some useful derived expressions are named in Fig. 2.

All derivations in G_L begin with the merger of lexical item 1 and 2, deriving the expression $\widetilde{t} := \mathsf{<^o}(c, \epsilon^{\mathsf{i\ -5\ -0\ -4\ -2\ -x}})$. There are three logically possible next steps, before lexical item 9 is merged. Setting $t = \widetilde{t}$, these possible next steps are:

1) $\langle \epsilon, \textsf{i -5 -0 -4 -2 -x}\rangle$ 2) $\langle c, \textsf{=i o}\rangle$	
3) $\langle \epsilon, \textsf{=x o}\rangle$ 5) $\langle \epsilon, \textsf{=o +1 o}\rangle$ 7) $\langle \epsilon, \textsf{=o +3 o}\rangle$ 9) $\langle b, \textsf{=o +x +2 +4 +0 +5 d}\rangle$	4) $\langle \epsilon, \textsf{=y e}\rangle$ 6) $\langle \epsilon, \textsf{=e +2 e}\rangle$ 8) $\langle \epsilon, \textsf{=e +4 e}\rangle$ 10) $\langle b, \textsf{=e +y +1 +3 +0 +5 n}\rangle$
11) $\langle \epsilon, \textsf{=d y -5 -0 -3 -1 -y}\rangle$ 13) $\langle d, \textsf{=d } \textbf{\textsf{s}}\rangle$	12) $\langle \epsilon, \textsf{=n x -5 -0 -4 -2 -x}\rangle$ 14) $\langle d, \textsf{=n } \textbf{\textsf{s}}\rangle$
15) $\langle \epsilon, \textsf{=}\textbf{\textsf{s}}\textsf{ +0 +5 }\textbf{\textsf{s}}\rangle$	
16) $\langle a, \textsf{=1 =3 } \approx\textsf{e}\rangle$ 18) $\langle c, \textsf{1 -1}\rangle$ 20) $\langle c, \textsf{3 -3}\rangle$ 22) $\langle a, \textsf{=1}' \textsf{ =3}' \approx\textsf{e}\rangle$ 24) $\langle e, \textsf{1' -0}\rangle$ 26) $\langle e, \textsf{3' -5}\rangle$	17) $\langle a, \textsf{=2 =4 } \approx\textsf{o}\rangle$ 19) $\langle c, \textsf{2 -2}\rangle$ 21) $\langle c, \textsf{4 -4}\rangle$ 23) $\langle a, \textsf{=2}' \textsf{ =4}' \approx\textsf{o}\rangle$ 25) $\langle e, \textsf{2' -0}\rangle$ 27) $\langle e, \textsf{4' -5}\rangle$

Fig. 1. The lexical items of G_L

$e_c := \textsf{>}^{\approx\textsf{e}}(c^{\textsf{3 -3}}, \textsf{<}(a, c^{\textsf{1 -1}}))$ $e_e := \textsf{>}^{\approx\textsf{e}}(e^{\textsf{3' -5}}, \textsf{<}(a, e^{\textsf{1' -0}}))$	$o_c := \textsf{>}^{\approx\textsf{o}}(c^{\textsf{4 -4}}, \textsf{<}(a, c^{\textsf{2 -2}}))$ $o_e := \textsf{>}^{\approx\textsf{o}}(e^{\textsf{4' -5}}, \textsf{<}(a, e^{\textsf{2' -0}}))$
$s_1 := \circ^{\textsf{d}}(\epsilon^{\textsf{i}}, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, \textsf{<}(b, \textsf{<}^{\circ}(c, \mathtt{t})))))))$	

Fig. 2. Derived expressions

1. t is merged with item 5, deriving the expression $\textsf{<}^{\textsf{+1 o}}(\epsilon, t)$. However, t does not contain any subexpression $s^{\gamma\textsf{-1}}$, and none of the possible late adjuncts o_i do either. Thus, the $\textsf{+1}$ feature at the root of this expression cannot be checked, and this expression can not participate in a derivation of a complete expression of any category.
2. t is merged with item 7. This case is similar in all relevant respects to the previous one.
3. One of the expressions o_i (late) adjoins to t, resulting in the expression $\bullet^{\circ}(o_i^{\epsilon}, t^{\epsilon})$, which contains a subtree $s^{\gamma\textsf{-k}}$ for some $k \in \{0, 2, 4, 5\}$, and also the subtree $\epsilon^{\textsf{i -5 -0 -4 -2 -x}}$. Merging this expression with lexical item 9 will ultimately result in an expression on which the **SMC**-restricted *move* operation is not defined.

Merging $\widetilde{t}$ with lexical item 9 results in $\textsf{<}^{\textsf{+x +2 +4 +0 +5 d}}(b, \textsf{<}^{\circ}(c, \epsilon^{\textsf{i -5 -0 -4 -2 -x}}))$. Continuing with this expression, five successive applications of *move* clearly result in s_1.

A cycle converts a complete expression s_i (t_i) of category $\textsf{d}$ ($\textsf{n}$) to a complete expression t_i (s_{i+1}) of category $\textsf{n}$ ($\textsf{d}$). For $s_i \in L^{\alpha}_{\textsf{d}}(G_L)$ ($t_i \in L^{\alpha}_{\textsf{n}}(G_L)$), where $\alpha = \textbf{+LA}, \textbf{+SMC}, \textbf{+SpIC}_{mv}$, we write $t_i \in \textsf{cycle}(s_i)$ ($s_{j+1} \in \textsf{cycle}(t_j)$) to indicate that t_i (s_{j+1}) is derivable from s_i (t_j) in one cycle. A cycle begins with the merger of lexical item 11 (12) with s_i (t_j), followed by the merger of lexical item 4 (3), and a cycle ends with the merger of lexical item 10 (9) and five successive applications of the *move* operation. In between, the adjunct o_c (e_c) is late adjoined. Then one of lexical item 6 (5) and 8 (7) is merged, *move* applies, the other of lexical item 6 (5) and 8 (7) is merged, and *move* applies again. Starting with $s = s_i$, we obtain the following expressions:[15]

merge 11:

$<^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, s)$

merge 4:

$<^{\mathsf{e}}(\epsilon, <^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, s))$

repeat *k* times:

lateAdjoin $\boldsymbol{o_c}$:

$<^{\mathsf{e}}(\epsilon, <^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, D[c^{\mathsf{2\ -2}}, c^{\mathsf{4\ -4}}]))$

merge 6:

$<^{\mathsf{+2\ e}}(<^{\mathsf{e}}(\epsilon, <^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, D[c^{\mathsf{2\ -2}}, c^{\mathsf{4\ -4}}])))$

move:

$\circ^{\mathsf{e}}(c^{\mathsf{2}}, <(\epsilon, <^{\mathsf{e}}(<^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, D[\mathtt{t}, c^{\mathsf{4-4}}]))))$

merge 8:

$<^{\mathsf{+4\ e}}(\epsilon, \circ^{\mathsf{e}}(c^{\mathsf{2}}, <(\epsilon, <^{\mathsf{e}}(<^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, D[\mathtt{t}, c^{\mathsf{4-4}}])))))$

move:

$\circ^{\mathsf{e}}(c^{\mathsf{4}}, <(\epsilon, \circ^{\mathsf{e}}(c^{\mathsf{2}}, <(\epsilon, <^{\mathsf{e}}(<^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, D[\mathtt{t}, \mathtt{t}]))))))$

merge 10:

$<^{\mathsf{+y\ +1\ +3\ +0\ +5\ n}}(b, \underbrace{\circ^{\mathsf{e}}(c^{\mathsf{4}}, <(\epsilon, \circ^{\mathsf{e}}(c^{\mathsf{2}}, \ldots}_{2k\ \textit{times}} <(\epsilon, <^{\mathsf{e}}(\epsilon, <^{\mathsf{y\ -5\ -0\ -3\ -1\ -y}}(\epsilon, E[\underbrace{\mathtt{t}, \ldots, \mathtt{t}}_{2k\ \textit{times}}]))) \ldots))))$

move 5 times:

$t_i = \circ^{\mathsf{n}}(\underbrace{<^{\mathsf{y}}(\epsilon, E[\mathtt{t}, \ldots, \mathtt{t}])}_{\textit{opaque by } \boldsymbol{SpIC_{mv}}}, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, <(b, \underbrace{\circ^{\mathsf{e}}(c^{\mathsf{4}}, <(\epsilon, \circ^{\mathsf{e}}(c^{\mathsf{2}}, \ldots <^{\mathsf{e}}(\epsilon, \mathtt{t}) \ldots)))}_{\textit{available for appropriate adjunction}}))))))$

Within the expression t_i (s_{j+1}) derived from s_i (t_j) in this manner, the only nodes where adjunction may take place in the next cycle without corrupting the cycle's completion are amongst the mother node of the most embedded trace `t` and the mother nodes of those instances of c introduced by (late) adjunction and moved to a specifier position during the just completed cycle. The number of these instances of c is $2k$, where k is the number of times o_c (e_c) was adjoined during the cycle. The selectee appearing in the label of the corresponding mother nodes and allowing for adjunction is `e` (`o`).

Note that nothing stops adjuncts other than o_c (e_c) from adjoining during the process above. But any such adjunct will introduce at least one feature `-k` for some $k \in \{0, 1, 3, 5\}$ ($k \in \{0, 2, 4, 5\}$) which will cause an **SMC**-violation in the last step (*move 5 times*), thereby banning the adjunct's containing expression from ever becoming complete.

A variation on the above which *is* acceptable (but which does not affect the finally derived expression) depends on the timing of the *lateAdjoin* operation.

[15] Starting a cycle with the merger of lexical item 12 and t_j works analogously.

The first adjunct o_c can be late adjoined before the merger of lexical item 11 or the merger of lexical item 4 without affect. The first adjunct might even be present already within s_i (t_j), having been introduced at the end of the preceding cycle. Put differently, regarding the cycle considered above, up to one of the adjuncts e_c (o_c) can be adjoined after the feature $\mathsf{+3}$ ($\mathsf{+4}$) has been checked in the last five movements, or alternatively up to one of the adjuncts e_e (o_e) can be adjoined after the feature $\mathsf{+5}$ has been checked in the last five movements. The latter forces the initiation of the final cycle (see below) in order to derive a complete expression.

Nothing else may happen during a cycle.

Proposition 1. *For $\alpha = \boldsymbol{+LA}, \boldsymbol{+SMC}, \boldsymbol{+SpIC_{mv}}$ consider a complete expression $s_i \in L^\alpha_{\mathsf{d}}(G_L)$ ($t_j \in L^\alpha_{\mathsf{n}}(G_L)$) with $\mathsf{yield}(s_i) = ubc^m$ ($\mathsf{yield}(t_j) = ubc^m$) for some $u \in (\boldsymbol{b}(\boldsymbol{a}|\boldsymbol{c})^*)^*$ and $m \in \mathbb{N}$. Then for every $t_i \in \mathsf{cycle}(s_i)$ ($s_{j+1} \in \mathsf{cycle}(t_j)$) it holds that $\mathsf{yield}(t_i) = ubvbc^{2n}$ ($\mathsf{yield}(s_{j+1}) = ubvbc^{2n}$) for some $n \in \mathbb{N}$ and $v \in (\boldsymbol{a}|\boldsymbol{c})^*$ with $|v|_c = m$ and $|v|_a = n$.*

Proof. By induction we actually prove a somewhat stronger proposition: we additionally show that for each $s_i \in L^\alpha_{\mathsf{d}}(G_L)$ ($t_j \in L^\alpha_{\mathsf{n}}(G_L)$) with $\mathsf{yield}(s_i) = ubc^m$ ($\mathsf{yield}(t_j) = ubc^m$) for some $u \in (\boldsymbol{b}(\boldsymbol{a}|\boldsymbol{c})^*)^*$ and $m \in \mathbb{N}$, we have $s_i = \circ^{\mathsf{d}}(r', r'')$ ($t_i = \circ^{\mathsf{d}}(r', r'')$) such that $\mathsf{yield}(r') = u$, and adjunction must take place within r'' to the right of the leaf b in order to derive a complete expression.

Because we have $s_1 = \circ^{\mathsf{d}}(\epsilon, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, \circ(\mathtt{t}, <(b, <^{\mathsf{o}}(c, \mathtt{t})))))))$, we, in fact, have $s_1 = \circ^{\mathsf{d}}(s', s'')$ such that $\mathsf{yield}(s') = \epsilon$, and adjunction can only take place within s'' to the right of the leaf b.

Now define $\mathsf{cycle}(t_0) := \{s_1\}$, and let $s_i \in \mathsf{cycle}(t_{i-1})$ be as in the assumption of the proposition, i.e., such that $\mathsf{yield}(s_i) = ubc^m$ for some $u \in (\boldsymbol{b}(\boldsymbol{a}|\boldsymbol{c})^*)^*$ and $m \in \mathbb{N}$. By hypothesis, we can even assume that $s_i = \circ^{\mathsf{d}}(s', s'')$ with $\mathsf{yield}(s') = u$, and adjunction must take place within s'' to the right of the b leaf in order to derive a complete expression. Inspection of the definition of $\mathsf{cycle}(s_i)$ therefore yields that every $t_i \in \mathsf{cycle}(s_i)$ has the desired form. The reasoning is similar for $s_{i+1} \in \mathsf{cycle}(t_i)$. □

The final cycle turns a complete expression s (t) of category d (n) into a complete expression of start category s. First the lexical item 13 (14) is merged, already generating a complete expression of category s, and then for each late adjunction of o_e (e_e), the lexical item 15 is merged, and the operation *move* applies twice, generating another complete expression of category s. Starting with a complete expression s of category d, we obtain the following expressions:

merge 13:
$<^{\mathsf{s}}(d, s)$
repeat:
lateAdjoin $\boldsymbol{o_e}$:
$<^{\mathsf{s}}(d, D[e^{\mathsf{2'}\ \mathsf{-0}}, e^{\mathsf{4'}\ \mathsf{-5}}])$
merge 15:
$<^{\mathsf{+0}\ \mathsf{+5}\ \mathsf{s}}(\epsilon, <^{\mathsf{s}}(d, D[e^{\mathsf{2'}\ \mathsf{-0}}, e^{\mathsf{4'}\ \mathsf{-5}}]))$

move:

$\circ^{\mathsf{+5\ s}}(e^2, \mathsf{<}(\epsilon, \mathsf{<^s}(d, D[\mathtt{t}, e^{\mathsf{4'\ -5}}])))$

move:

$\circ^{\mathsf{s}}(e^{\mathsf{4}}, \circ(e^{\mathsf{2}}\mathsf{<}(\epsilon, \mathsf{<^s}(d, D[\mathtt{t}, \mathtt{t}])))$

Starting a final cycle with the merger of lexical item 14 and a complete expression t of category n works analogously. Clearly, only o_e (e_e) can be late adjoined in the cycle and lead to a complete expression, as only $\mathsf{-0}$ and $\mathsf{-5}$ features can be checked.

Proof of Theorem 2. Let w.l.o.g. $s \in \bigcup_{i\in\mathbb{N}} \mathsf{cycle}(t_i)$. Then $s = \circ^{\mathsf{d}}(s', s'')$ with $\mathsf{yield}(s') = u \in (\boldsymbol{b}(\boldsymbol{a}|\boldsymbol{c})^*)^*$ and $\mathsf{yield}(s'') = bc^k$ for some $k \in \mathbb{N}$. Furthermore $r_0 = \mathit{merge}(13, s) \in L^{\alpha}_{\mathsf{s}}(G_L)$ with $\mathsf{yield}(r_0) = dubc^k$. Define now $R_0 = \{r_0\}$ and $R_{i+1} = \bigcup\{\mathit{move}(\mathit{move}(\mathit{merge}(15, \mathit{lateAdjoin}(o_e, r_i)))) : r_i \in R_i\}$. Then it holds that $R_i \subseteq L^{\alpha}_{\mathsf{s}}(G_L)$, and each $r_i \in R_i$ is such that $\mathsf{yield}(r_i) = e^{2i}dubv$ for some $v \in (\boldsymbol{a}|\boldsymbol{c})^*$ with $|v|_c = k$ and $|v|_a = i$. By Proposition 1, the desired relation between instances of a and instances of c obtains. $\square$

5 Conclusion

We have argued that the **SpIC** is best understood formally as three separate constraints on extraction, one for each 'kind' of specifier. Whatever the linguistic merits of this proposal, it has allowed us a clearer perspective on previous results, as well as given us a way forward to pursue novel ones. While our results underscore the conclusion of Gärtner and Michaelis [3, p. 127] that

> intuitions to the contrary notwithstanding, the imposition of [locality conditions] on grammars [. . .] does not automatically reduce their generative capacity

what is still missing is a 'big picture' perspective as to why exactly certain constraints interact with others in the way they do, which would allow us to reason at a high level about constraint behavior in general in a minimalist setting.

Another result of the present paper is the beginnings of a classification of the behavior of minimalist grammars with late adjunction. Late adjunction has similarities with the kind of second order substitution found in macro grammars [1], and thus minimalist grammars with late adjunction combine in a particular way macros with tupling, cf. [22], as do (but differently) multi-component tree adjoining grammars, cf. [28, 31]. We must leave it to future work to flesh out this perspective.

References

1. Fischer, M.J.: Grammars with Macro-like Productions. Ph.D. thesis, Harvard (1968)
2. Frey, W., Gärtner, H.M.: On the treatment of scrambling and adjunction in minimalist grammars. In: Proceedings of the Conference on Formal Grammar (FGTrento), Trento. pp. 41–52 (2002)

3. Gärtner, H.M., Michaelis, J.: A note on the complexity of constraint interaction. Locality conditions and minimalist grammars. In: Blache, P., Stabler, E., Busquets, J., Moot, R. (eds.) LACL 2005, LNAI, Vol. 3492, pp. 114–130. Springer, Berlin, Heidelberg (2005)
4. Gärtner, H.M., Michaelis, J.: Some remarks on locality conditions and minimalist grammars. In: Sauerland, U., Gärtner, H.M. (eds.) Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax and Semantics, pp. 161–195. Mouton de Gruyter, Berlin (2007)
5. Gärtner, H.M., Michaelis, J.: A note on countercyclicity and minimalist grammars. In: Penn, G. (ed.) Proceedings of FGVienna: The 8th Conference on Formal Grammar, pp. 95–109. CSLI Publications, Stanford, CA (2008), published within the corresponding volume of the CSLI Online Publications' series of FG Conference Proceedings available at `http://cslipublications.stanford.edu/FG/2003/index.html`
6. Girard, J.Y.: Linear logic (1987)
7. de Groote, P., Guillaume, B., Salvati, S.: Vector addition tree automata. In: 19th Annual IEEE Symposium on Logic in Computer Science (LICS '04), Turku. pp. 64–73. IEEE (2004)
8. de Groote, P., Morrill, G., Retoré, C. (eds.): LACL 2001, LNAI, Vol. 2099. Springer, Berlin, Heidelberg (2001)
9. Harkema, H.: A characterization of minimalist languages. In: de Groote et al. [8], pp. 193–211
10. Kanazawa, M.: The pumping lemma for well-nested multiple context-free languages. In: Diekert, V., Nowotka, D. (eds.) DLT 2009, LNCS, Vol. 5583, pp. 312–325. Springer, Berlin, Heidelberg (2009)
11. Kanazawa, M., Michaelis, J., Salvati, S., Yoshinaka, R.: Well-nestedness properly subsumes strict derivational minimalism. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011, LNAI, Vol. 6736, pp. 112–128. Springer, Berlin, Heidelberg (2011)
12. Kobele, G.M.: Formalizing mirror theory. Grammars 5, 177–221 (2002)
13. Kobele, G.M.: Features moving madly. Research on Language and Computation 3, 391–410 (2005)
14. Kobele, G.M., Michaelis, J.: Two type-0 variants of minimalist grammars. In: Rogers [23], pp. 81–91, published within the corresponding volume of the CSLI Online Publications' series of FG Conference Proceedings available at `http://cslipublications.stanford.edu/FG/2005/index.html`
15. Kracht, M.: The Mathematics of Language. Mouton de Gruyter, Berlin (2003)
16. Lincoln, P.: Deciding provability of linear logic formulas. In: Girard, J.Y., Lafont, Y., Regnier, L. (eds.) Proceedings of the Workshop on Advances in Linear Logic, pp. 197–210. Cambridge University Press, Cambridge (1995)
17. Michaelis, J.: Derivational minimalism is mildly context-sensitive. In: Moortgat, M. (ed.) LACL '98, LNAI, Vol. 2014, pp. 179–198. Springer, Berlin, Heidelberg (2001)
18. Michaelis, J.: On Formal Properties of Minimalist Grammars. Linguistics in Potsdam 13, Universitätsbibliothek, Publikationsstelle, Potsdam (2001), Ph.D. thesis
19. Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. In: de Groote et al. [8], pp. 228–244
20. Michaelis, J.: Observations on strict derivational minimalism. Electronic Notes in Theoretical Computer Science 53, 192–209 (2004), available at `http://www.sciencedirect.com/science/journal/15710661`
21. Michaelis, J.: An additional observation on strict derivational minimalism. In: Rogers [23], pp. 101–111, published within the corresponding volume of the CSLI Online Publications' series of FG Conference Proceedings available at `http://cslipublications.stanford.edu/FG/2005/index.html`

22. Mönnich, U.: Tupel vs. Makros. In: Gärtner, H.M., Beck, S., Eckardt, R., Musan, R., Stiebels, B. (eds.) Between 40 and 60 Puzzles for Krifka. ZAS, Berlin (2006), available at `http://www.zas.gwz-berlin.de/fileadmin/material/40-60-puzzles-for-krifka`
23. Rogers, J. (ed.): Proceedings of FG-MoL 2005: The 10th Conference on Formal Grammar and The 9th Meeting on Mathematics of Language. CSLI Publications, Stanford, CA (2009)
24. Salvati, S.: Minimalist grammars in the light of logic. Research Report, INRIA Bordeaux (2011), available at `http://hal.inria.fr/inria-00563807/en/`
25. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88, 191–229 (1991)
26. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) LACL '96, LNAI, Vol. 1328, pp. 68–95. Springer, Berlin, Heidelberg (1997)
27. Stabler, E.P.: Remnant movement and complexity. In: Bouma, G., Kruijff, G.J.M., Hinrichs, E., Oehrle, R.T. (eds.) Constraints and Resources in Natural Language Syntax and Semantics, pp. 299–326. CSLI Publications, Stanford, CA (1999)
28. Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: 25th Annual Meeting of the Association for Computational Linguistics (ACL '87)*, Stanford, CA*. pp. 104–111. ACL (1987)
29. Villemonte de la Clergerie, É.: Parsing MCS languages with thread automata. In: Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+6), Venezia. pp. 101–108 (2002)
30. Villemonte de la Clergerie, É.: Parsing mildly context-sensitive languages with thread automata. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING '02), Taipei. pp. 1–7 (2002)
31. Weir, D.J.: Characterizing Mildly Context-Sensitive Grammar Formalisms. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA (1988)