8

Two Type 0-Variants of Minimalist Grammars

Gregory M. Kobele and Jens Michaelis

Abstract

Minimalist grammars (Stabler 1997) capture some essential ideas about the basic operations of sentence construction in the Chomskyian syntactic tradition. Their affinity with the unformalized theories of working linguists makes it easier to implement and thereby to better understand the operations appealed to in neatly accounting for some of the regularities perceived in language. Here we characterize the expressive power of two, apparently quite different, variations on the basic minimalist grammar framework, gotten by:

1. adding a mechanism of 'feature percolation' (Kobele, forthcoming), or
2. instead of adding a central constraint on movement (the 'specifier island condition', Stabler 1999), using it to replace another one (the 'shortest move condition', Stabler 1997, 1999) (Gärtner and Michaelis 2005).

We demonstrate that both variants have equal, unbounded, computing power by showing how each can simulate straightforwardly a 2-counter automaton.

Keywords minimalist grammars, 2-counter automata, locality conditions, feature percolation

8.1 Introduction

Recently, two variants of the *minimalist grammar (MG)* formalism introduced in Stabler 1997 have been discussed w.r.t. the issue of generative capacity (see Gärtner and Michaelis 2005 and Kobele, forthcoming).

Seen from a linguistic perspective, the motivation for studying the two variants arose from two rather different starting points: Kobele (forthcoming), attempting to provide a formalization of mechanisms used to account for pied-piping (Ross 1967), considers MGs endowed

FG-MoL 2005.
James Rogers (ed.).

with feature percolation from specifiers to (attracting) heads. Gärtner and Michaelis (2005), as part of a larger project to better understand the effects of constraint interaction in minimalist syntax, study the behaviour of the *specifier island constraint (SPIC)* in isolation from Stabler's original *shortest move constraint (SMC)*.

What both variants have in common formally is that, in contrast to the original MG-type, they allow the generation of non-mildly context-sensitive languages: Kobele (forthcoming) shows how an arbitrary *(infinite) abacus* (Lambek 1961) can be simulated by an MG^{+perc}, an MG enriched with the corresponding mechanism of feature percolation. As a corollary he shows how any arbitrary recursively enumerable subset of the natural numbers can be derived as the string language of an $\mathrm{MG}^{+\mathrm{perc}}$. Thus, by means of an "MG-external" encoding (i.e. a computable, bijective function $f_\Sigma : \mathbb{N} \to \Sigma^*$ for any finite set Σ),[1] $\mathrm{MG}^{+\mathrm{perc}}$s can be seen as deriving the class of all type 0-languages over arbitrary finite alphabets. However, the question of how to define an $\mathrm{MG}^{+\mathrm{perc}}$ which directly derives an arbitrary type 0-language is left open.

Gärtner and Michaelis (2005) show that there is an $MG^{\text{-}SMC,+SPIC}$, an MG respecting the SPIC but not the SMC, which derives a non-semilinear string language, and they conjecture that each type 0-language is derivable by some $\mathrm{MG}^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$.

In this paper we prove the full Turing power of $\mathrm{MG}^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$s as well as $\mathrm{MG}^{+\mathrm{perc}}$s, showing that for each 2-counter automaton there is an $\mathrm{MG}^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$ as well as an $\mathrm{MG}^{+\mathrm{perc}}$ which both generate exactly the language accepted by the automaton. In fact, our construction of a corresponding $\mathrm{MG}^{+\mathrm{perc}}$ is fully general, holding for all variants of the feature percolation mechanism proposed in Kobele, forthcoming.

8.2 2-Counter Automata

Definition 13 A 2-*counter automaton (2-CA)* is given by a 7-tuple $M = \langle Q, \Sigma, \{\mathbf{1}, \mathbf{2}\}, \delta, \{\mathtt{\#_1}, \mathtt{\#_2}\}, q_0, Q_f\rangle$, where Q and Σ are the finite sets of *states* and *input symbols*, respectively. For $i \in \{1, 2\}$, $\mathbf{i}$ is the *i-th counter symbol*, and $\mathtt{\#_i}$ is the *i-th end of stack symbol*. $q_0 \in Q$ is the *initial state*, $Q_f \subseteq Q$ is the set of *final states*, δ is the *transition function* from $Q \times \Sigma_\epsilon \times \{\mathbf{1}, \mathtt{\#_1}\} \times \{\mathbf{2}, \mathtt{\#_2}\}$ to $\mathcal{P}_{\mathrm{fin}}(Q \times \mathbf{1}^* \times \mathbf{2}^*)$.[2] An *instantaneous configuration* is a 4-tuple from $Q \times \mathbb{N} \times \mathbb{N} \times \Sigma^*$. For $a \in \Sigma_\epsilon$ we write $\langle q, n_1, n_2, aw\rangle \vdash_M \langle q', n_1', n_2', w\rangle$ just in case $\langle q', \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, g_1, g_2\rangle)$,

[1] Throughout, we take $\mathbb{N}$ to denote the set of natural numbers, including 0. For every set X, X^* is the Kleene closure of X, including ϵ, the empty string.

[2] For each set X, X_ϵ denotes the set $X \cup \{\epsilon\}$. $\mathcal{P}_{\mathrm{fin}}(X)$ is the set of all finite subsets of X. For a singleton set $\{x\}$, we often write x, if this does not cause misunderstandings. Thus, x^* denotes $\{x\}^*$ under this convention.

where $g_i = \mathtt{\#_i}$ iff $n_i = 0$, and $n'_i = |\gamma_i| + (n_i \dot{-} 1)$.[3] $\vdash_M^*$ is the reflexive-transitive closure of $\vdash_M$. The *language determined by* M, $L(M)$, is the set $\{w \mid \langle q_0, 0, 0, w\rangle \vdash_M^* \langle q, 0, 0, \epsilon\rangle \text{ for some } q \in Q_f\}$.

It is known (cf. Hopcroft and Ullman 1979) that the class of languages determined by 2-CAs is exactly the class of type 0-languages.

8.3 Minimalist Grammars and Variations

It will be useful to explicitly mark the "outer complement line" and the corresponding specifiers of a minimalist expression. To do this we extend the notation from Stabler and Keenan 2003, introduced there in order to reduce the presentation of minimalist expressions to "the bare essentials." Throughout we let Σ and *Syn* be disjoint sets, a finite set of *non-syntactic features*, the *(terminal) alphabet*, and a finite set of *syntactic features*, respectively, in accordance with (F1) and (F2). We take *Feat* to be the set $\Sigma \cup Syn$. Furthermore, we let ::, :, `comp`, `spec` and $-$ be pairwise distinct new symbols, where, in particular, :: and : serve to denote *lexical* and *derived* types, respectively.

(F1) *Syn* is partitioned into five sets:[4]

Base		a set of *(basic) categories*
Select	$= \{ \mathtt{=x} \mid \mathtt{x} \in Base \}$	a set of *selectors*
Licensees	$= \{ \mathtt{-x} \mid \mathtt{x} \in Base \}$	a set of *licensees*
Licensors	$= \{ \mathtt{+x} \mid \mathtt{x} \in Base \}$	a set of *licensors*
P-Licensors	$= \{ \mathtt{+}\widehat{\mathtt{x}} \mid \mathtt{x} \in Base \}$	a set of *p-licensors*

(F2) *Base* includes at least the category `c`.

Definition 14 An element of $\Sigma^* \times \{::, :\} \times Syn^* \times \{\mathtt{comp}, \mathtt{spec}, -\}$ is a *chain (over Feat)*. *Chains* denotes the set of all chains over *Feat*.

An element of $Chains^* \setminus \{\epsilon\}$ is an *expression (over Feat)*. *Exp* denotes the set of all expressions over *Feat*.

For given $\phi \in Syn^*$, $\cdot_\alpha \in \{::, :\}$, and $z \in \{\mathtt{comp}, \mathtt{spec}, -\}$, the chain $\langle s, \cdot_\alpha, \phi, z\rangle$ is usually denoted as $\langle s \cdot_\alpha \phi, z\rangle$, and is said to *display (open) feature* f if $\phi = f\chi$ for some $f \in Syn$ and $\chi \in Syn^*$.

A classical MG in the sense of Stabler 1997 employs the functions *merge* and *move*, creating minimalist expressions from a finite lexicon *Lex*. The corresponding definitions are explicitly given w.r.t. trees, as in Stabler 1999 too. There a revised MG-type is introduced, obeying, besides the *shortest move condition (SMC)*, a particular implementation of the *specifier island condition (SPIC)*: to be movable, a constituent must

[3] For each set M and each $w \in M^*$, $|w|$ denotes the length of w. For all $x, y \in \mathbb{N}$, $x \dot{-} y$ is defined by $x \dot{-} y = x - y$ if $x > y$, and by $x \dot{-} y = 0$ otherwise.

[4] Elements from *Syn* will usually be typeset in `typewriter font`.

belong to the transitive complement closure of a given tree, or be a specifier of such a constituent.[5] The SPIC crucially implies that moving or merging a constituent α into a specifier position blocks checking (by later movement) of any licensee feature displayed by some proper subconstituent of α. Thus in order to avoid "crashing" derivations, only the lowest embedded complement within the complement closure displaying some licensee can move, and then only if it contains no specifier with an unchecked feature.[6]

The *minimalist grammar (MG)* types to be introduced below differ essentially in the definitions of their structure building functions. Accordingly, we first introduce those. We let $s, t \in \Sigma^*$, $\cdot_\alpha, \cdot_\beta \in \{::, :\}$, $\mathtt{f} \in \mathit{Base}$, $\phi, \chi \in \mathit{Syn}^*$, $z \in \{\mathtt{comp}, \mathtt{spec}, -\}$, and $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_l \in \mathit{Chains}$ for some $k, l \in \mathbb{N}$ such that $\cdot_\alpha =::$ implies $k = 0$, and we let $i \in \mathbb{N}$ with $i \leq k$. Also, relevant in (mo-fp$_\otimes$), we let $\psi \in (\mathit{Syn} \setminus \mathit{Licensees})^*$, and $\phi', \chi' \in \mathit{Licensees}^*$, and assume $\otimes$ to be some linear, non-deleting function from $\mathit{Syn}^* \times \mathit{Syn}^*$ to Syn^*, i.e., $\otimes$ neither deletes material nor inserts material not in its arguments.

(me) *merge* maps partially from $\mathit{Exp} \times \mathit{Exp}$ to Exp such that the pair $\langle \widehat{\alpha}, \widehat{\beta} \rangle$ built from the expressions $\widehat{\alpha} = \langle \langle s \cdot_\alpha \mathtt{=f}\phi, - \rangle, \alpha_1, \ldots, \alpha_k \rangle$ and $\widehat{\beta} = \langle \langle t \cdot_\beta \mathtt{f}\chi, - \rangle, \beta_1, \ldots, \beta_l \rangle$, belongs to $\mathit{Dom}(\mathit{merge})$,[7] and the value $\mathit{merge}(\widehat{\alpha}, \widehat{\beta})$ is defined as

(me.1) $\langle \langle st : \phi, - \rangle, \beta_1, \ldots, \beta_l \rangle$ if $\cdot_\alpha =::$ and $\chi = \epsilon$
(me.2) $\langle \langle s : \phi, - \rangle, \langle t \cdot \chi, \mathtt{comp} \rangle, \beta_1, \ldots, \beta_l \rangle$ if $\cdot_\alpha =::$ and $\chi \neq \epsilon$
(me.3) $\langle \langle ts : \phi, - \rangle, \beta_1, \ldots, \beta_l, \alpha_1, \ldots, \alpha_k \rangle$ if $\cdot_\alpha =:$ and $\chi = \epsilon$
(me.4) $\langle \langle s : \phi, - \rangle, \langle t \cdot \chi, \mathtt{spec} \rangle, \beta_1, \ldots, \beta_l, \alpha_1, \ldots, \alpha_k \rangle$ otherwise[8]

(me$^{+\mathbf{SPIC}}$) The partial mapping $\mathit{merge}^{+\mathrm{SPIC}}$ from $\mathit{Exp} \times \mathit{Exp}$ to Exp is defined such that the pair $\langle \widehat{\alpha}, \widehat{\beta} \rangle$ built from the expressions $\widehat{\alpha} = \langle \langle s \cdot_\alpha \mathtt{=f}\phi, - \rangle, \alpha_1, \ldots, \alpha_k \rangle$ and $\widehat{\beta} = \langle \langle t \cdot_\beta \mathtt{f}\chi, - \rangle, \beta_1, \ldots, \beta_l \rangle$, belongs to $\mathit{Dom}(\mathit{merge})$ iff the *specifier island condition on merger* as expressed in (SPIC.me) is satisfied, in which case $\mathit{merge}^{+\mathrm{SPIC}}(\widehat{\alpha}, \widehat{\beta}) = \mathit{merge}(\widehat{\alpha}, \widehat{\beta})$. The specifier island condition on

[5] It can be shown that, in terms of derivable string languages, this revised type defines a proper subclass of the original type (Michaelis 2005). That is to say, adding the SPIC to the SMC, in fact, reduces the weak generative capacity of the formalism.

[6] Note that, in (me.2), respectively (me.4) and (mo-$^{\mathrm{SMC}}$) below, a constituent displaying a further unchecked feature after *merge* or *move* has been applied, gets marked as complement or specifier, respectively.

[7] For a partial function f from a set A into a set B, $\mathit{Dom}(f)$ is the domain of f, i.e., the set of all $x \in A$ for which $f(x)$ is defined.

[8] Recall that by assumption, $\cdot_\alpha =::$ implies $k = 0$.

merger in effect enforces a constraint against proper left branch extraction, disallowing movement from inside a specifier (a left branch), by prohibiting the merger of specifiers which contain proper subconstituents potentially moving in a later derivation step:

$$\text{If } \cdot_\alpha =: \text{ then } l = 0. \qquad \textbf{(SPIC.me)}$$

(mo-SMC) $move^{\text{-SMC}}$ is a partial mapping from *Exp* to $\mathcal{P}_{\text{fin}}(Exp)$ such that $\widehat{\alpha} = \langle\langle s \cdot_\alpha \texttt{+f}\phi, -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \langle t \cdot_\beta \texttt{-f}\chi, z\rangle, \alpha_{i+1}, \ldots, \alpha_k\rangle$ belongs to $Dom(move)$, and the value $move^{\text{-SMC}}(\widehat{\alpha})$ includes

(mo-SMC.1) $\langle\langle ts : \phi, -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k\rangle$ if $\chi = \epsilon$

(mo-SMC.2) $\langle\langle s : \phi, -\rangle, \langle t : \chi, \texttt{spec}\rangle, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k\rangle$ if $\chi \neq \epsilon$

(mo-SMC,+SPIC) $move^{\text{-SMC,+SPIC}}$ maps partially from *Exp* to $\mathcal{P}_{\text{fin}}(Exp)$ with $\widehat{\alpha} = \langle\langle s \cdot_\alpha \texttt{+f}\phi, -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \langle t \cdot_\beta \texttt{-f}\chi, z\rangle, \alpha_{i+1}, \ldots, \alpha_k\rangle$ belonging to $Dom(move^{\text{-SMC,+SPIC}})$ iff the *specifier island condition on movement* as expressed in (SPIC.mo) is satisfied, and then $move^{\text{-SMC,+SPIC}}(\widehat{\alpha}) = move^{\text{-SMC}}(\widehat{\alpha})$. In conjunction with (SPIC.me), which ensures that the only way z can be `comp` is if it was introduced by (me.2), the specifier island condition on movement requires that all chains internal to the subtree whose root is the chain in question have themselves moved out before it is permitted to move.

$$\text{If } z = \texttt{comp} \text{ then } i = k. \qquad \textbf{(SPIC.mo)}$$

(mo) *move* is a partial mapping from *Exp* to $\mathcal{P}_{\text{fin}}(Exp)$ such that $\widehat{\alpha} = \langle\langle s \cdot_\alpha \texttt{+f}\phi, -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \langle t \cdot_\beta \texttt{-f}\chi, z\rangle, \alpha_{i+1}, \ldots, \alpha_k\rangle$ belongs to $Dom(move)$ iff the *shortest move constraint* as expressed in (SMC) holds, in which case $move(\widehat{\alpha}) = move^{\text{-SMC}}(\widehat{\alpha})$. The shortest move constraint disallows 'competition' for the same position—where there is competition, there is a loser, and thus something that will move farther then it had to.

None of the chains $\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k$ displays `-f`. **(SMC)**

(mo-fp$_\otimes$) *move-fp*$_\otimes$ is a partial mapping from *Exp* to $\mathcal{P}_{\text{fin}}(Exp)$ with $\widehat{\alpha}_\otimes = \langle\langle s : \widehat{\texttt{+f}}\psi\phi', -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \langle t : \texttt{-f}\chi', z\rangle, \alpha_{i+1}, \ldots, \alpha_k\rangle$ belonging to $Dom(move\text{-}fp_\otimes^{\text{-SMC}})$ iff (SMC) is satisfied, and with $move\text{-}fp_\otimes(\widehat{\alpha})$ including the expression

(—) $\langle\langle ts : \psi(\phi' \otimes \chi'), -\rangle, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_k\rangle$.[9]

(mo$^{+\mathbf{perc}}$) $move^{+\mathrm{perc}}_{\otimes}$ is the partial mapping from *Exp* to $\mathcal{P}_{\mathrm{fin}}(Exp)$ which results from the union of *move* and *move-fp*$_{\otimes}$.

In the following, *Lex* denotes a *lexicon (over Feat)*, i.e., *Lex* is a finite set of simple expressions of lexical type, more concretely, a finite subset of $\Sigma \times \{::\} \times Syn^* \times \{-\}$.

Definition 15 (Gärtner and Michaelis 2005) An *MG without SMC, but obeying the SPIC (MG$^{\text{-}SMC,+SPIC}$)* is a tuple $\langle \Sigma, Syn, \{::, :\}, Lex, \Omega, \mathsf{c}\rangle$ with Ω being the set $\{merge^{+\mathrm{SPIC}}, move^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}\}$.

Definition 16 (Kobele, forthcoming) For a linear, non-deleting mapping $\otimes$ from $Syn^* \times Syn^*$ to Syn^*, a 6-tuple $\langle \Sigma, Syn, \{::, :\}, Lex, \Omega, \mathsf{c}\rangle$ is called an *MG with percolation from specifiers to heads (MG^{+perc})* if $\Omega = \{merge, move^{+\mathrm{perc}}_{\otimes}\}$.

For $G = \langle \Sigma, Syn, \{::, :\}, Lex, \Omega, \mathsf{c}\rangle$, an MG$^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$, or MG$^{+\mathrm{perc}}$, the *closure of* G, $CL(G)$, is the set $\bigcup_{k\in\mathbb{N}} CL^k(G)$, where $CL^0(G) = Lex$, and for $k \in \mathbb{N}$, $CL^{k+1}(G) \subseteq Exp$ is recursively defined as

$$CL^k(G) \cup \{merge'(v, \phi) \mid \langle v, \phi\rangle \in Dom(merge') \cap CL^k(G) \times CL^k(G)\} \cup \bigcup_{v \in Dom(move') \cap CL^k(G)} move'(v)$$

with $merge' = merge^{+\mathrm{SPIC}}$ if G is an MG$^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$, and $merge' = merge$ otherwise, and with $move' \in \Omega \setminus \{merge'\}$. The *(string) language derivable by* G, $L(G)$, is the set $\{s \in \Sigma^* \mid \langle s \cdot \mathsf{c}, -\rangle \in CL(G) \text{ for } \cdot \in \{::, :\}\}$.

8.4 Simulating a 2-Counter Automata

Let $M = \langle Q, \Sigma, \{\mathbf{1}, \mathbf{2}\}, \delta, \{\#_{\mathbf{1}}, \#_{\mathbf{2}}\}, q_0, Q_f\rangle$ be a 2-CA. In constructing an MG$^{\text{-}\mathrm{SMC},+\mathrm{SPIC}}$, G_1, and an MG$^{+\mathrm{perc}}$, G_2, with $L(G_1) = L(G_2) = L(M)$, we take $\#_\Sigma$, $\mathtt{1}$ and $\mathtt{2}$ to be new, pairwise distinct symbols:

$$\begin{aligned} Base \;=\; & \{\mathsf{c}, \#_\Sigma\} \cup \{\mathsf{q} \mid q \in Q\} \cup \{\mathtt{1}, \mathtt{2}\} \\ \cup\; & \{\, \mathtt{0}_{qajkr\gamma_1\gamma_2}\,,\, \mathtt{1}^{(h)}_{qajkr\gamma_1\gamma_2}\,,\, \mathtt{2}^{(i)}_{qajkr\gamma_1\gamma_2} \mid \\ & \quad q, r \in Q\,, a \in \Sigma_\epsilon\,, j \in \{\mathbf{1}, \#_{\mathbf{1}}\}\,, k \in \{\mathbf{2}, \#_{\mathbf{2}}\}\,, \gamma_1 \in \mathbf{1}^*\,, \gamma_2 \in \mathbf{2}^* \\ & \quad \text{with } \langle r, \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, j, k\rangle)\,,\, 0 \le h \le |\gamma_1|\,,\, 0 \le i \le |\gamma_2| \,\} \end{aligned}$$

For $q, r \in Q$, $a \in \Sigma_\epsilon$, $j \in \{\mathbf{1}, \#_{\mathbf{1}}\}$, $k \in \{\mathbf{2}, \#_{\mathbf{2}}\}$, $\gamma_1 \in \mathbf{1}^*$, $\gamma_2 \in \mathbf{2}^*$ such that $\langle r, \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, j, k\rangle)$, the categories $\mathtt{1}^{(|\gamma_1|)}_{qajkr\gamma_1\gamma_2}$ and $\mathtt{2}^{(|\gamma_2|)}_{qajkr\gamma_1\gamma_2}$ are likewise denoted by $\mathtt{1}_{qajkr\gamma_1\gamma_2}$ and $\mathtt{2}_{qajkr\gamma_1\gamma_2}$, respectively.

[9] Note that $move(\widehat{\alpha})$ and $move\text{-}fp_{\otimes}(\widehat{\alpha})$, in (mo) and (mo-fp) respectively, both are singleton sets because of (SMC). Thus, these functions can easily be identified with one from *Exp* to *Exp*.

An MG$^{\text{-SMC},+\text{SPIC}}$-expression which represents an instantaneous configuration $\langle q, n_1, n_2, t\rangle$, derived from an initial configuration $\langle q_0, 0, 0, st\rangle$ in a 2-CA will have the following shape:

$$(*_1)\quad \langle\, \langle \epsilon : \mathtt{q}, -\rangle, \underbrace{\langle \epsilon : \mathtt{-2}, \mathtt{comp}\rangle, \ldots}_{n_2 \text{ times}}, \underbrace{\langle \epsilon : \mathtt{-1}, \mathtt{comp}\rangle, \ldots}_{n_1 \text{ times}}, \langle s : \mathtt{-\#_\Sigma}, \mathtt{comp}\rangle \,\rangle$$

Construction 1 $G_1 = \langle \Sigma, Syn, \{::, :\}, Lex_1, \Omega, c\rangle$ is the MG$^{\text{-SMC},+\text{SPIC}}$ with $L(G_1) = L(M)$ such that Lex_1 contains exactly the items:

$\boldsymbol{\phi_0} \;::\; \langle\, \boldsymbol{\epsilon} :: \mathtt{q_0\text{-}\#_\Sigma}\,,\, -\rangle$

For all $q, r \in Q, a \in \Sigma_\epsilon, j \in \{\mathbf{1}, \mathbf{\#_1}\}, k \in \{\mathbf{2}, \mathbf{\#_2}\}, \gamma_1 \in \mathbf{1}^*, \gamma_2 \in \mathbf{2}^*$ such that $\langle r, \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, j, k\rangle)$

$\boldsymbol{\chi_{qajkr\gamma_1\gamma_2}} \;::\; \langle\, \boldsymbol{a} :: \mathtt{=q\ +\#_\Sigma}\ 0_{qajkr\gamma_1\gamma_2}\ \mathtt{-\#_\Sigma}\,,\, -\rangle$

If $j = \#_1$ then

$\boldsymbol{\alpha^{0}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}0_{qajkr\gamma_1\gamma_2}\ 1^{(0)}_{qajkr\gamma_1\gamma_2}\,,\, -\rangle$

else

$\boldsymbol{\alpha^{0'}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}0_{qajkr\gamma_1\gamma_2}\ \mathtt{+1}\ 1^{(0)}_{qajkr\gamma_1\gamma_2}\,,\, -\rangle$

For $0 \le h < |\gamma_1|$

$\boldsymbol{\alpha^{h+1}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}1^{(h)}_{qajkr\gamma_1\gamma_2}\ 1^{(h+1)}_{qajkr\gamma_1\gamma_2}\ \mathtt{-1}\,,\, -\rangle$

$\boldsymbol{\alpha^{\sim}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}1_{qajkr\gamma_1\gamma_2}\ \mathtt{+1}\ 1_{qajkr\gamma_1\gamma_2}\ \mathtt{-1}\,,\, -\rangle$

If $k = \#_2$ then

$\boldsymbol{\beta^{0}_{qajkr\gamma_1\gamma_2}} \;::\; \langle\, \boldsymbol{\epsilon} :: \mathtt{=}1_{qajkr\gamma_1\gamma_2}\ 2^{(0)}_{qajkr\gamma_1\gamma_2}\,,\, -\rangle$

else

$\boldsymbol{\beta^{0'}_{qajkr\gamma_1\gamma_2}} \;::\; \langle\, \boldsymbol{\epsilon} :: \mathtt{=}1_{qajkr\gamma_1\gamma_2}\ \mathtt{+2}\ 2^{(0)}_{qajkr\gamma_1\gamma_2}\,,\, -\rangle$

For $0 \le i < |\gamma_2|$

$\boldsymbol{\beta^{i+1}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}2^{(i)}_{qajkr\gamma_1\gamma_2}\ 2^{(i+1)}_{qajkr\gamma_1\gamma_2}\ \mathtt{-2}\,,\, -\rangle$

$\boldsymbol{\beta^{\sim}_{qajkr\gamma_1\gamma_2}} \;::\; \langle \boldsymbol{\epsilon} :: \mathtt{=}2_{qajkr\gamma_1\gamma_2}\ \mathtt{+2}\ 2_{qajkr\gamma_1\gamma_2}\ \mathtt{-2}\,,\, -\rangle$

$\boldsymbol{\psi_{qajkr\gamma_1\gamma_2}} \;::\; \langle\, \boldsymbol{\epsilon} :: \mathtt{=}2_{qajkr\gamma_1\gamma_2}\ \mathtt{r}\,,\, -\rangle$

For each $q \in Q_f$,

$\boldsymbol{\phi_q} \;::\; \langle\, \boldsymbol{\epsilon} :: \mathtt{=q\ +\#_\Sigma\ c}\,,\, -\rangle$

Each derivation necessarily starts with ϕ_0 being selected, either by ϕ_{q_0} in case $q_0 \in Q_f$, or by $\chi_{q_0ajkr\gamma_1\gamma_2}$ for some $r \in Q$, $a \in \Sigma_\epsilon$, $j \in \{\mathbf{1}, \mathbf{\#_1}\}$, $k \in \{\mathbf{2}, \mathbf{\#_2}\}$, $\gamma_1 \in \mathbf{1}^*$ and $\gamma_2 \in \mathbf{2}^*$. Generally, when an expression is selected by a lexical item of the form $\chi_{qajkr\gamma_1\gamma_2}$ for some $q, r \in Q$, $a \in \Sigma_\epsilon$, $j \in \{\mathbf{1}, \mathbf{\#_1}\}$, $k \in \{\mathbf{2}, \mathbf{\#_2}\}$, $\gamma_1 \in \mathbf{1}^*$ and $\gamma_2 \in \mathbf{2}^*$, G_1 begins simulating the CA's application of δ to $\langle q, a, j, k\rangle$ with outcome $\langle r, \gamma_1, \gamma_2\rangle$ w.r.t. some matching instantaneous configuration: the currently recognized a from the input tape is introduced, and afterwards the already recognized prefix (being the alphabetic string of the last chain within the currently derived expression displaying $\mathtt{-\#_\Sigma}$, cf. $(*_1)$) is moved to the left of a.

This creates an expression whose first chain displays $\mathtt{0}_{qajkr\gamma_1\gamma_2}$. Depending on j, this expression is selected either by $\alpha^{0}_{qajkr\gamma_1\gamma_2}$, which does not consume an instance of **1** from the 1st CA-counter, or $\alpha^{0'}_{qajkr\gamma_1\gamma_2}$, which does by applying $move^{\text{-SMC},+\text{SPIC}}$ in the next step. The resulting expression is selected successively by $\alpha^{h+1}_{qajkr\gamma_1\gamma_2}$ for $0 \leq h < |\gamma_1|$, each time introducing a new complement displaying `-1`, and finally creating an expression whose first chain displays $\mathtt{1}_{qajkr\gamma_1\gamma_2}$. Thus, the correct number of new instances of **1** on the 1st CA-counter are introduced. Now, all the lower complements in the complement closure, which display `-1` (and which were created in an earlier cycle simulating another application of δ) are cycled through to the top of the expression, first merging with $\alpha^{\sim}_{qajkr\gamma_1\gamma_2}$, then applying $move^{\text{-SMC},+\text{SPIC}}$. This ends in a configuration in which all chains displaying `-1` are consecutive components within the expression derived, immediately following the first chain which still displays $\mathtt{1}_{qajkr\gamma_1\gamma_2}$. Analogously, proceeding from here by means of the items $\beta^{0'}_{qajkr\gamma_1\gamma_2}$, $\beta^{i}_{qajkr\gamma_1\gamma_2}$ for $0 \leq i \leq |\gamma_2|$, and $\beta^{\sim}_{qajkr\gamma_1\gamma_2}$, the operation of the 2nd CA-counter is simulated. Merging with $\psi_{qajkr\gamma_1\gamma_2}$, this procedure results in an expression, the first chain of which displays category `r` (which corresponds to the CA being in state r), first followed by the "true" number of consecutive chains displaying `-2`, then by the "true" number of consecutive chains displaying `-1`, and finally by a chain which displays $\mathtt{-\#}_\Sigma$, and contains as alphabetic material the prefix of the CA-input string recognized so far by the 2-CA simulated. Relying on this, and recalling that, in particular, (SPIC.mo) holds, it is rather easy to verify that a derivation ends in a single-chain expression of the form $\langle s \cdot \mathtt{c}, - \rangle$ for some $s \in \Sigma^*$ and $\cdot \in \{::, :\}$ if, and only if $s \in L(M)$.

Constructing the $\text{MG}^{+\text{perc}}$ G_2 (and arguing that it does its job) works much the same as for the $\text{MG}^{\text{-SMC},+\text{SPIC}}$ G_1, and turns out to be even somewhat more straightforward, since both the "α-" and "β-part" can be reduced to just two alternative chains. This is due to the fact that we can simulate the correct behavior of the i-th CA-counter, $i = 1, 2$, in terms of feature percolation: all **i**-instances currently belonging to the i-th counter appear as one string of `-i`-instances within one chain representing the counter. Because of this, each CA transition is simulated by just four applications of *merge*, and up to three applications of $move^{+\text{perc}}_{\otimes}$—one application of $move^{+\text{perc}}_{\otimes}$ is to properly order the previously parsed word with the currently scanned symbol, and each of the other two are to attract the chain with the appropriate licensee features for percolation (and thus happen only when such chains exist, cf. next paragraph including $(*_2)$ and footnote 10).

An $\mathrm{MG}^{+\mathrm{perc}}$-expression representing an instantaneous configuration $\langle q, n_1, n_2, t\rangle$, derived from an initial configuration $\langle q_0, 0, 0, st\rangle$ in a 2-CA looks like the following, where $(\texttt{-i})^n$, $i = 1, 2$, is the string consisting of n instances of $\texttt{-i}$:[10]

$(*_2)$ $\quad \langle\, \langle \epsilon : \texttt{q}, -\rangle, \langle \epsilon : (\texttt{-2})^{n_2}, \texttt{comp}\rangle, \langle \epsilon : (\texttt{-1})^{n_1}, \texttt{comp}\rangle, \langle s : \texttt{-\#}_\Sigma, \texttt{comp}\rangle \,\rangle$

Construction 2 Let $G_2 = \langle \Sigma, Syn, \{::, :\}, Lex_2, \Omega, c\rangle$ be the $\mathrm{MG}^{+\mathrm{perc}}$ with $L(G_2) = L(M)$ such that Lex_2 contains exactly the items below:

$\boldsymbol{\phi_0} \;=\; \langle\, \boldsymbol{\epsilon} :: \texttt{q}_\mathbf{0}\texttt{-\#}_\mathbf{\Sigma}\,,\, -\rangle$

For all $q, r \in Q, a \in \Sigma_\epsilon, j \in \{\mathbf{1}, \#_\mathbf{1}\}, k \in \{\mathbf{2}, \#_\mathbf{2}\}, \gamma_1 \in \mathbf{1}^*, \gamma_2 \in \mathbf{2}^*$ such that $\langle r, \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, j, k\rangle)$

$\boldsymbol{\chi_{qajkr\gamma_1\gamma_2}} \;=\; \langle\, \boldsymbol{a} :: \texttt{=q +\#}_\mathbf{\Sigma}\ 0_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ \texttt{-\#}_\mathbf{\Sigma}\,,\, -\rangle$

If $j = \#_1$ then

$\boldsymbol{\alpha_{qajkr\gamma_1\gamma_2}} \;=\; \langle \boldsymbol{\epsilon} :: \texttt{=}0_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ 1_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ (\texttt{-1})^{|\gamma_1|}\,,\, -\rangle$

else

$\boldsymbol{\alpha'_{qajkr\gamma_1\gamma_2}} \;=\; \langle \boldsymbol{\epsilon} :: \texttt{=}0_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ \widehat{+1}\ 1_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ (\texttt{-1})^{|\gamma_1|}\,,\, -\rangle$

If $k = \#_2$ then

$\boldsymbol{\beta_{qajkr\gamma_1\gamma_2}} \;=\; \langle\, \boldsymbol{\epsilon} :: \texttt{=}1_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ 2_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ (\texttt{-2})^{|\gamma_2|}\,,\, -\rangle$

else

$\boldsymbol{\beta'_{qajkr\gamma_1\gamma_2}} \;=\; \langle\, \boldsymbol{\epsilon} :: \texttt{=}1_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ \widehat{+2}\ 2_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ (\texttt{-2})^{|\gamma_2|}\,,\, -\rangle$

$\boldsymbol{\psi_{qajkr\gamma_1\gamma_2}} \;=\; \langle\, \boldsymbol{\epsilon} :: \texttt{=}2_{\boldsymbol{qajkr\gamma_1\gamma_2}}\ \texttt{r}\,,\, -\rangle$

For each $q \in Q_f$,

$\boldsymbol{\phi_q} \;=\; \langle\, \boldsymbol{\epsilon} :: \texttt{=q +\#}_\mathbf{\Sigma}\ \texttt{c}\,,\, -\rangle$

Again, a derivation begins either with with some $\chi_{q_0ajkr\gamma_1\gamma_2}$ selecting ϕ_0, or with ϕ_{q_0} doing so in case $q_0 \in Q_f$ (implying that ϵ belongs to $L(M)$). As in the case of G_1, in general, when—within a derivation—an expression is selected by a lexical item ϕ_q for some $q \in Q_f$, or a lexical item $\chi_{qajkr\gamma_1\gamma_2}$ for some $r \in Q$, $a \in \Sigma_\epsilon$, $j \in \{\mathbf{1}, \#_\mathbf{1}\}$, $k \in \{\mathbf{2}, \#_\mathbf{2}\}$, $\gamma_1 \in \mathbf{1}^*$ and $\gamma_2 \in \mathbf{2}^*$, the selected expression corresponds to an instantaneous configuration derived by the 2-CA from an initial configuration. In both cases $move_\otimes^{+\mathrm{perc}}$ applies to the resulting expression, checking an instance of $\texttt{-\#}_\Sigma$. In the former case this ends in a complete expression, if there is no chain left displaying an unchecked licensee feature. Otherwise further derivation steps are blocked. In the latter case G_2 begins simulating the CA's application of δ to $\langle q, a, j, k\rangle$ with outcome $\langle r, \gamma_1, \gamma_2\rangle$ w.r.t. some matching instantaneous configuration: after having moved by means of $move_\otimes^{+\mathrm{perc}}$ the already recognized prefix to the

[10] When $n_i = 0$, the respective chain is not present in the expression.

front of the newly scanned instance of a, the derivation continues by merging an α-expression—note that if $\alpha_{qajkr\gamma_1\gamma_2}$ is merged with an expression which itself contains a chain already in possession of -1 features, the SMC will disallow further operations on the resulting expression (in essence, crashing the derivation).[11] This ensures that only those derivations involving $\alpha_{qajk\gamma_1\gamma_2 r}$ succeed, in which it combines with an expression which is completely devoid of -1 features at the point of merger. If $\alpha'_{qajkr\gamma_1\gamma_2}$ is merged instead, then the chain containing the -1 features will move, and, as per the definition of $move_{\otimes}^{+\mathrm{perc}}$, percolate its features (modulo the one checked) to the initial chain. In either case, the initial chain comes to host all of the expression's -1 features. We then merge a β expression, where there transpires something similar. The simulation of the 2-CA transition is complete once $\psi_{qajkr\gamma_1\gamma_2}$ is merged, resulting in an expression which must once again be selected by some ϕ_r or $\chi_{rajk\gamma_1\gamma_2 s}$.

Note that, involving feature percolation, G_2 only creates chains in which all licensee instances result from the same licensee, i.e., either -1 or -2. Thus, G_2 is defined completely independently of the percolation function $\otimes$ from $Syn^* \times Syn^*$ to Syn^* underlying $move_{\otimes}^{+\mathrm{perc}}$.

8.5 Conclusion

We reviewed two apparently unrelated extensions to minimalist grammars, and showed that both of them can derive arbitrary r.e. sets of strings. Moreover, much the same construction sufficed to show this for each variant of MGs presented herein. This highlights that at least a subpart of each of these extensions have similar strong generative capacities. How similar these variants are is a matter left for another time. We note here only that while the 2-CA simulation given for $\mathrm{MG}^{\text{-SMC},+\mathrm{SPIC}}$ extends straightforwardly to one of a *queue automaton*, no similarly straightforward extension exists for the $\mathrm{MG}^{+\mathrm{perc}}$, as we were able to nullify our ignorance of the function $\otimes$ only by (in effect) reducing its domain to strings over a single alphabetic symbol.

References

Gärtner, Hans-Martin and Jens Michaelis. 2005. A note on the complexity of constraint interaction. Locality conditions and minimalist grammars.

[11] This is true so long as γ_1 is non-zero. To keep the presentation simple, we impose the condition on δ that for $\langle q', \gamma_1, \gamma_2\rangle \in \delta(\langle q, a, g_1, g_2\rangle)$, $\gamma_i = \epsilon$ entails that $g_i \neq \#_{\mathbf{i}}$. That this is a normal form can be seen by the fact that we can eliminate a 'forbidden' transition $\langle q', \epsilon, \gamma_2\rangle \in \delta(\langle q, a, \#_{\mathbf{1}}, g_2\rangle)$ by the pair of new transitions $\langle q'', \mathbf{1}, \gamma_2\rangle \in \delta(\langle q, a, \#_{\mathbf{1}}, g_2\rangle)$ and $\langle q', \epsilon, \gamma_2\rangle \in \delta(\langle q'', \epsilon, \mathbf{1}, g_2\rangle)$, where q'' is a new state not in Q. Analogously, we can eliminate a transition $\langle q', \gamma_1, \epsilon\rangle \in \delta(\langle q, a, g_1, \#_{\mathbf{2}}\rangle)$.

In P. Blache, E. P. Stabler, J. Busquets, and R. Moot, eds., *Logical Aspects of Computational Linguistics (LACL '05)*, Lecture Notes in Artificial Intelligence Vol. 3492, pages 114–130. Berlin, Heidelberg: Springer.

Hopcroft, John E. and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley.

Kobele, Gregory M. forthcoming. Features moving madly. *Research on Language and Computation* Draft version available at `http://www.linguistics.ucla.edu/people/grads/kobele/papers.htm`.

Lambek, Joachim. 1961. How to program an (infinite) abacus. *Canadian Mathematical Bulletin* 4:295–302.

Michaelis, Jens. 2005. An additional observation on strict derivational minimalism. In *FG-MoL 2005. The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language,* Edinburgh. This volume.

Ross, John R. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA. Appeared as *Infinite Syntax*. Ablex, Norwood, NJ, 1986.

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

Stabler, Edward P. 1999. Remnant movement and complexity. In G. Bouma, G. M. Kruijff, E. Hinrichs, and R. T. Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299–326. Stanford, CA: CSLI Publications.

Stabler, Edward P. and Edward L. Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science* 293:345–363.