Journal of Language Modelling

VOLUME 11 ISSUE 1
JUNE 2023

Institute of Computer Science
Polish Academy of Sciences
Warsaw

Journal of Language Modelling

VOLUME 11 ISSUE 1
JUNE 2023

Articles

JOURNAL OF
LANGUAGE MODELLING

ISSN 2299-8470 (electronic version)
ISSN 2299-856X (printed version)
http://jlm.ipipan.waw.pl/

Published by IPI PAN
Institute of Computer Science, Polish Academy of Sciences
ul. Jana Kazimierza 5, 01-248 Warszawa, Poland

Circulation: 50 + print on demand

Layout designed by Adam Twardoch.
Typeset in XeLaTeX using the typefaces: *Playfair*
by Claus Eggers Sørensen, *Charis SIL* by SIL International,
JLM monogram by Łukasz Dziedzic.

On regular copying languages

Yang Wang and Tim Hunter
University of California, Los Angeles

ABSTRACT

This paper proposes a formal model of regular languages enriched with unbounded copying. We augment finite-state machinery with the ability to recognize copied strings by adding an unbounded memory buffer with a restricted form of first-in-first-out storage. The newly introduced computational device, finite-state buffered machines (FS-BMs), characterizes the class of regular languages and languages derived from them through a primitive copying operation. We name this language class *regular copying languages* (RCLs). We prove a pumping lemma and examine the closure properties of this language class. As suggested by previous literature (Gazdar and Pullum 1985, p.278), regular copying languages should approach the correct characterization of natural language word sets.

Keywords: reduplication, copying, finite-state machinery, queue automata

INTRODUCTION 1

The aim of this paper is to introduce a formal model of possible natural language word forms which is restrictive enough to rule out many unattested patterns, but still expressive enough to allow for reduplication. Among the well-known existing classes of formal languages, there is a tension between these two goals. The overwhelming majority of attested phonological patterns fall within the finite-state class (Kaplan and Kay 1994), and perhaps within even more restrictive subclasses (Heinz 2007). Reduplication is the striking exception to this generalization. But at present, if we look for alternatives to the finite-state characterization which are powerful enough to express reduplication, we only find classes of formal languages which additionally allow a wide variety of unattested patterns – for example, nesting/mirror-image patterns, or arbitrary cross-serial dependency patterns significantly more general than reduplication itself. This gives us no way to retain the finite-state characterization's (apparently correct) prediction that mirror-image patterns and so on will be unattested, while avoiding the (apparently incorrect) prediction that reduplication will be unattested.

Jäger and Rogers (2012) review other cases where natural language generalizations do not appear to correspond neatly to degrees of complexity as defined by the formalisms of the classical Chomsky Hierarchy, and the "refinements" of the hierarchy that these findings have prompted. In the case of natural language syntax, for example, it is widely accepted that context-free grammars are insufficiently expressive (Huybregts 1984; Shieber 1985; Culy 1985); but the next level up on the classical hierarchy, context-sensitive grammars, are far too expressive to be a plausible characterization of possible natural languages. This situation prompted the development of many *mildly context-sensitive* formalisms (Joshi 1985; Kallmeyer 2010), whose generative capacity sits in between the context-free and context-sensitive levels. Another "mismatch" has been observed in phonology, where even the lowest level of the classical hierarchy, the finite-state languages, has been argued to be insufficiently restrictive. To address this, a number of researchers have developed *sub-regular* formalisms (e.g., Heinz *et al.* 2011; Chandlee 2014; Heinz 2018).

In this paper, the situation we are addressing is slightly less straightforward than the two mismatches just mentioned. The development of sub-regular formalisms was a response to a perception that *all* the levels of the classical hierarchy were too powerful. The mildly context-sensitive formalisms address the fact that, with regard to syntax, each of the classical levels is either too weak (finite-state, context-free) or too powerful (context-sensitive, recursively enumerable). The situation we address in this paper, in contrast, is one where the classical context-free class is both too powerful in some ways (since it allows mirror-image patterns) and too restrictive in other ways (since it disallows reduplication). We, therefore, seek a formalism that *cuts across* the levels of the classical hierarchy, rather than one which adds a level that sits within the existing hierarchical relationships.

We introduce *finite-state buffered machines (FSBMs)* as a step towards solving this problem. The idea is to preserve as much as possible of the restrictiveness of the finite-state class and add just what is necessary to generate copying patterns. FSBMs include unbounded memory in the form of a first-in-first-out buffer, but the use of this memory is restricted in two important ways. First, this memory buffer uses the alphabet of surface symbols, rather than a separate alphabet like the stack alphabet of a pushdown automaton (PDA). Second, the allowable ways of interacting with this memory buffer are closely tied to the surface string being generated: the only storage operation adds a copy of the current surface symbol to the memory buffer, and the only retrieval operation empties the entire memory buffer and adds its contents to the generated string. For example, in computing a string of the form $urrv$, an FSBM will proceed through three phases corresponding to the sub-strings u, r and v, much like a standard finite-state machine generating the string urv. But throughout the middle phase, a copy of each surface symbol of r will be stored in the FSBM's memory buffer, and at the transition from this middle phase to the third phase the buffer will be emptied and its contents appended to the computed string; thus ur has r appended to it, before the machine proceeds to compute the v portion in the third phase.

In Section 2 we discuss the computational challenge posed by reduplication in more detail, and outline the ways our approach differs from a number of other attempts to enrich otherwise restrictive formalisms with copying mechanisms. We present FSBMs in full in Section 3, give a pumping lemma in Section 4, and explore the mathematical properties of the generated class of languages in Section 5. Section 6 discusses some remaining issues, including various kinds of non-canonical reduplication, and a formal distinction between what we will call *symbol-oriented* generative mechanisms (such as string-copying) and the better-known mechanisms underlying the classical Chomsky Hierarchy. Section 7 concludes the paper.

2 BACKGROUND

Section 2.1 outlines the important empirical properties of reduplication that make it a poor fit to the classical Chomsky Hierarchy; in particular, we aim to show that an appropriate characterization of possible natural language word forms should include the pattern ww, for unboundedly many strings w, but not ww^R, where w^R is the reverse of w. Section 2.2 reviews various modifications to classical automata, like our proposal, that incorporate some form of unbounded queue-like memory. In Section 2.3 we discuss other modifications to finite-state automata that were motivated by reduplication, but do not accommodate the crucial property of unboundedness.

2.1 *The puzzle of reduplication*

2.1.1 Reduplication in natural languages

Reduplication, creating identity within word forms, is common cross-linguistically. Table 1 provides illustrative examples. Dyirbal exhibits *total reduplication*, with the plural form of a nominal comprised of two perfect copies of the full singular stem; whereas *partial reduplication* is exemplified in Agta, where plural forms only copy the first CVC sequence of the corresponding singular forms (Healey 1960; Marantz

Table 1: Total reduplication:Dyirbal plurals (top); partial reduplication:Agta plurals (bottom)

Total reduplication: Dyirbal plurals (Dixon 1972, p. 242; Inkelas 2008, p. 352)			
Singular	*Gloss*	*Plural*	*Gloss*
midi	‘little, small’	midi-midi	‘lots of little ones’
gulgiɽi	‘prettily painted men’	gulgiɽi-gulgiɽi	‘lots of prettily painted men’

Partial reduplication: Agta plurals (Healey 1960, p.7)			
Singular	*Gloss*	*Plural*	*Gloss*
labáng	‘patch’	lab-labáng	‘patches’
takki	‘leg’	tak-takki	‘legs’

1982).[1] In the sample reported by Rubino (2013) and further surveyed in Dolatian and Heinz (2020), 313 out of 368 natural languages exhibit productive reduplication, of which 35 languages have total reduplication but not partial reduplication. Moravcsik (1978, p. 328) hypothesized that all languages with attested partial reduplication would also use total reduplication.

By comparison, context-free palindrome patterns are rare in phonology and morphology (Marantz 1982) and appear to be confined to language games (Bagemihl 1989; Gil 1996), whose phonological status is unclear. Figure 1 illustrates the important difference between Dyirbal total reduplication (*midi-midi*) and the logically-possible but unattested palindrome pattern (*midi-idim*).

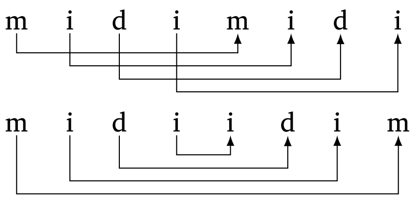


Figure 1:
Crossing dependencies in Dyirbal total reduplication *midi-midi* (top) versus nesting dependencies in unattested string reversal *midi-idim* (bottom)

From the perspective of a computational analysis, it will be important to establish that (at least some) reduplication constructions are *unbounded*, in the sense that they are usefully modeled by string-sets of the form $\{ww \mid w \in S\}$ for some infinite set S. A partial reduplication construction, such as the Agta case above where an initial CVC sequence is copied, is obviously not unbounded in this sense, since – assuming a finite alphabet – there are only finitely-many CVC sequences (Chandlee and Heinz 2012).[2] But as observed by Clark and Yoshinaka (2014) and Chandlee (2017), even amongst total reduplication constructions we must take care to distinguish between unrestricted, productive total reduplication (which is unbounded in the

[1] For clarity, we adopt a simplistic analysis here. When the bases start with a vowel, Agta copies the first VC sequence, as in *uffu* 'thigh' and *uf-uffu* 'thighs'. Thus, a more complete generalization is that Agta copies a (C)VC sequence.

[2] In principle, a reduplication operation which copied, for example, *half* of the relevant stem, would be a case of unbounded copying in this sense that would likely nonetheless be described as partial reduplication. But the attested cases of partial reduplication appear to all involve templates that do not depend on the length of the base (see the most frequent attested shapes in Moravcsik 1978; Rubino 2005; Dolatian and Heinz 2020), like the Agta examples above.

Table 2: Reduplication and bounded/unbounded copying

	Restricted to lexemes (not productive)	Not restricted to lexemes (productive)
Partial Reduplication	bounded	bounded
Total Reduplication	bounded	unbounded

relevant sense) and total reduplication on a *finite* set of bases. For example, it is important to establish that *midi-midi* is not simply part of a collection $\{ww \mid w \in S\}$ where S is some finite memorized set (e.g. the set of all lexemes of a particular category); in such a case, the resulting set of reduplicated forms would itself be finite, and therefore within most familiar language classes. Table 2 illustrates the relationship between productivity, the partial/total distinction, and unboundedness.

A famous case of reduplication that is unbounded in the relevant sense is the Bambara 'Noun *o* Noun' construction (Culy 1985). For example, the stem **wulu** *dog* can be copied to form **wulu o wulu** *whichever dog*. The important point about productivity comes from the interaction of this reduplication with the agentive *la* construction, illustrated in (1) (Culy 1985, pp. 346–347).

(1) a. wulu + nyini + la = wulunyinina
dog search for
"one who searches for dogs", i.e., "dog searcher"

b. wulu + filè + la = wulufilèla
dog watch
"one who watches dogs", i.e., "dog watcher"

This agentive construction itself is recursive, in the sense that it can build on its own outputs, as illustrated in (2); and the outputs of the agentive construction, including the recursively-formed ones, can be used in the 'Noun *o* Noun' reduplicative construction, as illustrated in (3).

(2) a. wulunyinina + nyini + la = wulunyininanyinina
dog searcher search for
"one who searches for dog searchers"

b. wulunyinina + filè + la = wulunyininafilèla
dog searcher watch
"one who watches dog searchers"

(3) a. wulunyinina **o** wulunyinina
dog searcher dog searcher
(1a) (1a)
"whichever dog searcher"

b. wulufilèla **o** wulufilèla
dog watcher dog watcher
(1b) (1b)
"whichever dog watcher"

c. wulunyininanyinina **o** wulunyininanyinina
(2a) (2a)
"whichever one who searches for dog searchers"

d. wulunyininafilèla **o** wulunyininafilèla
(2b) (2b)
"whichever one who watches dog searchers"

The set of all outputs of this reduplication process can therefore naturally be thought of as taking the form $\{ww \mid w \in S\}$, where S is the *infinite* set of nouns, including outputs of the agentive construction.

Further evidence that reduplication is productive in this sense comes from its applicability to borrowed words: Yuko (2001, p. 68) cites the totally-reduplicated plurals *teknik-teknik* 'techniques' and *teknologi-teknologi* 'technologies' attested in Malay, for example. Similarly, the code-switching data from Tagalog in (4) (Waksler 1999), shows the English word *swimming* being (partially) reduplicated.

(4) Saan si Jason? Nag-SWI-SWIMMING siya.
where DET Jason PRESENT-REDUP-SWIMMING he
'Where is Jason? He's swimming.'

In addition, in a few experiments that, either directly or indirectly, study the learnability of surface identity-based patterns, copying appears to be salient and easy to learn. The famous study by Marcus *et al.* (1999) shows that infants can detect and habituate to different identity-based patterns: ABA vs. ABB and AAB vs. ABB, where A and B are CV syllables. Crucially, the particular syllables used at test time were distinct from any seen during training.

Evidence that reduplication/copying (ww) patterns have an importantly different status than reversal (ww^R) patterns – converging with the typological absence of reversal patterns noted above – comes from one recent artificial grammar learning study (AGL) (Moreton *et al.* 2021). In this experiment, adult learners were trained to identify either a reduplication or a syllable reversal pattern. Participants were also asked to explicitly state the rule they had learned (if they could). Participants in the reduplication group showed final above-chance performance whether they could state the rule or not. However, in the syllable-reversal condition, only participants who could also correctly state the rule showed final above-chance performance; this suggests that learning the reversal pattern relied on some degree of explicit/conscious reasoning that the copying pattern did not. In further support of this distinction, correct syllable-reversal responses showed longer reaction times than correct copying responses. In a second variant of this experiment, the training phase was replaced with explicit instruction on the rule to apply; participants in the reduplication group still showed shorter reaction times. These results suggest that, to the extent that reversal patterns can be learned or applied at all, this is achieved more by conscious application of a rule rather than unconscious linguistic knowledge, in contrast to reduplication.

A significant aspect of this AGL study is that the stimuli used were auditory, "purely phonological", "meaningless" strings (Moreton *et al.* 2021, p. 9), chunks of which are identical. We take this to indicate that cognitively representable reduplication or reduplication-like patterns need not be realizations of meaning-changing operations: identity between sub-strings can contribute to the phonotactic well-formedness of a surface form, in ways that can be separated from any morphological paradigms in which that surface form appears. This aligns with the general tendency that Zuraw (2002) called *aggressive reduplication*: human phonological grammar is sensitive to output forms with self-similar subparts, regardless of morphosyntactic or semantic cues. Such sensitivity is formalized as the constraint REDUP which requires string-to-string correspondence by coupling sub-strings together.[3]

[3] Direct evidence supporting aggressive reduplication comes from pseudo-reduplication. A pseudo-reduplicated word has one portion identical to another portion. But the decomposed form cannot stand alone and thus does not bear proper morphosyntactic or semantic information. Zuraw (2002) studied the transparency of phonological rule application within pseudo-reduplicated words

Inadequacy of familiar language classes 2.1.2

Having established that the formal pattern *ww*, for unboundedly many strings *w*, is a reasonable model for reduplication, we can ask where this falls on the hierarchy of familiar language classes. The original Chomsky Hierarchy, shown in solid lines in Figure 2, classifies the *ww* pattern as properly context-sensitive; it is also included in the more recent *mildly context-sensitive* subclass (MCS; Joshi 1985; Stabler 2004), shown with a dashed line. This creates a puzzle with two parts.

The first part of the puzzle comes from the fact that reduplication is a counter-example to the otherwise overwhelming generalization that attested phonological and morphological patterns are regular. Aside from reduplication, it is very natural to hypothesize that the set of possible natural language word forms is regular (or even sub-regular). This is why the distinction above between bounded and unbounded copying is crucial: one way to save the regular hypothesis would be to demonstrate that reduplication is bounded, which would place it in the class of *finite* languages which is properly included in all of the classes shown in Figure 2. For example, Figure 3 shows a finite state automaton that successfully recognizes $\{ww \mid w \in S\}$ with a finite $S = \{aaa, aba, aab, abb, baa, bba, bab, bbb\}$. The finiteness makes it possible to essentially just memorize the desired list of surface forms.[4]

The second part of the puzzle comes from considering the classes in Figure 2 that do include *ww*. The most restrictive of these is the

---

in Tagalog loan words. For example, stem-final mid vowels in Tagalog usually raise to high vowels when suffixed, as in [kal**o**s] *grain leveller* but [kal**u**s-in] *to use a grain leveller on*. However, within English and Spanish loans, mid vowel raising is less frequently applied when a preceding mid vowel is present: /tod**o** + in/ *to include all* has /todo/ realized as [tod**o**] but not [tod**u**]. The hypothesized motivation is that speakers preserve sub-string similarity between /to/ and /do/. A recent MEG study on visual inputs (Wray *et al.* 2022) further supports the reduplication-like representation for those pseudo-reduplicated words that fail to undergo a process due to similarity preservation.

[4] Of course one might also dispute whether Figure 3, with its explosion in the number of states (Roark and Sproat 2007; Dolatian and Heinz 2020), represents a linguistically adequate model of even a bounded copying construction. The distinction between arguing that Figure 3 is linguistically inadequate and arguing that copying is unbounded is subtle (Savitch 1993).

Figure 2:
Familiar language classes

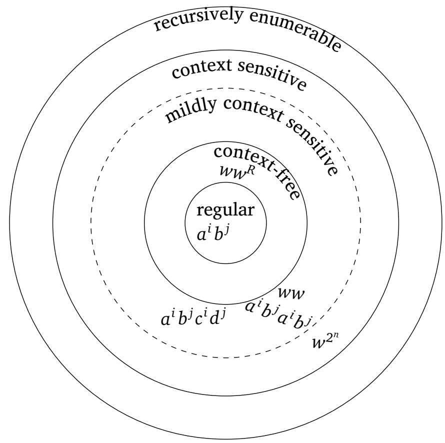
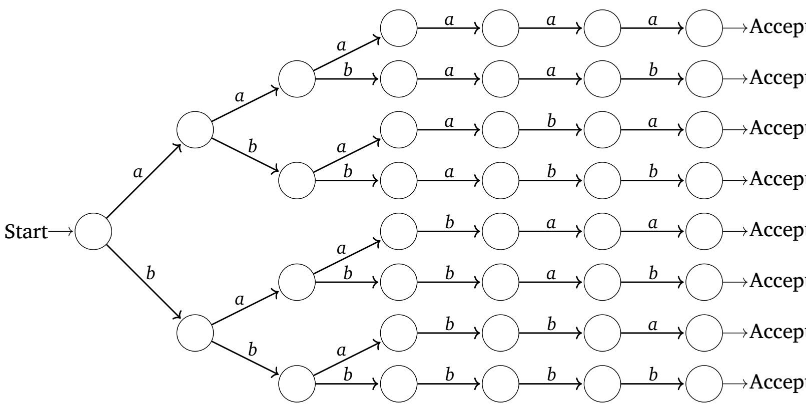


Figure 3:
A finite-state machine for whole-base copying with the set of bases = $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$



mildly context-sensitive class. This is not a good fit with natural language word forms because it also includes the ww^R pattern, which is unattested as discussed above; more generally, it includes *nesting* patterns as well as *crossing* patterns (recall Figure 1). But the problem is slightly more subtle than the simple distinction between nesting and crossing suggests: the MCS class includes very general crossing patterns such as $a^i b^j c^i d^j$, but reduplication represents a special case where the cross-serially dependent elements are identical symbols. MCS grammars are motivated by natural language syntax, where the more general kind of crossing patterns appear to be necessary[5] –

[5] And nesting patterns are at least as common as crossing patterns.

	Linear/regular	Nested	Cross-serial
Morphology and Phonology	✓	✗	✓ restricted to symbol identity
Syntax	✓	✓	✓

Figure 4: Attested types of dependencies in different language modules

the influential paper by Shieber (1985) on Swiss German appeals to exactly the aforementioned example $a^i b^j c^i d^j$ – but for the purposes of morphophonology, there is reason to distinguish crossing patterns that involve surface symbol identity (e.g. *ww* and $a^i b^j a^i b^j$) from those that do not. This situation is summarized in Figure 4. We return to the distinction between formalisms where symbol identity plays a role and those where it does not in Section 6.3.

2.2 *Language classes motivated by reduplication and queue automata*

In response to essentially the puzzle introduced above, Gazdar and Pullum (1985, p.287) made the remark that

> We do not know whether there exists an independent characterization of the class of languages that includes the regular sets and languages derivable from them through reduplication, or what the time complexity of that class might be, but it currently looks as if this class might be relevant to the characterization of NL [natural language] word-sets.

One such proposal is offered by Manaster-Ramer (1986, p.87), who introduces the idea – closely related to that underlying our own proposal below – as follows:[6]

> Rather than grudgingly clambering up the Chomsky Hierarchy towards Context-sensitive Grammars, we should consider

[6] Taken literally, this quotation seems to lead in the direction of unrestricted queue automata which are known to be equivalent to Turing machines. What Manaster-Ramer actually proposes is significantly more restricted. Also, see Kutrib *et al.* (2018) for a more complete review of the history of queue automata and investigations on restricted versions that computer scientists have conducted.

> going back down to Regular Grammars and striking out in a different direction. The simplest alternative proposal is a class of grammars which intuitively have the same relation to queues that CFGs have to stacks.

The *Context-free Queue Grammars* (CFQGs) that Manaster-Ramer proposes enriches the rules of a regular grammar (specifically in the form of right-linear rewrite rules) with the additional capacity to either (i) write a terminal symbol to a separate queue-based memory, or (ii) clear the queue and append its current contents to the output string. This is implemented in the form of a rewrite-rule system that effectively maintains two strings: rather than simply uX as in a standard right-linear grammar, uXv at an intermediate stage of a derivation represents having generated u as the output string which will grow on its right via rewrites of the nonterminal X, with v as the current queue contents.

There are significant similarities between CFQGs and the FSBM formalism that we introduce in this paper. Manaster-Ramer illustrates CFQGs via an example that generates $\{ww \mid w \in \{a,b\}^*\}$, and conjectures that they cannot generate the corresponding mirror-image (ww^R) language, but there is no careful exploration of the formalism's capacity or limitations. Also, it is clear that CFQGs can generate more general crossing patterns such as $a^i b^j c^i d^j$ along with reduplication-like patterns, so FSBMs are more restricted in at least this (linguistically well-motivated) respect.

Along similar lines to Manaster-Ramer's proposal, Savitch (1989) introduced *Reduplication PDAs* (RPDAs), which are pushdown automata (PDAs) augmented with the ability to match reduplicated strings by using a portion of the stack as a queue. RPDAs are more powerful than CFQGs, since the language class they define properly includes context-free languages, so they do not exclude nesting/mirror-image patterns. This aligns with the fact that the motivations Savitch discusses mainly involve crossing patterns found in syntax rather than identity-based reduplication which is our focus here. But the technical formulation of RPDAs has much in common with that of FSBMs below.

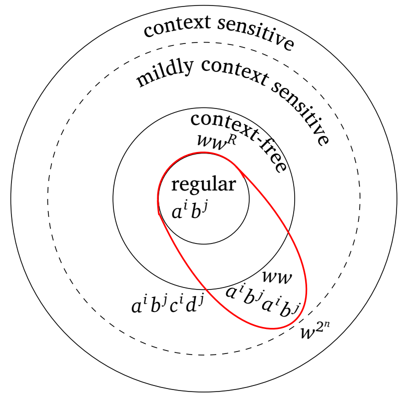
Finally, *Memory Automata* (MFAs; Schmid 2016; Freydenberger and Schmid 2019) introduce a kind of automata that is particularly similar to FSBMs. MFAs augment classical FSAs with a finite number of memory cells; each memory cell can store an unboundedly long sub-string of input, which can be matched against future input when it is recalled. The full class of MFAs can generate languages such as $\{a^i \mid i \text{ is not prime}\}$ (Câmpeanu *et al.* 2003, p.1013) and $\{a^{4^i} \mid i \geq 1\}$ (Freydenberger and Schmid 2019, p.21), and is therefore much too powerful to be suitable as a model for natural languages.[7] But these unusually complex languages all rely on either interactions between distinct memory cells, or the ability to recall a particular string from a memory cell more than once. The FSBM formalism that we introduce corresponds closely to a restricted version of MFAs where there is only one memory cell, and its contents are erased when recalled.

To summarize: our goal is to identify a formalism whose class of languages aligns with Gazdar and Pullum's motivating quotation above; RPDAs do not match this description because they extend upwards from the context-free languages, rather than the regular languages; CFQGs and MFAs do adopt the regular languages as the starting point, but extend too far and therefore overshoot the mark in different ways.

This paper introduces FSBMs as a way of examining what minimal changes can be brought to regular languages to include string-sets with two copies of the same sub-strings, while excluding some typologically unattested context-free patterns, such as reversals, and crossing dependencies other than reduplication. We name the resulting class of languages *regular copying languages* (RCLs). The intended relation of this language class to other existing language classes is shown in Figure 5.

[7] MFAs were introduced to provide an automaton-based characterization of the languages generated by regular expressions extended with back-references (Câmpeanu *et al.* 2002; Câmpeanu *et al.* 2003; Carle and Narendran 2009). There are some differences between the various definitions of these *extended regular expressions* in the literature; see Freydenberger and Schmid (2019, pp. 36–37) for discussion. We would like to thank an anonymous reviewer for pointing out the relevant research on extended regular expressions, which in turn led us to the literature on MFAs.

Figure 5:
The class of regular copying languages (oval shape) in the classical Chomsky Hierarchy



2.3 *Other computational models motivated by reduplication*

Now we review other computational models motivated by reduplication, which can be categorized into two groups: those that limit attention to bounded copying (Section 2.3.1) and those that consider transductions/mappings (Section 2.3.2).

2.3.1 Compact representations of bounded copying

The first line of work aims to improve upon the inelegant memorization strategy exemplified in Figure 3, while retaining the limitation to bounded copying. For example, Cohen-Sygal and Wintner (2006) introduce *finite-state registered automata* (FSRAs), which augment standard FSAs with finitely many memory registers. This allows for a more space-efficient representation of copying patterns, without the duplicating paths of Figure 3, by storing the symbols to be matched in registers rather than in the machine's central state. But because the registers themselves provide only a finite amount of additional memory, FSRAs do not extend upon the generative capacity of standard FSAs, and therefore do not accommodate productive total reduplication (i.e. unbounded copying).

An analogous proposal is the *compile-replace* algorithm (Beesley and Karttunen 2000). This run-time technique first maps a lexical item to a regular expression representation for either morphological generation or analysis. Then the desired output is obtained by re-evaluating the output regular expression. Similarly, Walther (2000) added different types of transitions to represent the lexicon: *repeat* (for copying), *skip* (for truncation) and *self-loops* (for infixation). Then, intersecting these enriched lexical items with an FSA encoding language-specific reduplication rules would derive the surface strings. Last but not least, Hulden (2009) introduced an EQ function, a filter on a finite-state transduction which excludes input-output pairs where the output string does not meet a sub-string identity condition. In principle, this idea allows for an unbounded-copying output language such as $\{ww \mid w \in \{\mathrm{a},\mathrm{b}\}^*\}$ to be specified, but in practice, Hulden's implementation restricts attention to cases where the equal sub-strings are bounded in length (p.125).

2-way Deterministic Finite-state Transducers 2.3.2

A finite-state device that computes unbounded copying elegantly and adequately is the *2-way deterministic finite-state transducer* (2-way D-FST) (Dolatian and Heinz 2018a,b, 2019, 2020), which differs from a conventional (1-way) FST in being able to move back and forth on the input.[8] 2-way D-FSTs have been proven to describe string transductions that are MSO-definable (Monadic Second-Order logic; Engelfriet and Hoogeboom 1999) and are equivalent to *streaming string transducers* (Alur and Černý 2010). In these formalisms, reduplication is modeled as a string-to-string *mapping* ($w \mapsto ww$). To avoid the mirror image function ($w \mapsto ww^R$), Dolatian and Heinz (2020) further studied sub-classes of 2-way D-FSTs which cannot output anything during right-to-left passes over the input (cf. *rotating transducers*: Baschenis *et al.* 2017).

The issue addressed in Dolatian and Heinz (2020) is distinct from, but related to, the main concern of this paper: these transducers model reduplication as a function mapping underlying forms to surface forms ($w \mapsto ww$), while this paper aims to characterize only the identical-substrings requirement on the corresponding surface forms (ww). There are at least two reasons to address the string-set problem

[8] 2-way FSTs are still more restricted than Turing machines since they cannot move back and forth on the output tape, only the input tape.

itself rather than considering only mappings between underlying and surface forms.

The first reason is a practical/strategic one, related to the problem of morphological *analysis* (rather than generation): the question of what kinds of transducers can implement the $ww \mapsto w$ mapping required for morphological analysis remains open, since 2-way D-FSTs (unlike standard 1-way FSTs) are not readily invertible as a class (Dolatian and Heinz 2020, p.235). Although we do not directly address the morphological analysis problem here, recognizing the reduplicated ww strings is plausibly an important first step: applying the mapping $ww \mapsto w$ to some string x requires at least *recognizing* whether x belongs to the ww string set.

The second reason stems from a full consideration of the linguistic facts surrounding reduplication: there is evidence supporting meaning-free, non-morphologically-generated reduplication-like structures, as mentioned in the discussion of aggressive reduplication above. This suggests that the phonological grammar involves a *phonotactic* constraint requiring sub-string identity, and the natural formal model for such a constraint is an automaton that generates/accepts the strings satisfying it. A constraint of this sort could play a role in mappings relating underlying forms to surface forms, so we may be missing a generalization if we only model those mappings directly.

3 FINITE-STATE BUFFERED MACHINES

The aim of proposing a new computing device is to add reduplication to FSAs and thereby gain a better understanding of the required computational operations. The new formalism is *finite-state buffered machines* (FSBMs), a summary of which is provided in Section 3.1. For ease of exposition, we introduce the new formalism by first presenting the general case of FSBMs in Section 3.2, along with illustrative examples. A clearer understanding of the formalisms' capacity for copying comes from identifying a subset of FSBMs that we call *complete-path FSBMs*, in Section 3.3; we show that the languages recognized by FSBMs are precisely the languages recognized by complete-path FSBMs in Section 3.4.

FSBM in a nutshell 3.1

FSBMs are two-taped automata with finite-state core control.[9] One tape stores the input, as in normal FSAs; the other serves as an unbounded memory buffer, storing reduplicants temporarily for future string matching. An FSBM can be thought of as an extension to the FSRAs discussed above (Cohen-Sygal and Wintner 2006) but equipped with unbounded memory. FSBMs with a *bounded* buffer would be as expressive as FSRAs, and therefore also standard FSAs.

The interaction of the queue-like buffer with the input is restricted in two important ways. First, the buffer stores symbols from the same alphabet as the input, unlike the stack in a PDA, for example. Second, once one symbol is removed from the buffer, everything else must also be emptied from the buffer before symbols can next be added to it. These restrictions together ensure the machine will not generate string reversals or other non-reduplicative non-regular patterns.

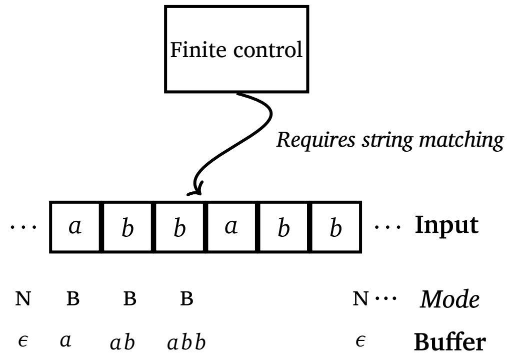
Unlike a standard FSA, an FSBM works with two possible modes: in *normal* (N) mode, M reads symbols and transits between states, functioning as a normal FSA; and in *buffering* (B) mode, besides consuming symbols from the input and taking transitions among states, M adds a copy of just-read symbols to the queue-like buffer. At a specific point, M exits buffering (B) mode, matching the stored string in the buffer against (a portion of) the remaining input. Provided this match succeeds, it switches back to normal (N) mode for another round of computation. Figure 6 provides a schematic diagram showing how the mode of an FSBM alternates when it determines the equality of sub-strings and how the buffer interacts with the input.

As presented here, FSBMs can only compute local reduplication with two adjacent, completely identical copies. They cannot handle non-local reduplication, multiple reduplication, or non-identical copies. We believe the current machinery can serve as the foundation for proposing different variants, and we discuss some potential modifications along these lines in Section 6.1.

Having introduced the important intuitions, we now turn to the formal definition of FSBMs.

[9] The presented model here is a modified version of the proposal of Wang (2021a) and Wang (2021b).

Figure 6:
Mode changes and input-buffer interaction of an FSBM M on ...$abbabb$.... The machine switches to B mode to temporarily store symbols in the queue-like buffer, and then at the point indicated by the arrow it compares the buffer contents against the remaining input. If the two strings match, the buffer is emptied, the matched input sub-string is consumed and the machine switches to N mode



3.2 *Preliminaries and Definitions*

For any finite alphabet Σ of symbols, we use Σ^* to denote the set of all finite strings over Σ. For a string w, $|w|$ denotes its length. ϵ is the null string and thus $|\epsilon| = 0$. We denote string union by '$+$', and denote string concatenation by simple juxtaposition, assuming implicit conversion between symbols and length-one strings where necessary. If $u = vw$, then $v\backslash u = w$; otherwise, $v\backslash u$ is undefined.

DEFINITION 1 *A **Finite-State Buffered Machine** is a 7-tuple*

$$\langle \Sigma, Q, I, F, G, H, \delta \rangle$$

where

- Σ*: a finite set of symbols*
- Q*: a finite set of states*
- $I \subseteq Q$*: initial states*
- $F \subseteq Q$*: final states*
- $G \subseteq Q$*: states where the machine must enter buffering mode*
- $H \subseteq Q - G$*: states requiring string matching*
- δ*:* $Q \times (\Sigma \cup \{\epsilon\}) \times Q$*: transition relation*

The specification of the two sets of special states, G and H, serves to control what portions of a string are copied. To avoid intricacies, G and H are defined to be disjoint. The special case where $G = H = \emptyset$ corresponds to a standard FSA.

DEFINITION 2 *A **configuration** of an FSBM is a four-tuple* $(u, q, v, t) \in \Sigma^* \times Q \times \Sigma^* \times \{\text{N}, \text{B}\}$*, where* u *is the input string;* q *is the current state;* v *is the string in the buffer; and* t *is the machine's current mode.*

DEFINITION 3 *Given an FSBM $M = (\Sigma, Q, I, F, G, H, \delta)$, the relation $\vdash_M$ on configurations is the smallest relation such that, for any $u, v, w \in \Sigma^*$:*

- *For every transition $(q_1, x, q_2) \in \delta$*
  $(xu,\ q_1,\ \epsilon,\ \text{N}) \vdash_M (u,\ q_2,\ \epsilon,\ \text{N})$ if $q_1 \notin G$ and $q_2 \notin H$ $\vdash_{\text{N}}$
  $(xu,\ q_1,\ v,\ \text{B}) \vdash_M (u,\ q_2,\ vx,\ \text{B})$ if $q_1 \notin H$ and $q_2 \notin G$ $\vdash_{\text{B}}$
- *For every $q \in G$*
  $(u,\ q,\ \epsilon,\ \text{N}) \vdash_M (u,\ q,\ \epsilon,\ \text{B})$ $\vdash_{\text{N}\to\text{B}}$
- *For every $q \in H$*
  $(vw,\ q,\ v,\ \text{B}) \vdash_M (w,\ q,\ \epsilon,\ \text{N})$ $\vdash_{\text{B}\to\text{N}}$

Thus, $\vdash_M = \vdash_{\text{N}} \cup \vdash_{\text{B}} \cup \vdash_{\text{N}\to\text{B}} \cup \vdash_{\text{B}\to\text{N}}$. When $D_1 \vdash_M D_2$, we say D_1 ***yields*** *D_2.*

As is standard, $\vdash^*$ denotes the reflexive and transitive closure of $\vdash$, while $\vdash^+$ is the corresponding irreflexive closure.

DEFINITION 4 *A* ***run*** *of M on w is a sequence of configurations $D_0, D_1, D_2 \ldots D_m$ such that*

- $\exists q_0 \in I,\ D_0 = (w, q_0, \epsilon, \text{N})$
- $\exists q_f \in F,\ D_m = (\epsilon, q_f, \epsilon, \text{N})$
- $\forall\, 0 \leq i < m,\ D_i \vdash_M D_{i+1}$

DEFINITION 5 *The language recognized by $M = \langle \Sigma, Q, I, F, G, H, \delta \rangle$, denoted by $L(M)$, is the set of all strings $w \in \Sigma^*$ such that there is a run of M on w. That is, $L(M) = \{w \in \Sigma^* \mid (w, q_0, \epsilon, \text{N}) \vdash_M^* (\epsilon, q_f, \epsilon, \text{N}), q_0 \in I, q_f \in F\}$.*

Notice that we do not impose any notion of determinism on the transitions of an FSBM. We return to some discussion of this point in Section 6.2.

Now, we give examples of FSBMs. In all illustrations, G states are drawn with diamonds and H states are drawn with squares.

Examples: Total reduplication 3.2.1

Figure 7 offers an FSBM M_1 for L_{ww}, with arbitrary strings over the alphabet $\Sigma = \{a, b\}$ as potential bases. The initial state q_1 is also a G state, and the only H state is q_3. The machine stores a copy of string computed in between q_1 and q_3 in the buffer and requires string matching at q_3. Since the states where the machine enters ($q_1 \in G$) and

Figure 7: M_1 with G $= \{q_1\}$ and H $= \{q_3\}$. $L(M_1) = \{ww \mid w \in \{a,b\}^*\}$

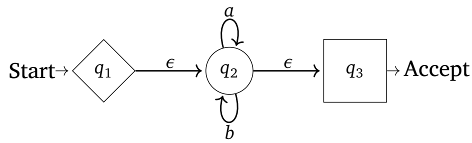
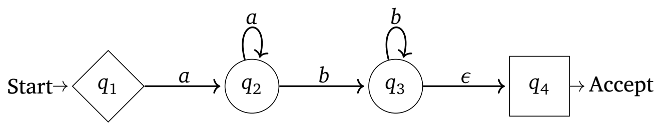


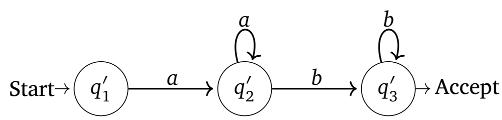
Table 3: M_1 in Figure 7 accepts *abbabb*

	Used arc or *state*	$\vdash$ types	*Configuration* (input, state, buffer, mode)
1.	*N/A*		(*abbabb*, q_1, ϵ, N)
2.	$q_1 \in G$	$\vdash_{N \to B}$	(*abbabb*, q_1, ϵ, B)
3.	(q_1, ϵ, q_2)	$\vdash_B$	(*abbabb*, q_2, ϵ, B)
4.	(q_2, a, q_2)	$\vdash_B$	(*bbabb*, q_2, *a*, B)
5.	(q_2, b, q_2)	$\vdash_B$	(*babb*, q_2, *ab*, B)
6.	(q_2, b, q_2)	$\vdash_B$	(*abb*, q_2, *abb*, B)
7.	(q_2, ϵ, q_3)	$\vdash_B$	(*abb*, q_3, *abb*, B)
8.	$q_3 \in H$	$\vdash_{B \to N}$	(ϵ, q_3, ϵ, N)
		Accept	

Figure 8: One example FSBM and the corresponding FSA for the base language



(a) An FSBM M_2 with G $= \{q_1\}$ and H $= \{q_4\}$; $L(M_2) = \{a^i b^j a^i b^j \mid i, j \geq 1\}$



(b) An FSA M_0; $L(M_0) = \{a^i b^j \mid i, j \geq 1\}$

leaves ($q_3 \in H$) buffering mode are also the initial and final states respectively, this machine will recognize simple total reduplication. Table 3 gives a complete run of M_1 on the string *abbabb*. As in Step 8, the string *abb* in the remaining input is consumed in one step.

For the rest of the illustration, we focus on the FSBM M_2 in Figure 8a. M_2 in Figure 8a recognizes the non-context-free language $\{a^i b^j a^i b^j \mid i, j \geq 1\}$. This language can be viewed as total reduplication added to the regular language $\{a^i b^j \mid i, j \geq 1\}$ (recognized by the

FSA M_0 in Figure 8b). q_1 is an initial state and more importantly a G state, forcing M_2 to enter B at the beginning of any run. Then M_2 in B mode always keeps a copy of consumed symbols until it proceeds to q_4, which is an H state and therefore requires M_2 to stop buffering and check for string identity to empty the buffer. Then, M_2 with a blank buffer can switch to N mode. It eventually ends at q_4, a legal final state. Table 4 shows one possible sequence of configurations of M_2 on *ababb*; this string is rejected because there is no way to reach a valid ending configuration.

Table 4: M_2 in Figure 8a rejects *ababb*

	Used arc or *state*	$\vdash$ types	*Configuration* (input, state, buffer, mode)
1.	*N/A*		(*ababb*, q_1, ϵ, N)
2.	$q_1 \in G$	$\vdash_{N \to B}$	(*ababb*, q_1, ϵ, B)
3.	(q_1, a, q_2)	$\vdash_B$	(*babb*, q_2, a, B)
4.	(q_2, b, q_3)	$\vdash_B$	(*abb*, q_3, ab, B)
5.	(q_3, ϵ, q_4)	$\vdash_B$	(*abb*, q_4, ab, B)
6.	$q_4 \in H$	$\vdash_{B \to N}$	(b, q_4, ϵ, N)
		Reject	

3.2.2 Examples: Partial reduplication

Assuming $\Sigma = \{b, t, k, ng, l, i, a\}$, the FSBM M_3 in Figure 9 serves as a simple model of Agta CVC reduplicated plurals, as illustrated earlier in Table 1. Given the initial state q_1 is in G, M_3 has to enter B mode before it takes any transitions. In B mode, M_3 transits to a plain state q_2, consuming a consonant from the input and keeping it in the buffer. Similarly, M_3 transits to a plain state q_3 and then to q_4. When M_3 first reaches q_4, the buffer would contain a CVC sequence; q_4, an H state, requires M_3 to match this CVC sequence in the buffer with the

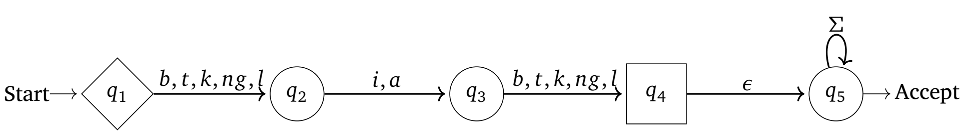


Figure 9: An FSBM M_3 for Agta CVC-reduplicated plurals: G = $\{q_1\}$ and H = $\{q_4\}$

Table 5:
M_3 in Figure 9 accepts *taktakki*

	Used arc	$\vdash$ types	*Configuration*
1.	*N/A*		(*taktakki*, q_1, ϵ, N)
2.	$q_1 \in G$	$\vdash_{\text{N}\rightarrow\text{B}}$	(*taktakki*, q_1, ϵ, B)
3.	(q_1, t, q_2)	$\vdash_{\text{B}}$	(*aktakki*, q_2, *t*, B)
4.	(q_2, a, q_3)	$\vdash_{\text{B}}$	(*ktakki*, q_3, *ta*, B)
5.	(q_3, k, q_4)	$\vdash_{\text{B}}$	(*takki*, q_4, *tak*, B)
6.	$q_4 \in H$	$\vdash_{\text{B}\rightarrow\text{N}}$	(ki, q_4, ϵ, N)
7.	(q_4, ϵ, q_5)	$\vdash_{\text{N}}$	(ki, q_5, ϵ, N)
8.	(q_5, k, q_5)	$\vdash_{\text{N}}$	(i, q_5, ϵ, N)
9.	(q_5, i, q_5)	$\vdash_{\text{N}}$	(ϵ, q_5, ϵ, N)
		Accept	

Table 6:
M_3 in Figure 9 rejects *tiktakki*

	Used arc	$\vdash$ types	*Configuration*
1.	*N/A*		(*tiktakki*, q_1, ϵ, N)
2.	$q_1 \in G$	$\vdash_{\text{N}\rightarrow\text{B}}$	(*tiktakki*, q_1, ϵ, B)
3.	(q_1, t, q_2)	$\vdash_{\text{B}}$	(*iktakki*, q_2, *t*, B)
4.	(q_2, i, q_3)	$\vdash_{\text{B}}$	(*ktakki*, q_3, *ti*, B)
5.	(q_3, k, q_4)	$\vdash_{\text{B}}$	(*takki*, q_4, *tik*, B)
	$q_4 \in H$: *checks for string identity and rejects*		

remaining input. Then, M_3 with a blank buffer can switch to N mode at q_4. It transitions to q_5 to process the rest of the input via the normal loops on q_5. A successful run should end at q_5, the only final state. Table 5 gives a complete run of M_3 on the string *taktakki*. Table 6 illustrates a case where the crucial step of returning from B mode to N mode is not possible, because of the non-matching sub-strings in *tiktakki*; this string is rejected by M_3.

3.3 *The copying mechanism and complete-path FSBMs*

The copying mechanism is realized by four essential components: 1) the unbounded memory buffer, which has queue-like storage; 2) added modalities; 3) added specifications of states requiring the machine to buffer symbols into memory, namely states in G; 4) added specifications of states requiring the machine to empty the buffer by matching sub-strings, namely states in H.

As shown in the definitions of configuration changes and the examples in Section 3.2, the machine must end in N mode to accept an input. There are two possible scenarios for a run to meet this requirement: either never entering B mode or undergoing full cycles of N → B → N mode changes. Correspondingly, the resulting languages reflect either no copying (functioning as plain FSAs) or full copying.

In any specific run, it is the states that inform a machine M of its modality. The first time M reaches a G state, it has to enter B mode and keeps buffering when it transits between plain states. The first time when it reaches an H state, M is supposed to match strings. Hence, it is clear that to go through full cycles of mode changes, once M reaches a G state and switches to B mode, it has to encounter some H state later. Then the buffer has to be emptied for N mode at the point when a H state transits to a plain state. A template for those machines performing full copying can be seen in Figure 10.

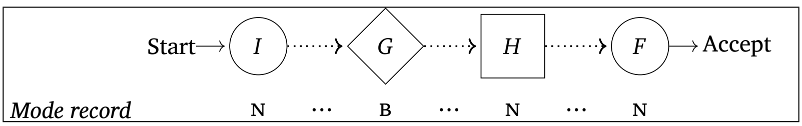


Figure 10: The template for the implementation of the copying in FSBMs. Key components: G state, H states, and strict ordering between G and H. Dotted lines represent a sequence of transitions

To allow us to reason about only the useful arrangements of G and H states, we impose an ordering requirement on G and H states in a machine. We define the *completeness restriction* on a path in Definition 7. We then identify those FSBMs in which all paths are complete as *complete-path FSBMs*. The machine M_1 in Figure 7, M_2 in Figure 8a and M_3 in Figure 9 are all complete-path FSBMs.

DEFINITION 6 *A* ***path*** *from one state p_1 to another state p_n in an FSBM M is a sequence of states $p_1, p_2, p_3, \ldots p_n$ such that for each $i \in \{1, \ldots, n-1\}$, there is a transition $(p_i, x, p_{i+1}) \in \delta_M$.*

DEFINITION 7 *A path in an FSBM M is* ***complete*** *if it is in the denotation of the regular expression $(P^*GP^*H)^*P^*$, where P represents any state* *in $Q-(G \cup H)$. A **complete-path FSBM** is an FSBM in which any path $p_1 \dots p_n$ with $p_1 \in I$ and $p_n \in F$ is complete.*

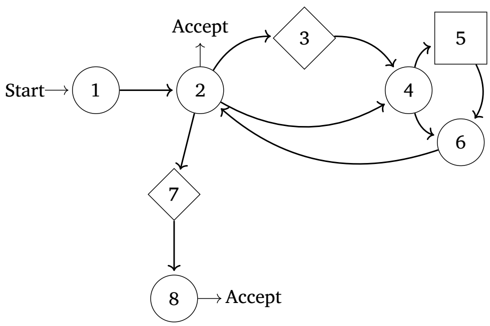
DEFINITION 8 *A path is said to be **a copying path** if it is complete and there is at least one G state (or at least one H state).*

3.4 *The sufficiency of complete-path FSBMs*

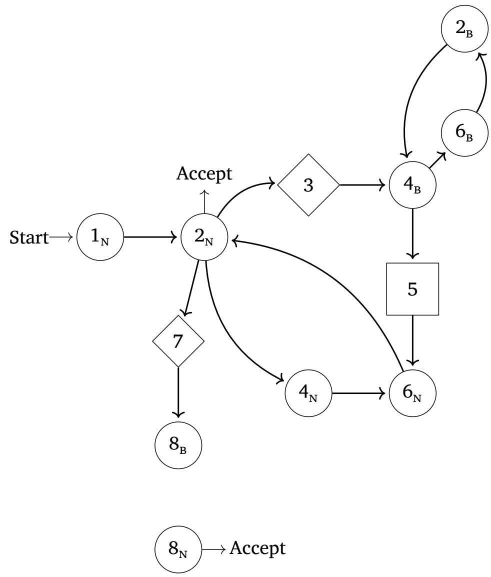
Now, we show that the languages recognized by FSBMs are precisely the languages recognized by complete-path FSBMs; this will allow us to restrict attention to complete-path FSBMs when studying the formal properties of these machines below.

PROPOSITION 1 *For any FSBM M, there exists a complete-path M' with $L(M) = L(M')$.*

Incomplete paths contribute nothing to the language generated by an FSBM, so showing this equivalence requires showing that, for any FSBM M_1, we can construct a new FSBM M_2 such that every path from an initial state to an accepting state in M_2 corresponds to some complete path from an initial state to an accepting state in M_1. The idea is that M_2 is a complete-path FSBM that keeps only those paths from M_1 that are indeed complete. The non-obvious cases of this construction involve scenarios where some plain state in M_1 might be reached either in normal (N) mode or in buffering (B) mode, depending on the path by which that plain state is reached. In Figure 11a, for example, this is the case for states 2, 4 and 6: intuitively, a path from state 2 back to itself might contain a G state (3) or an H state (5), or both or neither. To construct an equivalent complete-path FSBM M_2, we split each plain state q into two distinct states q_{N} and q_{B}. Transitions from a G state to q and transitions from q to an H state (i.e. transitions that only make sense in buffering mode) are carried over in M_2 for q_{B} but not for q_{N}. Similarly, transitions from an H state to q and transitions from q to a G state are carried over in M_2 for q_{N} but not for q_{B}. And the status of q as an initial and/or accepting state is carried over for q_{N} but not for q_{B}. Figure 11b shows the resulting complete-path FSBM for this example. In addition to keeping track of the mode in which states 2, 4 and 6 are visited, notice that this construction also prevents state 7 from occurring in any path from an initial state to an accepting state, since 8_{B} is not an accepting state and 8_{N} is unreachable.



(a) M_1



(b) M_2

Figure 11: Construction of a complete-path FSBM M_2 that is equivalent to M_1

PUMPING LEMMA 4

We define the *Regular Copying Languages* (RCLs) to be the set of all languages accepted by some (complete-path) FSBM. To be able to prove that some languages are not RCLs, we present a pumping lemma in this section. The idea is that if an FSBM produces a string $urrv$ via a copying run, and r is sufficiently long, then some subpart of r will be pumpable in the manner of the familiar pumping lemma for regular languages; that is, r can be broken into $x_1x_2x_3$ such that $ux_1x_2^ix_3x_1x_2^ix_3w$ is also accepted.[10]

THEOREM 1 *If $\mathscr{L}$ is a regular copying language, there is a positive integer k such that for every string $w \in \mathscr{L}$ with $|w| \geq 4k$, one of the following two conditions holds:*

1. *w can be rewritten as $w = xyz$ with*

[10] This idea is largely inspired by Savitch (1989, p.256), who proposes a pumping lemma for context-free languages augmented with copying.

(a) $|y| \geq 1$

(b) $|xy| \leq k$

(c) $\forall i \geq 0, xy^iz \in \mathscr{L}$

2. *w can be rewritten as* $w = ux_1x_2x_3x_1x_2x_3v$ *such that*

   (a) $|x_2| \geq 1$

   (b) $|x_1x_2| \leq k$

   (c) $\forall i \geq 0, ux_1x_2^ix_3x_1x_2^ix_3v \in \mathscr{L}$

PROOF Since $\mathscr{L}$ is a regular copying language, there is a complete-path FSBM M that recognizes $\mathscr{L}$. Let k be the number of states in M. For an arbitrary string $w \in \mathscr{L}$ with $|w| \geq 4k$, there is at least one path through M that generates w. Let p be the shortest such path (or if there are ties, choose arbitrarily). Note that p does not contain any ϵ-*loops*; if it did, its length would not be minimal among all candidate paths.

Suppose first that p is not a copying path. The length of p is at least $|w|+1$, and so since $|w| \geq 4k > k$, some state must occur twice in p, in fact in the first $k+1$ elements of p. As in the standard pumping lemma for regular languages, this means that w can be rewritten as xyz, with $|xy| \leq k$, in such a way that M can also generate xy^iz by repeating the loop, and $y \neq \epsilon$ since p contains no ϵ-loops. So in this case, w satisfies Condition 1.

If $p = p_0p_1 \ldots p_n$ is a copying path, then the run that generates $w = urrv$ must have the form $(urrv, p_0, \epsilon, \text{N}) \vdash_M^* (rrv, p_i, \epsilon, \text{N}) \vdash_M (rrv, p_i, \epsilon, \text{B}) \vdash_M^* (rv, p_j, r, \text{B}) \vdash_M (v, p_j, \epsilon, \text{N}) \vdash_M^* (\epsilon, p_n, \epsilon, \text{N})$ with $p_0 \in I$, $p_i \in G$, $p_j \in H$ and $p_n \in F$. Since $|w| \geq 4k$, at least one of $|u|, |r|, |v|$ is greater than or equal to k.

- If $|r| \geq k$, then $|p_i \ldots p_j| \geq |r|+1 \geq k+1$, so at least one state must appear twice in the first $k+1$ elements of the sequence $p_i \ldots p_j$, i.e. there are ℓ and ℓ' such that $i \leq \ell < \ell' \leq j$ and $p_\ell = p_{\ell'}$, with $\ell' - i < k$. Then it must be possible to rewrite r as $x_1x_2x_3$, with $|x_1x_2| \leq k$, such that repeating the subpath $p_\ell \ldots p_{\ell'}$ results in pumping x_2, and so any string of the form $x_1x_2^ix_3$ can be consumed from the input and stored in the buffer in the course of moving from $p_i \in G$ to $p_j \in H$, i.e. $(x_1x_2^ix_3x_1x_2^ix_3v, p_i, \epsilon, \text{B}) \vdash_M^* (x_1x_2^ix_3v, p_j, x_1x_2^ix_3, \text{B}) \vdash_M (v, p_j, \epsilon, \text{N})$. M will therefore generate all strings of the form $ux_1x_2^ix_3x_1x_2^ix_3v$, satisfying Condition 2.

- If $|u| \geq k$, then $|p_0 \ldots p_i| \geq |u| + 1 \geq k + 1$, so at least one state must appear twice in the sequence $p_0 \ldots p_i$, i.e. there are ℓ and ℓ' such that $0 \leq \ell < \ell' \leq i$ and $p_\ell = p_{\ell'}$, with $\ell' < k$. There are two cases to consider:
  - Suppose that M is in buffering mode throughout the part of the run from p_ℓ to $p_{\ell'}$. Therefore $p_\ell = p_{\ell'}$ is a plain state. Then it must be possible to rewrite u as $u'x_1x_2x_3x_1x_2x_3v'$, such that repeating the subpath $p_\ell \ldots p_{\ell'}$ results in pumping x_2. And since the repeated state must occur in the first $k+1$ elements of p, $|u'x_1x_2| \leq k$ and therefore $|x_1x_2| \leq k$. M will therefore generate all strings of the form
  $$u'x_1x_2^ix_3x_1x_2^ix_3v'rrv,$$
  satisfying Condition 2.
  - Otherwise, it must be possible to rewrite u as $x_1x_2x_3$ such that repeating this loop pumps x_2; since M is a complete-path FSBM, repeating the loop cannot create incomplete paths. And since the repeated state must occur in the first $k+1$ elements of p, $|x_1x_2| \leq k$. M will therefore generate all strings of the form $x_1x_2^ix_3rrv$, satisfying Condition 1.
- If $|v| \geq k$, an analogous argument shows that either Condition 1 or Condition 2 is satisfied. □

THEOREM 2 *$\mathscr{L}_{inv} = \{(a+b)^ic^j(a+b)^ic^j \mid i, j \geq 0\}$ is not an RCL.*

PROOF Suppose $\mathscr{L}_{inv}$ is an RCL. Let $w = a^kc^{k+1}b^kc^{k+1} \in \mathscr{L}_{inv}$, where k is the pumping length from Theorem 1. Given $|w| > 4k$, one of the conditions from Theorem 1 must hold.

1. Assume condition 1 holds. That is $w = xyz$ such that (i) $|y| \geq 1$, (ii) $|xy| \leq k$ and (iii) $\forall i \geq 0, xy^iz \in L$. Given $|xy| \leq k$, y must only contain as. Therefore $xyyz$ must have the form $a^{k+|y|}c^{k+1}b^kc^{k+1}$, so $xyyz \notin \mathscr{L}_{inv}$, a contradiction.
2. Assume condition 2 holds. Then, $w = ux_1x_2x_3x_1x_2x_3v$ such that (i) $|x_2| > 1$, (ii) $|x_1x_2| \leq k$ and (iii) $\forall i \geq 0$, $ux_1x_2^ix_3x_1x_2^ix_3v \in \mathscr{L}_{inv}$. The string x_1x_2 cannot contain the sub-string ac, because x_1x_2 occurs twice in w but ac does not; similarly, x_1x_2 cannot contain cb or bc. There remain three possible ways of choosing x_1x_2 with $|x_1x_2| \leq k$, each incurring a contradiction.

(a) If x_1x_2 contains only as, then x_3 must also contain only as because it occurs in between the two occurrences of x_1x_2 in w. Therefore $ux_1x_2^2x_3x_1x_2^2x_3v$ must have the form $a^\ell c^{k+1}b^kc^{k+1}$ with $\ell > k$, and is therefore not in $\mathscr{L}_{inv}$; a contradiction.

(b) Similarly, if x_1x_2 contains only bs, then $ux_1x_2^2x_3x_1x_2^2x_3v$ must have the form $a^kc^{k+1}b^\ell c^{k+1}$ with $\ell > k$, and is therefore not in $\mathscr{L}_{inv}$; a contradiction.

(c) Finally, suppose x_1x_2 contains only cs. If x_3 did not contain only cs, then it would need to cover the sub-string b^k since it appears in between the two occurrences of x_1x_2 in w; but if x_3 covered the sub-string b^k then this sub-string would occur twice in w, which it does not. So x_3 must also contain only cs. Therefore $ux_1x_2^2x_3x_1x_2^2x_3v$ must have the form either $a^kc^\ell b^kc^{k+1}$ or $a^kc^{k+1}b^kc^\ell$, with $\ell > k+1$; a contradiction.

□

EXAMPLE 1 *Some Non-RCL languages*

1. $\mathscr{L}_{SwissGerman} = \{a^ib^jc^id^j \mid i,j \geq 0\}$
2. $\mathscr{L} = \{a^nb^n \mid n \geq 0\}$
3. $\mathscr{L} = \{ww^R \mid w \in \Sigma^*\}$
4. $\mathscr{L} = \{www \mid w \in \Sigma^*\}$
5. $\mathscr{L} = \{w^{(2^n)} \mid n \geq 0\}$

To see that $\{w^{(2^n)} \mid n \geq 0\}$ is not an RCL, notice that the pumping lemma above requires that a constant-sized increase in the length of a string in the language can produce another string also in the language, but $w^{(2^n)}$ does not have this constant growth property (Joshi 1985).

5 CLOSURE PROPERTIES

The class of regular copying languages is closed under the following operations: intersection with a finite-state language (Section 5.1), some regular operations (union, concatenation, Kleene star; Section 5.2), and homomorphism (Section 5.3). But it is not closed under intersection, nor complementation (Section 5.4). More interestingly, it is not closed under inverse homomorphism (Section 5.5). In this section, we present proofs of these results.

Closure under intersection with regular languages 5.1

In this subsection, we write $\mathbf{0}$ for the zero matrix and $\mathbf{I}$ for the identity matrix, with the size of these matrices determined implicitly by context.

For any FSA $M = \langle Q, \Sigma, I, F, \delta \rangle$ and any symbol $x \in \Sigma$, $\mathbf{A}_x^M \in \{0,1\}^{|Q|\times|Q|}$ is the square matrix with rows and columns indexed by Q, whose (q_1, q_2) entry is 1 if $(q_1, x, q_2) \in \delta$ and is 0 otherwise. We will sometimes just write $\mathbf{A}_x$ where the FSA is clear from the context. We define $\mathbf{A}_\epsilon^M = \mathbf{I}$, and for any non-empty string $w = x_1 \dots x_n$ we define $\mathbf{A}_w^M = \mathbf{A}_{x_1}^M \dots \mathbf{A}_{x_n}^M$. Then it follows that the (q_1, q_2) entry of the matrix $\mathbf{A}_w^M$ is 1 if there is a path from q_1 to q_2 generating w, and is 0 otherwise.

We will assume, when we write any $\mathbf{A}_w^M$ in what follows, that the FSA M is supplemented with sink states as necessary to ensure that, for every $q_1 \in Q$ and every $x \in \Sigma$, there is at least one $q_2 \in Q$ such that $(q_1, x, q_2) \in \delta$. This ensures that, for any $w \in \Sigma^*$, there is at least one 1 on each row of $\mathbf{A}_w^M$, and therefore $\mathbf{A}_w^M \neq \mathbf{0}$.

We first define the relevant construction, then show below that it generates the desired intersection language. Without loss of generality, we assume that the FSA being intersected with the FSBM is ϵ-free.

DEFINITION 9 *Given an FSBM $M_1 = \langle Q_1, \Sigma, I_1, F_1, G_1, H_1, \delta_1 \rangle$, and an FSA $M_2 = \langle Q_2, \Sigma, I_2, F_2, \delta_2 \rangle$, we define $M_1 \cap M_2$ to be the FSBM $\langle Q, \Sigma, I, F, G, H, \delta \rangle$, where*

- $Q = Q_1 \times Q_2 \times \{0,1\}^{|Q_2|\times|Q_2|}$
- $I = I_1 \times I_2 \times \{\boldsymbol{0}\}$
- $F = F_1 \times F_2 \times \{\boldsymbol{0}\}$
- $G = G_1 \times Q_2 \times \{\boldsymbol{A}_\epsilon^{M_2}\}$
- $H = H_1 \times Q_2 \times \{\boldsymbol{0}\}$
- $\delta = \delta_{\text{N}} \cup \delta_{\text{B}} \cup \delta_{\text{N}\to\text{B}} \cup \delta_{\text{B}\to\text{N}}$, *where*
  - *(a)* $((q_1, q_1', \boldsymbol{0}), x, (q_2, q_2', \boldsymbol{0})) \in \delta_{\text{N}}$ *iff* $(q_1, x, q_2) \in \delta_1$ *with* $q_1 \notin G_1$ *and* $q_2 \notin H_1$*, and either*
    - $(q_1', x, q_2') \in \delta_2$*, or*
    - $x = \epsilon$ *and* $q_1' = q_2'$.
  - *(b)* $((q_1, q_1', \boldsymbol{0}), \epsilon, (q_1, q_1', \boldsymbol{A}_\epsilon^{M_2})) \in \delta_{\text{N}\to\text{B}}$ *iff* $q_1 \in G_1$
  - *(c)* $((q_1, q_1', \boldsymbol{A}), x, (q_2, q_2', \boldsymbol{A}\boldsymbol{A}_x^{M_2})) \in \delta_{\text{B}}$ *iff* $\boldsymbol{A} \neq \boldsymbol{0}$ *and* $(q_1, x, q_2) \in \delta_1$ *with* $q_1 \notin H_1$ *and* $q_2 \notin G_1$*, and either*

- $(q'_1, x, q'_2) \in \delta_2$, *or*
- $x = \epsilon$ *and* $q'_1 = q'_2$.

(d) $((q_1, q'_1, \boldsymbol{A}), \epsilon, (q_1, q'_2, \boldsymbol{0})) \in \delta_{\text{B}\to\text{N}}$ *iff* $q_1 \in H_1$ *and* $\boldsymbol{A} \neq \boldsymbol{0}$ *and the* (q'_1, q'_2) *entry of* $\boldsymbol{A}$ *is 1*

Notice that $|Q| = |Q_1| \times |Q_2| \times 2^{|Q_1| \times |Q_2|}$ is finite, since Q_1 and Q_2 are both finite.

The central challenge in setting up an FSBM to simulate the combination of an FSBM M_1 and an FSA M_2 is handling the effect on M_2 of $\vdash_{\text{B}\to\text{N}}$ transitions in M_1, where a string of arbitrary length is emptied from the buffer. Obviously the buffered string itself cannot be stored in the simulating FSBM's finite state. But, following an idea from Savitch (1989), any buffered string w determines a finite transition relation on the states of M_2, and it suffices to record this relation, which we encode in the form of the matrix $\mathbf{A}_w^{M_2}$.

The following lemma establishes the invariants that underpin the proof that this construction recognizes $L(M_1) \cap L(M_2)$.

LEMMA 1 *Suppose a non-empty sequence of configurations* $D_1 \ldots D_m$ *is the initial portion of a successful run (of any string) on an intersection FSBM* $M = M_1 \cap M_2$*, with each* $D_i = (u_i, (q_i, q'_i, \boldsymbol{A}_i), v_i, t_i)$*. Then one of the following is true:*

(i) $t_i = \text{N}$ *and* $\boldsymbol{A}_i = \boldsymbol{0}$

(ii) $t_i = \text{N}$ *and* $(q_i, q'_i, \boldsymbol{A}_i) \in (G_1 \times Q_2 \times \{\boldsymbol{A}_\epsilon^{M_2}\}) = G$

(iii) $t_i = \text{B}$ *and* $\boldsymbol{A}_i = \boldsymbol{A}_{v_i}^{M_2}$

(iv) $t_i = \text{B}$ *and* $(q_i, q'_i, \boldsymbol{A}_i) \in (H_1 \times Q_2 \times \{\boldsymbol{0}\}) = H$

PROOF By induction on the length m of the sequence. If $m = 1$, then $t_m = \text{N}$ and $(q_m, q'_m, \mathbf{A}_m) \in I = I_1 \times I_2 \times \{\mathbf{0}\}$, so $\mathbf{A}_m = \mathbf{0}$, satisfying (i). Now we consider a sequence $D_1 \ldots D_m D_{m+1}$ where we assume that the requirement holds of D_m. Since $D_m \vdash_{M_1 \cap M_2} D_{m+1}$, there are four cases to consider.

- Suppose $D_m \vdash_{\text{N}} D_{m+1}$. Then $t_m = t_{m+1} = \text{N}$, $(q_m, q'_m, \mathbf{A}_m) \notin G$, and $(q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \notin H$. The inductive hypothesis therefore implies that $\mathbf{A}_m = \mathbf{0}$. Now there are four subcases, depending on the critical element of δ that licenses $D_m \vdash_{\text{N}} D_{m+1}$.
  - If the critical transition is in δ_{N}, then immediately $\mathbf{A}_{m+1} = \mathbf{0}$, satisfying (i).

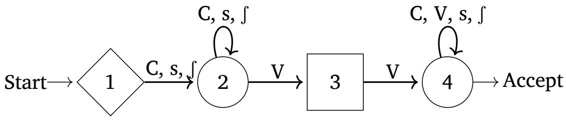
- If the critical transition is in $\delta_{\text{N}\to\text{B}}$, then $q_{m+1} \in G_1$ and $\mathbf{A}_{m+1} = \mathbf{A}_\epsilon^{M_2}$, satisfying (ii).
- The critical transition cannot be in δ_{B}, since $\mathbf{A}_m = \mathbf{0}$.
- The critical transition cannot be in $\delta_{\text{B}\to\text{N}}$, since $(q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \notin H$ which implies that either $q_{m+1} \notin H_1$ or $\mathbf{A}_{m+1} \neq \mathbf{0}$.

- Suppose $D_m \vdash_{\text{N}\to\text{B}} D_{m+1}$. Then $t_m = \text{N}$, $t_{m+1} = \text{B}$, $v_m = v_{m+1} = \epsilon$, and $(q_m, q'_m, \mathbf{A}_m) = (q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \in G = G_1 \times Q_2 \times \{\mathbf{A}_\epsilon^{M_2}\}$. Therefore $\mathbf{A}_{m+1} = \mathbf{A}_\epsilon^{M_2} = \mathbf{A}_{v_{m+1}}^{M_2}$, satisfying (iii).
- Suppose $D_m \vdash_{\text{B}} D_{m+1}$. Then $t_m = t_{m+1} = \text{B}$, $(q_m, q'_m, \mathbf{A}_m) \notin H$, $(q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \notin G$, and $v_{m+1} = v_m x$ for some $x \in \Sigma \cup \{\epsilon\}$. The inductive hypothesis therefore implies that $\mathbf{A}_m = \mathbf{A}_{v_m}^{M_2}$. Now there are four subcases, depending on the critical element of δ that licenses $D_m \vdash_{\text{B}} D_{m+1}$.
  - The critical transition cannot be in δ_{N}, since $\mathbf{A}_m = \mathbf{A}_{v_m}^{M_2} \neq \mathbf{0}$.
  - The critical transition cannot be in $\delta_{\text{N}\to\text{B}}$, since $(q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \notin G$ which implies that either $q_{m+1} \notin G_1$ or $\mathbf{A}_{m+1} \neq \mathbf{A}_\epsilon^{M_2}$.
  - If the critical transition is in δ_{B}, then $\mathbf{A}_{m+1} = \mathbf{A}_m \mathbf{A}_x^{M_2} = \mathbf{A}_{v_m}^{M_2} \mathbf{A}_x^{M_2} = \mathbf{A}_{v_m x}^{M_2} = \mathbf{A}_{v_{m+1}}^{M_2}$, satisfying (iii).
  - If the critical transition is in $\delta_{\text{B}\to\text{N}}$, then $q_{m+1} \in H_1$ and $\mathbf{A}_{m+1} = \mathbf{0}$, satisfying (iv).
- Suppose $D_m \vdash_{\text{B}\to\text{N}} D_{m+1}$. Then $t_m = \text{B}$, $t_{m+1} = \text{N}$, $v_{m+1} = \epsilon$, and $(q_m, q'_m, \mathbf{A}_m) = (q_{m+1}, q'_{m+1}, \mathbf{A}_{m+1}) \in H = H_1 \times Q_2 \times \{\mathbf{0}\}$. Therefore $\mathbf{A}_{m+1} = \mathbf{0}$, satisfying (i). □

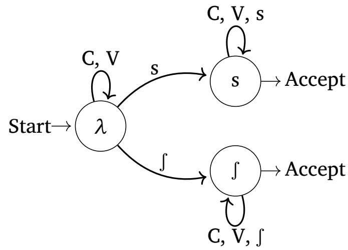
This lemma establishes that the matrix component of the constructed machine's state tracks the information necessary to determine the appropriate jump to make through M_2 when a string is emptied from the buffer: in a $\delta_{\text{B}\to\text{N}}$ transition from $(q_1, q'_1, \mathbf{A})$ to $(q_1, q'_2, \mathbf{0})$, the base FSBM M_1 is in state $q_1 \in H_1$ and therefore leaves buffering mode, and the matrix $\mathbf{A}$ determines the appropriate states q'_2 for M_2 to jump to. The rest of the proof that $L(M_1 \cap M_2) = L(M_1) \cap L(M_2)$ is standard, but is provided in Appendix A.

An example demonstrating how the intersection works can be found in Figure 12. The FSBM in Figure 12a computes the language that shows initial CC*V-copying. The FSA in Figure 12b, adapted from

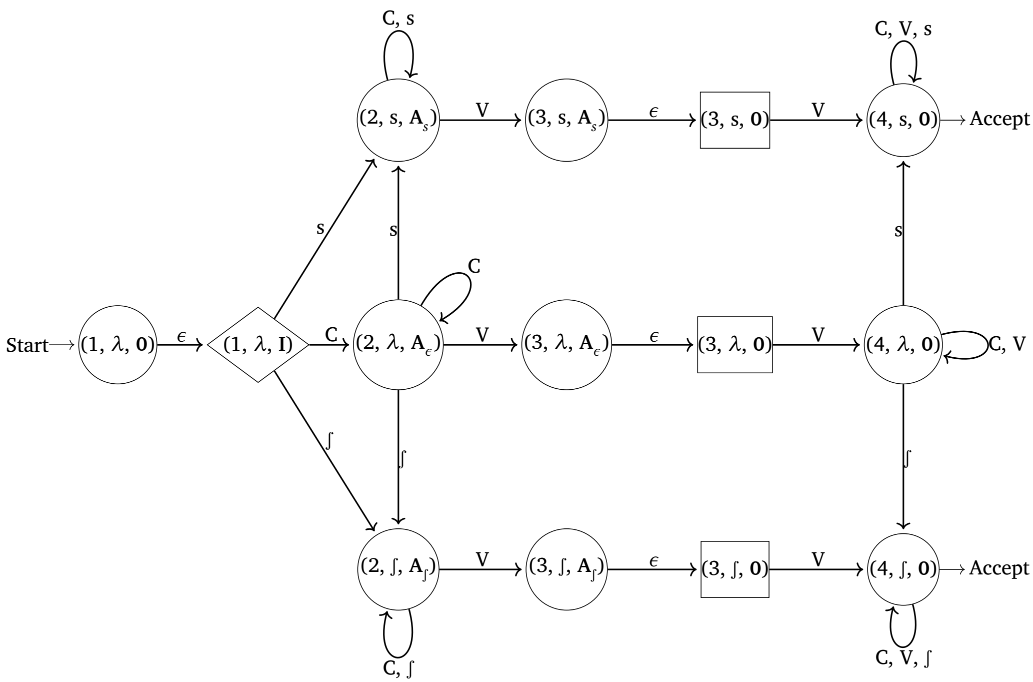
Heinz (2007, p.38), encodes Navajo sibilant harmony (Sapir and Hoijer 1967) on [anterior] features by banning *s…ʃ and *ʃ…s sequences.



(a) A complete-path FSBM M_1 recognizing initial CC*V-identity. $G = \{1\}$, $H = \{3\}$



(b) An FSA M_2 enforcing sibilant harmony. C indicates any non-sibilant consonant.



(c) The intersection FSBM $M_1 \cap M_2$, ignoring states from which no accepting state is reachable. $\mathbf{A}_\epsilon$ is the M_2 transition matrix for any string without any s or ʃ (equal to $\mathbf{I}$); $\mathbf{A}_s$ is the transition matrix for all strings with at least one *s* and no ʃ; and $\mathbf{A}_ʃ$ is the transition matrix for all strings with at least one ʃ and no *s*

Figure 12: An example intersection construction

The intersection FSBM is shown in Figure 12c, which recognizes the language of strings obeying both restrictions.

That FSBM-recognizable languages are closed under intersection with regular languages is an important step in clarifying the potential role of FSBMs for phonological theory. The overwhelming majority of phonotactic constraints that are not concerned with sub-string identity are regular (Heinz 2018), and so any such constraint can be combined with an FSBM-enforcable identity constraint to yield another FSBM-recognizable language. In fact, since the regular languages are closed under intersection, FSBMs can also express the intersection of any *collection* of normal phonotactic constraints with any single FSBM-enforcable substring-identity constraint.

An important issue that we leave open for future work is developing an algorithm for intersecting an FSA with an FSBM that assigns *weights* to strings expressing degrees of well-formedness. This kind of intersection algorithm has been used to implement the notion of competition between candidates from Optimality Theory (Smolensky and Prince 1993), where violable constraints are expressed by weighted FSAs (Ellison 1994; Eisner 1997; Albro 1998; Riggle 2004a). Such an intersection algorithm for weighted FSBMs would allow for FSBM-defined reduplication constraints to be incorporated into implemented OT grammars. In other words, the point from the preceding paragraph might generalize beyond the special case of binary constraints which combine via simple intersection.

Closed under regular operations 5.2

Noticeably, given complete-path FSBMs are finite-state machines with a copying mechanism, most of the proof ideas in this subsection are similar to the standard proofs for FSAs, which can be found in Hopcroft and Ullman (1979) and Sipser (2013).

THEOREM 3 *If L_1, L_2 are two FSBM-recognizable languages, then $L_1 \cup L_2$, $L_1 \circ L_2$ and L_1^* are also FSBM-recognizable languages.*

PROOF Assume there are complete-path FSBMs

$M_1 = \langle \Sigma, Q_1, I_1, F_1, G_1, H_1, \delta_1 \rangle$ and $M_2 = \langle \Sigma, Q_2, I_2, F_2, G_2, H_2, \delta_2 \rangle$

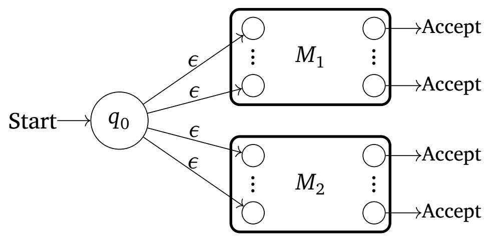
such that $L(M_1) = L_1$ and $L(M_2) = L_2$, then …

Union One can construct a new FSBM M that accepts an input w if either M_1 or M_2 accepts w. $M = \langle \Sigma, Q, I, F, G, H, \delta \rangle$ such that

- $Q = Q_1 \cup Q_2 \cup \{q_0\}$
- $I = \{q_0\}$
- $F = F_1 \cup F_2$
- $G = G_1 \cup G_2$
- $H = H_1 \cup H_2$
- $\delta = \delta_1 \cup \delta_2 \cup \{(q_0, \epsilon, q') \,|\, q' \in (I_1 \cup I_2)\}$

As illustrated in Figure 13, the construction of M keeps M_1 and M_2 unchanged, but adds a new state q_0. q_0 is the only initial state, branching into those previous initial states in M_1 and M_2 with ϵ-arcs. q_0 is a non-G, non-H plain state, so the constructed automaton is a complete-path FSBM.

Figure 13:
The construction used
in the union of two FSBMs



Concatenation There is a complete-path FSBM M that can recognize $L_1 \circ L_2$ by the normal concatenation of two automata. The new machine $M = \langle \Sigma, Q, I, F, G, H, \delta \rangle$ satisfies $L(M) = L_1 \circ L_2$.

- Q $= Q_1 \cup Q_2 \cup \{q_0\}$
- I $= \{q_0\}$
- F $= F_2$
- $G = G_1 \cup G_2$
- $H = H_1 \cup H_2$
- $\delta = \delta_1 \cup \delta_2 \cup \{(p_f, \epsilon, q_i) \,|\, p_f \in F_1, q_i \in I_2\} \cup \{(q_0, \epsilon, p_i) \,|\, p_i \in I_1\}$

As illustrated in Figure 14, the new machine adds a new plain state q_0 and makes it the only initial state, branching into those previous initial states in $M_1 \epsilon$-arcs. q_0 is not in H, nor in G. All final states in M_2 are the only final states in M. M also adds ϵ-arcs from all old final states in M_1 to all initial states in M_2.

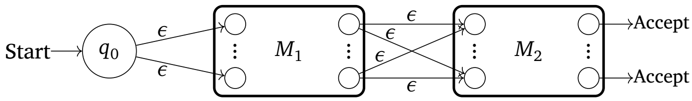
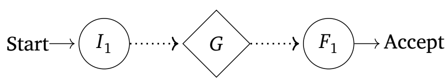
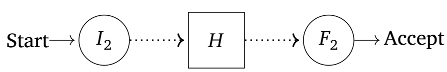


Figure 14: The construction used in the concatenation of two FSBMs

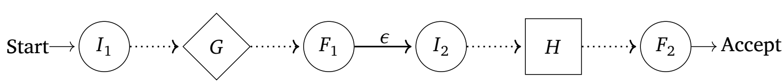
For this construction to work, it is important that we assume that M_1 and M_2 are complete-path FSBMs. Incomplete paths in two arbitrary machines might create a complete copying path, thus overgenerating under the construction of concatenation mentioned here. For example, as illustrated in Figure 15, imagine one path in M_1 only has G states but no H states, and another path in M_2 contains only H states. They both recognize the empty language $L_\emptyset = \emptyset$. Therefore, the concatenation of these two languages should also be $L_\emptyset$. The assumption that M_1 and M_2 are complete-path FSBMs ensures that the construction has this result.



(a) An incomplete path without H states; the language along this path ∅



(b) An incomplete path without G states; the language along this path is ∅



(c) Concatenation of two incomplete paths might lead to a copying path and result in a non-empty language

Figure 15: Problems arise in the concatenation of two incomplete paths. Dotted lines represent a sequence of transitions

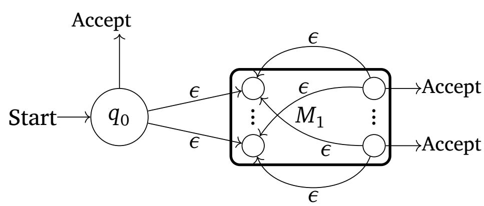
Kleene Star $(L_1)^*$ is a complete-path FSBM-recognizable language. The new machine $M = \langle \Sigma, Q, I, F, G, H, \delta \rangle$ satisfies $L(M) = (L_1)^*$.

- $Q = Q_1 \cup \{q_0\}$
- $I = \{q_0\}$

- $F = F \cup \{q_0\}$
- $G = G_1$
- $H = H_1$
- $\delta = \delta_1 \cup \{(p_f, \epsilon, q_i) \,|\, p_f \in F_1, q_i \in I_1\} \cup \{(q_0, \epsilon, q_i) \,|\, q_i \in I_1\}$

As illustrated in Figure 16, M is similar to M_1 with a new initial state q_0. q_0 is also a final state, branching into old initial states in M_1. In this way, M accepts the empty string ϵ. q_0 is never a G state nor an H state. Moreover, to make sure M can jump back to an initial state after it hits a final state, ϵ transitions from any final state to any old initial states are added. Since all paths in M_1 are complete, concatenations of these paths do not overgenerate. □

Figure 16:
The construction used in the star operation

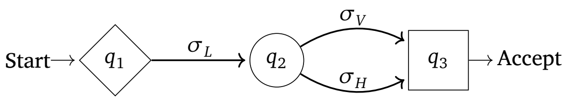


5.3 Closed under homomorphism

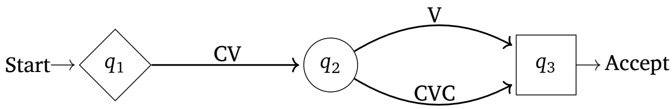
THEOREM 4 *The class of languages recognized by FSBMs is closed under homomorphisms.*

PROOF That complete-path FSBM languages are closed under homomorphism can be proved by constructing a new machine M_h based on the base machine M, such that $L(M_h) = h(L(M))$. The construction goes as follows. Relabel each transition that emits x in M with the string $h(x)$, and add states to split the transitions so that there is only one symbol or ϵ on each arc in M_h. States added for this purpose are not included in G or H. All paths in M_h are complete since the construction does not affect the arrangements G and H states in paths. □

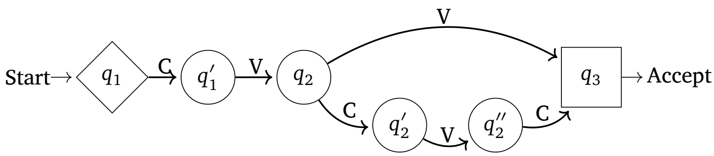
This construction is illustrated in Figure 17. The FSBM M uses the alphabet $\Sigma = \{\sigma_H, \sigma_L, \sigma_V\}$, and recognizes the finite language $\{\sigma_L\sigma_H\sigma_L\sigma_H, \sigma_L\sigma_V\sigma_L\sigma_V\}$. The constructed machine M_h recognizes



(a) $L(M) = \{\sigma_L\sigma_H\sigma_L\sigma_H, \sigma_L\sigma_V\sigma_L\sigma_V\}$



(b) $h(\sigma_L) = CV, h(\sigma_V) = V, h(\sigma_H) = CVC$. The intermediate step when the arcs are relabeled with mapped strings



(c) States q_1', q_2', q_2'' are added to split the arcs. $L(M_h) = \{CVVCVV, CVCVCCVCVC\}$

Figure 17: Constructions used for the homomorphic language

the image of this finite language under the homomorphism $h : \Sigma^* \to \{C, V\}^*$ defined by $h(\sigma_L) = CV$, $h(\sigma_V) = V$, and $h(\sigma_H) = CVC$.

The fact that FSBMs are closed under homomorphism allows theorists to perform analyses at convenient levels of abstraction.

Not closed under intersection and complementation 5.4

THEOREM 5 *The class of languages recognized by FSBMs is not closed under intersection, and thus not closed under complementation.*

PROOF $L_1 = \{wwx \,|\, w, x \in a^*b\}$ and $L_2 = \{xww \,|\, w, x \in a^*b\}$ are FSBM-recognizable languages. However, $L_1 \cap L_2 = \{www \,|\, w \in a^*b\}$ is not an FSBM-recognizable language. Given FSBM is closed under union but is not closed under intersection, by De Morgan's law, FSBM is not closed under complementation. □

Not closed under inverse homomorphism 5.5

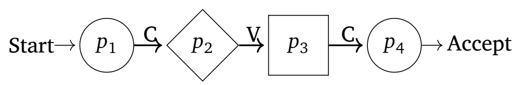
It is evident that the class of languages recognized by complete-path FSBMs is closed under *one-to-one* alphabetic inverse homomorphism.

One can directly relabel every mapped symbol in an FSBM to construct a new FSBM. But it is not closed under general inverse alphabetic homomorphisms and thus inverse homomorphism. Therefore, RCLs are not a trio.

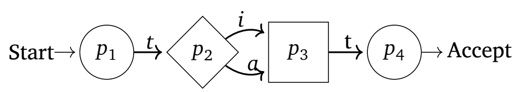
Consider the complete-path FSBM-recognizable language $L = \{a^i b^j a^i b^j \mid i, j \geq 1\}$ (see Figure 8a), and an alphabetic homomorphism $h : \{0,1,2\}^* \to \{a,b\}^*$ such that $h(0) = a$, $h(1) = a$ and $h(2) = b$. Then, the inverse homomorphic image of L is $h^{-1}(L) = \{(0+1)^i 2^j (0+1)^i 2^j \mid i, j \geq 1\}$, which is not an RCL by Theorem 2.

Even though RCLs are not closed under inverse homomorphisms, analyzing exactly why this is not the case highlights something that distinguishes the languages of FSBMs from many other well-known language classes. The pivotal point comes from the one-to-many mapping. At first glance, one might try to apply the conventional construction for showing closure under inverse homomorphism of FSAs, i.e. build a new machine M', which reads any symbol x in the new alphabet and simulates M on $h(x)$, as shown in Figure 18.

Figure 18:
The conventional construction of the inverse homomorphic image undergenerates



(a) $L(M) = \{CVVC\}$



(b) $h : \{a, i, t\} \to \{C, V\}^*$ with $h(a) = V$, $h(i) = V$ and $h(t) = C$. $L(M') = \{taat, tiit\}$ but $h^{-1}(L) = \{taat, tiit, tait, tiat\}$

.

But this construction fails to generate the full language $h^{-1}(L(M))$: the constructed machine M' still imposes an identity requirement, and therefore fails to accept strings such as *tait* where the two occurrences of V are mapped by h^{-1} to distinct symbols. The application of an inverse homomorphism – unlike the application of a homomorphism – can disrupt sub-string identity relationships that the construction of a new FSBM will necessarily maintain.

An equivalent extension of regular expressions 5.6

The standard class of regular languages can be defined either via FSAs or via regular expressions. FSBMs constitute a minimal enrichment of FSAs that allow for copying. Here we present a corresponding way to enrich regular expressions that leads to the same class of languages as FSBMs. This provides an alternative characterization of the RCL class in terms of language-theoretic closure properties.

DEFINITION 10 *Let Σ be an alphabet. The regular copying expressions (RCEs) over Σ and the languages they denote are defined as follows.*

- *$\emptyset$ is an RCE and $\mathscr{L}(\emptyset) = \emptyset$*
- *ϵ is an RCE and $\mathscr{L}(\epsilon) = \{\epsilon\}$*
- *$\forall a \in \Sigma$, a is an RCE and $\mathscr{L}(a) = \{a\}$*
- *If R_1 and R_2 are RCEs, then $R_1 + R_2$, R_1R_2, and R_1^* are RCEs, and $\mathscr{L}(R_1 + R_2) = \mathscr{L}(R_1) \cup \mathscr{L}(R_2)$, $\mathscr{L}(R_1R_2) = \{uv \mid u \in \mathscr{L}(R_1), v \in \mathscr{L}(R_2)\}$, and $\mathscr{L}(R_1^*) = (\mathscr{L}(R_1))^*$.*
- *(new copying operator) If R_1 is a **regular** expression, R_1^C is an RCE and $\mathscr{L}(R_1^C) = \{ww \mid w \in \mathscr{L}(R_1)\}$*

RCEs introduce two modifications to regular expressions. First, a $\cdot^C$ expression operator for the copying-derived language is added. Then, the closure under other regular operations is extended to all RCEs. Therefore, languages denoted by regular copying expressions are closed under concatenation, union and Kleene star. Second, the copying operation is only granted access to regular expressions, namely to regular sets formed *without* the use of copying. In other words, the languages denoted by RCEs are not closed under copying, thus restricting the denoted languages by excluding w^{2^n}.

Given Σ^* is a regular language, an RCE for the simplest copying language $L_{ww} = \{ww \mid w \in \Sigma^*\}$ with $\Sigma = \{a, b\}$ would be $((a+b)^*)^C$. Assume $\Sigma = \{C, V\}$, a naive RCE describing Agta plurals after CVC-reduplication without considering the rest of the syllable structures could be $(CVC)^C(V + C)^*$. This denotes a regular language, unlike $((a+b)^*)^C$. Note, $((CVC)^C(V+C)^*)^C$ is not a regular copying expression, because the copying operator cannot apply to the expressions containing copying.

As noted in footnote 7, there are a number of definitions of *extended regular expressions* in the literature that incorporate some form of back-references (e.g. Câmpeanu *et al.* 2002; Câmpeanu *et al.* 2003; Carle and Narendran 2009), and these motivated the development of Memory Automata (MFAs; Schmid 2016; Freydenberger and Schmid 2019). Just as FSBMs can be seen as a restricted special case of MFAs, RCEs correspond to a special case of extended regular expressions: essentially, an RCE of the form R^C is equivalent to $(R)\backslash 1$, where the back-reference necessarily immediately follows the captured group.

For further details of the equivalence of RCEs and FSBMs, see Appendix B.

6 DISCUSSION AND FURTHER IMPLICATIONS

6.1 *Typology of reduplication*

Here, we briefly consider some more complicated kinds of reduplication that are beyond the capacity of FSBMs as formulated in the present paper. We sketch some possible ways in which FSBMs might provide a starting point for future work that aims for a proper treatment of the full range of natural language reduplication phenomena.

Non-local Reduplication Non-local reduplication is the case when the surface phonological strings have non-adjacent copies, incurring non-local correspondence among symbols.[11] A more comprehensive typology and linguistic analysis on non-local reduplication can be found in Riggle (2004b). Examples from Creek are shown in Table 7.

[11] Bambara ‘Noun o Noun’ illustrates a particularly simple kind of non-local reduplication where the intervening string is *always* the fixed string ‘o’. This could be relatively easily handled by specifying a fixed string to each H state, to be inserted between the two copies when the buffer is emptied. The examples discussed in the main text are when the intervening elements are variable, different from the Bambara-like examples in important ways.

Non-local reduplication		
Creek plural		
Gloss	*Singular*	*Plural*
'precious'	a-cá:k-i:	a-ca:**ca**k-í:
'clean'	hasátk-i:	hasat**ha**k-í:
'soft'	lowáck-i:	lowac**lo**k-í:

Table 7:
Creek plural; CV-copying placed before the final consonant of the root (Booker 1979; Riggle 2004b)

Marantz (1982) described the adjacency between the reduplicant and the base as a general typological trend. There were proposals (e.g. Nelson 2005) arguing that Marantz's generalization is inviolable: the counter-examples could be analyzed either as non-reduplicative copying, or as results of interactions between adjacent reduplication and independently-motivated deletions. Riggle (2004b) used the Creek words in Table 7 to argue for true non-local correspondence relations.

FSBMs' current limitation to local reduplication comes from the requirement that B-mode computation has to be directly followed by the buffer-emptying process, and a filled buffer is not allowed in N mode. A possible modification to allow non-local reduplication would be to allow the buffer to be filled in N mode and encode such a possibility in another kind of special states, say J, which stops the machine from buffering, with the buffer only being matched against input and emptied when an H state is encountered. The transitions leading from a G state to a J state would consume symbols in the input tape and buffer symbols in the queue-like buffer. Then, if there is no adjacent H following the end of buffering, the machine can use plain transitions to plain states for only input symbols. The buffer with symbols in it should be kept unchanged. Ultimately, the machine has to encounter some H states to empty the buffer to accept the string, since no final configuration allows symbols on the buffer.

Such a modification might not affect much of the proof ideas of the theorems constructed so far. Regarding the pumping lemma, Condition 2 can be modified by including a sub-string of intervening segments in between two copies. That is, $w \in L$ with $|w| > 5k$ can be rewritten as $w = ux_1x_2x_3\mathbf{y}x_1x_2x_3v$ such that $\forall i \in \mathbb{N}, ux_1x_2^ix_3\mathbf{y}x_1x_2^ix_3v \in L$. It is worth pointing out that if the generalization in Creek is productive, the sub-string of intervening segments between copies could be unboundedly long.

Multiple Reduplication Here, multiple reduplication refers to the cases when two or more different reduplicative patterns appear in one word. One string can have multiple sub-strings identical to each other. Examples from Nlaka'pamux (previously known as Thompson), a Salish language, are listed in Table 8. See Zimmermann (2019) for a complete typological survey and classification.

Table 8:
Multiple reduplication in Nlaka'pamux

Multiple reduplication Nlaka'pamux (Broselow 1983, p.329)	
Gloss	Strings
calico	sil
DIM-calico	sí-sil'
DIST-calico	sil-síl
DIST-DIM-calico	sil-sí-sil'

While the computational nature of multiple reduplication in natural language phonology and morphology remains an open question,[12] FSBMs could be relatively easily modified to include multiple copies of the same base form ($\{w^n \mid w \in \Sigma^*, n \in \mathbb{N}\}$), where n might be tied to the number of copying operations in a language. Given a natural number n, an appropriate modification of FSBMs might allow for the buffered symbols to not be emptied until they have been matched n times against the input.

However, FSBMs cannot be easily modified to recognize the language $\{w^{2^n} \mid w \in \Sigma^*, n \in \mathbb{N}\}$, where *ww* strings are themselves copied (i.e. $\{w, ww, wwww, \dots\}$, excluding *www*).

It is worth carefully distinguishing between the sense of copying instantiated by *ww* and w^n on the one hand, and the sense instantiated by w^{2^n} on the other. The former sense highlights the fact that certain portions of a string are identical to certain other portions,

[12] For recent phonological analyses, see Zimmermann (2021a) and Zimmermann (2021b). For a more detailed discussion on the string-to-string function version of this problem, see Rawski *et al.* (2023).

whereas the latter is a natural interpretation of the idea that there is a copying *operation* that can apply to *its own outputs*. The kind of recursive copying exhibited by w^{2^n} means that this language does not have the constant growth property that Joshi (1985) identified as a criterion for mild context-sensitivity. Excluding this recursive copying from phonology seems relatively well-justified, on the grounds that triplication is attested (Zimmermann 2019; Rawski *et al.* 2023). But the situation may be different for syntax, where Kobele (2006), for example, has argued for recursive copying of the w^{2^n} sort on the basis of Yoruba relativized predicates. See also Clark and Yoshinaka (2014) on the relationship between parallel multiple context-free grammars (PMCFGs) and multiple context-free grammars (MCFGs); and Stabler (2004) on the comparison between what he calls *generating grammars* and *copying grammars*.

Reduplication with non-identical copies In natural languages, non-identical copies are prevalent. There are cases where other phonological processes apply to the base or the reduplicant to create nonidentical copies, such as onset cluster simplification in Tagalog partial reduplication (Zuraw 1996), e.g. 'X is working' [nag-ta-tɾabahoh], mapped from [tɾabahoh]. Another type of non-identical copies involves a fixed, memorized segment/sub-string (Alderete *et al.* 1999). Examples are given in Mongolian, illustrated in Table 9, where whole stems are copied to create forms with the meaning 'X and such things'. However, the initial consonant is always rewritten as [m].[13]

Table 9: Non-identical copies in Mongolian

Non-identical copies Mongolian Noun Reduplication (Svantesson *et al.* 2005, p. 60)		
Gloss	*Root*	*X and such things*
'gown'	teeɮ	teeɮ-meeɮ
'bread'	tʰaɮx	tʰaɮx-maɮx
'eye'	nut	nut-mut

[13] When the stem form starts with [m], it is always rewritten to [c]. For example, the reduplicated form of [maɮ] *cattle* is [maɮ-caɮ]

One way to modify FSBMs to accommodate non-identical copies would be to allow the machine to either store or empty not exactly the same input symbols, but the image of the inputs symbols under some alphabetic mapping, or finite-state transduction, f. For example, to account for the fixed consonant in Mongolian, we can introduce a finite state transduction $f_{C_1 \to m}$ that rewrites the first consonant to [m]. To empty the buffer, instead of checking the identity relation, it determines whether $f_{C_1 \to m}(x) = y$ where x is in the buffer and y is a prefix of the remaining input.

If no restrictions at all are imposed on the transduction, then the modified automata would recognize the context-free $\{a^n b^n \mid n \in \mathbb{N}\}$ with $f(a) = b$ in a manner that (unlike a context-free grammar) associates the first a with the first b and so on, though still excluding string reversals. Moreover, the resulting language set would also include $\{a^i b^j c^i d^j \mid i, j \geq 1\}$ with $f(a) = c, f(b) = d$. It could be fruitful for further studies to examine possible restrictions on the transduction.

6.2 *A note (and a conjecture) regarding determinism*

A natural question to consider is whether the non-determinism that we have allowed in FSBMs is essential.[14] A proper treatment of this issue turns out to be more subtle than it might initially appear, but we offer some initial observations here.

The FSBM in Figure 19 is non-deterministic in the sense that the string aa might lead the machine either to q_2 or to q_3. This familiar kind of non-determinism brings no additional expressive power in the case of standard FSAs, where the subset construction can be used to determinize any FSA. But this method for determinization cannot be straightforwardly applied to FSBMs, because of the distinguished status of G and H states. Applying the construction to the FSBM in Figure 19 would yield a new state corresponding to $\{q_2, q_3\}$, and then the question arises of whether this new state should be an H state (like q_3) or not (like q_2). Neither answer is sufficient: in the new machine, the string aa will deterministically lead to the state $\{q_2, q_3\}$, but the prefix aa may or may not be the entire string that needs to be buffered and copied.

[14] Thanks to two reviewers for drawing our attention to this.

Stated slightly more generally, the subset construction can eliminate non-determinism *between states* (state-nondeterminism), but in FSBMs there is also the possibility of nondeterminism *between modes* (mode-nondeterminism). The state-nondeterminism indicated in (5) could be eliminated, in a sense, by applying the subset construction to yield a new machine M' with transitions as in (6).

(5) a. $(aa\ldots, q_1, \text{N}, \epsilon) \vdash_M^* (\ldots, q_2, \text{B}, aa)$

b. $(aa\ldots, q_1, \text{N}, \epsilon) \vdash_M^* (\ldots, q_3, \text{B}, aa)$

(6) $(aa\ldots, \{q_1\}, \text{N}, \epsilon) \vdash_{M'}^* (\ldots, \{q_2, q_3\}, \text{B}, aa)$

But the two configurations reached in (5) differ in whether M will stop buffering after this prefix aa, and we suspect that there is no way to eliminate this kind of nondeterminism between modes. To bring out this important additional distinction, consider the transition sequences in (7) for the longer prefix $aaaa$.

(7) a. $(aaaa\ldots, q_1, \text{N}, \epsilon) \vdash_M^* (aa\ldots, q_2, \text{B}, aa)$
$\vdash_M^* (\ldots, q_2, \text{B}, aaaa)$

b. $(aaaa\ldots, q_1, \text{N}, \epsilon) \vdash_M^* (aa\ldots, q_3, \text{B}, aa) \vdash_M (\ldots, q_3, \text{N}, \epsilon)$

This indicates that there is something distinctive about the kind of non-determinism in Figure 19, which lies not in the fact that the prefix aa might lead to either state q_2 or state q_3, but rather the fact that the prefix $aaaa$ might lead to either state q_2 in mode B, or state q_3 in mode N.

The following definition makes a first attempt at pinpointing the distinctive kind of non-determinism in Figure 19.

DEFINITION 11 *An FSBM M is mode-deterministic if there do not exist three configurations $C = (w, q, m, v)$, $C_1 = (\epsilon, q_1, m_1, v_1)$ and $C_2 = (\epsilon, q_2, m_2, v_2)$, such that*

- *$C \vdash_M^* C_1$ and $C \vdash_M^* C_2$,*
- *$C_1 \not\vdash_M^* C_2$ and $C_2 \not\vdash_M^* C_1$, and*
- *$m_1 \neq m_2$.*

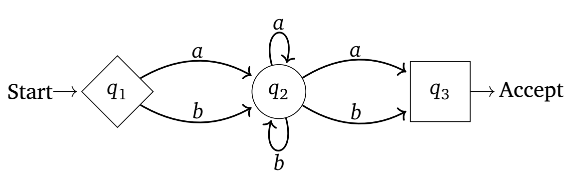
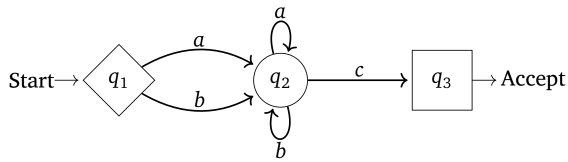


Figure 19:
An FSBM illustrating nondeterminism

Figure 20:
An FSBM illustrating
mode-determinism



The FSBM in Figure 20, for example, is mode-deterministic in this sense, whereas (7) demonstrates that the FSBM in Figure 19 is not. We conjecture that the mode-deterministic FSBMs are properly less powerful than the full class of FSBMs, and in particular that there is no mode-deterministic FSBM that generates the same language as the FSBM in Figure 19.

6.3 *The role of symbol identity*

A noteworthy trait of the RCL class is its non-closure under inverse homomorphisms. This distinguishes the RCL class from many of the familiar language classes that have played a role in the analysis of natural languages: the regular class and the context-free class are each closed under both homomorphisms and inverse homomorphisms, as are prominent classes in the mildly context sensitive region, such as the tree-adjoining languages and multiple context-free languages (Joshi 1985; Kallmeyer 2010).

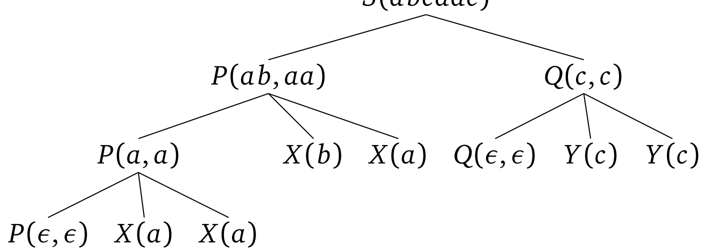
To illustrate, consider the relationship between the following two languages:

$$L_1 = (a+b)^i c^j (a+b)^i c^j$$
$$L_2 = a^i c^j a^i c^j$$

We showed above that L_1 is not an RCL, whereas L_2 obviously is. This sets the RCL class apart from the regular and context-free classes, which contain neither L_1 nor L_2, and from the tree-adjoining and multiple context-free classes, which contain both; recall Figure 5. For all these other formalisms, the surface differences between L_1 and L_2 are essentially irrelevant. For example, a multiple context-free grammar (MCFG; Seki *et al.* 1991; Kallmeyer 2010) for L_1 is given in (8), and (9) shows an illustrative derivation for the string $abcaac$. This grammar uses the nonterminals P and Q to control the assembly of (discontinuous) $(a+b)^i \ldots (a+b)^i$ and $c^j \ldots c^j$ portions respectively; P-portions can grow via the addition of X elements, and Q-portions can grow via the addition of Y elements.

(8) $S(u_1 v_1 u_2 v_2) \rightarrow P(u_1, u_2)\ Q(v_1, v_2)$
$P(\epsilon, \epsilon)$
$P(u_1 v, u_2 w) \rightarrow P(u_1, u_2)\ X(v)\ X(w)$
$Q(\epsilon, \epsilon)$
$Q(u_1 v, u_2 w) \rightarrow Q(u_1, u_2)\ Y(v)\ Y(w)$
$X(a)$
$X(b)$
$Y(c)$

(9)



Notice that to generate L_2 instead of L_1, we would simply omit the rule X(b) from (8). What this highlights is that for either L_1 or L_2, the significant work is done by the rules that arrange the yields of the nonterminals X and Y appropriately, and this work can be dissociated from the rules that specify the terminal symbols that can appear as the yields of X and Y. The nonterminals provide a grammar-internal mechanism for doing the book-keeping necessary to enforce the abstract pattern shared by L_1 and L_2, and the relationship between these grammar-internal symbols and the terminal symbols that make up the generated strings is opaque.

In an FSBM, however, the machinery that extends the formalism beyond the regular languages has no analogous grammar-internal book-keeping mechanism that can be dissociated from surface symbols: the non-regular effects of an FSBM's string-buffering mechanism are inherently tied to the identity of certain surface symbols. This is what underlies the crucial difference between L_1 and L_2 for FSBMs, and the non-closure under inverse homomorphisms of RCLs.[15]

[15] Of course the states of an FSBM are grammar-internal symbols in the relevant sense, and this is in effect what allows FSAs to be closed under both homo-

To put a label on this distinction, we might say that FSBMs are symbol-oriented (where by symbol we mean surface/terminal symbol), in contrast to the other formalisms mentioned above. Suppose, to make this precise, we say that a formalism (or a language class) is **symbol-oriented** iff it fails to be closed under both homomorphisms and inverse homomorphisms.

It is interesting to note that, while the symbol-oriented nature of FSBMs sets them apart from formalisms (such as MCFGs) motivated by the kinds of non-context-free cross-serial dependencies observed in syntax, this property of FSBMs is shared by other formalisms that have been argued to align well with observed phonological patterns. Many of the sub-regular language classes discussed by Heinz (2007), are also symbol-oriented in this sense. An easy example (Mayer and Major 2018; De Santo and Graf 2019) comes from the Strictly 2-Local (SL_2) languages: $(ab)^*$ is an SL_2 language, but applying the homomorphism h defined by $h(a) = c, h(b) = c$ yields $(cc)^*$, which is not an SL_2 language. So the SL_2 languages are not closed under homomorphisms.

The fact that the SL languages lack closure under homomorphisms, whereas the RCL class lacks closure under *inverse* homomorphisms, reflects the different role that symbol identity plays for the two formalisms. The move from $(ab)^*$ to $(cc)^*$ eliminates distinctions between surface symbols, which removes information that the SL_2 grammar for $(ab)^*$ was using to ensure that the length of each generated string was even. The move from L_2 to L_1, on the other hand, *introduces*

morphisms and inverse homomorphisms. But the point of the discussion here is to look at the distinctive additional capacities of FSBMs, which are brought out by considering a non-regular language such as L_2.

A comparison with Savitch's RPDAs (discussed above; Savitch 1989) is informative: RPDAs, while similar in some respects to FSBMs, generate a class of languages that *is* closed under both inverse homomorphism and homomorphism (in fact, under any finite-state transduction). This difference stems from the fact that an RPDA's queue-like memory arises from relaxing restrictions on a standard PDA's stack, and so the queue-like memory uses a distinct alphabet of stack symbols rather than surface symbols. These stack symbols are grammar-internal book-keeping devices whose relationship to surface symbols can be specified by the grammar-writer, as in the case of MCFGs such as (8) above.

distinctions between surface symbols which are incompatible with the string-buffering mechanism of an FSBM.[16]

But the broader point we wish to draw attention to here is the distinction between (i) the context-free class and various mildly context-sensitive classes, which are closed under both homomorphisms and inverse homomorphisms, and (ii) the RCL and SL classes, which are not and therefore exhibit a degree of sensitivity to surface symbol identity. It is intriguing that the insensitivity to surface symbol identity seems to be necessary for many important patterns found in natural language syntax – for example, the classic cross-serial dependencies in Swiss German (Shieber 1985) correspond to $a^i b^j c^i d^j$, rather than $a^i b^j a^i b^j$ – whereas many phonological patterns that have been studied computationally are compatible with symbol-oriented formalisms. This includes both the sub-regular patterns that motivate formalisms such as SL grammars, and the non-regular reduplication patterns that motivate FSBMs.

A complication to this clear picture may come from copying patterns in syntax, for example the Yoruba constructions discussed by Kobele (2006), mentioned above in Section 6.1. The languages generated by parallel multiple context-free grammars (PMCFGs) are not closed under inverse homomorphisms (Nishida and Seki 2000, p. 145, Corollary 12), for reasons analogous to what we have seen for FSBMs, and so this is an example of a symbol-oriented formalism that has been argued to be appropriate for syntax. But it is clear that syntax requires at least some *non*-symbol-oriented mechanisms to generate the well-known cross-serial dependencies of the Swiss-German sort ($a^i b^j c^i d^j$), whereas those cross-serial dependencies that we do observe in phonology are compatible with the more restricted, symbol-oriented notion of cross-serial dependencies that appear in reduplication.

[16] For similar reasons, the languages of regular expressions extended with back-references are also not closed under inverse homomorphism (Câmpeanu *et al.* 2003).

7 CONCLUSION

This paper has looked at the formal computational properties of unbounded copying on regular languages, including the simplest copying language L_{ww} where w can be any arbitrary string over an alphabet. We have proposed a new computational device: finite-state buffered machines (FSBMs), which add copying to regular languages by adding an unbounded queue-structured memory buffer, with specified states restricting how this memory buffer is used. As a result, we introduce a new class of languages, which is incomparable to context-free languages, named regular copying languages (RCLs).

This class of languages extends regular languages with *unbounded* copying but excludes non-reduplicative non-regular patterns. Context-free string reversals are excluded since the buffer is queue-like, and the mildly context-sensitive Swiss-German cross-serial dependency pattern, abstracted as $\{a^i b^j c^i d^j \mid i, j \geq 1\}$, is also excluded, since the buffer works on the same alphabet as the input tape and only matches *identical* sub-strings.

We have also surveyed the class's closure properties and proved a pumping lemma. This language set is closed under union, concatenation, Kleene Star, homomorphism, and intersection with regular languages. It is not closed under copying, inhibiting the recursive application of copying and excluding non-semilinear w^{2^n}. This class is also not closed under intersection, nor complementation. Finally, it is not closed under inverse homomorphism, given it cannot recover the possibility of non-identity among corresponding segments when the mapping is many-to-one (and the inverse homomorphic image is one-to-many); we suggested that this might reflect an important difference between the string-generating mechanisms of phonology and syntax.

One potential direction for future research is to connect FSBMs with the 2-way D-FSTs studied by Dolatian and Heinz (2018a,b, 2019, 2020), which successfully model unbounded copying as *function*s while excluding mirror image mappings. We briefly mention two possibilities along these lines. First, it will be interesting to compare the RCL class of languages with the image of the functions studied by Dolatian and Heinz (2020). Second, it is natural to consider adding to FSBMs another tape for output strings, extending from acceptors (as presented here) to finite-state buffered transducers (FSBTs). The morphological analysis ($ww \mapsto w$) problem is claimed to be difficult for 2-way D-FSTs, since they are not invertible. Our intuition is that FSBTs might help solve this issue: after reading the first w in input and buffering this string in memory, the machine can write ϵ to the output tape when it matches the buffered string against the contents of the input tape. But a more detailed and rigorous study is required in this direction.

We are currently investigating the learning and learnability of FSBMs and copying in sub-regular phonology. The RCL class itself cannot be identified in the limit, since it properly contains the regular class (Gold 1967). However, we take positive learning results from Clark and Yoshinaka (2014) and Clark *et al.* (2016) on PMCFGs with copying, and from Dolatian and Heinz (2018b) on Concatenated Output Strictly Local functions for reduplication, as suggestions for future directions towards learning results for FSBMs. In particular, one of the most attractive properties of the sub-regular classes is their Gold-learnability (e.g. Garcia *et al.* 1990; Heinz 2010; Chandlee *et al.* 2014; Jardine and Heinz 2016). We hope to explore whether the learnability property still holds once copying is added to these sub-regular classes.

Last but not least, the current class of languages excludes non-adjacent copies, multiple reduplication and reduplication with non-identical copies. We briefly sketched some possible modifications and their potential effects. We hope that our proposal here provides a useful framework for better understanding the formal issues raised by these more complex reduplication phenomena, and guiding empirical research into their typology.

ACKNOWLEDGMENT

We would like to thank Dylan Bumford, Hossep Dolatian, Bruce Hayes, Ed Keenan, Claire Moore-Cantwell, Kie Zuraw, the audience at the UCLA Phonology Seminar, the reviewers and audience at the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, the ESSLLI 2021 Student Session, and the 2021 Annual Meeting on Phonology for helpful input and discussion. Thanks also to the three anonymous reviewers for their valuable feedback, and the editor for noticing a mistake in Theorem 5. All errors remain our own.

APPENDICES

A PROOF OF THEOREM 1

LEMMA 2 *For any string w, if $w \in L(M_1 \cap M_2)$, then $w \in L(M_1)$ and $w \in L(M_2)$.*

PROOF Assume

$$M_1 = \langle Q_1, \Sigma, I_1, F_1, G_1, H_1, \delta_1 \rangle \quad \text{and} \quad M_2 = \langle Q_2, \Sigma, I_2, F_2, \delta_2 \rangle.$$

Let the run on $M_1 \cap M_2$ that generates w be $D_0, D_1, \ldots, D_m$, where each $D_i = (u_i, (p_i, q_i, \mathbf{A}_i), v_i, m_i)$. We define a sequence $C_0, C_1, \ldots, C_m$ of configurations of M_1, and a sequence $B_0, B_1, \ldots, B_m$ of configurations of M_2, as follows:

$$C_i = (u_i, p_i, v_i, m_i)$$

$$B_i = \begin{cases} (v_i \backslash u_i, q_i) & \text{if } m_i = \text{B} \\ & \text{and } (p_i, q_i, \mathbf{A}_i) \in (H_1 \times Q_2 \times \{\mathbf{0}\}) = H \\ (u_i, q_i) & \text{otherwise} \end{cases}$$

For the initial configuration $D_0 = (w, (p_0, q_0, \mathbf{A}_0), \epsilon, \text{N})$, we know that $(p_0, q_0, \mathbf{A}_0) \in I$, so $p_0 \in I_1$ and $q_0 \in I_2$. Therefore $C_0 = (w, p_0, \epsilon, n)$ is a valid starting configuration for a run of w on M_1, and $B_0 = (w, q_0)$ is a valid starting configuration for a run of w on M_2.

For the final configuration $D_m = (\epsilon, (p_m, q_m, \mathbf{A}_m), \epsilon, \text{N})$, we know that $(p_m, q_m, \mathbf{A}_m) \in F$, so $p_m \in F_1$ and $q_m \in F_2$. Therefore $C_m = (\epsilon, p_m, \epsilon, \text{N})$ is a valid ending configuration for a run on M_1, and $B_m = (\epsilon, q_m)$ is a valid ending configuration for a run on M_2.

To use the sequences $C_0, \ldots, C_m$ and $B_0, \ldots, B_m$ to establish that $w \in L(M_1)$ and $w \in L(M_2)$, we will show that, for every $i \in \{0, \ldots, m-1\}$, $C_i \vdash^*_{M_1} C_{i+1}$ and $B_i \vdash^*_{M_2} B_{i+1}$.

For each $i \in \{0, \ldots, m-1\}$, we know that $D_i \vdash_{M_1 \cap M_2} D_{i+1}$, so there are four cases to consider:

- Suppose $D_i \vdash_{\text{N}} D_{i+1}$. Then $D_i = (xu_{i+1}, (p_i, q_i, \mathbf{A}_i), \epsilon, \text{N})$ and $D_{i+1} = (u_{i+1}, (p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}), \epsilon, \text{N})$, with $((p_i, q_i, \mathbf{A}_i), x, (p_{i+1}, q_{i+1}, \mathbf{A}_{i+1})) \in \delta$, $(p_i, q_i, \mathbf{A}_i) \notin G$, and $(p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}) \notin H$. Then $C_i = (xu_{i+1}, p_i, \epsilon, \text{N})$, $C_{i+1} = (u_{i+1}, p_i, \epsilon, \text{N})$, $B_i = (xu_{i+1}, q_i)$ and $B_{i+1} = (u_{i+1}, q_{i+1})$. We want to show that $C_i \vdash^*_{M_1} C_{i+1}$ and that $B_i \vdash^*_{M_2} B_{i+1}$.
  - Suppose the critical transition is in δ_{N}. Then $(p_i, x, p_{i+1}) \in \delta_1$ and $p_i \notin G_1$ and $p_{i+1} \notin H_1$, so $C_i \vdash_{\text{N}} C_{i+1}$. Also either $(q_i, x, q_{i+1}) \in \delta_2$, or $x = \epsilon$ and $q_i = q_{i+1}$; so $B_i \vdash^* B_{i+1}$.
  - Suppose the critical transition is in $\delta_{\text{N}\to\text{B}}$. Then $x = \epsilon$, and $p_i = p_{i+1}$ and $q_i = q_{i+1}$. Therefore $C_i = C_{i+1}$ and $B_i = B_{i+1}$.
  - The critical transition cannot be in δ_{B}, because Lemma 1 implies that $\mathbf{A}_i = \mathbf{0}$.
  - The critical transition cannot be in $\delta_{\text{B}\to\text{N}}$, because Lemma 1 implies that $\mathbf{A}_i = \mathbf{0}$.
- Suppose $D_i \vdash_{\text{N}\to\text{B}} D_{i+1}$. Then $D_i = (u_i, (p_i, q_i, \mathbf{A}_i), \epsilon, \text{N})$ and $D_{i+1} = (u_i, (p_i, q_i, \mathbf{A}_i), \epsilon, \text{B})$, with $(p_i, q_i, \mathbf{A}_i) \in G$ and therefore $p_i \in G_1$. So $C_i = (u_i, p_i, \epsilon, \text{N})$ and $C_{i+1} = (u_i, p_i, \epsilon, \text{B})$, and therefore $C_i \vdash_{\text{N}\to\text{B}} C_{i+1}$. Furthermore $B_i = B_{i+1} = (u_i, q_i)$, since $\mathbf{A}_i = \mathbf{A}_\epsilon \neq \mathbf{0}$, so $B_i \vdash^* B_{i+1}$.
- Suppose $D_i \vdash_{\text{B}} D_{i+1}$. Then $D_i = (xu_{i+1}, (p_i, q_i, \mathbf{A}_i), v_i, \text{B})$ and $D_{i+1} = (u_{i+1}, (p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}), v_i x, \text{B})$, with $((p_i, q_i, \mathbf{A}_i), x$ $, (p_{i+1}, q_{i+1}, \mathbf{A}_{i+1})) \in \delta$, $(p_i, q_i, \mathbf{A}_i) \notin H$ and $(p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}) \notin G$. So $C_i = (xu_{i+1}, p_i, v_i, \text{B})$ and $C_{i+1} = (u_{i+1}, p_{i+1}, v_i x, \text{B})$, but B_i and B_{i+1} will depend on the sub-cases below. There are four sub-cases to consider.
  - The critical transition cannot be in δ_{N}, since Lemma 1 implies that $\mathbf{A}_i = \mathbf{A}^{M_2}_{v_i} \neq \mathbf{0}$.
  - The critical transition cannot be in $\delta_{\text{N}\to\text{B}}$, since Lemma 1 implies that $\mathbf{A}_i = \mathbf{A}^{M_2}_{v_i} \neq \mathbf{0}$.
  - Suppose the critical transition is in δ_{B}. Then $(p_i, x, p_{i+1}) \in \delta_1$ and $p_i \notin H_1$ and $p_{i+1} \notin G_1$. Therefore $C_i \vdash_{\text{B}} C_{i+1}$. Now consider B_i and B_{i+1}. Since $(p_i, q_i, \mathbf{A}_i) \notin H$ we know that $B_i = (xu_{i+1}, q_i)$. Also, we know $\mathbf{A}_{i+1} = \mathbf{A}_i \mathbf{A}_x \neq \mathbf{0}$, so $(p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}) \notin H$ and $B_{i+1} = (u_{i+1}, q_{i+1})$. Finally, either

$(q_i, x, q_{i+1}) \in \delta_2$, or $x = \epsilon$ and $q_i = q_{i+1}$; so in either case $B_i \vdash^* B_{i+1}$.

- Suppose the critical transition is in $\delta_{\text{B}\to\text{N}}$. Then $x = \epsilon$ and $p_i = p_{i+1}$, so $C_i = C_{i+1}$. Also $\mathbf{A}_i \neq \mathbf{0}$, so $B_i = (u_{i+1}, q_i)$. Furthermore, $p_{i+1} \in H_1$ and $\mathbf{A}_{i+1} = \mathbf{0}$, so $B_{i+1} = (v_i \backslash u_{i+1}, q_{i+1})$. And we know that $v_i \backslash u_{i+1}$ is defined, because the configuration D_{i+1} is part of a successful run and its state $(p_{i+1}, q_{i+1}, \mathbf{A}_{i+1}) \in H$, so the step to D_{i+2} must involve matching an initial portion of the string u_{i+1} against the buffered string v_i. Finally, we also know from the definition of $\delta_{\text{B}\to\text{N}}$ that the (q_i, q_{i+1}) entry of $\mathbf{A}_i = \mathbf{A}_{v_i}^{M_2}$ is 1, so $q_{i+1} \in \delta_2^*(q_i, v_i)$. Therefore $B_i = (u_{i+1}, q_i) \vdash^*_{M_2} (v_i \backslash u_{i+1}, q_{i+1}) = B_{i+1}$.

- Suppose $D_i \vdash_{\text{B}\to\text{N}} D_{i+1}$. Then $D_i = (vu_{i+1}, (p_i, q_i, \mathbf{A}_i), v, \text{B})$ and $D_{i+1} = (u_{i+1}, (p_i, q_i, \mathbf{A}_i), \epsilon, \text{N})$, with $(p_i, q_i, \mathbf{A}_i) \in H$. Therefore $C_i = (vu_{i+1}, p_i, v, \text{B})$ and $C_{i+1} = (u_{i+1}, p_i, \epsilon, \text{N})$, and $p_i \in H_1$, so $C_i \vdash_{\text{B}\to\text{N}} C_{i+1}$. Since $(p_i, q_i, \mathbf{A}_i) \in H$, $B_i = (v \backslash vu_{i+1}, q_i) = (u_{i+1}, q_i)$. But also $B_{i+1} = (u_{i+1}, q_i)$. So $B_i = B_{i+1}$.

Therefore $C_0 \vdash^*_{M_1} C_m$, so $w \in L(M_1)$. Similarly, $B_0 \vdash^*_{M_2} B_m$, so $w \in L(M_2)$. □

LEMMA 3 *For any string w, if $w \in L(M_1)$ and $w \in L(M_2)$, then $w \in L(M_1 \cap M_2)$.*

PROOF

Assume $w = x_1 x_2 x_3 \ldots x_n \in L_1$ and $w \in L_2$, N.T.S that $w \in L_M$.
$\because w \in L_1$ and $w \in L_2$
$\therefore$ there exists a sequence of configurations $C_0, C_1, C_2 \ldots C_m$ with

- $C_0 = (w, p_0, \epsilon, \text{N})$ with $p_0 \in I_1$
- $C_m = (\epsilon, p_m, \epsilon, \text{N})$ with $p_m \in F_1$
- $\forall 0 \leq i < m$, $C_i \vdash_{M_1} C_{i+1}$

and there's a function $f : \text{SUFFIX}(w) \to Q_2$ such that $f(w) \in I_2$ and $f(\epsilon) \in F_2$ and $\forall x \in \Sigma, v \in \Sigma^*$, $(f(xv), x, f(v)) \in \delta_2$.

For each $i \in \{0, \ldots, m\}$, we take $C_i = (u_i, p_i, v_i, m_i)$, and define D_i to be a configuration of $M_1 \cap M_2$ as follows:

$$D_i = \begin{cases} (u_i, (p_i, f(u_i), \mathbf{0}), v_i, \text{N}) & \text{if } m_i = \text{N} \\ (u_i, (p_i, f(u_i), \mathbf{A}_{v_i}^{M_2}), v_i, \text{B}) & \text{if } m_i = \text{B} \end{cases}$$

First, notice that $D_0 = (w, (p_0, f(w), \mathbf{0}), \epsilon, \textsc{n})$, where $p_0 \in I_1$ and $f(w) \in I_2$, so D_0 is a valid starting configuration for a run of w on $M_1 \cap M_2$. Similarly, $D_m = (\epsilon, (p_m, f(\epsilon), \mathbf{0}), \epsilon, \textsc{n})$, where $p_m \in F_1$ and $f(\epsilon) \in F_2$, so D_m is a valid ending configuration for a run on $M_1 \cap M_2$. To show that $w \in L(M_1 \cap M_2)$, we will show that for each $i \in \{0, \ldots, m-1\}$, $D_i \vdash^*_{M_1 \cap M_2} D_{i+1}$, which implies that $D_0 \vdash^*_{M_1 \cap M_2} D_m$.

For each $i \in \{0, \ldots, m-1\}$, we know that $C_i \vdash_{M_1} C_{i+1}$, so there are four cases to consider.

- Suppose $C_i \vdash_{\textsc{n}} C_{i+1}$. Then $C_i = (xu_{i+1}, p_i, \epsilon, \textsc{n})$ and $C_{i+1} = (u_{i+1}, p_{i+1}, \epsilon, \textsc{n})$ where $(p_i, x, p_{i+1}) \in \delta_1$ and $p_i \notin G_1$ and $p_{i+1} \notin H_1$. Therefore $D_i = (xu_{i+1}, (p_i, f(xu_{i+1}), \mathbf{0}), \epsilon, \textsc{n})$ and $D_{i+1} = (u_{i+1}, (p_{i+1}, f(u_{i+1}), \mathbf{0}), \epsilon, \textsc{n})$, with $(f(xu_{i+1}), x, f(u_{i+1})) \in \delta_2$. So $D_i \vdash_{\textsc{n}} D_{i+1}$, since $(p_i, f(xu_{i+1}), \mathbf{0}) \notin G$ and $(p_{i+1}, f(u_{i+1}), \mathbf{0}) \notin H$.
- Suppose $C_i \vdash_{\textsc{n}\to\textsc{b}} C_{i+1}$. Then $C_i = (u, p, \epsilon, \textsc{n})$ and $C_{i+1} = (u, p, \epsilon, \textsc{b})$, where $p \in G_1$. Therefore $D_i = (u, (p, f(u), \mathbf{0}), \epsilon, \textsc{n})$ and $D_{i+1} = (u, (p, f(u), \mathbf{A}^{M_2}_{\epsilon}), \epsilon, \textsc{b})$, and we need to show that $D_i \vdash^*_{M_1 \cap M_2} D_{i+1}$.
  - Since $p \in G_1$, the automaton $M_1 \cap M_2$ has a transition $((p, f(u), \mathbf{0}), \epsilon, (p, f(u), \mathbf{A}^{M_2}_{\epsilon})) \in \delta_{\textsc{n}\to\textsc{b}}$.
    Therefore $D_i \vdash_{\textsc{n}} (u, (p, f(u), \mathbf{A}^{M_2}_{\epsilon}), \epsilon, \textsc{n})$.
  - Since $p \in G_1$, we know that $(p, f(u), \mathbf{A}^{M_2}_{\epsilon}) \in G$, and therefore $(u, (p, f(u), \mathbf{A}^{M_2}_{\epsilon}), \epsilon, \textsc{n}) \vdash_{\textsc{n}\to\textsc{b}} (u, (p, f(u), \mathbf{A}^{M_2}_{\epsilon}), \epsilon, \textsc{b}) = B_{i+1}$.

  Therefore $D_i \vdash^*_{M_1 \cap M_2} D_{i+1}$.
- Suppose $C_i \vdash_{\textsc{b}} C_{i+1}$. Then $C_i = (xu_{i+1}, p_i, v_i, \textsc{b})$ and $C_{i+1} = (u_{i+1}, p_{i+1}, v_i x, \textsc{b})$, with $p_i \notin H_1$ and $p_{i+1} \notin G_1$. Therefore
  $$D_i = (xu_{i+1}, (p_i, f(xu_{i+1}), \mathbf{A}^{M_2}_{v_i}), v_i, \textsc{b})$$
  and
  $$D_{i+1} = (u_{i+1}, (p_{i+1}, f(u_{i+1}), \mathbf{A}^{M_2}_{v_i x}), v_i x, \textsc{b}),$$
  with $(f(xu_{i+1}), x, f(u_{i+1})) \in \delta_2$. Since $p_i \notin H_1$ and $p_{i+1} \notin G_1$ and $\mathbf{A}^{M_2}_{v_i} \neq \mathbf{0}$, the automaton $M_1 \cap M_2$ has a transition $((p_i, f(xu_{i+1}), \mathbf{A}^{M_2}_{v_i}), x, (p_{i+1}, f(u_{i+1}), \mathbf{A}^{M_2}_{v_i}\mathbf{A}^{M_2}_{x})) \in \delta_{\textsc{b}}$.
- Suppose $C_i \vdash_{\textsc{b}\to\textsc{n}} C_{i+1}$. Then $C_i = (v_i u_{i+1}, p, v_i, \textsc{b})$ and $C_{i+1} = (u_{i+1}, p, \epsilon, \textsc{n})$, with $p \in H_1$. Therefore
  $$D_i = (v_i u_{i+1}, (p, f(v_i u_{i+1}), \mathbf{A}^{M_2}_{v_i}), v_i, \textsc{b})$$
  and
  $$D_{i+1} = (u_{i+1}, (p, f(u_{i+1}), \mathbf{0}), \epsilon, \textsc{n}),$$

with $f(u_{i+1}) \in \delta_2^*(f(v_i u_{i+1}), v_i)$. We need to show that $D_i \vdash^*_{M_1 \cap M_2} D_{i+1}$.

- Since $p \in H_1$ and the $(f(v_i u_{i+1}), f(u_{i+1}))$ entry of the matrix $\mathbf{A}^{M_2}_{v_i}$ must be 1, we know that the automaton $M_1 \cap M_2$ has a transition $((p, f(v_i u_{i+1}), \mathbf{A}^{M_2}_{v_i}), \epsilon, (p, f(u_{i+1}), \mathbf{0})) \in \delta_{\text{B}\to\text{N}}$. Therefore $D_i \vdash_{\text{B}} (v_i u_{i+1}, (p, f(u_{i+1}), \mathbf{0}), v_i, \text{B})$.
- Since $p \in H_1$, we know that $(p, f(u_{i+1}), \mathbf{0}) \in H$, and therefore

$$(v_i u_{i+1}, (p, f(u_{i+1}), \mathbf{0}), v_i, \text{B}) \vdash_{\text{B}\to\text{N}} u_{i+1}, (p, f(u_{i+1}), \mathbf{0}), \epsilon, \text{N}) = D_{i+1}.$$

Therefore $D_i \vdash^*_{M_1 \cap M_2} D_{i+1}$.

Therefore $D_0 \vdash^*_{M_1 \cap M_2} D_m$, i.e.

$$(w, (p_0, f(w), \mathbf{0}), \epsilon, \text{N}) \vdash^*_{M_1 \cap M_2} (\epsilon, (p_m, f(\epsilon), \mathbf{0}), \epsilon, \text{N}),$$

and so $w \in L(M_1 \cap M_2)$. □

B EQUIVALENCE OF REGULAR-COPYING EXPRESSIONS TO FSBMS

We show here that RCEs and FSBMs are equivalent in terms of expressivity: namely, the languages accepted by FSBMs are precisely the languages denoted by RCEs. We prove this statement in two directions: 1) every RCE has a corresponding FSBM; 2) every language recognized by FSBMs can be denoted by an RCE.

THEOREM 6 *Let R be a regular copying expression. Then, there exists an FSBM that recognizes $\mathscr{L}(R)$.*

PROOF We complete our proof by induction on the number of operators in R.

Base case: zero operators R must be ϵ, $\emptyset$, a for some symbol a in Σ. Then, standard method to construct corresponding FSAs, thus FSBMs, meet the requirements.

Inductive step: One or more operators In induction, we assume this theorem holds for RCEs with less than n operators with $n \geq 1$. Let R have n operators. There are two cases: 1): $R = R_1^C$; 2): $R \neq R_1^C$;

- Case 1: $R = R_1^C$. Then, we know R_1 must be a regular expression and we can construct an FSA for R_1. Assume there's an FSA $M_0 = \langle Q', \Sigma, I', F', \delta' \rangle$ that recognizes $L(R_1)$. Let $M = \langle Q, \Sigma, I, F, \delta, G, H \rangle$ with
  - $Q = Q' \cup \{q_0, q_f\}$
  - $G = I = \{q_0\}$
  - $H = F = \{q_f\}$
  - $\delta = \delta' \cup \{(q_0, \epsilon, q) \mid q \in I'\} \cup \{(q, \epsilon, q_f) \mid q \in F'\}$

  As part of this construction, we add another initial state q_0 and a final state q_f and use them as the *only* initial and final states in the new machine. We add ϵ-arcs 1) from the new initial state q_0 to the previous initial states, and 2) from the previous final states to the new final state q_f. The key component is to add the copying mechanism: G and H. Let G contain only the initial state q_0, which would put the machine to B mode before it takes any transitions. Let H contain only the final state q_f, which stops the machine from buffering and sends it to string matching. Thus, if w is in $L(R_1)$, ww must be in the language accepted by this complete-path FSBM and nothing beyond. Figure 21 shows such a construction. The proof showing L(M) = L(R) is suppressed here.

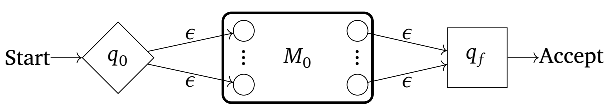


Figure 21:
The construction used in converting the copy expression R_1^C to a finite-state buffered machine. $L(M_0) = L(R_1)$.

- Case 2: when $R \neq R_1^C$ for some R_1, we know it has to be made out of the three operations: for some R_1 and R_2, $R = R_1 + R_2$, or $R = R_1R_2$ or $R = R_1^*$. Because R_1 and R_2 have operators less than i, from the induction hypothesis, we can construct FSBMs for R_1 and R_2 respectively. Using the constructions mentioned in the main text, we can construct the new FSBM for R. □

THEOREM 7 *If a language L is recognized by an FSBM, then L could be denoted by a RCE.*

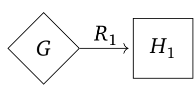
Instead of diving into proof details, we introduce the most crucial fragments to the full FSBM-to-RCE conversion: how the copying mechanism in a complete-path FSBM is converted into a copy expression. We leave out parts that use basic ideas of FSA-to-RE conversion, which can be found in Hopcroft and Ullman (1979, pp. 33–34).

The previous discussion on the realization of the copying mechanism in complete-path FSBMs concluded with three aspects 1) the specification of G states, 2) the specification of H states, and 3) the *completeness restriction* which imposes ordering requirements on G and H. Thus, to start with, we want to concentrate on the areas selected by G states and H states in a machine, as they are closely related to the copying mechanism.

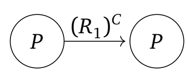
The core is to treat any G state and H state pair as an small FSA: if the paths along the pair do not cross other special states, borrow the FSA-to-RE conversion to get a regular expression R_1, denoting the languages possible to be stored in the buffer temporarily. Importantly, there are only finitely many (G, H) pairs. Iterating through all possible paths between these two states and getting a general RE R_1 by union, we use two plain states with the RCE R_1 along the arc to denote the languages from that specific G to H. Then we plug them back into the starting FSBM.

All special states are eliminated. Thus, we get an intermediate representation with only plain states. Similar ideas as FSA-to-RE conversion could be applied again to get the final regular copying expression for this FSBM. The described conversion of the copying mechanism in a machine to a copy expression is depicted in Figure 22.

Figure 22:
The conversion of the copying mechanism in an FSBM to a corresponding RCE. P represents the plain, non-H, non-G states



(a) Goal for the possible (G, H) in the first steps of the FSBM-to-RCE conversion



(b) Next step after Figure 22a

REFERENCES

Daniel M. ALBRO (1998), *Evaluation, implementation, and extension of primitive optimality theory*, Master's thesis, University of California, Los Angeles.

John ALDERETE, Jill BECKMAN, Laura BENUA, Amalia GNANADESIKAN, John MCCARTHY, and Suzanne URBANCZYK (1999), Reduplication with fixed segmentism, *Linguistic Inquiry*, 30(3):327–364, `https://doi.org/10.1162/002438999554101`.

Rajeev ALUR and Pavol ČERNÝ (2010), Expressiveness of streaming string transducers, in Kamal LODAYA and Meena MAHAJAN, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 1–12, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, `http://drops.dagstuhl.de/opus/volltexte/2010/2853`.

Bruce BAGEMIHL (1989), The crossing constraint and 'backwards languages', *Natural Language & Linguistic Theory*, 7(4):481–549.

Félix BASCHENIS, Olivier GAUWIN, Anca MUSCHOLL, and Gabriele PUPPIS (2017), Untwisting two-way transducers in elementary time, in *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–12, Reykjavik, Iceland, doi:10.1109/LICS.2017.8005138.

Kenneth R. BEESLEY and Lauri KARTTUNEN (2000), Finite-state non-concatenative morphotactics, in *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 191–198, Association for Computational Linguistics, Hong Kong, doi:10.3115/1075218.1075243, `https://www.aclweb.org/anthology/P00-1025`.

Karen M. BOOKER (1979), *Comparative Muskogean: Aspects of Proto-Muskogean verb morphology*, Ph.D. thesis, University of Kansas, Lawrence, KS.

Ellen I. BROSELOW (1983), Salish double reduplications: Subjacency in morphology, *Natural Language & Linguistic Theory*, 1:317–346.

Cezar CÂMPEANU, Kai SALOMAA, and Sheng YU (2002), Regex and extended regex, in Jean-Marc CHAMPARNAUD and Denis MAUREL, editors, *Implementation and Application of Automata, 7th International Conference, CIAA 2002, Tours, France, July 3–5, 2002, Revised Papers*, volume 2608 of *Lecture Notes in Computer Science*, pp. 77–84, Springer, doi:10.1007/3-540-44977-9_7, `https://doi.org/10.1007/3-540-44977-9_7`.

Cezar CÂMPEANU, Kai SALOMAA, and Sheng YU (2003), A formal study of practical regular expressions, *International Journal of Foundations of Computer Science*, 14(06):1007–1018.

Benjamin CARLE and Paliath NARENDRAN (2009), On extended regular expressions, in *Language and Automata Theory and Applications: Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings 3*, pp. 279–289, Springer.

Jane CHANDLEE (2014), *Strictly local phonological processes*, Ph.D. thesis, University of Delaware, Newark, DE.

Jane CHANDLEE (2017), Computational locality in morphological maps, *Morphology*, 27:599–641.

Jane CHANDLEE, Rémi EYRAUD, and Jeffrey HEINZ (2014), Learning strictly local subsequential functions, *Transactions of the Association for Computational Linguistics*, 2:491–504, doi:10.1162/tacl_a_00198, https://aclanthology.org/Q14-1038.

Jane CHANDLEE and Jeffrey HEINZ (2012), Bounded copying is subsequential: Implications for metathesis and reduplication, in *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pp. 42–51, Association for Computational Linguistics, Montréal, Canada, https://www.aclweb.org/anthology/W12-2306.

Alexander CLARK, Makoto KANAZAWA, Gregory M KOBELE, and Ryo YOSHINAKA (2016), Distributional learning of some nonlinear tree grammars, *Fundamenta Informaticae*, 146(4):339–377.

Alexander CLARK and Ryo YOSHINAKA (2014), Distributional learning of parallel multiple context-free grammars, *Machine Learning*, 96(1–2):5–31, ISSN 0885-6125, doi:10.1007/s10994-013-5403-2, https://doi.org/10.1007/s10994-013-5403-2.

Yael COHEN-SYGAL and Shuly WINTNER (2006), Finite-state registered automata for non-concatenative morphology, *Computational Linguistics*, 32(1):49–82.

Christopher CULY (1985), The complexity of the vocabulary of Bambara, *Linguistics and Philosophy*, 8(3):345–351.

Aniello DE SANTO and Thomas GRAF (2019), Structure sensitive tier projection: Applications and formal properties, in Raffaella BERNARDI, Greg KOBELE, and Sylvain POGODALLA, editors, *Formal Grammar*, pp. 35–50, Springer Berlin Heidelberg, ISBN 978-3-662-59648-7.

Robert M. W. DIXON (1972), *The Dyirbal language of North Queensland*, volume 9 of *Cambridge Studies in Linguistics*, Cambridge University Press, Cambridge.

Hossep DOLATIAN and Jeffrey HEINZ (2018a), Learning reduplication with 2-way finite-state transducers, in Olgierd UNOLD, Witold DYRKA, and Wojciech WIECZOREK, editors, *Proceedings of the 14th International Conference on Grammatical Inference*, volume 93 of *Proceedings of Machine Learning Research*, pp. 67–80, PMLR.

Hossep DOLATIAN and Jeffrey HEINZ (2018b), Modeling reduplication with 2-way finite-state transducers, in *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 66–77, Association for Computational Linguistics, Brussels, Belgium, doi:10.18653/v1/W18-5807.

Hossep DOLATIAN and Jeffrey HEINZ (2019), RedTyp: A database of reduplication with computational models, in *Proceedings of the Society for Computation in Linguistics*, volume 2, article 3.

Hossep DOLATIAN and Jeffrey HEINZ (2020), Computing and classifying reduplication with 2-way finite-state transducers, *Journal of Language Modelling*, 8(1):179–250.

Jason EISNER (1997), Efficient generation in primitive Optimality Theory, in *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 313–320, Association for Computational Linguistics, Madrid, Spain, doi:10.3115/976909.979657, `https://www.aclweb.org/anthology/P97-1040`.

T. Mark ELLISON (1994), Phonological derivation in Optimality Theory, in *Proceedings of the 15th Conference on Computational Linguistics – Volume 2*, COLING 94, pp. 1007–1013, Association for Computational Linguistics, USA, doi:10.3115/991250.991312, `https://doi.org/10.3115/991250.991312`.

Joost ENGELFRIET and Hendrik Jan HOOGEBOOM (1999), MSO definable string transductions and two-way finite state transducers, doi:10.48550/ARXIV.CS/9906007, `https://arxiv.org/abs/cs/9906007`.

Dominik D. FREYDENBERGER and Markus L. SCHMID (2019), Deterministic regular expressions with back-references, *Journal of Computer and System Sciences*, 105:1–39.

Pedro GARCIA, Enrique VIDAL, and José ONCINA (1990), Learning locally testable languages in the strict sense, in *Proceedings of the Workshop on Algorithmic Learning Theory*, pp. 325–338.

Gerald GAZDAR and Geoffrey K. PULLUM (1985), Computationally relevant properties of natural languages and their grammars, *New Generation Computing*, 3(3):273–306.

David GIL (1996), How to speak backwards in tagalog, in *Pan-Asiatic Linguistics, Proceedings of the Fourth International Symposium on Language and Linguistics, January 8–10, 1996, Institute of Language and Culture for Rural Development, Mahidol University at Salaya*, volume 1, pp. 297–306.

E. Mark GOLD (1967), Language identification in the limit, *Information and Control*, 10(5):447–474.

Phyllis M. HEALEY (1960), *An Agta grammar*, Bureau of Printing, Manila.

Jeffrey HEINZ (2007), *The inductive learning of phonotactic patterns*, Ph.D. thesis, University of California, Los Angeles.

Jeffrey HEINZ (2010), String extension learning, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 897–906, Association for Computational Linguistics, Uppsala, Sweden.

Jeffrey HEINZ (2018), The computational nature of phonological generalizations, in Larry HYMAN and Frans PLANK, editors, *Phonological Typology*, Phonetics and Phonology, chapter 5, pp. 126–195, De Gruyter Mouton.

Jeffrey HEINZ, Chetan RAWAL, and Herbert G TANNER (2011), Tier-based strictly local constraints for phonology, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies*, pp. 58–64.

John E HOPCROFT and Jeffrey D ULLMAN (1979), Introduction to automata theory, languages, and computation, *Addison-Welsey, NY*.

Mans HULDEN (2009), *Finite-state machine construction methods and algorithms for phonology and morphology*, Ph.D. thesis, University of Arizona, Tucson, USA, http://hdl.handle.net/10150/196112.

Riny HUYBREGTS (1984), The weak inadequacy of context-free phrase structure grammars, *Van periferie naar kern*, pp. 81–99.

Sharon INKELAS (2008), The dual theory of reduplication, *Linguistics*, 46(2):351–401, https://doi.org/10.1515/LING.2008.013.

Gerhard JÄGER and James ROGERS (2012), Formal language theory: Refining the Chomsky hierarchy, *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970.

Adam JARDINE and Jeffrey HEINZ (2016), Learning tier-based strictly 2-local languages, *Transactions of the Association for Computational Linguistics*, 4:87–98, ISSN 2307-387X, https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/694.

Aravind K. JOSHI (1985), *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?*, pp. 206–250, Studies in Natural Language Processing, Cambridge University Press, doi:10.1017/CBO9780511597855.007.

Laura KALLMEYER (2010), *Parsing beyond context-free grammars*, Cognitive Technologies, Springer, ISBN 978-3-642-14845-3, https://doi.org/10.1007/978-3-642-14846-0.

Ronald M. KAPLAN and Martin KAY (1994), Regular models of phonological rule systems, *Computational Linguistics*, 20(3):331–378.

Gregory M. KOBELE (2006), *Generating copies: An investigation into structural identity in language and grammar*, Ph.D. thesis, University of California, Los Angeles.

Martin KUTRIB, Andreas MALCHER, and Matthias WENDLANDT (2018), *Queue automata: Foundations and developments*, pp. 385–431, Springer International Publishing, Cham, ISBN 978-3-319-73216-9, https://doi.org/10.1007/978-3-319-73216-9_19.

Alexis MANASTER-RAMER (1986), Copying in natural languages, context-freeness, and queue grammars, in *Proceedings of the 24th Annual Meeting on Association for Computational Linguistics*, pp. 85–89, Association for Computational Linguistics, USA, https://doi.org/10.3115/981131.981145.

Alec MARANTZ (1982), Re reduplication, *Linguistic Inquiry*, 13(3):435–482.

Gary F. MARCUS, Sugumaran VIJAYAN, Shoba Bandi RAO, and Peter M. VISHTON (1999), Rule learning by seven-month-old infants, *Science*, 283(5398):77–80, http://www.jstor.org/stable/2897195.

Connor MAYER and Travis MAJOR (2018), A challenge for tier-based strict locality from Uyghur backness harmony, in Annie FORET, Greg KOBELE, and Sylvain POGODALLA, editors, *Formal Grammar 2018*, pp. 62–83, Springer Berlin Heidelberg, ISBN 978-3-662-57784-4.

Edith A. MORAVCSIK (1978), Reduplicative constructions, in Joseph GREENBERG, editor, *Universals of Human Language*, pp. 297–334, Stanford University Press, Stanford, CA, USA.

Elliott MORETON, Brandon PRICKETT, Katya PERTSOVA, Joshua FENNELL, Joe PATER, and Lisa SANDERS (2021), Learning reduplication, but not syllable reversal, in Ryan BENNETT, Richard BIBBS, Mykel Loren BRINKERHOFF, Stephanie Rich MAX J. KAPLAN, Amanda RYSLING, Nicholas Van HANDEL, and Maya Wax CAVALLARO, editors, *Supplemental Proceedings of the 2020 Annual Meeting on Phonology*, https://journals.linguisticsociety.org/proceedings/index.php/amphonology/article/view/4912/4634.

Nicole NELSON (2005), *Wrong side reduplication is epiphenomenal: Evidence from Yoruba*, pp. 135–160, De Gruyter Mouton, Berlin, Boston, https://doi.org/10.1515/9783110911466.135.

Taishin Y. NISHIDA and Shigeko SEKI (2000), Grouped partial et0l systems and parallel multiple context-free grammars, *Theoretical Computer Science*, 246(1):131–150, ISSN 0304-3975, https://www.sciencedirect.com/science/article/pii/S0304397599000766.

Jonathan RAWSKI, Hossep DOLATIAN, Jeffrey HEINZ, and Eric RAIMY (2023), Regular and polyregular theories of reduplication, *Glossa: a Journal of General Linguistics*, 8(1).

Jason RIGGLE (2004a), *Generation, recognition, and learning in finite state Optimality Theory*, Ph.D. thesis, University of California, Los Angeles.

Jason RIGGLE (2004b), Nonlocal reduplication, in *Proceedings of the 34th Meeting of the North-East Linguistics Society (NELS 34)*, pp. 485–496, GLSA, University of Massachusetts, USA, https://doi.org/doi:10.7282/T3GT5PZF.

Brian ROARK and Richard SPROAT (2007), *Computational approaches to morphology and syntax*, Oxford University Press, Oxford.

Carl RUBINO (2005), *Reduplication: Form, function and distribution*, pp. 11–30, De Gruyter Mouton, Berlin, Boston, https://doi.org/10.1515/9783110911466.11.

Carl RUBINO (2013), Reduplication, in Matthew S. DRYER and Martin HASPELMATH, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig, https://wals.info/chapter/27.

Edward SAPIR and Harry HOIJER (1967), *The phonology and morphology of the Navaho language*, 50, University of California Publication.

Walter SAVITCH (1989), A formal model for context-free languages augmented with reduplication, *Computational Linguistics*, 15(4):250–261, https://aclanthology.org/J89-4003.

Walter J. SAVITCH (1993), Why it might pay to assume that languages are infinite, *Annals of Mathematics and Artificial Intelligence*, 8:17–25.

Markus L. SCHMID (2016), Characterising REGEX languages by regular languages equipped with factor-referencing, *Information and Computation*, 249:1–17.

Hiroyuki SEKI, Takashi MATSUMURA, Mamoru FUJII, and Tadao KASAMI (1991), On multiple context-free grammars, *Theoretical Computer Science*, 88(2):191–229, https://www.sciencedirect.com/science/article/pii/030439759190374B.

Stuart M. SHIEBER (1985), Evidence against the context-freeness of natural language, in *Philosophy, Language, and Artificial Intelligence*, pp. 79–89, Springer.

Michael SIPSER (2013), *Introduction to the theory of computation*, The MIT Press, Boston, MA, third edition, ISBN 113318779X.

Paul SMOLENSKY and Alan PRINCE (1993), Optimality Theory: Constraint interaction in generative grammar, *Optimality Theory in Phonology*, 3.

Edward P. STABLER (2004), Varieties of crossing dependencies: Structure dependence and mild context sensitivity, *Cognitive Science*, 93(5):699–720.

Jan-Olof SVANTESSON, Anna TSENDINA, Anastasia KARLSSON, and Vivan FRANZÉN (2005), *The phonology of Mongolian*, OUP Oxford.

Rachelle WAKSLER (1999), Cross-linguistic evidence for morphological representation in the mental lexicon, *Brain and Language*, 68(1):68–74, https://www.sciencedirect.com/science/article/pii/S0093934X9992117X.

Markus WALTHER (2000), Finite-state reduplication in one-level prosodic morphology, in *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, https://www.aclweb.org/anthology/A00-2039.

Yang WANG (2021a), Recognizing reduplicated forms: Finite-State Buffered Machines, in *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 177–187, Association for Computational Linguistics, Online, doi:10.18653/v1/2021.sigmorphon-1.20, https://aclanthology.org/2021.sigmorphon-1.20.

Yang WANG (2021b), *Regular languages extended with reduplication: Formal models, proofs and illustrations*, Master's thesis, University of California, Los Angeles.

Samantha WRAY, Linnaea STOCKALL, and Alec MARANTZ (2022), Early form-based morphological decomposition in Tagalog: MEG evidence from reduplication, infixation, and circumfixation, *Neurobiology of Language*, 3(2):235–255, ISSN 2641-4368, https://doi.org/10.1162/nol_a_00062.

Fujimura YUKO (2001), Reduplication in standard Malay and Japanese, *Journal of Modern Languages*, 13(1):65–92.

Eva ZIMMERMANN (2019), Database of the DFG project multiple reduplication: Typology and theory, online available at http://www.evazimmermann.org/multiple-reduplication-data.html.

Eva ZIMMERMANN (2021a), Faded copies: Reduplication as distribution of activity, *Glossa: a Journal of General Linguistics*, 6(1).

Eva ZIMMERMANN (2021b), A phonological account of unfaithful multiple reduplication, *The Linguistic Review*, 38(3):537–572, https://doi.org/10.1515/tlr-2021-2075.

Kie ZURAW (1996), *Floating phonotactics: Infixation and reduplication in Tagalog loanwords*, Master's thesis, University of California, Los Angeles.

Kie ZURAW (2002), Aggressive reduplication, *Phonology*, 19(3):395–439.

Yang Wang
0000-0003-0575-2626
yangwangx@g.ucla.edu

Department of Linguistics
University of California, Los Angeles
Los Angeles, CA, USA

Tim Hunter
0000-0003-1587-9049
timhunter@ucla.edu

Department of Linguistics
University of California, Los Angeles
Los Angeles, CA, USA

Yang Wang and Tim Hunter (2023), *On regular languages*, Journal of Language Modelling, 11(1):1–66
https://dx.doi.org/10.15398/jlm.v11i1.342

Evaluating syntactic proposals using minimalist grammars and minimum description length*

Marina Ermolaeva
Lomonosov Moscow State University

ABSTRACT

Many patterns found in natural language syntax have multiple possible explanations or structural descriptions. Even within the currently dominant Minimalist framework (Chomsky 1995, 2000), it is not uncommon to encounter multiple types of analyses for the same phenomenon proposed in the literature. A natural question, then, is whether one could evaluate and compare syntactic proposals from a quantitative point of view. In this paper, we show how an evaluation measure inspired by the minimum description length principle (Rissanen 1978) can be used to compare accounts of syntactic phenomena implemented as minimalist grammars (Stabler 1997), and how arguments for and against this kind of analysis translate into quantitative differences.

Keywords: syntax, evaluation measure, minimum description length, minimalist grammars, double object construction

INTRODUCTION 1

Even within the same framework, different proposals often seem equally capable of capturing observed linguistic phenomena, which creates a need for an additional criterion to choose between them.

*This paper is based on some parts of the author's PhD thesis (Ermolaeva 2021).

This idea is prominent in early generative grammar. An *evaluation procedure* – a method of determining which of the two given grammars is better, given a corpus of data – is discussed in Chomsky 1957. It makes another appearance in Chomsky 1965, where an *explanatory theory* of language is defined as one capable of selecting a descriptively adequate grammar based on linguistic data. The components of such a theory mirror those of an *acquisition model*, i.e. how a child learns a language, and are listed below:

i. a universal phonetic theory that defines the notion "possible sentence"
ii. a definition of "structural description"
iii. a definition of "generative grammar"
iv. a method for determining the structural description of a sentence, given a grammar
v. a way of evaluating alternative proposed grammars

(Chomsky 1965, p. 31)

The last requirement (v) is described as being twofold: it calls for a formal *evaluation measure*, some sort of quantitative indication of how good a grammar is, but also demands that the class of possible grammars be small enough so the evaluation measure can realistically choose between them. In this framework, a precise and rich definition of "generative grammar" serves to tighten the class of grammars. However, the theory still permits multiple grammars compatible with the same data set; the choice of grammar is under-determined by the language data alone. This is where the evaluation measure comes in: the correct grammar is the highest-valued one among those that describe the data correctly. Of course, exactly how to construct a reasonable evaluation measure is a major issue by itself. Chomsky and Halle (1968) make some specific steps in this direction (for phonological rules), including a proposal of an evaluation procedure based on rule length measured in symbols.

Chomsky's later work takes the idea of restricting what counts as a candidate grammar much further. By Chomsky 1986, the description of a grammar has shifted away from rule systems and is split into two components: an innate universal system of principles and parameters and a language-specific lexicon of items defined by their phonological form and semantic properties, with the former getting most of the attention. Assuming a finite number of principles, parameters, and parameter values, the number of possible languages (apart from the lexicon) is also finite. This move sharply reduces the role of the evaluation measure or even dispenses with it altogether, as long as the universal grammar can be designed to permit only a single grammar compatible with the data.[1] The most recent and currently dominant iteration of generative grammar, the Minimalist Program (Chomsky 1995, 2000), continues this trend. Much of the system is assumed to be universal and innate, leaving no need or place for an evaluation measure; and language-specific properties that must be learned are largely shifted into the features of lexical items.

To summarize, the framework of Chomsky (1965) allows for multiple descriptions of a given language, one of which is the correct grammar, and these descriptions can be compared based on some quantitative measure. On the other hand, another framework he proposed (Chomsky 1986), as well as his later work, allow for a small number of descriptions of a given language, or even a single one; the correct grammar follows from the formal properties of the system and the language data. This stance can be considered a special case of the previous one, where the set of candidate grammars is made sufficiently small to eliminate the need for an evaluation measure.

Even though these two approaches are often thought of as mutually exclusive, they can be reconciled. Goldsmith (2011) and Katzir (2014) argue in favor of an evaluation measure based on the principle of minimum description length (MDL, Rissanen 1978), which takes into account both how good a grammar is by itself and how well it fits the data. MDL is compatible with any theory of universal grammar – as long as the grammars permitted by it are capable of *parsing*, or assigning structural descriptions to sentences as per (iv), and their description length can be compared. In line with these ideas, we combine the learning focus of (Chomsky 1965) with the simplifying developments

[1] The *strong learning* approach of Clark 2013, 2015 can be thought of as a formalization of this idea. For each set of strings, it requires the existence of a unique description called the *canonical grammar*. A strong learning algorithm is required to converge to this target grammar for each (formal) language.

of Minimalism, applying an evaluation measure to Minimalist lexical items.

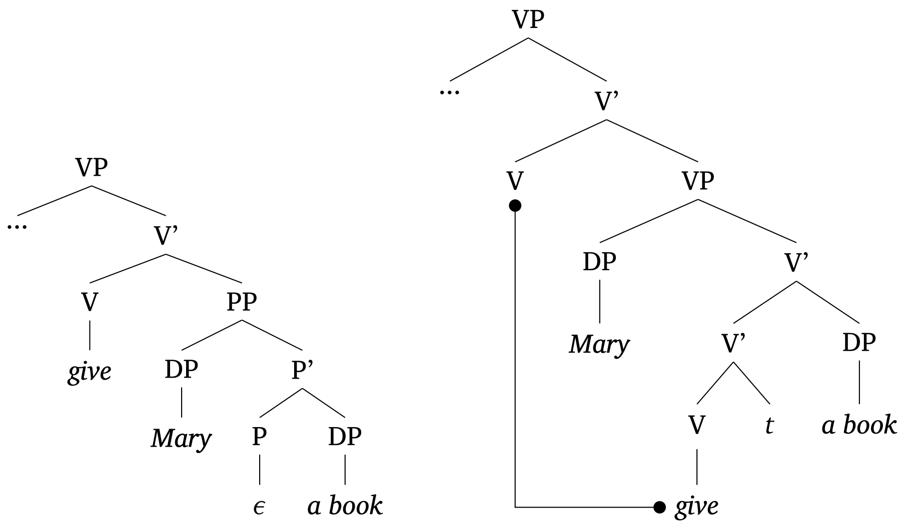
One major issue we have to tackle right from the start is that of formalization. Marr (1982) distinguishes between three levels of description of complex cognitive systems, including language:

- Computational: abstract specification of what the system computes;
- Algorithmic: structures representing the data and algorithms that manipulate them;
- Implementational: concrete realization of the algorithms in the hardware or wetware.

Johnson (2017) considers linguistic theories to be computational-level, while Peacocke (1986) places them at a "level 1.5", between the computational and algorithmic level. Syntactic literature in particular tends to abstract away from algorithmic-level details such as full specifications of lexical items involved in derivations or syntactic features being checked by each application of a structure-building operation. At the same time, differences between competing analyses of the same phenomenon seem to fall closer to the algorithmic level.

For a specific example, consider the double object construction (e.g. *John gave Mary a book*) in English (Figure 1). Any analysis of a syntactic phenomenon encodes two kinds of information: relatively theory-neutral, high-level facts that directly follow from the data, such as relations between words based on argument structure and linear order; and a proposed explanation of these facts – for instance, a specific configuration of lexical items constructed by structure-building operations. Descriptively, ditransitive verbs such as *give* appear in active sentences with three arguments: a subject, a direct object, and an indirect object. This is (apparently) non-controversial. On the lower level,[2] disregarding the subject, one option is to combine the two internal arguments together and have the verb select the resulting constituent as its complement (Figure 1a). The arguments are described in

[2] Work concerning these structures also tends to assume and try to explain a connection between them and prepositional constructions, as in *John gave a book to Mary*. This too is a nontrivial analytical choice; see Goldsmith 1980. A sketch of comparison between grammars along this dimension is given in Appendix A.



(a) Null P (adapted from Pesetsky 1996)　(b) VP-shells (adapted from Larson 1988)

Figure 1: The double object construction

terms of Williams' (1975) "small clauses" or taken to be connected by a silent preposition-like element (Kayne 1984; Pesetsky 1996; Harley 2002; Harley and Jung 2015). The alternative is to have the verb form a constituent with one of its internal arguments and then select the other one (Figure 1b). This option gives rise to VP-shells (Larson 1988) and analyses inspired by them (Kawakami 2018).

Existing treatments of the double object construction generally fall into one of the two categories mentioned above, as there are only so many conceivable ways to form a binary-branching structure containing a verb and two arguments. That said, the abundance of recent literature on the topic indicates that this is far from a closed issue.

Given a disagreement in the literature over a specific linguistic puzzle, how can the competing solutions be compared in terms of Chomsky's evaluation procedure? In order to take on this question, one needs to capture precisely what makes them different. This requires formalizing syntactic proposals at the algorithmic level, expressing them as a clearly defined set of building blocks and rules for putting them together. This paper adopts the formalism known as *minimalist grammars* (MGs), introduced by Stabler (1997). On the one hand, minimalist grammars were expressly designed as an implementation of Chomsky's Minimalist Program[3] and offer a way to state analyses of syntactic phenomena in terms familiar to a linguist: lexical items defined by features and structure-building operations that combine them.[4] On the other hand, they are explicit in spelling out assumptions about syntactic units and operations, and their formal properties – such as the complexity of string languages they generate and relation to other grammar formalisms – are relatively well understood.

This paper is structured as follows. Section 2 discusses the MDL principle, along with a toy example to demonstrate it in action. Section 3 provides a semi-formal, example-driven description of minimalist grammars. Section 4 builds on the previous sections to outline an encoding scheme for MGs and show how various intuitive notions translate into MDL values. In Section 5 we move away from toy examples and look at how MDL and MGs can be used to approach the problem of the double object construction. Finally, Section 6 offers some higher-level discussion and indicates some directions for future work.

2 THE MINIMUM DESCRIPTION LENGTH PRINCIPLE

Minimum description length (Rissanen 1978) is a principle for selecting a model to explain a dataset, which takes into account the simplic-

[3] The choice of capitalization – uppercase for "Minimalist Program" and lowercase for "minimalist grammars" – follows the sources that introduced these terms, Chomsky (1995) and Stabler (1997), respectively.

[4] Why minimalist grammars? A fully fleshed-out formalism is necessary to compute a quantitative measure such as MDL for each proposal in a self-contained way, independently from other candidates. That said, *which* formalism to use is a nontrivial decision, as any choice involves a tradeoff between conceptual simplicity and faithfulness to the original theoretical proposals. MGs do appear to diverge from mainstream Minimalist syntax with respect to the feature calculus, implementation of movement, locality, and other issues. However, as discussed in depth by Graf (2013, pp. 96–125), many of these apparent points of disagreement are a matter of convenience rather than an integral part of the formalism. We will briefly return to the problem of choosing a formalism in Section 6.

ity of both the model itself and the explanation of the dataset it offers. In the MDL framework, the best grammar to describe a corpus is the one that minimizes the sum of the following:

- the length of the grammar, measured in bits;
- the length of the description assigned by the grammar to the corpus, measured in bits.

Within linguistics, MDL has been used as a method of comparing candidate analyses of a given dataset, for example, for induction of phonological constraints (Rasin and Katzir 2016) and ordered rules (Rasin *et al.* 2018), morphological segmentation (Goldsmith 2001, 2006), and inferring syntactic categories given known morphological patterns (Hu *et al.* 2005).

Context-free grammars 2.1

To demonstrate this idea in action, we will use the formalism of context-free grammars (CFGs), also called phrase-structure grammars (Chomsky 1956). CFGs were developed for describing syntactic structure in natural language and serve as the starting point of Chomsky's (1965) Standard Theory. A context-free grammar is defined by specifying the following components:

- N, a finite set of *nonterminal symbols*. By convention, $\mathtt{S} \in N$ is the *start symbol*;
- Σ, a finite set of *terminal symbols* disjoint from N;
- R, a finite set of *(rewrite) rules*. Each member of R is a pair $\langle \alpha, \beta \rangle$ (usually written as $\alpha \rightarrow \beta$), where $\alpha \in N$ and β is a (potentially empty) string of terminal and nonterminal symbols.

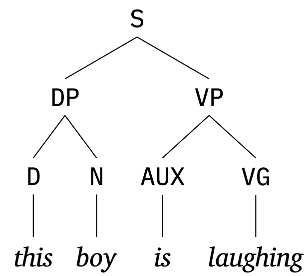
Rules are applied by replacing the nonterminal symbol on the left-hand side with the sequence on the right-hand side. The derivation begins with the start symbol and proceeds by applying rules until no nonterminal symbols are left in the string.

For a specific example, consider a CFG with $N = \{\mathtt{S}, \mathtt{DP}, \mathtt{VP}, \mathtt{D}, \mathtt{N}, \mathtt{AUX}, \mathtt{VG}\}$, $\Sigma = \{\textit{this}, \textit{boy}, \textit{laughs}, \textit{is}, \textit{laughing}\}$, and R as given in Figure 2a. CFGs are often represented simply as a list of rewrite rules, since N and Σ are recoverable from R. The phrase-structure tree, or

Figure 2: A toy context-free grammar

S → DP VP
DP → D N
D → *this*
N → *boy*
VP → *laughs*
VP → AUX VG
AUX → *is*
VG → *laughing*

(a) Rules



(b) Parse tree of *this boy is laughing*

parse tree, associated with the derivation of the string *this boy is laughing*, is shown in Figure 2b. In a phrase-structure tree for a context-free derivation, each internal node corresponds to the left-hand side of a rule, and its children to symbols on the rule's right-hand side.

Context-free grammars have been shown by Shieber (1985) to be insufficiently powerful to describe patterns found in natural language syntax. Nevertheless, they have useful connections to other grammar formalisms that will be discussed in Subsection 3.3.

2.2 *Encoding FGs*

Now let us consider a corpus of three strings over $\Sigma = \{$*this*, *boy*, *girl*, *laughs*, *jumps*, *and*$\}$:

this boy laughs;
this girl jumps;
this boy jumps and this girl laughs.

The three CFGs in Figure 3 all generate these strings but assign different phrase-structure trees to them (Figure 4). The first one (Figure 3a) is too permissive and *overgenerates* by producing every non-empty string in Σ^*, including those that are not grammatical sentences in English, such as **laughs jumps girl and this this*. In linguistic terms, Figure 3a assigns the same syntactic category to every word without regard to their distribution. The second grammar (Figure 3b) is too constraining and *overfits* the corpus: it generates the three sentences

Figure 3: Three context-free grammars

(a) Overgenerating

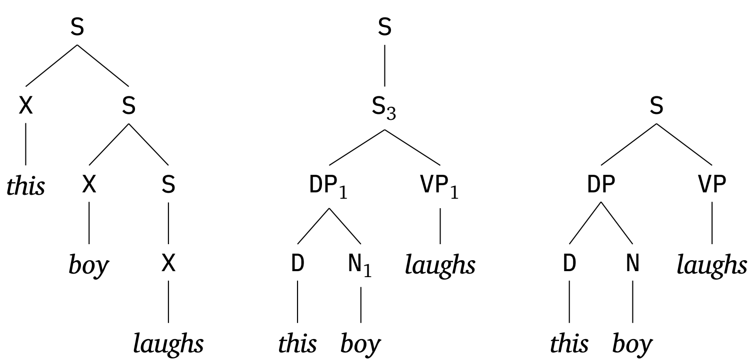
$$\begin{aligned}
\mathtt{S} &\rightarrow \mathtt{X\ S}\\
\mathtt{S} &\rightarrow \mathtt{X}\\
\mathtt{X} &\rightarrow \textit{this}\\
\mathtt{X} &\rightarrow \textit{boy}\\
\mathtt{X} &\rightarrow \textit{girl}\\
\mathtt{X} &\rightarrow \textit{laughs}\\
\mathtt{X} &\rightarrow \textit{jumps}\\
\mathtt{X} &\rightarrow \textit{and}
\end{aligned}$$

(b) Overfitting

$$\begin{aligned}
\mathtt{S} &\rightarrow \mathtt{S_1\ CONJ\ S_2}\\
\mathtt{S} &\rightarrow \mathtt{S_3}\\
\mathtt{S} &\rightarrow \mathtt{S_4}\\
\mathtt{S_1} &\rightarrow \mathtt{DP_1\ VP_2}\\
\mathtt{S_2} &\rightarrow \mathtt{DP_2\ VP_1}\\
\mathtt{S_3} &\rightarrow \mathtt{DP_1\ VP_1}\\
\mathtt{S_4} &\rightarrow \mathtt{DP_2\ VP_2}\\
\mathtt{DP_1} &\rightarrow \mathtt{D\ N_1}\\
\mathtt{DP_2} &\rightarrow \mathtt{D\ N_2}\\
\mathtt{D} &\rightarrow \textit{this}\\
\mathtt{N_1} &\rightarrow \textit{boy}\\
\mathtt{N_2} &\rightarrow \textit{girl}\\
\mathtt{VP_1} &\rightarrow \textit{laughs}\\
\mathtt{VP_2} &\rightarrow \textit{jumps}\\
\mathtt{CONJ} &\rightarrow \textit{and}
\end{aligned}$$

(c) Balanced

$$\begin{aligned}
\mathtt{S} &\rightarrow \mathtt{S\ CONJ\ S}\\
\mathtt{S} &\rightarrow \mathtt{DP\ VP}\\
\mathtt{DP} &\rightarrow \mathtt{D\ N}\\
\mathtt{D} &\rightarrow \textit{this}\\
\mathtt{N} &\rightarrow \textit{boy}\\
\mathtt{N} &\rightarrow \textit{girl}\\
\mathtt{VP} &\rightarrow \textit{laughs}\\
\mathtt{VP} &\rightarrow \textit{jumps}\\
\mathtt{CONJ} &\rightarrow \textit{and}
\end{aligned}$$



(a) Overgenerating (b) Overfitting (c) Balanced

Figure 4: Phrase-structure trees for *this boy laughs*

above and nothing else. Finally, Figure 3c strikes a balance by making a number of correct generalizations – for instance, that *boy* and *girl* have the same distribution and should be generated by the same nonterminal symbol. This grammar generates every sentence in the corpus, but also an infinite set of grammatical sentences absent from the corpus such as *this boy laughs and this girl jumps and this girl laughs*.

We will now adopt a straightforward encoding scheme and notation after Katzir (2014) and Rasin and Katzir (2019) to see how this intuition translates into MDL values. The first step is to convert each nonterminal in N and each terminal in Σ, along with an additional *delimiter* symbol, #, into a binary string. Then the number of bits needed to represent each symbol is :

$$\lceil \log_2(|N| + |\Sigma| + 1) \rceil,$$

where $\lceil \; \rceil$ indicates rounding up to the nearest integer. For simplicity, this encoding scheme assigns binary strings of equal length to all symbols; see Section 6 for discussion of alternatives. It takes four bits to encode a symbol in Figure 3a or 3c, while the symbols of 3b require five bits each (Figure 5).

We can now use these binary representations to encode each grammar. Since context-free rewrite rules follow a very specific format (one nonterminal symbol on the left-hand side, a sequence of terminal and nonterminal symbols on the right-hand side), a grammar can be unambiguously represented by concatenating all symbols in each rule and concatenating all rules together, separated by delimiters, as shown in Figure 6.

This step converts a grammar into a single binary string. Formalizing, the length of this string equals

$$\sum_{\langle \alpha, \beta \rangle \in R} \big(|\alpha| + |\beta| + 1\big) \times \lceil \log_2(|N| + |\Sigma| + 1) \rceil$$

and represents the size of the entire grammar in bits.

2.3 *Encoding corpora*

Our next step is to encode the data, which is done by using phrase-structure trees of sentences in the corpus. We start at the root (labeled with the start symbol, S) and traverse the tree in preorder – i.e. read the current node, then recursively traverse its children in the same

#	0000
S	0001
X	0010
this	0011
boy	0100
girl	0101
laughs	0110
jumps	0111
and	1000

(a) Overgenerating

#	00000
S	00001
S_1	00010
S_2	00011
S_3	00100
S_4	00101
DP_1	00110
DP_2	00111
D	01000
N_1	01001
N_2	01010
VP_1	01011
VP_2	01100
CONJ	01101
this	01110
boy	01111
girl	10000
laughs	10001
jumps	10010
and	10011

(b) Overfitting

#	0000
S	0001
DP	0010
D	0011
N	0100
VP	0101
CONJ	0110
this	0111
boy	1000
girl	1001
laughs	1010
jumps	1011
and	1100

(c) Balanced

Figure 5: Encoding tables for symbols

S	→	X	S	#	S	→	X	#	X	→	*this*	#	…
0001		0010	0001	0000	0001		0010	0000	0001		0011	0000	

Figure 6: Encoding of the overgenerating grammar (Figure 3a)

way, from left to right. At each internal node, the number of possible choices equals the number of different rules whose left-hand side corresponds to the node's label. Formally, given the left-hand side α, the cost of encoding a rule in bits is: $\lceil \log_2(|\{\beta : \langle \alpha, \beta \rangle \in R\}|) \rceil$.

Using the overfitting grammar (Figure 3b) as an example, the cost of using the rule $S \rightarrow S_3$ given the left-hand side S is $\lceil \log_2 3 \rceil = 2$ bits, because there are 3 different rules whose left-hand side is S. If there is

$\mathtt{S} \to \mathtt{X}\ \mathtt{S}$	0
$\mathtt{S} \to \mathtt{X}$	1
$\mathtt{X} \to \textit{this}$	000
$\mathtt{X} \to \textit{boy}$	001
$\mathtt{X} \to \textit{girl}$	010
$\mathtt{X} \to \textit{laughs}$	011
$\mathtt{X} \to \textit{jumps}$	100
$\mathtt{X} \to \textit{and}$	101

(a) Overgenerating

$\mathtt{S} \to \mathtt{S}_1\ \mathtt{CONJ}\ \mathtt{S}_2$	00
$\mathtt{S} \to \mathtt{S}_3$	01
$\mathtt{S} \to \mathtt{S}_4$	10
$\mathtt{S}_1 \to \mathtt{DP}_1\ \mathtt{VP}_2$	ϵ
$\mathtt{S}_2 \to \mathtt{DP}_2\ \mathtt{VP}_1$	ϵ
$\mathtt{S}_3 \to \mathtt{DP}_1\ \mathtt{VP}_1$	ϵ
$\mathtt{S}_4 \to \mathtt{DP}_2\ \mathtt{VP}_2$	ϵ
$\mathtt{DP}_1 \to \mathtt{D}\ \mathtt{N}_1$	ϵ
$\mathtt{DP}_2 \to \mathtt{D}\ \mathtt{N}_2$	ϵ
$\mathtt{D} \to \textit{this}$	ϵ
$\mathtt{N}_1 \to \textit{boy}$	ϵ
$\mathtt{N}_2 \to \textit{girl}$	ϵ
$\mathtt{VP}_1 \to \textit{laughs}$	ϵ
$\mathtt{VP}_2 \to \textit{jumps}$	ϵ
$\mathtt{CONJ} \to \textit{and}$	ϵ

(b) Overfitting

$\mathtt{S} \to \mathtt{S}\ \mathtt{CONJ}\ \mathtt{S}$	0
$\mathtt{S} \to \mathtt{DP}\ \mathtt{VP}$	1
$\mathtt{DP} \to \mathtt{D}\ \mathtt{N}$	ϵ
$\mathtt{D} \to \textit{this}$	ϵ
$\mathtt{N} \to \textit{boy}$	0
$\mathtt{N} \to \textit{girl}$	1
$\mathtt{VP} \to \textit{laughs}$	0
$\mathtt{VP} \to \textit{jumps}$	1
$\mathtt{CONJ} \to \textit{and}$	ϵ

(c) Balanced

Figure 7: Encoding tables for rules

Table 1: Encoding costs for Figure 3a–3c (bits)

	Grammar	Corpus	MDL
Overgenerating (Figure 3a)	100	52	152
Overfitting (Figure 3b)	265	6	271
Balanced (Figure 3c)	124	13	137

only one possible right-hand side, as with the rule $\mathtt{S}_3 \to \mathtt{DP}_1\ \mathtt{VP}_1$, the cost is 0 bits because there is no choice to make, and the corresponding encoding is ϵ, the empty string.

In this way, we can now give binary string representations to all rules, as shown in Figure 7. To encode a tree, we concatenate all rule encodings in the order in which the nodes are traversed (Figure 8).

This explicit encoding scheme highlights the differences in how each grammar describes the data. Overall costs for the three grammars and data are given in Table 1. The overgenerating grammar (Figure 3a) is very short but requires a lengthy encoding of the corpus.

$$\underbrace{\mathtt{S} \rightarrow \mathtt{X}\ \mathtt{S}}_{0}\ \underbrace{\mathtt{X} \rightarrow \textit{this}}_{000}\ \underbrace{\mathtt{S} \rightarrow \mathtt{X}\ \mathtt{S}}_{0}\ \underbrace{\mathtt{X} \rightarrow \textit{boy}}_{001}\ \underbrace{\mathtt{S} \rightarrow \mathtt{X}}_{1}\ \underbrace{\mathtt{X} \rightarrow \textit{laughs}}_{011}$$

(a) Overgenerating

$$\underbrace{\mathtt{S} \rightarrow \mathtt{S}_3}_{01}\ \underbrace{\mathtt{S}_3 \rightarrow \mathtt{DP}_1\ \mathtt{VP}_1}_{\epsilon}\ \underbrace{\mathtt{DP}_1 \rightarrow \mathtt{D}\ \mathtt{N}_1}_{\epsilon}\ \underbrace{\mathtt{D} \rightarrow \textit{this}}_{\epsilon}\ \underbrace{\mathtt{N}_1 \rightarrow \textit{boy}}_{\epsilon}\ \underbrace{\mathtt{VP}_1 \rightarrow \textit{laughs}}_{\epsilon}$$

(b) Overfitting

$$\underbrace{\mathtt{S} \rightarrow \mathtt{DP}\ \mathtt{VP}}_{1}\ \underbrace{\mathtt{DP} \rightarrow \mathtt{D}\ \mathtt{N}}_{\epsilon}\ \underbrace{\mathtt{D} \rightarrow \textit{this}}_{\epsilon}\ \underbrace{\mathtt{N} \rightarrow \textit{boy}}_{0}\ \underbrace{\mathtt{VP} \rightarrow \textit{laughs}}_{0}$$

(c) Balanced

Figure 8: Encoding of *this boy laughs*

The overfitting grammar (Figure 3b) makes describing the corpus extremely easy at the cost of a long encoding of the grammar itself.

The sum of the grammar and corpus encoding favors the balanced grammar (Figure 3c) – which aligns with a linguistic intuition of which of the three grammars is best.[5]

3 MINIMALIST GRAMMARS

3.1 *Lexical items, Merge, and Move*

Minimalist grammars (MGs, Stabler 1997) provide a formal implementation of Minimalist syntax (Chomsky 1995, 2000), which is used throughout the paper. In order to keep the paper fully self-contained, this section introduces the MG formalism and provides examples of derivations.

[5] An editor has pointed out that the following "extremely overfitting" grammar would outperform the balanced grammar given the corpus discussed above:

$\mathtt{S} \rightarrow$ *this boy laughs*

$\mathtt{S} \rightarrow$ *this girl jumps*

$\mathtt{S} \rightarrow$ *this boy jumps and this girl laughs*

This grammar introduces no nonterminal symbols other than S, which works well for the three-sentence corpus. However, we can easily construct an example over

An MG specifies a finite set of lexical items and encodes their selectional properties in the form of *syntactic features*. A feature of the form `x` corresponds to a syntactic *category*, whereas `=x`, `=>x`, and `x=` are selecting features which indicate that an expression is looking to merge (on the right, on the right with head movement, or on the left, respectively[6]) with something of that category. Similarly, `-x` indicates

---

the same Σ that would make better use of additional nonterminals and show extreme overfitting underperform on a slightly larger dataset.

Let us add to the original corpus a sentence containing $n+1$ clauses (for some n) of the form: *this boy laughs* $\underbrace{\textit{and this girl jumps ... and this girl jumps}}_{n \text{ times}}$. On the grammar side, the overgenerating and balanced grammar can already generate it. The overfitting grammar needs to add two new rules: $\mathtt{S} \rightarrow \mathtt{S}_5$ and $\mathtt{S}_5 \rightarrow \mathtt{S}_3\ \underbrace{\mathtt{CONJ}\ \mathtt{S}_4 \ldots \mathtt{CONJ}\ \mathtt{S}_4}_{n \text{ times}}$. The extremely overfitting grammar needs one rule, $\mathtt{S} \rightarrow$ *this boy laughs* $\underbrace{\textit{and this girl jumps ... and this girl jumps}}_{n \text{ times}}$, costing three bits per symbol to encode. On the corpus side, the overgenerating grammar would pay one bit per word in the sentence to choose between $\mathtt{S} \rightarrow \mathtt{S}\ \mathtt{X}$ and $\mathtt{S} \rightarrow \mathtt{X}$ and three bits per word to pick the terminal. For the balanced grammar, the additional cost is n instances of $\mathtt{S} \rightarrow \mathtt{S}\ \mathtt{CONJ}\ \mathtt{S}$, $n+1$ instances of $\mathtt{S} \rightarrow \mathtt{DP}\ \mathtt{VP}$, and two more bits per clause to pick the noun and the verb. Both the overfitting and the extremely overfitting grammar would see a flat 2-bit increase.

	Grammar cost increase	Corpus cost increase
Overgenerating	0	$(3+4n)+3\times(3+4n)$
Balanced	0	$n+(n+1)+2\times(n+1)$
Overfitting	$5\times 3+5\times(3+2n)$	2
Extremely overfitting	$3\times(5+4n)$	2

It is easy to see that the balanced approach and even the overfitting one outperform extreme overfitting at higher values of n. While not very natural (as the number of distinct words is limited to keep it simple), this example shows how the initial investment of setting up syntactic structure (as additional nonterminal symbols and rules) takes more than a toy corpus to pay off.

[6] The choice to distinguish between left and right selection puts linear order under lexical control. One alternative, commonly adopted in the literature on MGs, is to have the first dependent of a head merge on the right, and all subsequent dependents on the left – a version of the Linear Correspondence Axiom (Kayne 1994).

the requirement to move, and +x and ⋆x mean that the expression attracts a sub-expression with that feature into its specifier position (overtly or covertly).

In order to define an MG, one has to specify the following:

- *Base*, a finite set of syntactic *categories*. The set *Syn* of syntactic features is defined as the union of *Base* and the following sets:

$$\begin{aligned} Sel &= \{\texttt{=x} : \texttt{x} \in Base\} \cup && (\textit{right selectors}) \\ &\quad \{\texttt{=>x} : \texttt{x} \in Base\} \cup && (\textit{morphological selectors}) \\ &\quad \{\texttt{x=} : \texttt{x} \in Base\} && (\textit{left selectors}) \\ Lic &= \{\texttt{+x} : \texttt{x} \in Base\} \cup && (\textit{overt licensors}) \\ &\quad \{\star\texttt{x} : \texttt{x} \in Base\} && (\textit{covert licensors}) \\ Lee &= \{\texttt{-x} : \texttt{x} \in Base\} && (\textit{licensees}) \end{aligned}$$

  Each syntactic feature is then characterized by its *name* (drawn from *Base*) and *type* (category, right/morphological/left selector, overt/covert licensor, or licensee). Selectors and licensors together are called *attractors*, and categories and licensees are called *attractees*;

- Σ, a finite alphabet of phonological segments;

- *Lex*, a lexicon, or finite set of *lexical items*. Each lexical item (LI) is a pair $\langle s, \delta \rangle$ (written as $s :: \delta$), where $s \in \Sigma^*$ is a (phonological) *string component* and $\delta \in Syn^*$ is a list of syntactic features, or *feature bundle*. In cases that are not ambigous, we will sometimes refer to specific lexical items by their string components.

MGs are commonly defined by simply stating a lexicon, which also implicitly fixes a set of categories and an alphabet of segments. Because of this, and for the sake of convenience, we will use the terms "grammar" and "lexicon" interchangeably when referring to MGs. An example grammar of five lexical items is given in Figure 9.[7]

Syntactic *expressions* generated by an MG are binary trees whose terminal nodes are labeled with LIs (which themselves are referred to

[7] In this example, all complements are merged on the right. The subject DP then moves to the position to the left of the finite verb.

Figure 9:
A toy MG

this :: =n d -k
boy :: n
is :: =g +k t
laugh :: =d v
-ing :: =>v g
-s :: =>v +k t

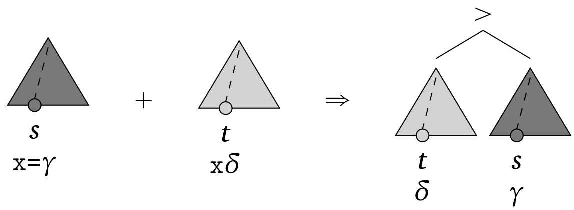
as *atomic expressions*). The first feature of each LI is *syntactically active*, i.e. accessible to structure building operations. These operations, **merge** and **move**, consume matching attractors and attractees to generate complex expressions from *Lex*.

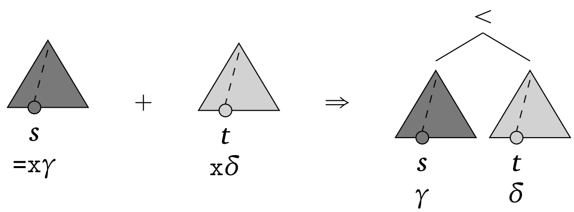
Following Stabler (2001), head movement is implemented as a subtype of **merge**, driven by features of the form =>x , which we will call *morphological selectors*. This version of head movement is defined in terms of head-complement relations, which means that this type is restricted to the first feature in the bundle. This addition allows minimalist lexica to reflect structure within complex words.[8] We will refer to lexical items bearing these selector features as *affixes* and write their string components starting with a hyphen, following a common notational convention.

The set of expressions *Exp* is defined as the closure of *Lex* under **merge** and **move**.

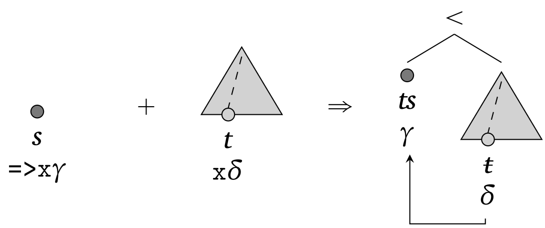
[8]Regarding the issue of complex words, multiple options have been explored in the literature. Head movement creates a chain of heads that is pronounced in the highest head position. Lowering or affix hopping, on the other hand, allows an affix to attach to the head of its complement, with the whole word being pronounced in the lower position. Unification of head movement and lowering is one of the defining features of Brody's (2000) Mirror Theory. In a similar vein, Arregi and Pietraszko (2018) propose a generalized account of head movement and lowering as high and low spellouts of a single syntactic operation, *unified head movement*. Stabler (2001) incorporates both head movement and lowering into MGs as subtypes of selector features. Brody's framework was adapted into minimalist grammars by Kobele (2002), and was proven not to affect the weak generative capacity of the formalism. Arregi and Pietraszko's (2018) proposal is similarly implemented by Kobele (to appear). In this paper, we consider all complex words to be formed by head movement. This decision is explicitly treated as a simplifying assumption.

- **merge** : $(Exp \times Exp) \rightarrow Exp$ is a binary function that targets selectors and categories and combines two syntactic expressions into a new one. The dependent is merged on the left if the selector is of the form x=, and on the right if it is of the form =x:





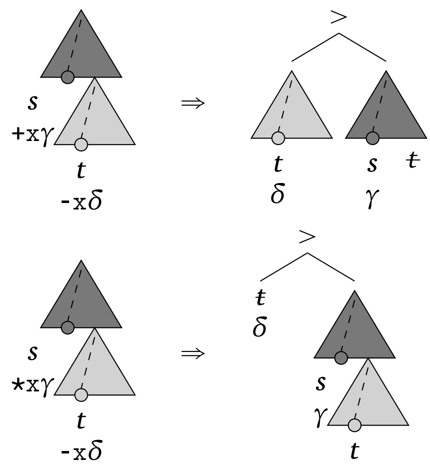
- **merge** with head movement is triggered by selectors of the form =>x. It proceeds as right **merge** and concatenates the string component of the head of the complement with that of the resulting expression:[9]



- **move** : $Exp \times Exp$ is a relation that matches a licensor with a licensee within the same expression. Overt licensors (+x) cause the moving subtree to become a (left) sister of the head, leaving behind an empty node without a string component or syntactic features. Covert **move** (⋆x) leaves the string component behind:[10]

[9]We indicate a moved string t as ~~*t*~~. This is a notational convenience; formally, the empty node contains ϵ, the empty string.

[10]This version of covert movement, which displaces syntactic features but leaves the string component in its base position, is in line with Stabler 1997. It fixes the position of a sub-expression once it has been covertly moved, rendering its string component inaccessible to future instances of (overt) **move**. Though



While there are many ways to limit the number of features which may be syntactically active at any given time, a simple one with desirable computational properties stipulates that only one feature of each name may be the first feature of any feature bundle in an expression. In particular, this means that the number of movable subtrees in any expression is limited by the size of *Base*. This restriction is known as the Shortest Move Constraint, or SMC. With the SMC in place, **move** becomes a function.

A single lexical item (atomic expression) is considered its own *head*. For complex structures formed by **merge** or **move**, the expression with the attractor becomes the head of the new expression; and the one with the attractee becomes its *dependent*. We label the parent node with $<$ if the head is on the left or $>$ if the head is on the right. The dependent introduced by the first attractor of an LI is its *complement*, and all subsequent dependents are *specifiers*. Matched features are *checked*, or deleted, making the next feature in the bundle accessible for syntactic operations. Checked features are no longer visible to syntax. We will sometimes keep them in representations for clarity, in which case they will be marked as $\boxed{\texttt{x}}$.

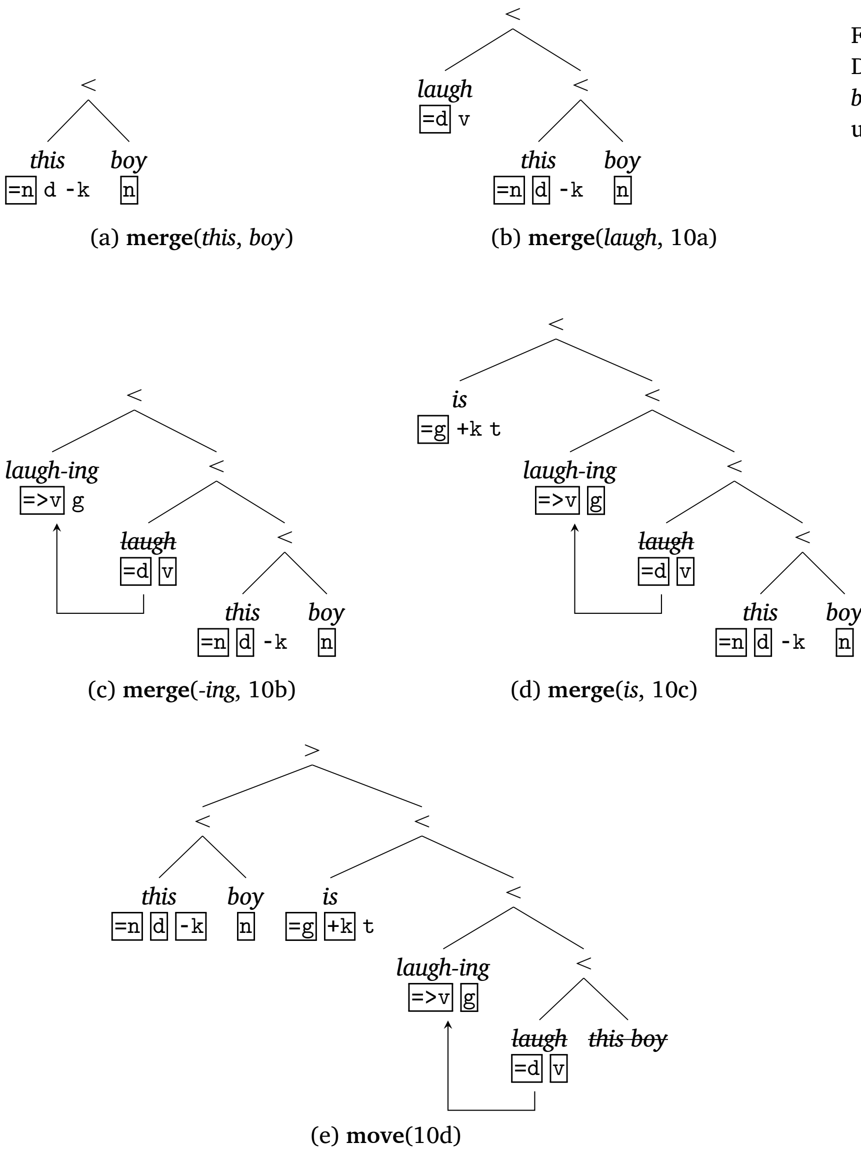
An expression with no unchecked features except for some category x on its head is called a *complete expression* of that category.

---

restricted, this implementation has been used in previous work on MGs (see e.g. Torr and Stabler 2016) and is sufficient for our purposes.

We will be primarily concerned with complete expressions of category `t` (for Tense) or `c` (for Complementizer) and their string yields (*sentences*).

The lexicon in Figure 9 generates, among others, the five expressions in Figure 10. In Figure 10a, **merge** applies to *this* and *boy*, whose



(a) **merge**(*this*, *boy*)

(b) **merge**(*laugh*, 10a)

(c) **merge**(-*ing*, 10b)

(d) **merge**(*is*, 10c)

(e) **move**(10d)

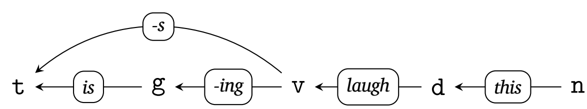
Figure 10: Derivation of *this boy is laugh-ing* using Figure 9

feature bundles start with the matching features =n and n, respectively. Both =n and n are deleted. In Figure 10b, **merge** once again targets two expressions: *laughing*'s feature bundle starts with =d, and Figure 10a has d as its first feature. Next, we **merge** in *-ing*. Its selector feature, =>v, triggers head movement, concatenating *laugh* and *-ing* together (Figure 10c). Another **merge** step (Figure 10d) checks the =g and g features, combining *is* with Figure 10c. In Figure 10e, the matching features are +k on *is* and -k on *this*. The DP is **moved** into the specifier position of *is*, which becomes the head of the new expression. This is a complete expression of category t, whose string yield is *this boy is laugh-ing*.

3.2 *Grammar graphs*

When it comes to visualizing an entire grammar, the default option is to list all lexical items, as in Figure 9. As mentioned before, such a list contains all information required to define an MG. However, it does not provide a good overview of expressions generated by the grammar in question. While it works for very small toy examples, larger grammars with dozens or hundreds of LIs can become difficult to read quickly. A convenient alternative for showing the head-complement relations within a set of lexical items is a directed multigraph whose vertices correspond to category features, and edges to lexical items. To better understand, consider Figure 11 which illustrates this representation using the same data as Figure 9.

Figure 11: Head-complement relations within Figure 9



This graph does not reflect all relations in the lexicon, since it ignores any **move** relations as well as any specifiers formed by **merge**. Lexical items without any selectors (such as *boy* :: n) don't contribute an edge to the graph. Instead, it focuses on a subset of relations which are relevant for morphologically complex words. Each path from n to t indicates a possible sequence of LIs along the clausal spine. Multiple paths between vertices indicate that there is more than one option available at that point in the derivation. For instance, there is an edge connecting `v` and `t`, as well as an alternative path between these categories. This reflects the fact that an expression of category `v` can be selected either by *-s* :: `=>v +k t` or by *-ing* :: `=>v g`, in the latter case producing a valid complement for *is* :: `=g +k t`.

Relation to CFGs 3.3

By definition, the two structure-building operations of MGs – **merge** and **move** – can only target subtrees whose heads bear an unchecked syntactic feature. Therefore, much of the derived structure is *syntactically inert*: once all features of a lexical item have been deleted, its position in the structure is fixed. The only elements that matter for syntax are those still capable of rearranging with respect to each other – namely, the head of the entire expression (via head movement) and any *movers*, or subtrees headed by lexical items with an unchecked licensee feature. With the SMC in place, the number of such subtrees in any given expression is finite, limited by the number of distinct licensee features in the grammar. Thus, a derived tree can be flattened into a much more compact structure containing all information relevant for **merge** and **move** – a sequence of strings annotated with unchecked features.

This insight gives rise to the so-called *chain notation* for MGs (Stabler 2001; Stabler and Keenan 2003). In short, each expression sans movers is represented as an *initial chain* – a triple of strings corresponding to the head and material to its left and right, annotated with features of the head. Movers within the tree are represented by separate *non-initial chains*, the number of which cannot be greater than the size of *Base* (see Figure 12).

$$\underbrace{(\textit{left},\ \textit{head},\ \textit{right}) : \texttt{features}}_{\text{Initial chain}},\ \underbrace{\textit{mover}_1 : \texttt{licensees},\ \textit{mover}_2 : \texttt{licensees},\ \ldots}_{\text{Non-initial chains}}$$

Figure 12: Schematic representation of a chain-based expression

Lexical items consist of only an initial chain, and their first and last components are empty strings, as shown in Figure 13.

The structure-building operations are redefined in terms of string tuples. Informally, the outcome of **merge** depends on whether the dependent has reached its final position in the structure or is going to

Figure 13:
Chain-based counterpart of Figure 9

$\langle \epsilon, \textit{this}, \epsilon \rangle$ :: `=n d -k`

$\langle \epsilon, \textit{boy}, \epsilon \rangle$ :: `n`

$\langle \epsilon, \textit{is}, \epsilon \rangle$ :: `=g +k t`

$\langle \epsilon, \textit{laugh}, \epsilon \rangle$ :: `=d v`

$\langle \epsilon, \textit{-ing}, \epsilon \rangle$ :: `=>v g`

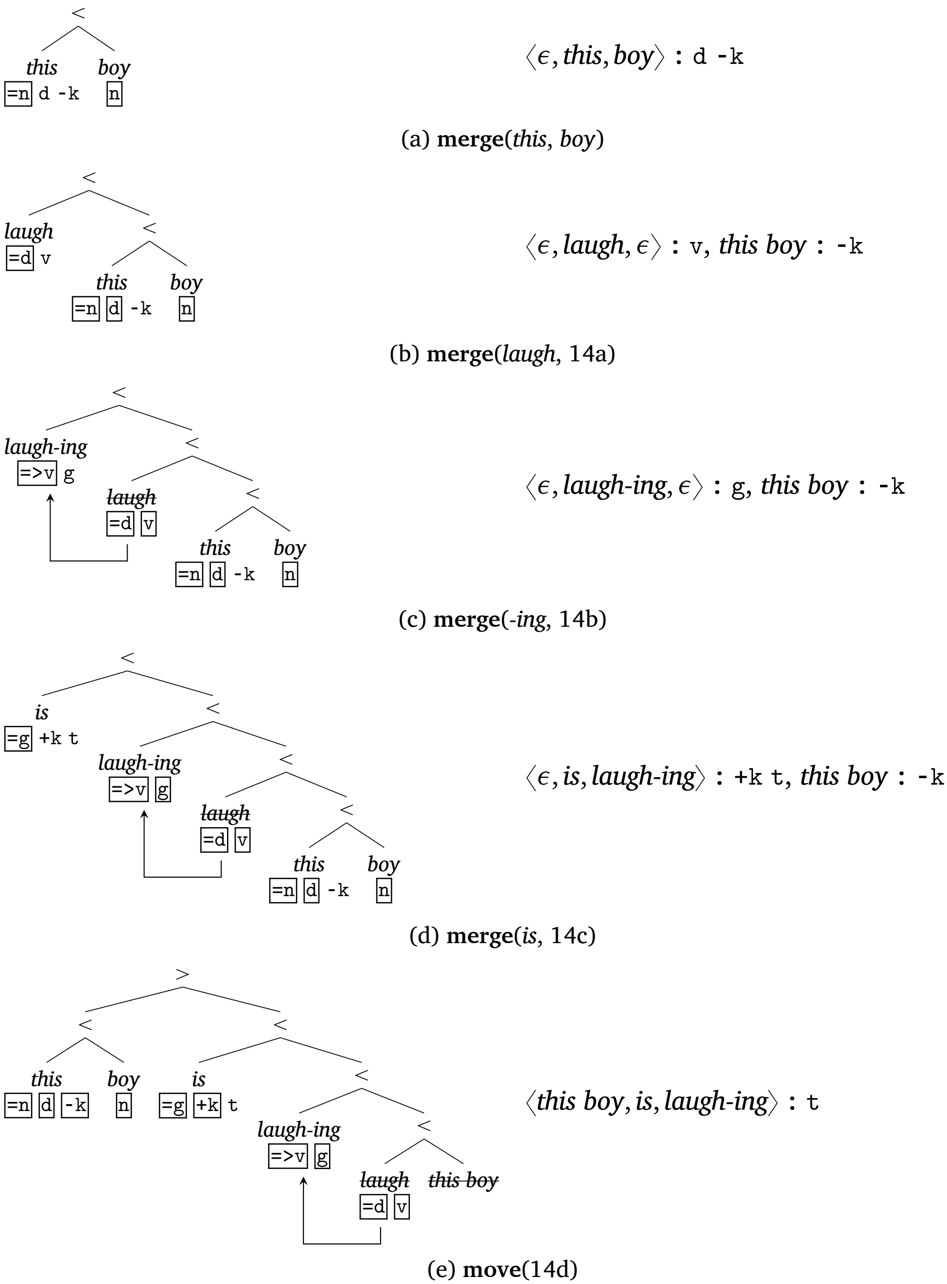
$\langle \epsilon, \textit{-s}, \epsilon \rangle$ :: `=>v +k t`

move later in the derivation. In the former case, its initial chain is concatenated together and attached to the leftmost (for left **merge**) or rightmost (for right **merge**) component of the initial chain. In the latter case, the dependent forms a non-initial chain ready to be targeted by **move**. Similarly, **move** comes in multiple varieties depending on whether the moving subtree has reached its surface position. A complete expression of category `x` consists of just an initial chain annotated with only the feature `x`.

The derivation of *this boy is laugh-ing*, shown before in Figure 10, is repeated in Figure 14, with each derivation step given as a derived tree and in chain notation side by side. In Figure 14a, *this* and *boy* are **merge**d, and the string component of the latter is concatenated into the third component of the initial chain. Next, *laugh* is **merge**d with the resulting structure (Figure 14b). Since the dependent still carries a licensee feature (`-k`), it forms a non-initial chain *this boy* annotated with `-k`. The next two steps continue building up the initial chain, leaving the single non-initial chain unaffected. Finally, Figure 14e **move**s *this boy* into the first component of the initial chain, arriving at a complete expression of category `t`.

Because chain notation is so compact, all intermediate steps in a derivation can be visualized as a single *derivation tree* by labeling each internal node with the chain-based expression corresponding to the step in question, as shown in Figure 15. Each internal node corresponds to a step in the derivation, an instance of **merge** or **move**, and the order of its children reflects their role in that step: the head precedes its dependent regardless of their relative order in the derived structure.

Derivation trees don't reflect displacement of leaves caused by **move** in the way derived trees do. For any MG, its derivation trees are



(a) **merge**(*this*, *boy*)

(b) **merge**(*laugh*, 14a)

(c) **merge**(-*ing*, 14b)

(d) **merge**(*is*, 14c)

(e) **move**(14d)

Figure 14: Derivation steps of *this boy is laugh-ing* as chain-based expressions

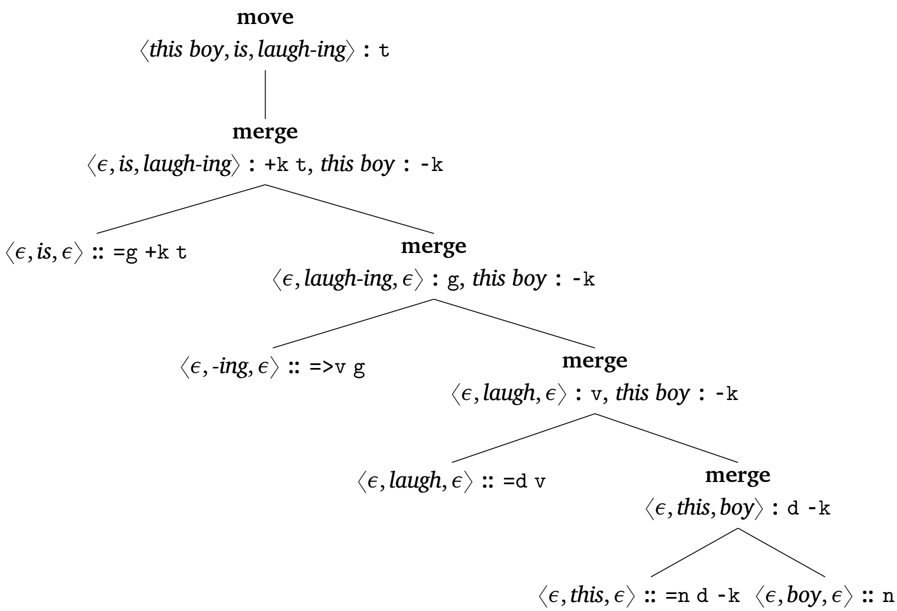


Figure 15: Chain-based derivation tree of *this boy is laugh-ing*

parse trees of a CFG; a clear presentation of this result is given in (Hale and Stabler 2005). Intuitively, constructing this CFG can be thought of as pre-computing all possible derivation steps that can be performed by the MG. The central concept here is that of a *feature configuration*, which is obtained from a chain-based expression by omitting string components;[11] the SMC guarantees that the number of such configurations is finite. The set of feature configurations is obtained as the closure of the lexicon under **merge** and **move**. Informally, the conversion process is as follows:

- Each feature configuration (written in round brackets) becomes a nonterminal symbol;
- For each feature configuration formed by **merge** or **move**, there is a rule rewriting it as the operation's argument or arguments;
- For each LI, there is a rule rewriting its feature configuration as its string component;

[11] For covert movement, feature configurations should also indicate the non-initial chains whose string components have been left behind.

• An additional rule rewrites the start symbol `S` as `(t)` or `(c)`.[12]

Derivation is then viewed as proceeding in the top-down manner of CFGs (starting with `t` and rewriting until lexical items in the leaves are reached), rather than the bottom-up manner characteristic of MGs. The CFG obtained from Figure 9 is shown in Figure 16.

`S` →	`(t)`	`(=n d -k)` →	*this*
`(t)` →	`(+k t, -k)`	`(n)` →	*boy*
`(+k t, -k)` →	`(=g +k t)  (g, -k)`	`(=g +k t)` →	*is*
`(+k t, -k)` →	`(=>v +k t)  (v, -k)`	`(=d v)` →	*laugh*
`(g, -k)` →	`(=>v g)  (v, -k)`	`(=>v g)` →	*-ing*
`(v, -k)` →	`(=d v)  (d -k)`	`(=>v +k t)` →	*-s*
`(d -k)` →	`(=n d -k)  (n)`		

Figure 16: CFG counterpart of Figure 9

ENCODING MINIMALIST GRAMMARS 4

With these definitions in place, we will now discuss how the approach of Section 2 can be adapted to implement an MDL-based metric for MGs. Consider the following four sentences:

Mary laughs;
Mary laughed;
Mary jumps;
Mary jumped.

[12]The method given in Hale and Stabler 2005 is itself an adaptation of Michaelis 1998, which shows how to convert an MG into an equivalent multiple context-free grammar (MCFG) generating the same language of sentences – yields of derived trees. MCFGs are a generalization of CFGs which operates on tuples instead of strings. Converting an MG into an equivalent MCFG is similar to the CFG construction, with a few differences. First, terminal rules rewrite feature bundles as triples of strings, corresponding to initial chains. Second, each non-terminal rule comes with a map describing how components of the argument tuples are rearranged and/or concatenated, in a way closely following chain-based **merge** and **move**.

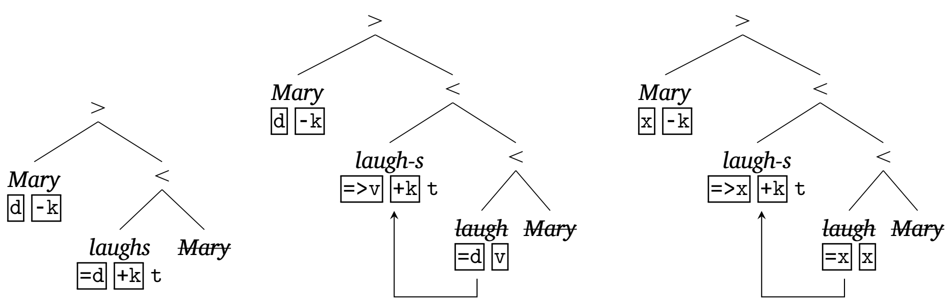
There are multiple (in fact, infinitely many) ways to construct a minimalist grammar accounting for this small corpus. Three of them are given in Figure 17.

Figure 17: Three minimalist grammars

Mary :: `d -k`	*Mary* :: `d -k`	*Mary* :: `x -k`
laughs :: `=d +k t`	*laugh* :: `=d v`	*laugh* :: `=x x`
laughed :: `=d +k t`	*jump* :: `=d v`	*jump* :: `=x x`
jumps :: `=d +k t`	*-s* :: `=>v +k t`	*-s* :: `=>x +k t`
jumped :: `=d +k t`	*-ed* :: `=>v +k t`	*-ed* :: `=>x +k t`
(a) Atomic verbs	(b) Complex verbs	(c) Overgenerating

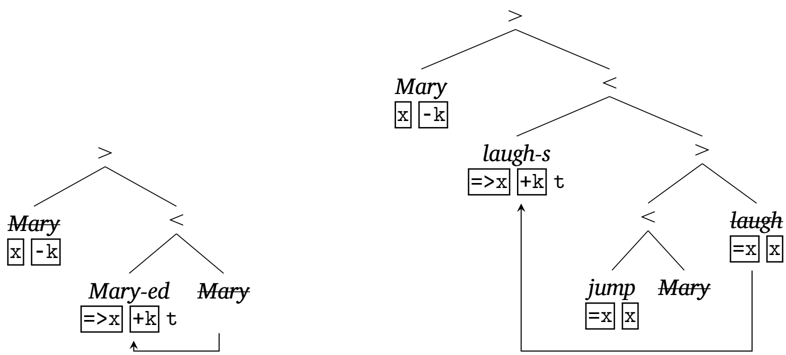
The first two grammars, Figure 17a and 17b, generate the four sentences above and no others. While they are are *weakly equivalent*, i.e. generate exactly the same set of strings, the structures they assign to these strings are different. In linguistic terms, the former treats each sentence as a single tP headed by an unsegmented verb. The latter reanalyzes each finite verb form as a complex head formed by head movement. The lexical verb directly selects its argument and forms a vP, while the affix takes the vP as its complement and is responsible for the movement of the subject into its specifier position (Figure 18b). The third grammar, Figure 17c, is also capable of generating inflected verbs in two derivation steps (Figure 18c). However, it conflates the category feature of lexical verbs with that of DPs, producing ungrammatical strings like **Mary-ed* and **Mary laugh-s* (*jump*)$^+$ (Figure 19).

To further help visualize the differences between these grammars, their graph representations are given in Figure 20.



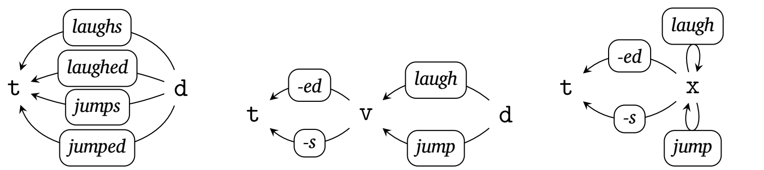
(a) Atomic verbs (b) Complex verbs (c) Overgenerating

Figure 18: Structural differences



(a) (b)

Figure 19: Overgeneration by Figure 17c



(a) Atomic verbs (b) Complex verbs (c) Overgenerating

Figure 20: Graph representations of the grammars in Figure 17

For instance, *laughed* in the atomic-verb grammar corresponds to one of the edges from `d` to `t` in Figure 20a. Its counterpart in the grammar with complex verbs is a bimorphemic word, which translates into a pair of adjacent edges: *laugh* from `d` to `v` and *-ed* from `v` to `t` (Figure 20b). In the overgenerating grammar, lexical verbs correspond to loops (Figure 20c).

Intuitively, complex verbs are an improvement over atomic verbs. By recognizing internal structure within verbs, it captures the similarities within verbal paradigms (*laughs*, *laughed* vs. *jumps*, *jumped*) and across paradigms (*laughs*, *jumps* vs. *laughed*, *jumped*). On the other hand, atomic verbs miss all these generalizations. For each new verbal paradigm encountered in the corpus (e.g. *walks*, *walked*), we would need to add two new lexical items to Figure 17a, but only one to Figure 17b. Finally, Figure 17c is a subpar choice: it shares the desirable generalizations of Figure 17b but also conflates a crucial distinction between two syntactic categories, leading to overgeneration.

What quantitative data can be used to back up this intuition? We can define an encoding scheme for MGs closely mirroring the one for context-free rules from Section 2. Since we are interested in the length of the encoding rather than the binary string itself (Grünwald 2007), we no longer round up to the nearest integer. Let *Types* = {*category, right selector, left selector, morphological selector, overt licensor, covert licensor, licensee*} denote the set of syntactic feature types, and let Σ be the set of English letters. For simplicity, as with context-free rules, we treat each LI as a sequence of symbols from the same encoding table. Then the size of a minimalist lexicon *Lex* over a set of categories *Base* is given by

$$\underbrace{\sum_{s\,::\,\delta\,\in\,Lex}\big(|s| + 2\times|\delta| + 1\big)}_{\text{total number of symbols}} \times \underbrace{\log_2(|\Sigma| + |\mathit{Types}| + |\mathit{Base}| + 1)}_{\text{cost of encoding per symbol}}.$$

Assuming that both Σ and *Types* are fixed (with $|\mathit{Types}| = 7$ and $|\Sigma| = 26$, without distinguishing between uppercase and lowercase letters), this is a function of the number of LIs and the following three metrics:

- $|\mathit{Base}|$, the number of unique category features in *Lex*;
- $\sum_{syn} = \sum_{s\,::\,\delta\,\in\,Lex}\big(|\delta|\big)$, the total count of syntactic features in *Lex*;
- $\sum_{phon} = \sum_{s\,::\,\delta\,\in\,Lex}\big(|s|\big)$, the total length of all string components in *Lex*.

Regardless of the specific encoding scheme,[13] all three values above contribute to the size difference between grammars. Table 2 summarizes the differences between the grammars with respect to individual metrics, as well as grammar size.

Table 2: Grammar metrics

	$\lvert Base \rvert$	$\sum_{syn}$	$\sum_{phon}$	Grammar (bits)
Atomic verbs (Figure 17a)	3	14	28	317.78
Complex verbs (Figure 17b)	4	12	16	236.16
Overgenerating (Figure 17c)	3	12	16	234.43

All three grammars have the same number of lexical items. However, splitting verbs into roots and affixes in Figure 17b comes at the

[13] The solution used here serves to keep the example straightforward. The choice of an encoding scheme is a meaningful decision that can lead to different grammars being optimal for the same corpus; see also discussion in Section 6.

cost of an extra category feature. This pays off by eliminating redundant strings, which almost halves $\sum_{phon}$. Moreover, four instances of `+k` are collapsed into two, yielding a small reduction of $\sum_{syn}$. The differences would be much more noticeable with larger datasets, especially with respect to open-class words, since adding a new verb to Figure 17a would have a higher cost (in both syntactic features and string components) compared to Figure 17b.

It is also easy to see how a complexity measure based solely on grammar encoding would fail to penalize overgeneration. It would incorrectly favor Figure 17c over Figure 17b, given that it achieves the same reduction of $\sum_{phon}$ and $\sum_{syn}$ without increasing $|Base|$. Similar to the results observed with CFGs, the MDL component expected to rule out the overgenerating grammar is the corpus size given the grammar. In order to calculate it, for each MG we construct a CFG generating its derivation trees, as we did in Subsection 3.3, and then reuse the encoding scheme from Section 2. The CFGs are given in Figure 21. Parse trees for *Mary laughs* as well as Figure 21c's ungrammatical structures are shown in Figure 22 and Figure 23 respectively .

The cost of encoding the corpus given Figure 21a is straightforward to calculate: there is only one choice with four options to be made in the derivation, namely rewriting (`=d +k t`) as *laughs*, *laughed*, *jumps*, or *jumped*. In Figure 21b this corresponds to two binary choices: rewriting (`=d v`) as *laugh* or *jump*, and (`=>x +k t`) as *-s* or *-ed*. Both cost 2 bits per sentence. The third grammar (Figure 21c), however, has two options for rewriting (`+k t, -k`) and two ways to expand (`x, -k`). These are the choices that make possible the ungrammatical strings in Figure 23, but they also drive up the cost of encoding each grammatical sentence to 4 bits. This is summarized in Table 3.

	Grammar	Corpus	MDL
Atomic verbs (Figure 17a)	317.78	8	325.78
Complex verbs (Figure 17b)	236.16	8	244.16
Overgenerating (Figure 17c)	234.43	16	250.43

Table 3:
Encoding costs (bits)

Once we take the length of corpus encoding into account, the overgenerating grammar is outperformed by the intuitively superior grammar with complex verbs.

S → (t)
(t) → (+k t, -k)
(+k t, -k) → (=d +k t) (d -k)
(d -k) → *Mary*
(=d +k t) → *laughs*
(=d +k t) → *laughed*
(=d +k t) → *jumps*
(=d +k t) → *jumped*

(a) Atomic verbs

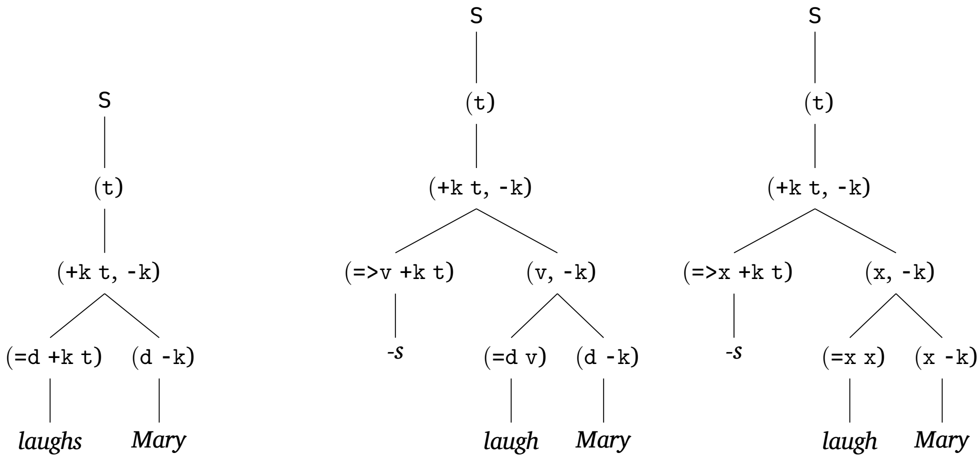
S → (t)
(t) → (+k t, -k)
(+k t, -k) → (=>v +k t) (v, -k)
(v, -k) → (=d v) (d -k)
(d -k) → *Mary*
(=d v) → *laugh*
(=d v) → *jump*
(=>v +k t) → *-s*
(=>v +k t) → *-ed*

(b) Complex verbs

S → (t)
(t) → (+k t, -k)
(+k t, -k) → (=>x +k t) (x, -k)
(+k t, -k) → (=>x +k t) (x -k)
(x, -k) → (=x x) (x -k)
(x, -k) → (=x x) (x, -k)
(x -k) → *Mary*
(=x x) → *laugh*
(=x x) → *jump*
(=>x +k t) → *-s*
(=>x +k t) → *-ed*

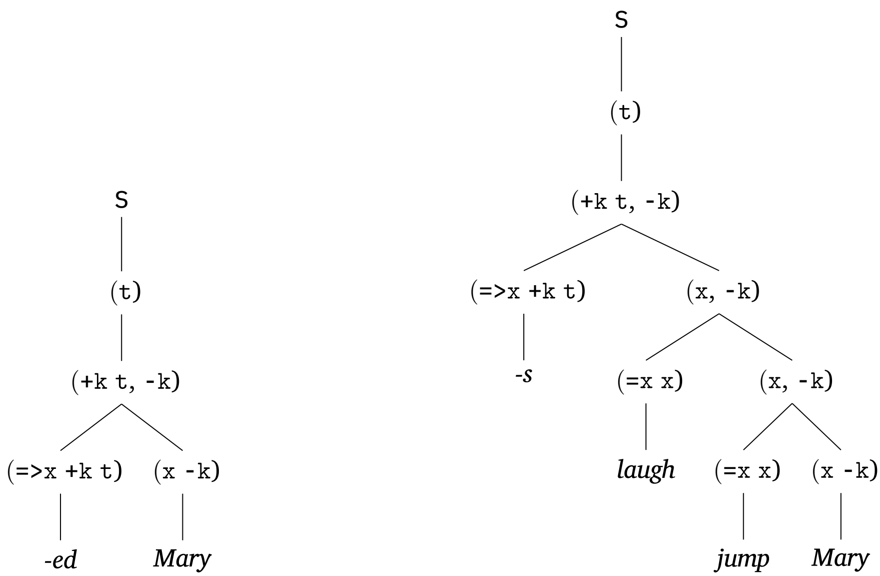
(c) Overgenerating

Figure 21: CFG counterparts of Figure 17



(a) Atomic verbs (b) Complex verbs (c) Overgenerating

Figure 22: CFG parse trees: structural differences



(a) (b)

Figure 23: CFG parse trees: overgeneration by Figure 21c

5 DOUBLE OBJECT CONSTRUCTION REVISITED

We will now take a step up from toy examples towards more interesting applications of the technique introduced above and re-examine the double object construction in the light of MDL. As pointed out in Section 1, there are two groups of approaches to sentences like *John gave Mary a book*: those which postulate a small clause complement of *give*, and those which maintain that the double object construction is monoclausal. Enumerating and analyzing all known arguments from both sides in a comprehensive way falls outside the scope of this paper. Instead, this section serves as proof of concept. In what follows, we convert a small sample of these arguments into the MG formalism and examine how the predictions of each analysis translate into higher or lower MDL values.

Let us focus on two facts regarding the English double object construction coming from two different sources. The first one is Harley and Jung (2015), who point out multiple parallels between double object structures with *give* and sentences with *have*. These are used to motivate an analysis where both *have* and *give* contain a possessive small clause headed by the abstract silent element P_{HAVE}. One of these parallels is an animacy restriction. Both possessors in *have*-clauses (1a, 1c) and Goal arguments in *give*-clauses (1b, 1d) are required to be animate, as long as the possession is alienable.

(1) a. John has a book.
  b. Brenda gave John a book.
  c. #The car has a flyer.
  d. #The advertiser gave the car a flyer.

(Harley and Jung 2015, p. 704)

The second source is Kawakami (2018), who argues against the small-clause analysis, citing a number of discrepancies between the properties of known small clause constructions (e.g. *John considers Mary angry*) and those of *give*-clauses. One of the arguments supporting this stance comes from wh-movement and ambiguity. For sentences with *consider* (2a), both the matrix clause and the small clause can be modified by *why*, yielding two different interpretations. On the other hand, the double object construction behaves as monoclausal, allowing only one reading where *why* modifies the matrix clause (2b).

(2) a. Why did John consider Mary angry at Bill?
READING: asking the reason of considering
asking the reason of being angry

b. Why did John give Mary a book?
READING: asking the reason of giving
#asking the reason of having

(Kawakami 2018, pp. 220–221)

Which of these two arguments is stronger with respect to encoding costs? We start by translating each of them into an MG. Assuming a consensus on all issues other than the double object construction, the two grammars should share most of their LIs. Since this example involves wh-movement, we consider complete expressions of category `c` rather than `t`. The shared lexical items are given in Figure 24a, and the additional LIs for *have* and *give* in the monoclausal and SC account are presented in Figure 24b and Figure 24c respectively.[14] In accordance with the simplifying assumptions stated in Section 3, we ignore non-concatenative morphology and assume a separate set of morphological rules which realize *have-s* as *has* and *do-s* as *does*.

[14]These grammars rely on using multiple lexical items with ϵ as the string component. Such empty LIs have been widely used in MGs since their conception in Stabler 1997 and can be thought of as a method of compressing the grammar. Consider, for instance, ϵ :: $\texttt{=d}_\texttt{a}$ `+k d -k`, which allows any DP of category $\texttt{d}_\texttt{a}$ to become a `d`, but not vice versa. The same restriction can be enforced without an empty LI by having two versions of each of its possible complements (*John* :: $\texttt{d}_\texttt{a}$ `-k`, *Mary* :: $\texttt{d}_\texttt{a}$ `-k`, *John* :: `d -k`, and *Mary* :: `d -k`), at the cost of introducing some redundancy into the lexicon.

More generally, empty LIs are how MGs express subcategorization requirements that are based on a hierarchy of projections rather than exact category matches. One alternative to this approach is an explicit hierarchy encoded as a partial order over selectors (Fowlie 2013) – although the cost of such an addition to the formalism would also need to be taking into account when calculating MDL. That said, certain empty LIs correspond to empty heads introduced in theoretical literature and are therefore necessary to formalize them faithfully. For example, ϵ :: `=d +k` $\texttt{d}_\texttt{a}$`= sc` represents the empty element P_{HAVE} central to the analysis of Harley and Jung (2015).

John :: $\mathtt{d_a}$ -k
Mary :: $\mathtt{d_a}$ -k
the car :: d -k
a flyer :: d -k
ϵ :: =$\mathtt{d_a}$ +k d -k

consider :: =sc V
-ϵ :: =>V +k d= v
-ϵ :: =>v x
do :: =x do
-ϵ :: =>x do
-*s* :: =>do +k t

angry :: a
ϵ :: =a d= sc
why :: w -wh
ϵ :: =sc w= sc
-ϵ :: =>t +wh c
ϵ :: =t c

(a) Shared lexical items

ϵ :: =d +k $\mathtt{d_a}$= sc
have :: =sc v
give :: =d +k d= V

(b) Monoclausal *give*

ϵ :: =d +k $\mathtt{d_a}$= sc
have :: =sc v
give :: =sc V

(c) Uniform SC *give*

ϵ :: =d +k $\mathtt{d_a}$= $\mathtt{sc_{poss}}$
-ϵ :: =>$\mathtt{sc_{poss}}$ sc
have :: =$\mathtt{sc_{poss}}$ v
give :: =$\mathtt{sc_{poss}}$ V

(d) Refined SC *give*

Figure 24: MG implementations of the double object construction

The simple solution in Figure 24c views all small clauses as having the same syntactic category, sc. This validates Kawakami's (2018) objections to the small clause analysis based on multiple differences between small clauses selected by *consider* and arguments of *give*. However, Harley and Jung (2015, p. 718) point out a way to reconcile the two groups of phenomena, suggesting a typology of small clauses. Under this view, small clauses embedded under *consider* (unlike those under *give*) include an additional projection, which explains different properties. Translating this idea into MGs yields the set of LIs given in Figure 24d. Possessive small clauses ($\mathtt{sc_{poss}}$) are selected by both *have* and *give*, and may merge with an empty LI to form expressions of category sc, which are selected by *consider*.

The animacy restriction is implemented by giving animate DPs a category feature distinct from d, $\mathtt{d_a}$. An animate DP can freely become a normal DP by merging with ϵ :: =$\mathtt{d_a}$ +k d -k, but the opposite is not possible. In other words, $\mathtt{d_a}$ occurs in all contexts that allow d, and also in some contexts where d is prohibited. The restriction on modification by *why* is added by only allowing *why* to merge with small clauses – expressions of category sc. This is done via two LIs: *why* :: w -wh and ϵ :: =sc w= sc. This fragment allows *why* to modify small clauses but not matrix clauses, since only the former are relevant for the example.[15]

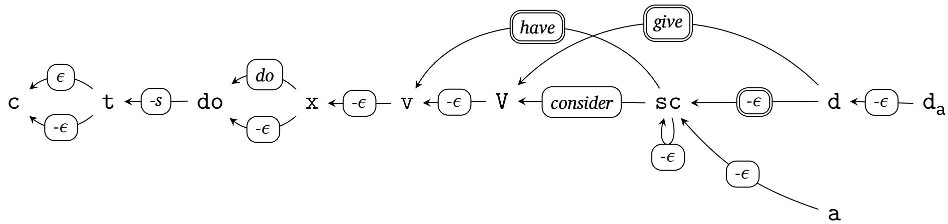
Note that all three grammars are associated with some overgeneration. First, there is no restriction requiring *do*-support in interrogative contexts, which gives rise to examples like **why consider-s John Mary angry*. In addition, all grammars except refined SC treat all small clauses as uniform, producing strings like **John have-s angry* (and, in the case of the uniform small clause analysis, **John give-s Mary angry*). As we have seen before, overgeneration does not affect grammar encoding, but will contribute to a higher cost of encoding some grammatical sentences.

Consider the head-complement graphs in Figure 25. The monoclausal *give* (Figure 25a) selects its arguments directly, whereas the uniform SC *give* (Figure 25b) takes as its complement the same small clause as *have* and shares its restriction on animacy. On the other hand, the loop at the `sc` vertex represents the position modifiable by *why*. The monoclausal *give* bypasses the category `sc`, unlike *have*; the latter, but not the former, is compatible with *why*. However, the uniform SC *have* merges with expressions of category `sc`, incorrectly allowing modification by *why*. Finally, the refined SC analysis (Figure 25c) gets around both problems by distinguishing between `sc` and $\mathtt{sc}_{\mathrm{poss}}$.

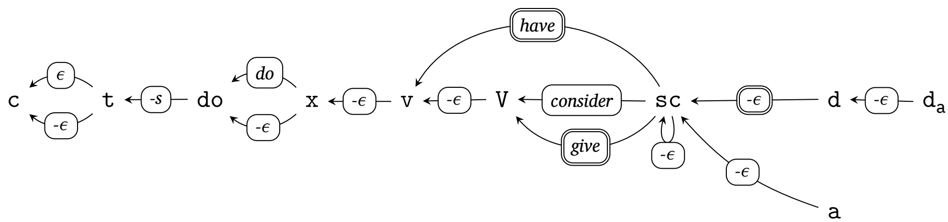
As a further illustration, some derived tree examples are given in Figure 26.

Grammar encoding costs (Table 4) reflect generalizations made by each grammar, as well as the number of category distinctions it makes. Both monoclausal and uniform SC approaches require 13 distinct categories; however, the latter has a lower cost as it reuses the abstract element heading a small clause, ϵ :: $\mathtt{=d\ +k\ d_a\!=\ sc}$, to provide arguments to both *have* and *give*. Refined SCs require an extra category, $\mathtt{sc}_{\mathrm{poss}}$, as well as an additional lexical item, -ϵ :: $\mathtt{=sc_{poss}\ sc}$, so this grammar ends up having the highest encoding cost.

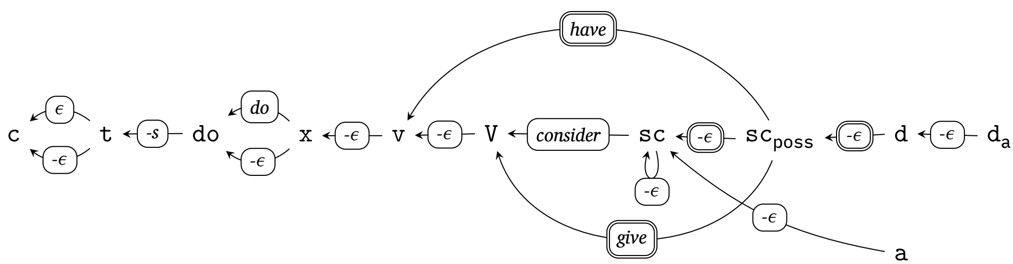
[15] For the sake of completeness, it would be easy to add modification of matrix clauses by introducing one more empty lexical item: -ϵ :: `=>v w= v`. Then the grammar would generate different structures corresponding to different readings of *consider*-clauses: *why* [*do-s John consider* [*Mary angry*] ~~*why*~~] vs. *why* [*do-s John consider* [*Mary angry* ~~*why*~~]] (cf. item 2a).



(a) Monoclausal



(b) Uniform SC

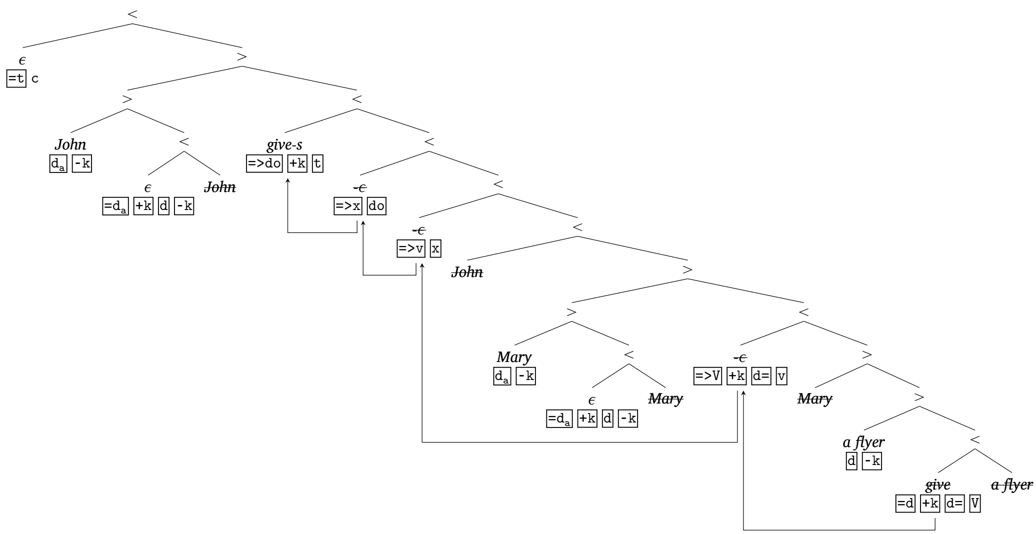


(c) Refined SC

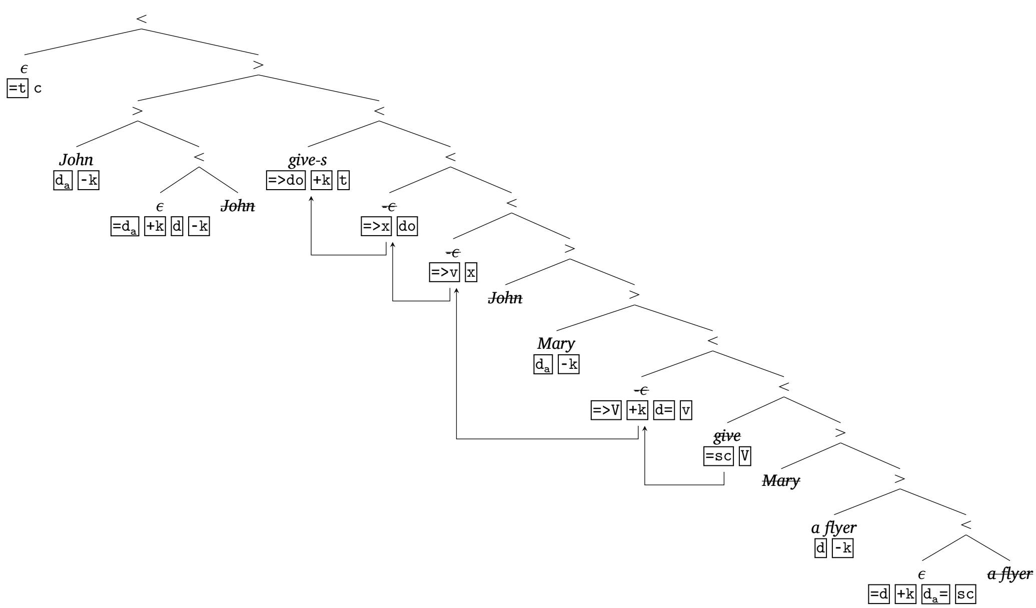
Figure 25: Head-complement graphs of MGs in Figure 24; LIs not shared by all grammars are highlighted with double frames

Table 4: Grammar metrics for the double-object construction

	$\lvert Base \rvert$	$\sum_{syn}$	$\sum_{phon}$	Grammar (bits)
Monoclausal	13	51	50	955.39
Uniform SC	13	49	50	933.17
Refined SC	14	51	50	966.20



(a) Monoclausal



(b) Uniform SC

Figure 26: Derived trees for *John give-s Mary a flyer*

Table 5: Sentence encoding costs for the double-object construction (bits)

	Monoclausal	Uniform SC	Refined SC
John give-s Mary a flyer	$6\log_2 2 + 3\log_2 3 \approx 10.75$	$7\log_2 2 + 2\log_2 3 \approx 10.17$	$6\log_2 2 + 2\log_2 3 \approx 9.17$
Mary have-s a flyer	$5\log_2 2 + \log_2 3 \approx 6.58$	$5\log_2 2 + \log_2 3 \approx 6.58$	$4\log_2 2 + \log_2 3 \approx 5.58$
John consider-s Mary angry	$7\log_2 2 + 2\log_2 3 \approx 10.17$	$7\log_2 2 + 2\log_2 3 \approx 10.17$	$7\log_2 2 + 2\log_2 3 \approx 10.17$
why do-s John consider Mary angry	$6\log_2 2 + 2\log_2 3 \approx 9.17$	$7\log_2 2 + 2\log_2 3 \approx 10.17$	$5\log_2 2 + 2\log_2 3 \approx 8.17$

In order to see how individual analysis choices contribute to corpus encoding, consider the costs of four different sentences shown in Table 5. Note that these four sentences are not meant to represent the entire corpus (and we do not calculate the final corpus cost or MDL value for this case study), but rather to illustrate how various data points contribute to the differences between our grammars with respect to corpus cost. Partial CFGs are given in Figure 27; for space reasons, it only includes rules with non-zero cost, i.e. those which share the left-hand side with at least one other rule.

As expected, the monoclausal approach pays a higher cost to encode examples with *give*, because of its lack of animacy restrictions, whereas the uniform SC grammar overpays for grammatical sentences involving modification by *why*. The third option, refined SCs, does not overpay in either case. In addition, it pays a lower cost to encode *Mary has a flyer*, because of its distinction between small clause types. This corresponds to the fact that this grammar, unlike the other two, does not generate strings like **John has angry*.

For a closer look at individual rules' contribution to these values, let us examine detailed costs of encoding a double object construction, provided in Table 6. All three grammars must pay the cost of picking *a flyer* as the object. The monoclausal approach, which lacks animacy restrictions, pays the extra cost of picking an animate Goal, in the form of an additional use of `(d -k)` → `(+k d -k, -k)`. Next, all three grammars use a rule to select the right complement type for the verb. However, since the uniform SC grammar assigns the same feature bundle to *give* and *consider*, it has to pay an additional bit to use `(=sc V)` → *give* and pick the former. Refined SCs pay for each

$\texttt{(d -k)} \rightarrow \textit{the car}$

$\texttt{(d -k)} \rightarrow \textit{a flyer}$

$\texttt{(d -k)} \rightarrow \texttt{(+k d -k, -k)}$

$(\texttt{d}_\texttt{a}\ \texttt{-k}) \rightarrow \textit{John}$

$(\texttt{d}_\texttt{a}\ \texttt{-k}) \rightarrow \textit{Mary}$

$\texttt{(sc, -k)} \rightarrow \texttt{(d= sc)}\ \texttt{(d -k)}$

$\texttt{(v, -k)} \rightarrow \texttt{(d= v)}\ \texttt{(d -k)}$

$\texttt{(do, -k)} \rightarrow \texttt{(=x do)}\ \texttt{(x, -k)}$

$\texttt{(do, -k)} \rightarrow \texttt{(=>x do)}\ \texttt{(x, -k)}$

$\texttt{(c)} \rightarrow \texttt{(=t c)}\ \texttt{(t)}$

$\texttt{(c)} \rightarrow \texttt{(+wh c, -wh)}$

$\texttt{(do, -wh, -k)} \rightarrow \texttt{(=x do)}\ \texttt{(x, -wh, -k)}$

$\texttt{(do, -wh, -k)} \rightarrow \texttt{(=>x do)}\ \texttt{(x, -wh, -k)}$

(a) Shared rules

$\texttt{(sc, -k)} \rightarrow (\texttt{d}_\texttt{a}\texttt{= sc})\ (\texttt{d}_\texttt{a}\texttt{, -k})$

$\texttt{(V, -k)} \rightarrow \texttt{(=sc V)}\ \texttt{(sc, -k)}$

$\texttt{(V, -k)} \rightarrow \texttt{(d= V)}\ \texttt{(d -k)}$

$\texttt{(v, -k)} \rightarrow \texttt{(=sc v)}\ \texttt{(sc, -k)}$

$\texttt{(do, -k, -wh)} \rightarrow \texttt{(=x do)}\ \texttt{(x, -k, -wh)}$

$\texttt{(do, -k, -wh)} \rightarrow \texttt{(=>x do)}\ \texttt{(x, -k, -wh)}$

$\texttt{(t, -wh)} \rightarrow \texttt{(+k t, -k, -wh)}$

$\texttt{(t, -wh)} \rightarrow \texttt{(+k t, -wh, -k)}$

(b) Monoclausal

$\texttt{(=sc V)} \rightarrow \textit{give}$

$\texttt{(=sc V)} \rightarrow \textit{consider}$

$\texttt{(sc, -k)} \rightarrow (\texttt{d}_\texttt{a}\texttt{= sc})\ (\texttt{d}_\texttt{a}\texttt{, -k})$

$\texttt{(v, -k)} \rightarrow \texttt{(=sc v)}\ \texttt{(sc, -k)}$

$\texttt{(do, -k, -wh)} \rightarrow \texttt{(=x do)}\ \texttt{(x, -k, -wh)}$

$\texttt{(do, -k, -wh)} \rightarrow \texttt{(=>x do)}\ \texttt{(x, -k, -wh)}$

$\texttt{(t, -wh)} \rightarrow \texttt{(+k t, -k -wh)}$

$\texttt{(t, -wh)} \rightarrow \texttt{(+k t, -wh -k)}$

(c) Uniform SC

$\texttt{(sc, -k)} \rightarrow (\texttt{=>sc}_\texttt{poss}\ \texttt{sc})\ (\texttt{sc}_\texttt{poss}\texttt{, -k})$

$\texttt{(V, -k)} \rightarrow \texttt{(=sc V)}\ \texttt{(sc, -k)}$

$\texttt{(V, -k)} \rightarrow (\texttt{=sc}_\texttt{poss}\ \texttt{V})\ (\texttt{sc}_\texttt{poss}\texttt{, -k})$

$\texttt{(v, -k)} \rightarrow (\texttt{=sc}_\texttt{poss}\ \texttt{v})\ (\texttt{sc}_\texttt{poss}\texttt{, -k})$

(d) Refined SC

Figure 27: Nonzero-cost CFG rules for Figure 24

Table 6: Nonzero-cost rules deriving *John give-s Mary a flyer* and their costs (bits)

	Rule	Cost	Total
Shared rules	(c) → (=t c) (t)	$\log_2 2$	≈ 6.58
	(v, -k) → (d= v) (d -k)	$\log_2 2$	
	(do, -k) → (=x do) (x, -k)	$\log_2 2$	
	(d_a -k) → *John*	$\log_2 2$	
	(d_a -k) → *Mary*	$\log_2 2$	
	(d -k) → *a flyer*	$\log_2 3$	
Monoclausal	(d -k) → (+k d -k, -k)	$2\log_2 3$	≈ 4.17
	(V, -k) → (d= V) (d -k)	$\log_2 2$	
Uniform SC	(d -k) → (+k d -k, -k)	$\log_2 3$	≈ 3.58
	(sc, -k) → (d_a= sc) (d_a, -k)	$\log_2 2$	
	(=sc V) → *give*	$\log_2 2$	
Refined SC	(d -k) → (+k d -k, -k)	$\log_2 3$	≈ 2.58
	(V, -k) → (=sc_{poss} V)	$\log_2 2$	

distinction only once, resulting in the lowest cost of encoding the sentence. Thus, the cost of this more complex grammar is offset by the lower cost of encoding the data.

Essentially, what the two positions exemplified by Harley and Jung 2015 and Kawakami 2018 disagree on is exactly what properties *have* shares with *give*. MGs can represent these shared properties as syntactic features within LIs which are reused in multiple constructions. The technique outlined here offers a way to examine and directly compare insights from multiple literature sources while accounting for possible overgeneration.

6 DISCUSSION AND FUTURE WORK

We have investigated the possibility of comparing syntactic analyses on quantitative grounds. Even within the same framework, such as Chomsky's (1995, 2000) Minimalist Program, there is enough room for alternative accounts of the same observed language data. We have shown how specific proposals stated as minimalist grammars (Stabler

1997) can be compared with the help of an evaluation measure inspired by minimum description length (Rissanen 1978), and how different predictions made by these proposals translate into quantifiable differences.

Examples throughout the paper have demonstrated how, overall, correct generalizations lead to smaller grammars, while overgeneration increases the corpus cost. The case study of the double-object costruction presented here is a proof of concept demonstrating how MDL can offer a quantitative perspective on various issues that are a matter of debate, or simply a topic of interest, for syntacticians. A few potential examples are listed below.

Hierarchy of adjectives vs. unordered adjuncts: One could ask whether complex cartography-style structures are "worth it" for a given set of data. For instance, Bayırlı (2018) argues that Turkish adjectives obey the adjective hierarchy of Cinque (1994) and Cinque (2010), whereas Grashchenkov and Isaeva (2023) suggest a lack of strict ordering of adjectives having used a corpus of Turkish and other Turkic data in their study. From the MDL perspective, choosing to implement a hierarchy of adjectives (through empty LIs or as a Fowlie (2013)-style extension to standard MGs) would carry the cost of encoding it. Conversely, allowing adjectives in any order may lead the grammar to overgenerate, increasing the cost of encoding adjective orderings that *are* attested in the corpus.

Acategorial roots: The idea of roots being category-neutral and having to merge with a categorizing head is a general assumption in Distributed Morphology (Marantz 1997; Embick and Marantz 2008). An MG implementation of this strategy would have root LIs carrying a single category feature and categorizing heads as LIs selecting expressions of that category, as opposed to having multiple lexical items with the same root. Quantitatively, we can expect this to be beneficial for the grammar cost, as long as the number of roots is large enough to justify the cost of the extra features needed to merge the roots and categorizing heads.

***There-insertion* in English:** Ermolaeva and Kobele (2022) use an MG-like formalism to compare various analyses of expletive-*there* constructions in English, describing their differences in terms of syntactic dependencies; MDL could translate these differences into quantitative terms. The high-origin account, where *there* merges directly into the specifier of TP (Chomsky 2000), requires fewer LIs and syntactic features to encode than one where *there* merges low and moves to its surface position (Deal 2009; Alexiadou and Schäfer 2011) or starts out in a constituent with its associate (Basilico 1997; Sabel 2000). At the same time, it would have trouble expressing any restrictions on lexical verbs (*There arrived a man in the station* vs. *There laughed a man in the hallway*; see Deal 2009), leading to overgeneration and consequences to the corpus cost.

All of that said, there is plenty of room for refinement. MDL can disagree with a linguistic intuition on what constitutes a simpler explanation of the data, if some aspect of the analysis is not taken into account by the encoding scheme, or cannot be expressed by the chosen formalism, or requires an overhead cost that does not pay off in the case of the chosen corpus. Let us briefly discuss each of these variables in turn.

Encoding schemes: The method of encoding MGs defined in Section 4, where each LI is considered a sequence of symbols from the same encoding table, is straightforward but naive. A few of the possible modifications include switching from the fixed-length code presented here to a variable-length code, with shorter binary sequences for more frequent symbols (Lee 2001); rethinking how elements of LIs are parsed into symbols to be encoded (for instance, it may be useful to combine the type and category of syntactic features into a single symbol); encoding each string component and/or feature bundle only once and using pointers to refer to them (Xanthos *et al.* 2006) to incentivize (i.e. lower the cost of) reusing elements that have already been introduced. Some proposals in the linguistics literature are motivated by patterns that could only be translated into cost reduction under a sophisticated encoding scheme; see Appendix A for an example.

Formalism-related choices: The version of minimalist grammars defined in Section 3 is a relatively simple but detailed one for the sake of conceptual clarity. Treating a lexical item as a string of phonological segments (approximated by orthography) followed by syntactic features is explicitly a simplification, as is using head movement as the sole operation for building complex words; but a more sophisticated formalism could be used to bring the results closer to those of theoretical syntax.

In order to compare analyses constructed with different formal machinery in mind (or to compare different formalisms), one would have to consider the cost of encoding this machinery along with the grammar and corpus. For example, compare MGs as defined in this paper vs. a version without covert movement. The grammar cost of an analysis using the latter formalism might be lower because of fewer distinct symbols involved; or it might be higher due to additional lexical items needed to compensate for the missing machinery. At the same time, the two versions would also differ in how **merge** and **move** are defined, the latter being cheaper to encode as it lacks the syntactic operation of covert movement. Chater *et al.* (2015) goes even further, proposing a higher-order version of MDL computed as the sum of four terms representing encoding lengths of (i) a Turing machine capable of describing Universal Grammars; (ii) a Universal Grammar ($\approx$ formalism) capable of stating specific grammars; (iii) a grammar generating the given corpus; and (iv) the corpus as encoded by the grammar.

Corpora: As shown in Footnote 5, extremely small datasets can favor overfitting grammars, if the reduction in corpus cost provided by introducing syntactic generalizations is insufficient to justify the initial investment in the grammar. This also applies to large but repetitive datasets; an extreme case would be a corpus containing the same sentence repeated an arbitrarily large number of times. Conversely, with a very large corpus of diverse sentences (which is a better representation of natural language as a whole) the MDL value is decided primarily by the corpus cost. At the same time, the grammar cost still contributes to the choice of the grammar, since many distinctions between grammars (that a linguist would consider important) have no effect on corpus cost. Consider a set of m roots, each compatible with any of n suffixes, for the total of $m \times n$ words, all found in the same syntactic contexts and attested in the corpus.[16] A minimalist grammar can encode these as whole words, with $m \times n$ lexical items, or as separate roots and suffixes, resulting in $m + n$ lexical items carrying shorter string components. On the corpus side, the former option

[16] This generalizes the observation illustrated by the toy grammar in Section 4, where $m = 2$ and $n = 2$; it is easy to see how this would scale with more lexical verbs in the corpus.

corresponds to a single choice out of $m \times n$ options, costing $\log_2 (m \times n)$ bits. The latter requires picking the root and the suffix separately, for $\log_2 (m) + \log_2 (n) = \log_2 (m \times n)$ bits. The corpus cost is the same for both options, whereas the grammar cost may be significantly different.

More fundamentally, linguists put a lot of emphasis on obtaining independent evidence for their proposals to justify the theoretical cost of postulating a new structure or operation. In an informal setting, this evidence would be brought in as a set of examples. With the proposal translated into a formal grammar, reusing a lexical item in multiple structures translates into a measurable reduction to the grammar cost, while failure to capture an observed contrast would lead to overgeneration and result in a higher corpus cost. If a consensus is reached on the set of data we care about (the corpus), and if we fix or take into account what shape analyses may take (the formalism) and how they are quantified (the encoding scheme), we can keep track of the strength of every relevant argument and counter-argument.[17] The minimum description length principle works as a natural evaluation measure, bringing the notion of "intuitive goodness" of syntactic descriptions a step closer to the more easily definable notion of "quantitative goodness".

ACKNOWLEDGMENTS

The author was supported by the Fellowship from Non-commercial Foundation for the Advancement of Science and Education INTELLECT.

[17] A reviewer has noted that there is little agreement in mainstream Minimalism with respect to the corpus and formalism, and a consensus has been impossible to reach so far. While this is a very valid concern, the MDL-based approach does not create a new problem but rather highlights one already present. The literature is replete with different (sometimes incompatible) assumptions of what grammars are allowed to look like, beliefs regarding the content of the universal grammar, and analyses developed for overlapping but non-identical datasets. The approach outlined in this paper makes this problem explicit, defining more precisely what needs to be settled in order to solve it.

APPENDIX A

The case study in Section 5 examines the distinction between the ditransitive verb selecting its arguments directly vs. selecting a constituent that is also found in other constructions and shares some properties with them. In what follows, we sketch a comparison along another dimension – namely, whether the double-object construction (*give Mary a book*) is related to the *to*-dative (*give a book to Mary*), illustrated by the original VP-shell analysis of Larson (1988) and the refined small clauses of Harley and Jung (2015).

Larson (1988) postulates a relation between the two constructions in question. Under his analysis, the structure of the VP containing the internal arguments of a ditransitive verb is parallel to that of a clause, with the Theme (*a letter*) corresponding to the subject and the Goal (*Mary*) to the object. The double object construction is derived from the *to*-dative via an operation analogous to passivization.

In order to see how this can be formalized, let us first implement passives in MGs. We start with the lexicon from Figure 24a (repeated in Figure 29a) and add two new LIs: *-ed* :: `=>V pass` and *be* :: `=pass v` (adapted from Kobele 2006). Then the passive construction is derived as shown in Figure 28. The expression of category `V`, with its topmost DP *Mary* still carrying its `-k` feature, is merged with *-ed* :: `=>V pass`, and the result with *be* :: `=pass v`. A subject is never merged in; instead, *Mary* is promoted to the subject position by having its `-k` checked by *-s* :: `=>do +k t`.

There are two issues preventing a faithful translation of Larson's (1988) solution into a minimalist grammar. First, the SMC requires that the movement of one argument be resolved before merging in another carrying the same licensee feature. A structure such as [[*give Mary*] *a letter*] with both DPs still carrying an unchecked `-k` would violate the SMC. We can bypass this problem (in a somewhat unsatisfying way) by constructing a version of *a letter* with its `-k` feature already checked. Second, the original analysis treats *to* as an instance of Case marking, which cannot be expressed in terms of standard MGs.[18] With the exception of *to*-as-Case, this analysis

[18] This aspect of the analysis is out of reach for basic MGs but could be captured by an extended version of the formalism. See e.g. Ermolaeva 2018 and Ermolaeva

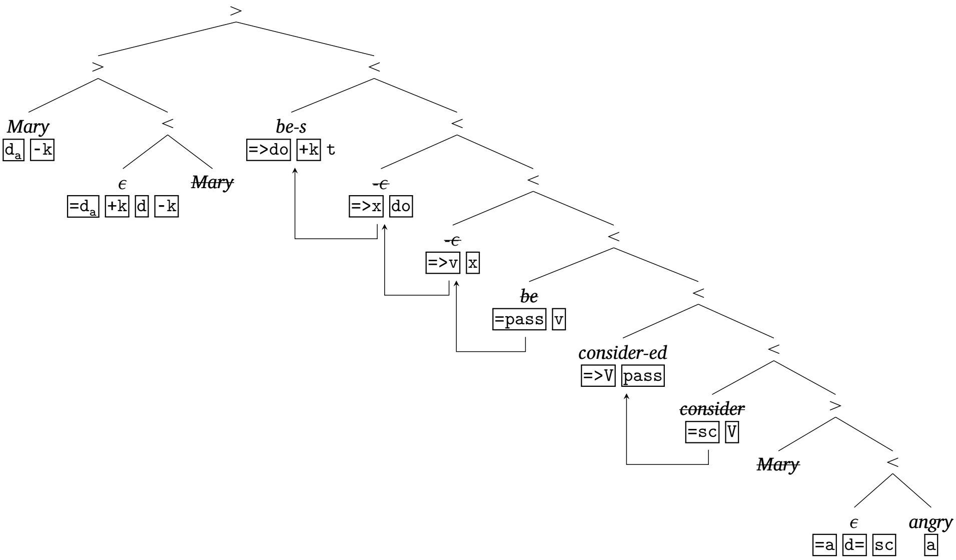


Figure 28: Derived tree for *Mary be-s consider-ed angry*

John :: $\mathtt{d_a}$ -k	*consider* :: =sc V	*angry* :: a
Mary :: $\mathtt{d_a}$ -k	-ϵ :: =>V +k d= v	ϵ :: =a d= sc
the car :: d -k	-ϵ :: =>v x	*why* :: w -wh
a flyer :: d -k	*do* :: =x do	ϵ :: =sc w= sc
ϵ :: =$\mathtt{d_a}$ +k d -k	-ϵ :: =>x do	-ϵ :: =>t +wh c
	-s :: =>do +k t	ϵ :: =t c

(a) Shared lexical items (=24a)

		ϵ :: =d +k $\mathtt{d_a}$= $\mathtt{sc_{poss}}$
	ϵ :: =d +k $\mathtt{d_a}$= sc	-ϵ :: =>$\mathtt{sc_{poss}}$ sc
	have :: =sc v	*have* :: =$\mathtt{sc_{poss}}$ v
	give :: =d y	*give* :: =$\mathtt{sc_{poss}}$ V
	ϵ :: =d +k $\mathtt{d_t}$	*to* :: p
-ed :: =>V pass	-ϵ :: =>y +k d= V	ϵ :: =d +k p= pp
be :: =pass v	-ϵ :: =>y =$\mathtt{d_t}$ V	*give* :: =pp d= V
(b) Shared passives	(c) Quasi-Larsonian *give*	(d) Extended refined SC *give*

Figure 29: MG implementations of the double object construction and *to*-datives

is recreated in Figure 29c; we will refer to this grammar as “quasi-Larsonian” to acknowledge its limitations. There is one lexical item, *give* :: `=d y`, shared by both constructions, which takes the Goal argument as its complement. To form a *to*-dative, the result then merges with -ϵ :: `=>y +k d= V`, checking the Goal’s `-k` and merging in the Theme argument. To form a double object construction, it merges instead with -ϵ :: `=>y` $=d_t$ `V`, which selects the Theme argument with its `-k` already checked, leaving the Goal’s `-k` to be checked later in the derivation – similar to the object in a passive construction.

In contrast, Harley and Jung (2015) cite the approach to *to*-datives proposed in Harley 2007, which treats them as separate from the double-object construction, with the verb base-generated low in the structure. This solution can be translated into MGs by adding a separate version of *give* (which selects a prepositional phrase and a DP), as well as a method of constructing PPs. We refer to the result, given in Figure 29d, as “extended refined SCs”.

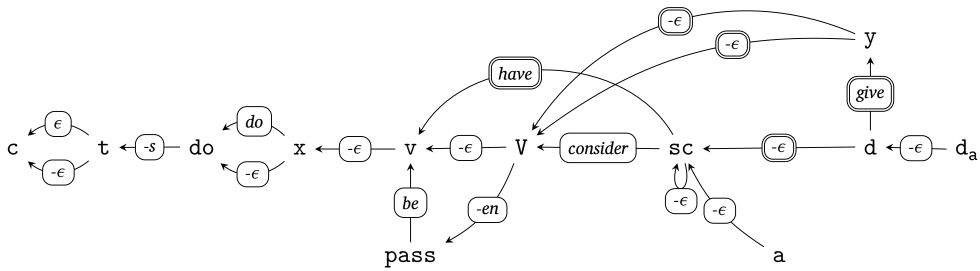
Both sets of LIs are compatible with the passive (Figure 29b) and ensure the promotion of the correct argument to the subject position (*Mary was given a letter* vs. *A letter was given to Mary*). Head-complement graphs for both grammars are given in Figure 30.

Let us first revisit the data points from Section 5 and assess their impact on the corpus cost. The quasi-Larsonian analysis performs largely like the monoclausal one. With respect to *why*-modification, no small clause in the constructions in question means no overgeneration.[19] The animacy restriction on the Goal is an argument against the quasi-Larsonian solution, since it applies to the double

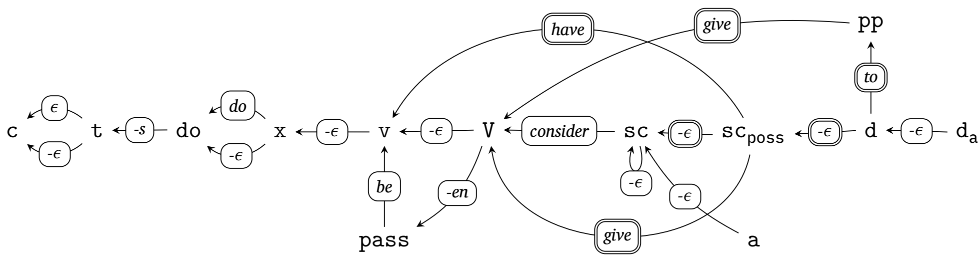
---

and Kobele 2022 for an MG-compatible treatment of Agree as transmission of morphological information along syntactic dependencies. In their framework, *to* could be implemented as data transmitted to the Goal when the -ϵ :: `=>y +k d= V` checks its `-k` feature.

[19]For completeness, the addition of passives does by itself affect some sentences we have *not* considered. The CFG for the quasi-Larsonian solution involves the same feature configuration on the left-hand side of two rules, (`v, -k, -wh`) $\rightarrow$ (`=pass v`) (`pass, -k, -wh`) and (`v, -k, -wh`) $\rightarrow$ (`=sc v`) (`sc, -k, -wh`), as it generates both *why be-s Mary considered angry* and *why do-s Mary have a flyer* (with *why* modifying the small clause). This applies to the monoclausal solution as well. The (extended) refined SC approach simply does not generate the latter of these sentences, so the remaining rule has a zero cost.



(a) Quasi-Larsonian



(b) Extended refined SC

Figure 30: Head-complement graphs of MGs in Figure 29

object construction and the *to*-dative to a different extent (Oehrle 1976). This is not a problem for extended refined SCs. However, the quasi-Larsonian solution relies on the LI which introduces the Goal, *give* :: =d y, being the same in both constructions – which is incompatible with the Goal animacy contrast between the two. While the MG fragments presented here simplify the contrast to "no restriction" vs. "animate only", the same logic would apply to more nuanced distinctions. In terms of corpus cost, this instance of overgeneration means that the quasi-Larsonian analysis would overpay for each double object construction in the corpus, same as the monoclausal analysis.

For the grammar cost, a proper comparison is impossible without tweaking the formalism itself, due to the limitations discussed above. As is, the quasi-Larsonian grammar is shorter than the extended refined SC one (with the caveat that the former would also need to pay the cost of encoding *to*). This difference is small enough to be negated if we argue that the implementation of PPs should be shared by both grammars (as the LIs *to* :: p and ϵ :: =d +k p= pp would be required for other constructions involving PPs). On the other hand, our basic encoding scheme is unable to take into account some elements of Larson's analysis – in particular, the parallel between passivization and the operation deriving the double object construction from the *to*-dative. This is reflected in the MG, for instance, by the similarities between -ϵ :: `=>V +k d= v` (which checks the object's `-k` and merges in the subject to derive the active construction) and -ϵ :: `=>y +k d= V` (which checks the Goal's `-k` and merges in the Theme to derive the *to*-dative). Both LIs are of the form -ϵ :: `=>_ +k d= _`, with _ standing for the two category names that constitute their differences. A more refined encoding scheme capable of reusing, rather than reencoding, repeated *parts* of lexical items would be able to capitalize on this.

REFERENCES

Artemis ALEXIADOU and Florian SCHÄFER (2011), There-insertion: An unaccusativity mismatch at the syntax-semantics interface, online proceedings of West Coast Conference on Formal Linguistics 28.

Karlos ARREGI and Asia PIETRASZKO (2018), Generalized head movement, in Patrick FARRELL, editor, *Proceedings of the Linguistic Society of America*, volume 3, pp. 1–15.

David BASILICO (1997), The topic is "there", *Studia Linguistica*, 51(3):278–316.

İsa Kerem BAYIRLI (2018), Does Turkish have adjective ordering restrictions?, *IULC Working Papers*, 18(2):1–26.

Michael BRODY (2000), Mirror theory: Syntactic representation in perfect syntax, *Linguistic Inquiry*, 31(1):29–56.

Nick CHATER, Alexander CLARK, John GOLDSMITH, and Amy PERFORS (2015), Towards a new empiricism for linguistics, in *Empiricism and Language Learnability*, pp. 58–105, Oxford University Press, Oxford, UK.

Noam CHOMSKY (1956), Three models for the description of language, *IRE Transactions on Information Theory*, 2(3):113–124.

Noam CHOMSKY (1957), *Syntactic structures*, De Gruyter Mouton, The Hague, Netherlands.

Noam CHOMSKY (1965), *Aspects of the theory of syntax*, MIT Press, Cambridge, MA.

Noam CHOMSKY (1986), *Knowledge of language: Its nature, origin, and use*, Praeger, New York, NY.

Noam CHOMSKY (1995), *The minimalist program*, MIT Press, Cambridge, MA.

Noam CHOMSKY (2000), Minimalist Inquiries: the framework, in Roger MARTIN, David MICHAELS, and Juan URIAGEREKA, editors, *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, pp. 89–156, MIT Press, Cambridge, MA.

Noam CHOMSKY and Morris HALLE (1968), *The sound pattern of English*, Harper & Row, New York, NY.

Guglielmo CINQUE (1994), On the evidence for partial N-movement in the Romance DP, in Guglielmo CINQUE, Jan KOSTER, Jean-Yves POLLOCK, and Rafaella ZANUTTINI, editors, *Paths towards universal grammar: Studies in Honor of Richard S. Kayne*, pp. 85–110, Georgetown University Press, Washington, DC.

Guglielmo CINQUE (2010), *The syntax of adjectives: A comparative study*, MIT Press, Cambridge, MA.

Alexander CLARK (2013), Learning trees from strings: A strong learning algorithm for some context-free grammars, *Journal of Machine Learning Research*, 14:3537–3559.

Alexander CLARK (2015), Canonical context-free grammars and strong learning: two approaches, in Marco KUHLMANN, Makoto KANAZAWA, and Gregory M. KOBELE, editors, *Proceedings of the 14th Meeting on the Mathematics of Language (MOL 2015)*, pp. 99–111, Association for Computational Linguistics, Chicago, IL.

Amy Rose DEAL (2009), The origin and content of expletives: Evidence from “selection”, *Syntax*, 12(4):285–323.

Marina ERMOLAEVA (2018), Morphological agreement in minimalist grammars, in *Formal Grammar: 22nd International Conference, FG 2017, Toulouse, France, July 22-23, 2017, Revised Selected Papers*, pp. 20–36, Springer, Berlin, Germany.

Marina ERMOLAEVA (2021), *Learning syntax via decomposition*, Ph.D. thesis, University of Chicago, Chicago, IL.

Marina ERMOLAEVA and Gregory M. KOBELE (2022), Agree as information transmission over dependencies, *Syntax*, 25(4):466–507.

Meaghan FOWLIE (2013), Order and optionality: Minimalist grammars with adjunction, in *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pp. 12–20, Association for Computational Linguistics, Sofia, Bulgaria.

John GOLDSMITH (1980), Meaning and mechanism in grammar, *Harvard studies in syntax and semantics*, 3:423–449.

John GOLDSMITH (2001), Unsupervised learning of the morphology of a natural language, *Computational Linguistics*, 27(2):153–198.

John GOLDSMITH (2006), An algorithm for the unsupervised learning of morphology, *Natural Language Engineering*, 12(4):353–372.

John GOLDSMITH (2011), The evaluation metric in generative grammar, presented at 50th Anniversary of the MIT Linguistics Department, Cambridge MA, December 2011.

Thomas GRAF (2013), *Local and transderivational constraints in syntax and semantics*, Ph.D. thesis, UCLA, Los Angeles, CA.

Pavel GRASHCHENKOV and Ulyana ISAEVA (2023), Vzaimnoe raspolozhenie atributivnyh prilagatel'nyh po dannym tyurkskih tekstov [Adjectival ordering on the data of Turkic texts], *Uralo-altayskie issledovaniya*, 48(1):22–32.

Peter D. GRÜNWALD (2007), *The minimum description length principle*, MIT Press, Cambridge, MA.

John T. HALE and Edward P. STABLER (2005), Strict deterministic aspects of minimalist grammars, in Philippe BLACHE, Edward STABLER, Joan BUSQUETS, and Richard MOOT, editors, *Logical Aspects of Computational Linguistics. LACL 2005. Lecture Notes in Computer Science*, pp. 162–176, Springer, Berlin, Germany.

Heidi HARLEY (2002), Possession and the double object construction, *Linguistic Variation Yearbook*, 2(1):31–70.

Heidi HARLEY (2007), The bipartite structure of verbs cross-linguistically (or: Why Mary can't exhibit John her paintings), in Thaïs Cristófaro SILVA and Heliana MELLO, editors, *Conferências do V Congresso Internacional da Associação Brasileira de Lingüística*, pp. 45–84, Belo Horizonte, Brazil.

Heidi HARLEY and Hyun Kyoung JUNG (2015), In support of the P_{HAVE} analysis of the double object construction, *Linguistic Inquiry*, 46(4):703–730.

Yu HU, Irina MATVEEVA, John GOLDSMITH, and Colin SPRAGUE (2005), Using morphology and syntax together in unsupervised learning, in *Proceedings of the Workshop on Psychocomputational Models of Human Language Acquisition*, pp. 20–27, Association for Computational Linguistics, Ann Arbor, MI.

Mark JOHNSON (2017), Marr's levels and the minimalist program, *Psychonomic Bulletin & Review*, 24(1):171–174.

Roni KATZIR (2014), A cognitively plausible model for grammar induction, *Journal of Language Modelling*, 2(2):213–248, doi:10.15398/jlm.v2i2.85, `https://jlm.ipipan.waw.pl/index.php/JLM/article/view/85`.

Masahiro KAWAKAMI (2018), Double object constructions: Against the small clause analysis, *Journal of Humanities and Social Sciences*, 45:209–226.

Richard S. KAYNE (1984), *Connectedness and binary branching*, Foris Publications, Dordrecht, Netherlands.

Richard S. KAYNE (1994), *The antisymmetry of syntax*, MIT Press, Cambridge, MA.

Gregory M. KOBELE (2002), Formalizing mirror theory, *Grammars*, 5(3):177–221.

Gregory M. KOBELE (2006), *Generating copies: An investigation into structural identity in language and grammar*, Ph.D. thesis, UCLA, Los Angeles, CA.

Gregory M. KOBELE (to appear), Minimalist grammars and decomposition, in Kleanthes K. GROHMANN and Evelina LEIVADA, editors, *The Cambridge Handbook of Minimalism*, Cambridge University Press, Cambridge, MA.

Richard K. LARSON (1988), On the double object construction, *Linguistic Inquiry*, 19(3):335–391.

Thomas C.M. LEE (2001), An introduction to coding theory and the two-part minimum description length principle, *International Statistical Review*, 69(2):169–183.

David MARR (1982), *Vision: A computational investigation into the human representation and processing of visual information*, Henry Holt and Co., New York, NY.

Jens MICHAELIS (1998), Derivational minimalism is mildly context-sensitive, in Michael MOORTGAT, editor, *International Conference on Logical Aspects of Computational Linguistics*, pp. 179–198, Springer, Berlin, Germany.

Richard Thomas OEHRLE (1976), *The grammatical status of the English dative alternation*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Christopher PEACOCKE (1986), Explanation in computational psychology: Language, perception and level 1.5, *Mind & Language*, 1(2):101–123.

David Michael PESETSKY (1996), *Zero syntax: Experiencers and cascades*, MIT Press, Cambridge, MA.

Ezer RASIN, Iddo BERGER, Nur LAN, and Roni KATZIR (2018), Learning phonological optionality and opacity from distributional evidence, in Sherry HUCKLEBRIDGE and Max NELSON, editors, *Proceedings of the North East Linguistic Society 48 (NELS 48)*, volume 48, pp. 269–282, Amherst, MA.

Ezer RASIN and Roni KATZIR (2016), On evaluation metrics in optimality theory, *Linguistic Inquiry*, 47(2):235–282.

Ezer RASIN and Roni KATZIR (2019), Simplicity-based learning in constraint-based and rule-based phonology, mini-course at the University of Leipzig.

Jorma RISSANEN (1978), Modeling by shortest data description, *Automatica*, 14(5):465–471.

Joachim SABEL (2000), Expletives as features, in Roger BILLEREY and Brook Danielle LILLEHAUGEN, editors, *Proceedings of the 19th West Coast Conference on Formal Linguistics*, pp. 411–424, Cascadilla Press, Somerville, MA.

Stuart M. SHIEBER (1985), Evidence against the context-freeness of natural language, *Linguistics and Philosophy*, 8:333–343.

Edward P. STABLER (1997), Derivational minimalism, in Christian RETORÉ, editor, *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 Nancy, France, September 23–25, 1996 Selected Papers*, pp. 68–95, Springer, Berlin, Germany.

Edward P. STABLER (2001), Recognizing head movement, in *Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics*, LACL '01, pp. 245–260, Springer-Verlag, Berlin, Germany.

Edward P. STABLER and Edward L. KEENAN (2003), Structural similarity within and among languages, *Theoretical Computer Science*, 293(2):345–363.

John TORR and Edward STABLER (2016), Coordination in minimalist grammars: Excorporation and across the board (head) movement, in David CHIANG and Alexander KOLLER, editors, *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 12)*, pp. 1–17, Düsseldorf, Germany.

Edwin S. WILLIAMS (1975), Small clauses in English, in John P. KIMBALL, editor, *Syntax and Semantics, volume 4*, pp. 249–273, Brill, Leiden, Netherlands.

Aris XANTHOS, Yu HU, and John GOLDSMITH (2006), Exploring variant definitions of pointer length in MDL, in *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology at HLT-NAACL 2006*, pp. 32–40, Association for Computational Linguistics, New York, NY.

Marina Ermolaeva
0000-0001-7796-7963
mail@mermolaeva.com

Lomonosov Moscow State University
Department of Theoretical and
Applied Linguistics
1-51 Leninskie Gory, Moscow, Russia

Marina Ermolaeva (2023), *Evaluating syntactic proposals using minimalist grammars and minimum description length*, Journal of Language Modelling, 11(1):67–119
https://dx.doi.org/10.15398/jlm.v11i1.334

An algebraic approach to translating Japanese

Valentin Boboc
University of Manchester

ABSTRACT

We use Lambek's pregroups and the framework of compositional distributional models of language ("DisCoCat") to study translations from Japanese to English as pairs of functors. Adding decorations to pregroups, we show how to handle word order changes between languages.

Keywords: pregroups, type grammars, translations, functorial

INTRODUCTION 1

Language has the purpose of conveying meaning. It is traditionally viewed as possessing both an empirical aspect – one learns language by practising language – and a compositional aspect – the view that the meaning of a complex phrase is fully determined by its structure and the meanings of its constituent parts.

In order to efficiently exploit the compositional nature of languages, a popular way of modelling natural languages is a categorical compositional distributional model, abbreviated "DisCoCat" (Coecke *et al.* 2010). Languages are modelled as functors from a category that interprets grammar ("compositional") to a category that interprets semantics ("distributional").

The compositional part is responsible for evaluating whether phrases or sentences are well-formed by calculating the overall grammatical type of a phrase from the grammatical types of its individual parts. There are several algebraic methods for modelling the grammar of a natural language. In the present article we choose the well-established model of pregroup grammars. Pregroups were introduced in Lambek 1997 to replace the algebra of residuated monoids in order to model grammatical types, their juxtapositions, and reductions. Pregroup calculus has been applied to formally represent the syntax of several natural languages such as: French (Bargelli and Lambek 2001a), German (Lambek and Preller 2004), Persian (Sadrzadeh 2007), Arabic (Bargelli and Lambek 2001b), Japanese (Cardinal 2002), and Latin (Casadio and Lambek 2005).

The distributional part assigns meanings to individual words by associating to them, for example, statistical co-occurrence vectors (Mitchell and Lapata 2008). The DisCoCat model is thus a way of interpreting compositions of meanings via grammatical structure.

In this article we study the notion of translating between compositional distributional models of language by analysing translation from Japanese into English. On the compositional side, a translation is a strong monoidal functor. It is easy to demonstrate that such a functor is too rigid to handle the translation of even simple phrases between languages which have different word order. We show that one can keep using the gadget of monoidal functors as long as the underlying pregroup grammars are decorated with additional structure.

We begin by introducing basic notions about the compact closed categories we work with – namely pregroups and finitely generated vector spaces – and define our notion of translation functor. Next, we give an introduction to basic Japanese grammar and the pregroup structure we use to model it. Finally, we introduce notions of pregroup decorations and use them to give a structured framework for translating Japanese sentences.

2 THEORETICAL BACKGROUND

2.1 *Compact closed structures*

The key to the DisCoCat model is that both the category of pregroups and the category of finitely generated vector spaces are *compact closed* *categories*. This allows for compositional characteristics of grammar to be incorporated into the distributional spaces of meaning.

For completeness, we provide here a definition of compact closure. The reader is encouraged to consult (Kelly and Laplaza 1980) for a more complete and technical reference.

DEFINITION 1 *A* compact closed category *is a category* $\mathscr{C}$ *together with a bifunctor*

$$- \otimes - : \mathscr{C} \times \mathscr{C} \to \mathscr{C},$$

called tensor product, which is associative up to natural isomorphism and possesses a two-sided identity element I*, and each object* $A \in \mathscr{C}$ *has a right dual* A^r *and a left dual* A^ℓ *with the following morphisms*

$$A \otimes A^r \xrightarrow{\varepsilon_A^r} I \xrightarrow{\eta_A^r} A^r \otimes A,$$

$$A^\ell \otimes A \xrightarrow{\varepsilon_A^\ell} I \xrightarrow{\eta_A^\ell} A \otimes A^\ell.$$

Moreover, the ε *and* η *maps satisfy the "yanking" conditions:*

$$(1_A \otimes \varepsilon_A^\ell) \circ (\eta_A^\ell \otimes 1_A) = 1_A \qquad (\varepsilon_A^r \otimes 1_A) \circ (1_A \otimes \eta_A^r) = 1_A$$

$$(\varepsilon_A^\ell \otimes 1_A) \circ (1_{A^\ell} \otimes \eta_A^\ell) = 1_{A^\ell} \qquad (1_{A^r} \otimes \varepsilon_A^r) \circ (\eta_A^r \otimes 1_{A^r}) = 1_{A^r}.$$

The upshot of compact closure is that we want to have elements which "cancel each other out" and we can decompose the identity into a product.

Recalling pregroups 2.2

DEFINITION 2 *A* pregroup *is a tuple* $(P, \cdot, 1, -^\ell, -^r, \leq)$ *where* $(P, \cdot, 1, \leq)$ *is a partially ordered monoid and the unary operations* $-^\ell, -^r$ *(the left and the right dual) satisfy for all* $x \in P$ *the following relations:*

$$x \cdot x^r \leq 1 \qquad x^\ell \cdot x \leq 1$$

$$1 \leq x^r \cdot x \qquad 1 \leq x \cdot x^\ell.$$

The operation sign $\cdot$ is omitted unless it is relevant. It is immediate to check that the following relations hold in every pregroup:

$$1^\ell = 1 = 1^r \qquad (x^\ell)^r = x = (x^r)^\ell$$

$$(xy)^\ell = y^\ell x^\ell \text{ and } (xy)^r = y^r x^r \qquad \text{if } x \leq y \text{ then } y^\ell \leq x^\ell \text{ and } y^r \leq x^r.$$

We model the grammar of a natural language by freely generating a pregroup from a set of grammatical types. Each word in the dictionary is assigned an element of the pregroup which corresponds to its linguistic function, e.g. noun, verb, adjective, etc. A string of words is interpreted by multiplying the elements assigned to the constituent parts in syntactic order. If a string of words satisfies the relation $w_1 w_2 \ldots w_n \leq s$ we say that the string *reduces* to the type s.

EXAMPLE 1 Suppose there are two grammatical types: noun n and sentence s. Grammar is modelled as the free pregroup $\text{PGrp}(\{n,s\})$. Consider the sentence *Pigeons eat bread*. We assign the type n to *pigeons* and *bread* and the type $n^r s n^\ell$ to the transitive verb *eat*. The sentence overall has type $n(n^r s n^\ell)n$ and the following reductions hold:

$$n(n^r s n^\ell)n = (nn^r)s(n^\ell n) \leq (1)s(n^\ell n) \leq s(1) \leq s.$$

In this case we say that *Pigeons eat bread* is a well-formed sentence since in the pregroup $\text{PGrp}(\{n,s\})$ the phrase reduces to the correct type.

The two individual reductions could have been performed in a different order. Lambek's *Switching Lemma* (Lambek 1997, Proposition 2) tells us that in any computation performed in a freely generated pregroup, we may assume without loss of generality that all contractions precede all expansions.

A pregroup can be viewed as a compact closed category. The objects of the category are the elements of the pregroup. There is an arrow $x \to y$ if and only if $x \leq y$, and the tensor product is given by the pregroup operation: $x \otimes y = xy$. The morphisms $\varepsilon^r, \varepsilon^\ell, \eta^r, \eta^\ell$ are defined in the obvious way. In terms of the ε and η maps, the reductions in this example can be represented as:

$$(\varepsilon_n^\ell \otimes 1_s \otimes \varepsilon_n^\ell)(n \otimes (n^r \otimes s \otimes n^\ell) \otimes n) \to s.$$

2.3 *Meaning space*

We encode the semantic structure of a natural language into the category of finitely generated vector spaces, which we denote by FVect. The arrows are linear transformations, and there is a natural monoidal structure given by the linear algebraic tensor product with unit $\mathbb{R}$, which also happens to be symmetric: $V \otimes W \simeq W \otimes V$. This implies that $V^\ell \simeq V^r \simeq V^*$, where the latter denotes the dual vector space.

Fixing a basis $\{\mathbf{v}_i\}$ for the vector space V we get moreover that $V \simeq V^*$ and the structure morphisms of compact closure are given by

$$\varepsilon_V = \varepsilon_V^r = \varepsilon_V^\ell : V^* \otimes V \to \mathbb{R}$$

$$\text{where } \sum_{i,j} a_{ij} v_i \otimes v_j \mapsto \sum_{i,j} a_{ij} \langle v_i | v_j \rangle$$

$$\eta_V = \eta_V^r = \eta_V^\ell : \mathbb{R} \to V \otimes V^*$$

$$\text{where } 1 \mapsto \sum_i v_i \otimes v_i \text{ extended linearly.}$$

If we denote by P both the pregroup and the corresponding category, the bridge between grammar and semantics is given by a strong monoidal functor

$$F : P \to \text{FVect},$$

which we call a *functorial language model*. The functor assigns vector spaces to atomic types: $F(1) = I$, $F(n) = N$ (the vector space of nouns), $F(s) = S$ (the vector space of sentences), etc. For words in P, monoidality tells us that $F(x \otimes y) = F(x) \otimes F(y)$. The compact closure is also preserved: $F(x^\ell) = F(x^r) = F(x)^*$. For example, we can interpret the transitive verb *eat* with type $n^r s n^\ell$ as a vector in

$$\begin{aligned} F(n^r \otimes s \otimes n^\ell) &= F(n^r) \otimes F(s) \otimes F(n^\ell) \\ &= F(n)^* \otimes F(s) \otimes F(n)^* = N \otimes S \otimes N. \end{aligned}$$

Pregroup reductions in P can be interpreted as semantic reductions in FVect using the corresponding ε and η maps. The reductions associated to a transitive verb are then given by

$$\begin{aligned} F(\varepsilon_n^r \otimes 1_s \otimes \varepsilon_n^\ell) &= F(\varepsilon_n^r) \otimes F(1_s) \otimes F(\varepsilon_n^\ell) \\ &= F(\varepsilon_n)^* \otimes F(1_s) \otimes F(\varepsilon_n)^* = \varepsilon_N \otimes 1_S \otimes \varepsilon_N. \end{aligned}$$

The meaning of a sentence or phrase is derived by interpreting the pregroup reduction as the correponding semantic reduction of the tensor product of distributional meanings of individual words in the phrase. The previous example *Pigeons eat bread* is interpreted as

$$F(\varepsilon_n^r \otimes 1_s \otimes \varepsilon_n^\ell)(\textbf{Pigeons} \otimes \textbf{eat} \otimes \textbf{bread}).$$

2.4 *Translating between functorial language models*

Bradley *et al.* (2018) formalised the notion of a translation between functorial language models. We illustrate this construction with an example on translating simple noun phrases and the problems one may encounter.

DEFINITION 3 *Let $(\mathscr{C}, \otimes, 1_{\mathscr{C}})$ and $(\mathscr{D}, \odot, 1_{\mathscr{D}})$ be monoidal categories. A* monoidal functor $F : \mathscr{C} \to \mathscr{D}$ *is a functor equipped with a natural isomorphism $\Phi_{x,y} : F(x) \odot F(y) \to F(x \otimes y)$ for every pair of objects $x, y \in \mathscr{C}$ and an isomorphism $\phi : 1_{\mathscr{D}} \to F(1_{\mathscr{C}})$ such that for any triple of objects $x, y, z \in \mathscr{C}$, the following diagram commutes*

$$\begin{array}{ccccc} (F(x) \odot F(y)) \odot F(z) & \xrightarrow{\Phi_{x,y} \odot 1_{F(z)}} & F(x \otimes y) \odot F(z) & \xrightarrow{\Phi_{x \otimes y, z}} & F((x \otimes y) \otimes z) \\ \downarrow & & & & \downarrow \\ F(x) \odot (F(y) \odot F(z)) & \xrightarrow{1_{F(x)} \odot \Phi_{y,z}} & F(x) \odot F(y \otimes z) & \xrightarrow{\Phi_{x, y \otimes z}} & F(x \otimes (y \otimes z)) \end{array}$$

where the vertical arrows apply the associativity in their respective categories. Moreover, for every object $x \in \mathscr{C}$, the following two squares commute:

$$\begin{array}{ccc} 1_{\mathscr{D}} \odot F(x) & \longrightarrow & F(x) \\ \downarrow & & \uparrow \\ F(1_{\mathscr{C}}) \odot F(x) & \rightarrow & F(1_{\mathscr{C}} \otimes x) \end{array} \qquad \begin{array}{ccc} F(x) \odot 1_{\mathscr{D}} & \longrightarrow & F(x) \\ \downarrow & & \uparrow \\ F(x) \odot F(1_{\mathscr{C}}) & \rightarrow & F(x \otimes 1_{\mathscr{C}}). \end{array}$$

DEFINITION 4 *Let (F, Φ, ϕ) and (G, Ψ, ψ) be monoidal functors between the monoidal categories $\mathscr{C}$ and $\mathscr{D}$. A* monoidal natural transformation $\alpha : F \Rightarrow G$ *is a natural transformation where the following diagrams commute:*

$$\begin{array}{ccc} F(x) \odot F(y) & \xrightarrow{\alpha(x) \odot \alpha(y)} & G(x) \odot G(y) \\ \downarrow{\Phi_{x,y}} & & \downarrow{\Psi_{x,y}} \\ F(x \otimes y) & \xrightarrow{\alpha_{x \otimes y}} & G(x \otimes y) \end{array} \qquad \begin{array}{ccc} 1_{\mathscr{D}} & & \\ \downarrow{\phi} & \searrow^{\psi} & \\ F(1_{\mathscr{C}}) & \xrightarrow{\alpha(1_{\mathscr{C}})} & G(1_{\mathscr{C}}). \end{array}$$

DEFINITION 5 *Let $\mathscr{A} : P \to$* FVect *and $\mathscr{B} : Q \to$* FVect *be two functorial language models. A* translation *from F to G is a tuple (T, α), where $T : P \to Q$ is a monoidal functor and $\alpha : \mathscr{A} \Rightarrow \mathscr{B} \circ T$ is a monoidal natural transformation.*

EXAMPLE 2 We attempt to translate simple phrases of the type *adjective* + *noun* from Japanese to English. We work on a restricted model. Let $J = \mathrm{PGrp}(\{s_J, n_J\})$ be the free pregroup (or category) generated by the sentence and noun types in Japanese and let $E = \mathrm{PGrp}(\{s_E, n_E\})$ be the free pregroup generated by the sentence and noun types in English.

The functorial language models are denoted by $\mathscr{J} : J \to \mathrm{FVect}$ and $\mathscr{E} : E \to \mathrm{FVect}$, respectively. The semantic assignment is straightforward: $\mathscr{J}(n_J) = N_J, \mathscr{J}(a_J) = A_J, \mathscr{E}(n_E) = N_E, \mathscr{E}(a_E) = A_E$.

The translation will consist of the monoidal functor $T : J \to E$, which sends $s_J \mapsto s_E$ and $n_J \mapsto n_E$. Automatically, we have that the type reduction is preserved in the corresponding languages, i.e. $T\left((n_J n_J^\ell) n_J\right) = T(n_J) = n_E$. Due to monoidality, it suffices to define the components $\alpha_{n_J}, \alpha_{s_J}$ of the natural transformation $\alpha : \mathscr{J} \Rightarrow \mathscr{E} \circ T$ in order to parse semantics.

Additionally, the natural transformation α must commute with the monoidal functor T. Pictorially, we have a commutative square:

$$\begin{array}{ccc} (N_J \otimes N_J) \otimes N_J & \xrightarrow{\mathscr{J}(\varepsilon^\ell_{N_J} \otimes 1_{\mathscr{J}})} & N_J \\ \Big\downarrow \alpha_{(n_J n_J^\ell) n_J} & & \Big\downarrow \alpha_{n_J} \\ (N_E \otimes N_E) \otimes N_E & \xrightarrow{\mathscr{E}(\varepsilon^\ell \otimes 1_{\mathscr{E}})} & N_E. \end{array}$$

Consider the concrete words **red** $\in N_E \otimes N_E$, **cat** $\in N_E$, **akai** $\in N_J \otimes N_J$, and **neko** $\in N_J$. The diagram says that if we first use Japanese grammar rules to reduce **akai** ⊗ **neko** to **akai neko** and then translate to **red cat** is the same thing as first translating component-wise **akai** ⊗ **neko** to **red** ⊗ **cat** and then using English grammar rules to reduce to **red cat**.

Since there is no discrepancy in word order, this example of phrasal translation works in the desired way. If we instead wanted to translate the phrase *akai neko* from Japanese into *pisică roşie* in Romanian, we would encounter some difficulties. The latter is a "noun + adjective" phrase, as the natural word order in Romanian for such phrases is the opposite to the word order in Japanese.

The reduction rule in Romanian is given by

$$n_R(n_R^r n_R) \to n_R.$$

Suppose there exists a monoidal functor $T' : J \to R$ that takes Japanese grammar types to Romanian grammar types. Then we want to preserve the reduction rules, i.e.

$$T'\left((n_J n_J^\ell) n_J\right) = n_R(n_R^r n_R).$$

We thus obtain the condition: $T'\left(n_J^\ell\right) = n_R^r$. However, left and right adjoints must be preserved by a strong monoidal functor. Hence this condition cannot be fulfilled.

Section 4 introduces techniques that can help us overcome such problems with word order changes.

3 JAPANESE CRASH COURSE

3.1 *Generalities*

Japanese is a synthetic and agglutinative language. The usual word order is subject-object-verb (SOV) with topic-comment sentence structure. There are no definite/indefinite articles. Nouns possess neither grammatical gender nor number. Verbs and adjectives are conjugated for tense, voice, and aspect, but not person or number. Particles are attached to words to identify their grammatical role. We write sentences natively and employ the *Nihon-siki* romanisation system.

The sentence *The cat eats fish* can be represented in two different but closely related ways.

(1)	猫	が	魚	を	食べる
	neko	ga	sakana	wo	taberu
	cat	NOM	fish	ACC	eat
(2)	猫	は	魚	を	食べる
	neko	ha	sakana	wo	taberu
	cat	TOP	fish	ACC	eat

Note the use of the subject particle *ga*, the topic particle *ha*, and the direct object particle *wo*. Remark that Japanese distinguishes between topic and subject. The topic generally needs to be explicitly introduced at the beginning of a discourse, but as the discourse carries on, the topic need not be the grammatical subject of every sentence. Both sentences translate into English as *The cat eats fish*, or *Cats eat fish*. However, a more pertinent interpretation of the second sentence is *As for the cat/Speaking of the cat, it eats fish.*

Another important aspect of word order in Japanese is head finality. Phrases can be broadly described as consisting of a **head** and a *modifier*. English is generally a head initial language. Consider for example the phrases: ***to** school*, ***in** England*, and *red **cat***. The word that gets modified tends to come before the modifiers, the main exception being that nouns succeed the adjectives that modify them. In contrast, Japanese is a canonical example of a head final language. Our example phrases become

(3)

学校	へ
gakkō	**he**
school	**to**

(4)

イギリス	に
igirisu	**ni**
England	**in**

(5)

赤い	猫
akai	**neko**
red	**cat**

Head finality is also encountered in the case of relative clauses, which usually occur before the part of speech they modify. This phenomenon is demonstrated by the following pair of phrases.

(6)

女	が	赤い	ワンピース	を	着た
onna	ga	akai	wanpîsu	wo	kita
woman	NOM	red	dress	ACC	wore

'The woman wore a red dress'

(7)

赤い	ワンピース	を	着た	女
akai	wanpîsu	wo	kita	**onna**
red	dress	ACC	wore	**woman**

'**The woman**, who wore a red dress'

This is a prime example of a structure where the word order is changed during translation. The following section will develop the algebraic machinery to interpret such translations.

Subjects are habitually dropped when they are clear from context, and personal pronouns are used sparingly. We conclude this section with an example, which demonstrates how a very common reflexive/personal pronoun *zibun* 'oneself' can lead to ambiguous interpretations. *Zibun* is often used as a way for the speaker to refer either to themselves or to their interlocutor. The sentence

(8)	自分	が	嘘つき	か
	zibun	ga	usotuki	ka
	oneself	NOM	liar	QUESTION

can be translated as either *Am I a liar?* or *Are you a liar?* in the absence of further context.

3.2 *Compositional model*

Define $J = \mathrm{PGrp}(\{\pi, n, s_1, s_2, s, o_1, \ldots\})$ to be the pregroup of grammar types associated to Japanese. Following Cardinal 2002 with slight modifications, we define the following atomic types:

- π pronoun,
- n noun,
- s_1, s_2 imperfective / perfective sentence,
- $\bar{s}$ topicalised sentence,
- s sentence,
- o_1 nominative case,
- o_2 accusative case,
- o_3 dative case,
- o_4 genitive case,
- o_5 locative case,
- o_6 lative case,
- o_7 ablative case,
- etc.

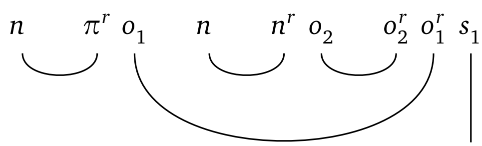
We also impose the following reductions in J:

$$s_i \to s \qquad \bar{s} \to s \qquad n \to \pi.$$

We now discuss how to assign types to various parts of speech. Revisiting the example sentence *neko ga sakana wo taberu* 'the cat eats fish', the words *neko* and *sakana* are both nouns and thus have type n. The subject particle *ga* has type $\pi^r o_1$, the direct object particle *wo* has type $n^r o_2$ and the transitive verb *taberu* then has type $o_2^r o_1^r s_1$. The sentence then has type $n(\pi^r o_1)n(n^r o_2)(o_2^r o_1^r s_1)$ and we can derive the following type reductions:

$$\begin{aligned} n(\pi^r o_1)n(n^r o_2)(o_2^r o_1^r s_1) &\to (n\pi^r)o_1(\pi\pi^r)o_2(o_2^r o_1^r s_1) \\ &\to (\pi\pi^r)o_1(\pi\pi^r)o_2(o_2^r o_1^r s_1) \\ &\to o_1 o_2(o_2^r o_1^r s_1) \\ &\to o_1(o_2 o_2^r)o_1^r s_1 \\ &\to o_1 o_1^r s_1 \\ &\to s_1 \\ &\to s \end{aligned}$$

to see that the sentence is well-formed and reduces to the correct grammatical type. Here we used the reductions $n \to \pi$ and $s_1 \to s$ together with different applications of the contraction morphism ε. Graphically, this type reduction can be seen in the following diagram, where a lower bracket indicates that a contraction morphism of the type ε was applied.



Since word order is flexible, the same sentence could have been written as *sakana wo neko ga taberu*, and then *taberu* would have been assigned the type $o_1^r o_2^r s_1$. As we want to take advantage of the *Switching Lemma* while performing computations, we want to restrict ourselves to working with freely generated pregroups. Situations where certain words or verbs can be assigned different types are generally handled by adding *metarules*. Informally, a metarule stipulates that if a grammar contains rules that match a specified pattern, then it also contains rules that match some other specified pattern. In our concrete example, we could impose the following metarule.

METARULE 1 *Any transitive verb that has type $o_1^r o_2^r s_i$ also has type $o_2^r o_1^r s_i$.*

Moving away from transitive verbs, the ablative particle *kara* has type $\pi^r o_7$ and the lative particle *he* has type $\pi^r o_6$. In the following example, the verb *untensita* has type $o_6^r o_7^r s_2$.

(9)	家	から	駅	へ	運転した
	ie	kara	eki	he	untensita
	house	ABL	station	LAT	drove

'(I) drove from home to the train station.'

Causative passive verbs take a subject and an indirect object marked with the dative particle *ni* of type $\pi^r o_3$. For instance, the verb *yomaseta* 'x made y read' has type $o_2^r o_3^r o_1^r s_2$.

(10)	先生	が	私	に	本	を	読ませた
	sensei	ga	watasi	ni	hon	wo	yomaseta
	teacher	NOM	I	DAT	book	ACC	read-CAUSE-PAS

'The teacher made me read the book.'

The genitive particle *no* has type $\pi^r o_4$ together with a metarule that states that type o_4 is equivalent to type nn^ℓ. The possessor is always on the left in a genitive construction. The topic particle *ha* is distinguished from the subject particle *ga* and has type $\pi^r \bar{s} s^\ell$, i.e. *ha* requires a topic on the left and a sentence about the topic on the right.

(11)	私	の	車	は	箸	を	渡れない
	watasi	no	kuruma	ha	hasi	wo	watarenai
	I	GEN	car	TOP	bridge	ACC	cross-POT-NEG

'I cannot cross the bridge with my car/About my car, it cannot cross the bridge.'

In the latter example, the type reduction goes as follows:

$\pi(\pi^r o_4)n(\pi^r \bar{s} s^\ell)n(\pi^r o_2)(o_2^r s_1)$	
$\rightarrow(\pi\pi^r)o_4 n(\pi^r \bar{s} s^\ell)(n\pi^r)(o_2 o_2^r)s$	associativity
$\rightarrow(1)(nn^\ell)n(\pi^r \bar{s} s^\ell)(n\pi^r)(1)s$	contractions + genitive metarule
$\rightarrow n(n^\ell n)(\pi^r \bar{s} s^\ell)(\pi\pi^r)s$	$n \rightarrow \pi$
$\rightarrow(n\pi^r)\bar{s}(s^\ell s)$	associativity
$\rightarrow\bar{s}$	contractions
$\rightarrow s.$	

4 TRANSLATION AND DECORATED PREGROUPS

4.1 Decorated pregroups

As Example 2 shows, our initial machinery is not suited to translating phrases between languages with different word orders. The morphism of pregroups (or monoidal functor) $T : P \to Q$ that transfers information from the source language to the target language happens to be too rigid. We decorate pregroups with additional structures so that we can have more control over the monoid's operation. To this end, we define anti-homomorphisms for the purpose of inverting word order and pregroups with braces and β-pregroups to get more refined control over associativity.

DEFINITION 6 *An* anti-homomorphism of monoids *is a map* $\Phi : P \to Q$ *such that for all elements* $x, y \in P$ *we have* $\Phi(xy) = \Phi(y)\Phi(x)$.

DEFINITION 7 *Let* $(P, \cdot)$ *be a monoid. The opposite monoid* $(P^{op}, *)$ *is the monoid which has the same elements as* P *and the operation for all* $x, y \in P^{op}$ *is given by* $x * y = y \cdot x$. *Observe that* $(P, \cdot) \simeq (P^{op}, *)$.

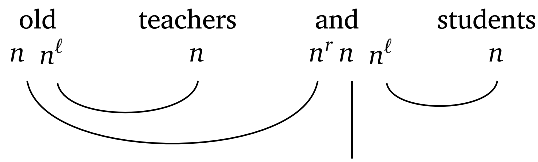
In light of this, an anti-homomorphism can be viewed as a morphism from the opposite monoid $\Phi : P^{\text{op}} \to Q$. Additionally, an anti-homomorphism of pregroups takes left adjoints to right adjoints and vice-versa.

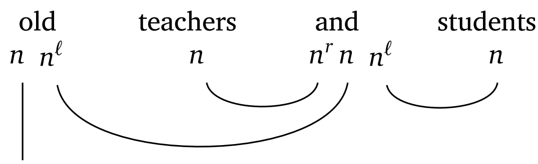
EXAMPLE 3 In Example 2 the problem of translating "adjective + noun" phrases from Japanese into Romanian can be solved by setting the translation functor to be an anti-homomorphism that sends $n_J \mapsto n_R$. Then the functor T preserves the desired reductions

$$T((n_J n_J^{\ell})n_J) = T(n_J)\big(T(n_J^{\ell})T(n_J)\big) = n_R\left(n_R^r n_R\right) \to n_R.$$

Parsing longer phrases and full sentences adds new layers of complexity. For instance, in simple short phrases there often is exactly one way of performing type reductions in order to assess the syntactic type of a phrase. Associativity can introduce ambiguity while parsing phrases. The following example demonstrates this.

EXAMPLE 4 Consider the phrase *old teachers and students*. We assign type n to *teachers* and *students*. We assign the type nn^{ℓ} to the adjective *old*. The conjunction *and* in this phrase requires two inputs of noun type to produce a noun phrase and is thus assigned $n^{r}nn^{\ell}$. We can use the associativity of the monoid operation to perform two distinct type reductions.





Both type reductions give the desired noun phrase. However, the two interpretations are slightly different. The first one attributes the adjective *old* to *teachers* only, and so the sentence is parsed as *(old teachers) and students*, while the second type reduction attributes *old* to both *teachers* and *students*, giving the phrase *old (teachers and students)*.

One can construct examples where changing the order of reductions can make the difference between reducing down to a well-formed sentence and reducing down to a phrase that cannot be grammatically accepted. For this reason, one can add a modality or a β-structure to the pregroup to locally suppress associativity. This is to ensure that our phrases reduce to the correct type or that we distribute modifiers in a prescribed way.

Pregroups with modalities were first introduced in Fadda 2002 and their logic was more extensively studied in Kiślak-Malinowska 2007.

DEFINITION 8 *A* β-pregroup *is a pregroup* $(P, \cdot, 1, -^{\ell}, -^{r}, \leq)$ *together with a monotone mapping* $\beta : P \to P$ *such that* β *has a right adjoint* $\hat{\beta} : P \to P$*, i.e. for all* $x, y \in P$ *we have* $\beta(x) \leq y$ *if and only if* $x \leq \hat{\beta}(y)$.

In practice, we enrich our pregroup grammars with types with modalities to indicate certain reductions must be performed first.

EXAMPLE 5 In our previous example, we can prescribe the parsing *(old teachers) and students* by assigning the types

$$n\,[\boldsymbol{\beta}(n)]^{\ell} \cdot [\boldsymbol{\beta}(n)] \cdot n^{r} n n^{\ell} \cdot n$$

and the parsing *old (teachers and students)* by assigning the types

$$n n^{\ell} \cdot [\boldsymbol{\beta}(n)] \cdot [\boldsymbol{\beta}(n)]^{r}\, n n^{\ell} \cdot n.$$

We now have ways to invert word order globally and block associativity locally. We conclude this section by introducing a new type of decoration which allows us to locally control word order.

The reader is also encouraged to consult Stabler 2008 for an introduction to tupled pregroups and Lambek 2010 for an analysis of French sentences using products of pregroups.

Next, we introduce a new pregroup decoration.

DEFINITION 9 *A* monoid with k-braces $(P, \cdot, 1)$ *is a free monoid in which every word is a prescribed concatenation of* $k > 0$ *distinguished subwords. Extending this and subsequent definitions to* pregroups with k-braces *is immediate.*

EXAMPLE 6 Consider the free monoid on two letters

$$F = \mathrm{Mon}(\{a, b\}).$$

Viewing F as a monoid with 2-braces, $\langle abba\rangle\langle b\rangle$ and $\langle abb\rangle\langle ab\rangle$ are distinct words because they have distinct distinguished subwords.

DEFINITION 10 *A* morphism of monoids with k-braces $f : (P, \cdot) \to (Q, *)$ *is a morphism of monoids* $f : (P, \cdot) \to (Q, *)$ *which preserves distinguished subwords. In symbols:*

$$f\,(\langle w_1\rangle \cdot \ldots \cdot \langle w_k\rangle) = \langle f(w_1)\rangle * \langle f(w_2)\rangle * \ldots * \langle f(w_k)\rangle.$$

EXAMPLE 7 [Some useful constructions] We define two morphisms of monoids with braces which are useful in understanding translations.

First, let P be a monoid with 2-braces and consider a word

$$w = \langle w_1\rangle\langle w_2\rangle.$$

Since the underlying monoid of P is free, we can view w as an element of the free product $P * P \simeq P$ where the distinguished subword w_i

belongs to the i-th factor. Take the following sequence of monoid morphisms

$$\Psi : P \simeq P * P \xrightarrow{f} P \times P \xrightarrow{g} P^{\mathrm{op}} \times P^{\mathrm{op}} \xrightarrow{h}$$
$$\xrightarrow{h} P^{\mathrm{op}} * P^{\mathrm{op}} \xrightarrow{i} Q$$

Here, f is the canonical surjection sending $\langle w_1\rangle\langle w_2\rangle \mapsto (w_1, w_2)$, g is a pair of anti-isomorphisms which act like the identity on atomic types $(w_1, w_2) \mapsto (w_1^{\mathrm{op}}, w_2^{\mathrm{op}})$, h is the canonical injection sending $(w_1^{\mathrm{op}}, w_2^{\mathrm{op}}) \mapsto \langle w_1^{\mathrm{op}}\rangle\langle w_2^{\mathrm{op}}\rangle$ and i is some fixed homomorphism of monoids with 2-braces.

Secondly, let P be a monoid with 3-braces. We construct in a similar fashion the following morphism.

$$\Xi : P \simeq P * P * P \longrightarrow P \times P \times P \longrightarrow P \times P^{\mathrm{op}} \times P \longrightarrow$$
$$\longrightarrow P * P^{\mathrm{op}} * P \longrightarrow P * P * P \simeq P \longrightarrow Q$$

We now proceed with concrete examples of phrasal translations. Throughout the remainder of the section, we work with two functorial language models: $\mathscr{J} : J \to \mathrm{FVect}$ for Japanese and $\mathscr{E} : E \to \mathrm{FVect}$ for English. We also impose the following useful metarule.

METARULE 2 *Any verb of type $so_1^r w$ also has type $o_1^\ell sw$, where w stands for all the remaining required complements.*

4.2 There is/There exists

Japanese has two verbs of existence, *iru* and *aru*, which are used for animate and inanimate beings, respectively. They both roughly mean 'to be', although a more common English translation is 'there is/there exists'.

Consider the following sentence.

(12)	森	に	猫	が	いる
	mori	ni	neko	ga	iru
	forest	LOC	cat	NOM	be

A human translator has numerous ways of approaching this sentence. A standard and literal SVO translation is *A cat is in the forest.* An easy SVO upgrade would be *A cat lives in the forest.* Considering that this is a short story meant for children, one could even opt for *In the forest lives a cat* to induce a fairy tale type atmosphere to the text. In this article, we choose to translate this using a straightforward anti-homomorphism and thus we aim for *There is a cat in the forest.*

We work with the following reduced models for grammar:

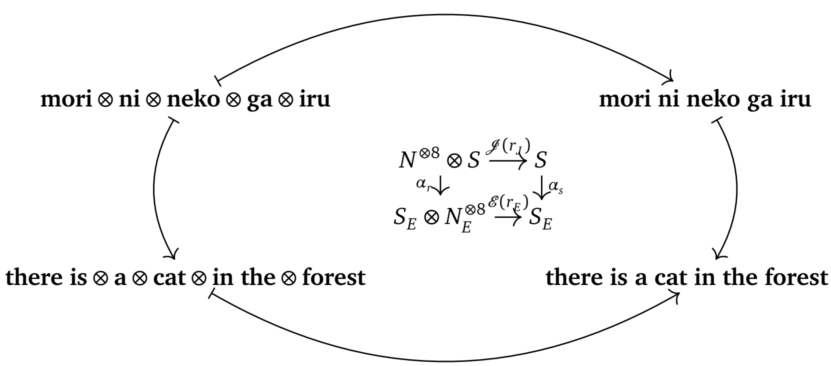
$$J = \mathrm{PGrp}(\{n, o_1, o_5, s\}) \text{ and } E = \mathrm{PGrp}(\{n_E, o_{1E}, o_{5E}, s_E\}).$$

The translation functor at the level of syntax is given by the anti-homomorphism $T : J \to E$ which sends $n \mapsto n_E, o_1 \mapsto o_{1E}, o_5 \mapsto o_{5E}, s \mapsto s_E$. At the level of semantics we have $F(n) = F(o_1) = F(o_5) = N, F(s) = S$ and $G(n_E) = G(o_{1E}) = G(o_{5E}) = N_E, G(s_E) = S_E$.

In J we have the type reduction $r = n(n^r o_5)n(n^r o_1)(o_1^r o_5^r s) \leq s$. After applying the translation functor T we get:

$$\begin{aligned} &T(n(n^r o_5)n(n^r o_1)(o_1^r o_5^r s)) \\ &\qquad = T(s)T(o_5^r)T(o_1^r)T(o_1)T(n^r)T(n)T(o_5)T(n^r)T(n) \\ &\qquad = (s_E o_{5E}^\ell o_{1E}^\ell) o_{1E} (n_E^\ell n_E) o_{5E} (n_E^\ell n_E) \\ &\qquad \to s_E o_{5E}^\ell (o_{1E}^\ell o_{1E}) o_{5E} \\ &\qquad \to s_E (o_{5E}^\ell o_{5E}) \\ &\qquad \to s_E. \end{aligned}$$

At the level of semantics we define the natural transformation $\alpha : \mathscr{J} \Rightarrow \mathscr{E} \circ T$ to act in the expected way, i.e. the map $N \to N_E$ sends **neko** $\mapsto$ **cat**, **mori** $\mapsto$ **forest** and the map $S \to S_E$ sends **iru** $\mapsto$ **there is**. We also impose **ga** $\mapsto$ **a** and **ni** $\mapsto$ **in the**. The commutativity of the following diagram is immediate.



4.3 Simple SOV sentences

We describe a procedure for translating the following sentence.

(13) 医者 は 手紙 を 書く
issya ga tegami wo kaku
doctor NOM letter ACC write
'The doctor writes a letter.'

We work with the grammars

$$J = \text{PGrp}(\{n, o_1, o_2, s\}) \text{ and } E = \text{PGrp}(\{n_E, o_{1E}, o_{2E}, s_E\}).$$

The words *issya* and *tegami* are assigned the noun type n, the particles *ga* and *wo* have the usual types $n^r o_1$ and $n^r o_2$, respectively, and the transitive verb *kaku* has type $o_2^r o_1^r s$. The sentence is clearly well-formed: $n(n^r o_1)n(n^r o_2)(o_2^r o_1^r s) \to s$.

Here we employ the notion of a pregroup with 2-braces. In principle, for an SOV sentence we assign braces as follows: $\langle S\rangle\langle OV\rangle$. In our particular sentence, this becomes

$$\Big\langle n(n^r o_1)\Big\rangle\Big\langle n(n^r o_2)(o_2^r o_1^r s)\Big\rangle.$$

We define our translation functor Ψ to be the morphism of monoids with braces defined in Example 7. Together with Metarule 2 this gives:

$$\begin{aligned}\Psi\Big\langle n(n^r o_1)\Big\rangle\Big\langle n(n^r o_2)(o_2^r o_1^r s)\Big\rangle \\ &= \Big\langle (o_{1E} n_E^\ell) n_E\Big\rangle\Big\langle (s_E o_{1E}^\ell o_{2E}^\ell)(o_{2E} n_E^\ell) n\Big\rangle \\ &= \Big\langle (o_{1E} n_E^\ell) n_E\Big\rangle\Big\langle (o_{1E}^r s_E o_{2E}^\ell)(o_{2E} n_E^\ell) n\Big\rangle\end{aligned}$$

Then α can be defined on atomic types as follows: **issya** $\mapsto$ **doctor**, **tegami** $\mapsto$ **letter**, **kaku** $\mapsto$ **write**, and the translation (Ψ, α) gives

(A/The) doctor write(s) (a/the) letter.

Again, the articles and the conjugation of *write* into third person singular can either be added by brute force in our model by adding meanings to the particles *ga* and *wo*, or one can verify agreement and articles separately as a different step in the translation process.

Interpreting relative pronouns in various languages in terms of pregroups proves to be quite challenging. In Sadrzadeh *et al.* 2013 and Sadrzadeh *et al.* 2014, the authors add the additional structure of a Frobenius algebra on the pregroup. Informally, a Frobenius algebra structure enriches the ε, η functorial yoga with additional maps, the most important of which are called "copying map" and "uncopying map." These new morphisms allow one to better keep track of information inside a phrase. For instance, in the English sentence

The woman, who drove from Tokyo today, was late to the party

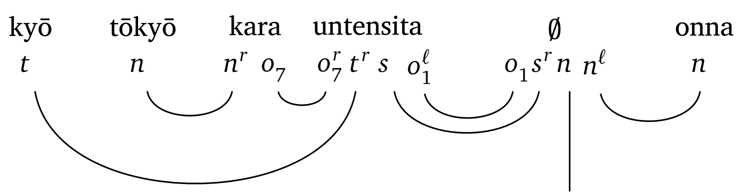
the new morphisms can formalise the fact that the subject of the main clause *The woman was late to the party* and the subject of the relative clause *who drove from Tokyo today* are one and the same. The relative pronoun *who* acts as a bridge that "copies" the subject into the relative clause and then transfers it back into the main clause.

We translate the following relative clause.

(14)	今日	東京	から	運転した	女
	kyō	tōkyō	kara	untensita	onna
	today	Tokyo	ABL	drove	woman

'The woman who drove from Tokyo today.'

We assign types in a less straightforward way. We first insert an empty word between the modifier *tōkyō kara untensita* and the head *onna*. We assign the following types: *tōkyō* and *onna* are both type n, the ablative particle *kara* has type $n^r o_7$, *kyō* has type t (temporal adverb), the verb *untensita* has type $o_7^r t^r s o_1^\ell$ and the empty word acts like a phantom relative pronoun with type $o_1 s^r n n^\ell$.



This construction generates a noun phrase and it can be translated using a straightforward anti-homomorphism. The advantage of this underhanded construction is that now we can translate the empty word as the relative pronoun *who* or *that*. This ties in perfectly with the Frobenius algebra approach of Sadrzadeh *et al.* (2013). In this example we modelled our relative clause as what the authors of the reference call a subject relative clause.

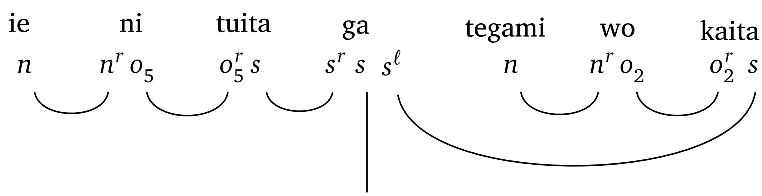
4.5 *Coordinate sentences*

The simplest way of coordinating sentences is by connecting them with the particle *ga* ‘and’ to which we assign the type $s^r s s^\ell$. We translate the following sentence where subjects are omitted.

(15)	家	に	着いた	が	手紙	を	書いた
	ie	ni	tuita	ga	tegami	wo	kaita
	house	LOC	arrived	and	letter	ACC	wrote

‘I arrived home and wrote a letter.’

In Japanese we have the following reduction diagram.



We decorate the pregroup with braces and assign the following type

$$\left\langle n \cdot n^r o_5 \cdot o_5^r s \right\rangle \left\langle s^r s s^\ell \right\rangle \left\langle n \cdot n^r o_2 \cdot o_2^r s \right\rangle.$$

Extending the morphism Ψ from Example 7 to monoids with 3-braces, we obtain

$$\begin{aligned}\Psi\langle n \cdot n^r o_5 \cdot o_5^r s\rangle \langle s^r s s^\ell\rangle \langle n \cdot n^r o_2 \cdot o_2^r s\rangle \\ = \langle s_E o_{5E}^\ell \cdot o_{5E} n_E^\ell \cdot n_E\rangle \langle s_E^r s_E s_E^\ell\rangle \langle s_E o_{2E}^\ell \cdot o_{2E} n_E^\ell \cdot n_E\rangle.\end{aligned}$$

Working under the assumption that an omitted subject refers to the first person singular, the translation, after applying a suitably defined α, is

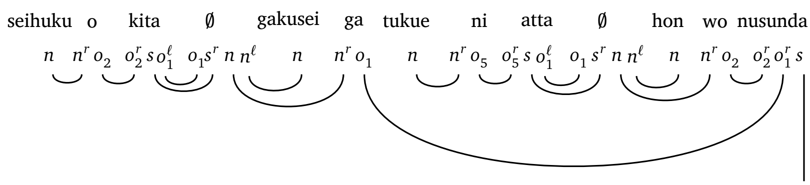
(I) arrived home and (I) wrote (a) letter.

Putting it all together 4.6

We combine all our techniques to study a more complex sentence.

(16) 制服 を 着た 学生 が 机 に あった
seihuku o kita gakusei ga tukue ni atta
uniform ACC wore student NOM desk LOC was
本 を 盗んだ
hon wo nusunda
book ACC stole
'The student, who wore a uniform, stole the book, which was on the desk.'

This is a standard SOV sentence, where both the subject and the direct object are modified by relative clauses. In the Japanese pregroup grammar, we have the following straightforward reductions.



One may observe that in the diagram above we use associativity to our advantage to prove that the sentence reduces to the correct syntactic type. To get a fail-safe reduction and translation we decorate our pregroup grammar with braces and a β-structure. The sentence is then assigned the type

$$\left\langle n \cdot n^r o_2 \cdot o_2^r o_1^\ell \cdot o_1 s^r n \boldsymbol{\beta}(n^\ell) \cdot \boldsymbol{\beta}(n) \cdot n^r o_1 \right\rangle$$
$$\left\langle n \cdot n^r o_5 \cdot o_5^r s o_1^\ell \cdot o_1 s^r n \boldsymbol{\beta}(n^\ell) \cdot \boldsymbol{\beta}(n) \cdot n^r o_2 \cdot o_2^r o_1^r s \right\rangle$$

and after applying the morphism Ψ from Example 7 together with Metarule 2, the sentence translates to

*(A/The) student, who wore (a/the) uniform, stole (a/the) book, which was (*LOC*) desk.*

4.7 *A Farsi to Japanese example*

Farsi has certain similarities to Japanese which make translations (at the syntactic level, at least) somewhat simpler. For instance, Farsi also has SOV word order, nouns do not possess grammatical gender, and it is a pro-drop language. A key structural difference is that Farsi uses both prepositions and postpositons as case markers.

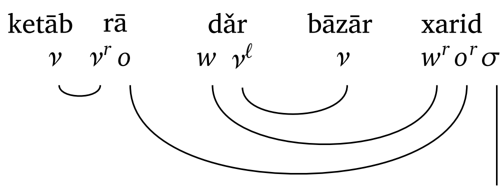
Following Sadrzadeh (2007), we use the following (reduced) pregroup to model Farsi grammar $F = \mathrm{PGrp}(\{v, \sigma, o, w\})$, where atomic types represent nouns, sentences, direct objects, and prepositional phrases respectively. For Japanese, we use $J = \mathrm{PGrp}(\{n, s, o_2, o_5\})$, with the usual meanings. Denote the two functorial language models as $\mathscr{F} : F \to \mathrm{FVect}$ and $\mathscr{J} : J \to \mathrm{FVect}$.

We are interested in translating the following sentence from Farsi to Japanese.

(17)	*ketāb*	*rā*	*dăr*	*bāzār*	*xarid*
	book	ACC	PREP	market	bought

‘He/She bought a book from the market.’

Here *ketāb rā* is the direct object, *dăr bāzār* is the prepositional phrase and *xarid* is the transitive verb in the past tense. This example sentence drops the subject and uses both a postposition *rā* and a preposition *dăr* to mark cases. In Farsi, we have the following reduction.



The functorial language models $\mathscr{F}, \mathscr{J}$ send $v, o, w \mapsto N_F$ (Farsi nouns) and $\sigma \mapsto S_F$ (Farsi sentences), and also $n, o_2, o_5 \mapsto N_J$ (Japanese nouns) and $s \mapsto S_J$ (Japanese sentences). The natural transformation $\alpha : \mathscr{F} \Rightarrow \mathscr{J} \circ i$ is given by **ketāb** $\mapsto$ **hon**, **rā** $\mapsto$ **o**, **dă** $\mapsto$ **de**, **bāzār** $\mapsto$ **itiba** and **xarid** $\mapsto$ **kaimasita**. At the syntactic level, we define some monoidal translation functor $T : F \to J$ which takes $v \mapsto n, \sigma \to s, o \to o_2$, and $w \to o_5$.

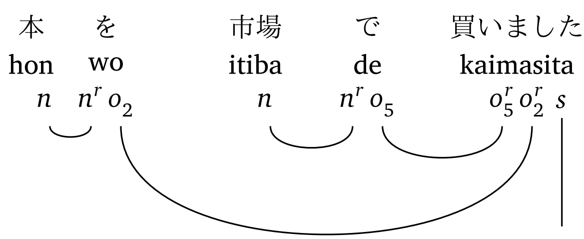
The pregroups are decorated with 3-braces. The sentence is assigned the type

$$\left\langle \nu \cdot \nu^r o \right\rangle \left\langle w \nu^\ell \cdot \nu \right\rangle \left\langle w^r o^r \sigma \right\rangle.$$

Syntactically, the translation functor is taken to be Ξ from Example 7. The word order is altered as follows

$$\Xi \left\langle \nu \cdot \nu^r o \right\rangle \left\langle w \nu^\ell \cdot \nu \right\rangle \left\langle w^r o^r \sigma \right\rangle = \left\langle n \cdot n^r o_2 \right\rangle \left\langle n \cdot n^r o_5 \right\rangle \left\langle o_5^r o_2^r s \right\rangle$$

which leads to the following type reduction in Japanese.



5 FUTURE WORK

In this article, we introduced decorated pregroups and used them as a means of constructing a compositional notion of translation between natural languages with different word order. The aim was to demonstrate that one can maintain a categorical approach to modelling translation without compromising on functoriality altogether. Some of our constructions are ad-hoc and there is room for improving most of them.

First, there is the issue of translating between a language where nouns do not have grammatical gender and number to a language that does. Using product pregroups or tupled pregroups to handle grammatical agreement could be a way forward, although a straightforward model for achieving this appears elusive.

Secondly, one could study translations between languages which have more featural and structural differences. For example, how could we interpret (functorially) translations between a language which has nominative-accusative alignment and a language that has ergative-absolutive (or split-ergative) alignment?

Thirdly, this article focuses heavily on syntax. It would be interesting to model how meaning in translation can be negotiated between different speakers and how one can keep track of their evolving semantic spaces. On a more technical note, one could change the meaning space from FVect to a category that possesses more substantial structure such as ConvexRel, the category where the objects are *convex algebras* and the morphisms as *convex relations*. In Bolt *et al.* 2019 the authors showed that ConvexRel is a compact closed symmetric monoidal category and is thus suitable for modelling semantics in a compositional distributional functorial language model.

Finally, separate from the question of translation, some attention could be dedicated to expanding the work of Cardinal (2002; 2006; 2007) and producing a more complete pregroup approach to analysing other aspects of grammar that are typical to Japanese. In particular, the structure of coordinate and subordinate sentences and internally-headed relative clauses are of particular interest to the author.

REFERENCES

Daniele Bargelli and Joachim Lambek (2001a), An algebraic approach to French sentence structure, in *International Conference on Logical Aspects of Computational Linguistics*, pp. 62–78, Springer, DOI:10.1007/3-540-48199-0_4.

Donna Bargelli and Joachim Lambek (2001b), An algebraic approach to Arabic sentence structure, *Linguistic Analysis*, 31(3):301–315, Corpus ID: 56575156.

Joe Bolt, Bob Coecke, Fabrizio Genovese, Martha Lewis, Dan Marsden, and Robin Piedeleu (2019), *Interacting conceptual spaces I: Grammatical composition of concepts*, Springer, DOI:10.1007/978-3-030-12800-5_9.

Tai-Danae Bradley, Martha Lewis, Jade Master, and Brad Theilman (2018), Translating and evolving: Towards a model of language change in discocat, *ECPTS*, DOI:10.4204/EPTCS.283.4.

Kumi Cardinal (2002), *An algebraic study of Japanese grammar*, PhD Thesis.

Kumi Cardinal (2006), Type grammar meets Japanese particles, in *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, pp. 142–149.

Kumi Cardinal (2007), A pregroup analysis of Japanese causatives, in *Proceedings of the Korean Society for Language and Information Conference*, pp. 96–104, Korean Society for Language and Information.

Claudia Casadio and Jim Lambek (2005), A computational algebraic approach to Latin grammar, *Research on Language and Computation*, 3(1):45–60, DOI:10.1007/s11168-005-1286-0.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark (2010), Mathematical foundations for a compositional distributional model of meaning, *Linguistic Analysis*, DOI:10.48550/arXiv.1003.4394.

Mario Fadda (2002), Towards flexible pregroup grammars, *New Perspectives in Logic and Formal Linguistics*, pp. 95–112.

Gregory M. Kelly and Miguel L. Laplaza (1980), Coherence for compact closed categories, *Journal of Pure and Applied Algebra*, 19:193–213, DOI:10.1016/0022-4049(80)90101-2.

Aleksandra Kiślak-Malinowska (2007), On the logic of β-pregroups, *Studia Logica*, 87(2):323–342, DOI:10.1007/s11225-007-9090-5.

Joachim Lambek (1997), Type grammar revisited, in *International conference on logical aspects of computational linguistics*, pp. 1–27, Springer, DOI:10.1007/3-540-48975-4_1.

Joachim Lambek (2010), Exploring feature agreement in French with parallel pregroup computations, *Journal of Logic, Language and Information*, 19(1):75–88, DOI:10.1007/s10849-009-9098-5.

Joachim Lambek and Anne Preller (2004), An algebraic approach to the German sentence, *Linguistic Analysis*, HAL ID: irmm-00108541.

Jeff Mitchell and Mirella Lapata (2008), Vector-based models of semantic composition, in *Proceedings of ACL-08: HLT*, pp. 236–244, https://aclanthology.org/P08-1028.pdf.

Mehrnoosh Sadrzadeh (2007), *Pregroup analysis of Persian sentences*, Citeseer, https://www.cs.ox.ac.uk/files/2416/PersPreGroup.pdf.

Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke (2013), The Frobenius anatomy of word meanings I: subject and object relative pronouns, *Journal of Logic and Computation*, 23(6):1293–1317, DOI:10.1093/logcom/ext044.

Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke (2014), The Frobenius anatomy of word meanings II: possessive relative pronouns, *Journal of Logic and Computation*, 26(2):785–815, DOI:10.1093/logcom/exu027.

Edward P. Stabler (2008), Tupled pregroup grammars, *Computational Algebraic Approaches to Natural Language*, pp. 23–52.

Valentin Boboc
0000-0003-0661-5404
valentinboboc@icloud.com

University of Manchester

Valentin Boboc (2023), *An algebraic approach to translating Japanese*, Journal of Language Modelling, 11(1):121–146
https://dx.doi.org/10.15398/jlm.v11i1.315

We thought the eyes of coreference were shut to multiword expressions and they mostly are

Agata Savary[1], *Jianying Liu*[2], *Anaëlle Pierredon*[2], *Jean-Yves Antoine*[3], *and Loïc Grobol*[4]
[1] Paris-Saclay University, CNRS, LISN, France
[2] Inalco, Paris, France
[3] University of Tours, LIFAT, France
[4] Paris-Nanterre University, MoDyCo, CNRS, France

ABSTRACT

Multiword expressions are combinations of words that exhibit peculiar semantic properties, such as different degrees of non-compositionality, decomposability, transparency and figuration. Long-standing linguistic debates suggest that such semantic idiosyncrasy can condition the morpho-syntactic configurations in which a given multiword expression can occur. Here, we extend this argumentation to a particular semantic and pragmatic phenomenon: nominal coreference. We hypothesise that the internal components of a multiword expression are unlikely to occur in coreference chains. While previous work has identified the rareness of coreference-related phenomena in presence of multiword expressions, this observation has never been quantified, to the best of our knowledge. We bridge this gap by performing an automated corpus-based study of the intersections between verbal multiword expressions and nominal coreference in French. The results largely corroborate our hypothesis but also display various tendencies depending on the type of multiword expression and the corpus genre. The analysis of the corpus examples highlights interesting properties of coreference, notably in speech.

Keywords: *multiword expressions, coreference, corpus linguistics*

1 INTRODUCTION

Multiword expressions (MWEs), such as ***every so often*** 'from time to time', ***top dog*** 'a person who is successful or dominant in their field', ***beyond recall*** 'impossible to retrieve', ***saw logs*** 'to snore', or ***strike while the iron is hot*** 'to make use of an opportunity immediately' are combinations of words that exhibit idiosyncratic behavior. Most prominently, they are semantically non-compositional, i.e. their meaning cannot be deduced in a way deemed regular from the meanings of their components and their syntactic structure.

Linguistic studies argue that semantic non-compositionality is a matter of scale rather than a binary phenomenon (Gross 1988) and is mitigated by other semantic properties such as decomposability, figuration and transparency (Nunberg 1978; Gibbs and Nayak 1989; Moon 1998; Sheinfux *et al.* 2019). These properties should be the reasons behind lexical, morphological and/or syntactic inflexibility of MWEs, i.e. the fact that certain constructions or transformations, normally allowed in a language, are blocked or infrequent in MWEs. For instance in *work while the kids are asleep*, which is a regular compositional construction, a lexical replacement of the verb and a modification of the adjective lead to an expression whose meaning shift with respect to the original expression is predictable from the formal change, as in *study while the kids are fast asleep*. However, a similar change in the weakly decomposable MWE ***strike while the iron is hot*** leads to the loss of the idiomatic reading, as in *hit while the iron is very hot*.

Some studies show that MWEs impose limitations also on semantic and pragmatic phenomena such as coreference, i.e. the process in which several discourse entities refer to the same discourse world referent. For instance in example (1),[1] the expression *sawing logs* has a compositional meaning and coreference occurs between the object (*logs*) and the pronoun (*them*). If this expression were used

[1] The presentation of inline and numbered examples follows the conventions put forward by the *Phraseology and Multiword Expressions* book series, see `https://gitlab.com/parseme/pmwe/-/blob/master/Conventions-for-MWE-examples/PMWE_series_conventions_for_multilingual_examples.pdf`.

idiomatically (meaning 'to snore'), then coreference would be prohibited, as in (2).

(1) By sawing logs you transform them into lumber. (en)

(2) *He was **sawing logs** for the whole night – I could hardly sleep! He should ask a doctor how to get rid of them. (en)

Such relationships and constraints at the crossroads between MWEs and coreference are the object of this work. More precisely, we are interested in the likelihood that internal components of MWEs (rather than whole MWEs) occur in coreference chains. Isolated examples of this kind, such as (3),[2] are cited in previous works but this phenomenon seems not to have been quantified in the past. We aim to bridge this gap through an automated corpus study in which MWEs and coreference chains are identified and studied jointly. More precisely, we focus on verbal MWEs, such as ***saw logs*** 'snore' and ***keep tabs on someone*** 'carefully watch someone', and on nominal coreference (i.e. coreference occurring among nominal phrases and/or pronouns). Our language of study is French.

(3) We thought tabs were being **kept** on us but they weren't. (en)

'We thought we were being carefully watched but we weren't.'
(Nunberg *et al.* 1994, our paraphrasing)

This paper is organized as follows. In Section 2, we present linguistic debate on interactions between the semantic and morpho-syntactic properties of MWEs, including reference and coreference. In Section 3, we introduce basic definitions related to MWEs and coreference, and we define the scope of our work. In Section 4, we describe the experimental setting of our corpus study. In Section 5, we present its quantitative and qualitative results and discuss the initial hypothesis and objectives in the light of these results. In Section 6, we discuss some phenomena highlighted by the experiments and we suggest perspectives for future work. Finally, we conclude in Section 7.

[2] Examples found in previous works and in corpora are documented with their sources, as in (3) and (19). All other examples are ours.

2 RELATED WORK

Explicit links between multiword expressions and coreference do not appear to have been studied extensively. However, linguistic debates about correlations between the semantic properties of MWEs and their morpho-syntactic behavior have important implications for our work.

2.1 *Decomposability and reference*

One such debate touches upon the hypothesis that the morpho-syntactic flexibility of idioms (a subtype of the MWEs considered in this work) is conditioned by their degree of semantic *decomposability*.

Following Nunberg (1978), Gibbs and Nayak (1989) claim that, despite the overall semantic non-compositionality of idioms, the components of some idioms can be assigned non-standard meanings, each of which may contribute to the expression's figurative interpretation. For instance, within the idiom *to* ***spill*** *the* ***beans*** 'to reveal a secret', the individual components *spill* and *beans* can be assigned metaphorical interpretations ('reveal' and 'secret', respectively). Each of them then contributes its 'abnormal' interpretation to the meaning of the idiom, which may thus be termed decomposable. Importantly for our work on coreference, Gibbs and Nayak (1989) stress the fact that decomposability touches upon the question of *reference*, as components of decomposable idioms "refer in some way to the components of their figurative referents". This is very explicit in example (4).

(4) To regard savings as the animating force in this scheme of things is to **put the cart before the horse**. The horse is the growth of national income […]; the harness linking horse and cart the financial system, and bringing up the rear is the cart of saving. (en)

(Moon 1998)

Further, for Gibbs and Nayak (1989), decomposability of idioms is a rationale behind their morpho-syntactic flexibility. Another flexibility facet, directly related to coreference, is *pronominalization* (cf. Section 2.3).

Figuration and transparency 2.2

Two other semantic properties of idioms are figuration and transparency (Gibbs and Nayak 1989; Sheinfux *et al.* 2019), which describe the relationship between their idiomatic and literal readings. *Figuration*[3] refers to the degree to which the idiom can be assigned a literal meaning. For instance, *to **skate on thin ice*** 'to be in a precarious situation' evokes a vivid image that is easy to imagine (the idiom is strongly figurative). Conversely, *to **drop a line*** 'to write a letter' and to ***take umbrage*** 'to take offense' have barely conceivable literal meanings (are non-figurative), especially when they contain so-called *cranberry words* (tokens having no status as standalone words but only occurring in MWEs) such as *umbrage.*[4] *Transparency* relates to how understandable the link is between the literal and the idiomatic reading. For instance, since ***skating on thin ice*** is literally dangerous, it is easy to understand the motivation behind its idiomatic reading 'to be in a precarious situation' (the idiom is transparent). Conversely, without expert historical knowledge it is hard to understand why ***kicking the bucket*** means 'to die' (the idiom is opaque). Gibbs and Nayak (1989) show a significant positive correlation between transparency and syntactic flexibility.

While the experiments of Gibbs and Nayak (1989) focus on 36 English idioms in artificially constructed utterances, Sheinfux *et al.* (2019) performed large-scale corpus studies. First, in a 20-billion word English corpus, they identified examples of syntactic flexibility for ***kick the bucket*** 'die', which questions the decomposability hypothesis (Section 2.1). They further used a 1-billion word Hebrew corpus to query occurrences of 15 specific verbal idioms. They show that transparent figurative idioms like (he) ***yarad me-ha-ʕec*** (lit. 'descended from the tree') 'conceded' are highly syntactically flexible, since the referent in the literal meaning (a tree) is easy to capture. Conversely, opaque figurative idioms like (he) ***ṭaman yad-o ba-calaħat*** (lit. 'buried his hand in the plate') 'refrained from acting' are syntactically rigid. Surprisingly, opaque non-figurative idioms, like (he) ***ʔavad ʕal-av*** (***ha-***)***kelaħ***

[3] Gibbs and Nayak (1989) use the term *well-formedness* instead.

[4] The word *umbrage* seems to be a cranberry word in British English but less so in American English, where it has synonyms like *shadow* or *foliage*.

(lit. '(the-)KELAH was lost on him') 'became outdated', exhibit some flexibility, which the authors interpret as the ability of the speakers to attribute semantic content to the meaningless cranberry words (*kelaħ*). Although Sheinfux *et al.* (2019) do not explicitly study coreference with MWE components, the examples of flexibility they found do include related phenomena like pronominalization and extraction, as discussed in the following section.

2.3 *Pronominalization and extraction*

Several studies have viewed the pronominalization of internal components of MWEs as a facet of their morpho-syntactic flexibility (or variation).

Moon (1998) studied fixed expressions and idioms in several English corpora, totalling 18 million words, using a knowledge base of 6,776 MWEs. She addressed various transformations and variations in which MWEs can occur, including pronominalization stating that "it is normally the case that fixed nominal groups in [fixed expressions and idioms] are not pronominalized". She found isolated examples in which a pronoun does corefer with an extracted nominal group occurring in the immediately preceding context, as in (5) and (6).

(5) Mr Lawson was **swimming with** <u>that **tide**</u>. Mrs Thacher was swimming against <u>it</u>. (en)

'Mr Lawson was acting in accordance with the prevailing opinion. Mrs Thacher was acting against it.'
(Moon 1998, paraphrasing is ours)

(6) If there is <u>**ice**</u>, Mr Clinton is **breaking** <u>it</u>. (en)

'If there is tension, Mr Clinton is relieving it.'
(Moon 1998, paraphrasing is ours)

Gibbs and Nayak (1989) hypothesised pronominalization as evidence of decomposability (cf. Section 2.1). They carried out experiments with human acceptability ratings of utterances containing English idioms whose components were pronominalized, as in (7) and (8). The results show higher rankings for pronominalization with semantically decomposable (7) than with nondecomposable (8) idioms.

(7) After they were divorced, Tony began to **hit the sauce**, but Cathy didn't begin to hit it. (en)

'After they were divorced, Tony began to drink heavily, but Cathy didn't begin to.' (Gibbs and Nayak 1989)

(8) The guys **chewed the fat** over coffee, but the girls didn't chew it. (en)

'The guys talked over coffee, but the girls didn't.' (Gibbs and Nayak 1989)

Moon (1998) and Sheinfux *et al.* (2019) also cite examples of *extraction* (also called *embedding*) of the lexicalized nominal group that leads to a relative clause. This introduces a relative or personal pronoun that can be considered as coreferent with the NP, as shown in examples (9) and (10)

(9) [The escapees] have a work **habit** which is hard to **kick**. (en)

'[The escapees] have a harmful habit which is hard to give up' (Moon 1998, paraphrasing is ours)

(10) ze lo **ʕec** gavoha [ʃe-nitan **laredet** mime-no]. (he)
this not tree tall that-possible to.descend from-him

'This is not an unrealistic stance that it is possible to withdraw from.' (Sheinfux *et al.* 2019)

In sum, the works covered in this section do provide examples of the MWE and coreference intersections that are our focus here, but which are either rare (and not quantified) or artificially constructed for the sake of the experiments.

2.4 *Coreference as an MWE classification criterion*

Laporte (2018) argued that, since MWEs encompass heterogeneous linguistic phenomena, their computational modeling and processing call for classifications. He advocated clear-cut syntactically motivated classification features, in the spirit of the Lexicon-Grammar (Gross 1994), against fuzzy semantic features, such as decomposability (Section 2.1). He claimed that decomposability is reliably approximated by a combination of tests, two of which are based on coreference.

Firstly, in a decomposable MWE, a component "can be the first in a chain of coreferring expressions, and then the syntactic markers of the coreference: determiners, pronouns, etc., follow the same rules as when the noun is not part of the idiom". For instance, in (11), the object *témoin* 'witness' is the first mention in a coreference chain and its coreferring pronoun *il* 'he' is the same as in (12), where no idiom occurs.

(11) La défense a **cité** un **témoin**. Il vient de s'exprimer. (fr)
lit. 'The defense quoted a witness. He has just expressed himself.'
'The defense called a witness. He has just spoken.'
(Laporte 2018)

(12) La défense a un témoin. Il vient de s'exprimer. (fr)
lit. 'The defense has a witness. He has just expressed himself.'
'The defence has a witness. He has just spoken.'
(Laporte 2018)

Conversely, in a non-decomposable idiom, as in (13), the object *mauvaise posture* 'bad posture' admits an indirect coreference[5] (with *ces difficultés* 'this trouble')[6] but not a direct one (with *cette posture* 'this posture'), as shown in (14). This is despite the fact that direct coreference is admitted in a non-idiomatic use of the same nominal group, as in (15).

(13)

Kathy	**était**	**en**	**mauvaise**	**posture**.	Ces	difficultés	auraient
Kathy	was	in	bad	posture.	These	difficulties	have

pu	être	évitées.	(fr)
could	be	avoided.	

'Kathy was in trouble. This trouble could have been avoided.'
(Laporte 2018, gloss and translation slightly adjusted)

[5] Direct coreference occurs when two coreferent mentions have lexically the same head (*a witness …, the witness*). Otherwise a coreference is pronominal (*a witness …, he*), or indirect (*a witness …, the person*) – see Section 3.

[6] Alternatively to this analysis by Laporte (2018), it could be argued that *ces difficultés* 'these troubles' corefer with the whole event *était en mauvaise posture* (lit. 'was in bad posture') 'was is trouble' rather than with *mauvaise posture* 'bad posture' alone (see also Section 4.2).

(14) *Kathy **était en mauvaise posture**. Cette posture aurait pu
Kathy was in bad posture. This posture has could
être évitée. (fr)
be avoided.
'Kathy was in trouble. This trouble could have been avoided.'
(Laporte 2018)

(15) Kathy avait une posture fière. Cette posture a été
Kathy had a proud posture. This posture has been
commentée. (fr)
commented.'
'Kathy had a proud posture. This posture has been commented on.' (Laporte 2018)

Laporte's ideas provided a direct inspiration for our study. They suggest a strong correlation between the idiomaticity of an expression and the impossibility of coreferring to its components, to the point of considering this correlation a defining property of MWEs. The main difference in our approach is to quantify this correlation via a corpus study, rather than to test it introspectively.

To summarize, in the light of the state of the art presented above, it appears that various MWEs have various degrees of semantic non-compositionality, decomposability, figuration and transparency (Sections 2.1-2.2). These semantic properties condition the morpho-syntactic configurations in which MWEs are likely to occur. As a result, testing the acceptability of morpho-syntactic variants is a good approximation for defining idiomaticity, as also advocated by the PARSEME guidelines for verbal MWE annotation (Savary *et al.* 2018).

Some of the syntactic configurations that are more or less acceptable in MWEs include coreference-related phenomena such as pronominalization and extraction (Section 2.3). Therefore, precise coreference-related tests might belong to MWE definition and classification criteria (Section 2.4).

3 DEFINITIONS AND SCOPE

In this work, concepts related to MWEs are defined as in the PARSEME framework (Savary *et al.* 2018). The MWE is understood as a combination of words that contains at least two *lexicalized component* words, and displays some degree of lexical, morphological, syntactic and/or semantic idiosyncrasy. Lexicalized components, highlighted in bold throughout this paper, are those components of the MWE that are always realized by the same lexemes, as opposed to *open slots*, i.e. arguments that are compulsory but not lexically constrained. For instance, in (en) *he* ***took*** *me* ***by surprise***, the verb and the prepositional objects are lexicalized, while the subject and the object are open slots. Multiword expressions can occur in corpora as morpho-syntactic variants, e.g. (en) *he was* ***taking*** *me* ***by surprise***, *I was* ***taken by surprise***, etc. The *canonical form* of the MWE is defined as the least syntactically marked variant that preserves the idiomatic reading.[7] For instance, the first example above is less syntactically marked than the other two since it contains a finite verb in active voice rather than a participle with passive voice.

A *verbal MWE* (VMWE) is an MWE whose canonical form is headed by a verb. The PARSEME annotation guidelines[8] distinguish 5 VMWE categories, 4 of which are annotated in the French PARSEME corpus. First, *light verb constructions* (*LVCs*) are verb(-preposition)-noun combinations in which the verb is semantically void or bleached, and the noun is predicative. There are two subcategories: *LVC.full*, where the verb's subject is the noun's semantic argument, as in (fr) *la chanson* ***connut*** *un grand* ***succès*** (lit. 'the song knew a big success') 'the song was a big success'; *LVC.cause*, where the noun is not a semantic argument of the verb, but adds a causative meaning to it, as in (fr) *il* ***donne espoir*** *aux soldats* 'he gives hope to the soldiers'. Second, a *verbal idiom* (*VID*) is a verbal construction of any syntactic structure that contains a cranberry word or exhibits lexical, morphological, or

[7] A singular form is less marked than a plural; active voice is less marked than passive; a finite verb is less marked than an infinitive; a form with an extraction is more marked than one without it, etc.

[8] `https://parsemefr.lis-lab.fr/parseme-st-guidelines/1.2/`

syntactic inflexibility (cf. Sections 2.1–2.2), as in (fr) *ces textes* ***font foi*** (lit. 'these texts do faith') 'these texts apply'. Third, an *inherently reflexive verb* (*IRV*) is an idiomatic combination of a verb and a reflexive clitic, as in (fr) ***se comporter*** (lit. 'to contain oneself') 'to behave'. Fourth, a *multi-verb construction* (*MVC*) is an idiomatic combination of two verbs, such as (fr) ***laisser tomber*** (lit. 'to let fall') 'to abandon'.

As with coreference, we do not commit to a particular framework: we simply call *mentions* linguistic elements (usually constituents) that refer to discourse *entities* (that might be real-world or fictional objects or individuals, concepts or events). Throughout this paper, mentions are highlighted with straight underlining. Mentions are said to be *coreferent* if they refer to the same entity, and the set of all mentions referring to a given entity is called a *coreference chain*. If a coreference chain consists of at least two mentions, it is called *non-trivial*. Otherwise, it is called *trivial* and the sole mention it contains is referred to as a *singleton*. The term *chain* underlines that the order of occurrence of the mentions of a non-trivial chain is usually significant, since the interpretation of a given mention *m* in a chain depends on the interpretation of the preceding mentions of the chain, called *antecedents* of *m*.

In natural language processing, the *coreference resolution* task is usually understood as a process with two steps: detecting the mentions in a document, and partitioning their set into coreference chains. For practical considerations, *nominal* coreference resolution – limited to mentions that are either noun phrases or pronouns – and *event* coreference resolution – limited to verb phrases and pronouns referring to events – are usually treated as different tasks. Within nominal coreference, we identify three cases for a pair of coreferent mentions:

Pronominal coreference if one of the mentions is a pronoun, as in (16).

Direct coreference if both mentions are noun phrases sharing a syntactico-semantic head, as in (17).

Indirect coreference if both mentions are noun phrases that *do not* share a syntactico-semantic head, as in (18).

(16) The crow was perched in a tree. It had a white feather. (en)

(17) I saw a man with a beautiful cat. The cat was deeply asleep. (en)

(18) Do not wander in the western forest! No one ever came back from these dark woods. (en)

The state of the art presented in Section 2 addresses (more or less explicitly) interactions between idiomaticity and coreference. None of these works, however, quantifies these interactions on real corpus data. Our work aims to contribute to bridging this gap. More precisely, we put forward the following hypothesis:

$\mathscr{H}$ Proper subsets of lexicalized components of MWEs are unlikely to occur in non-trivial coreference chains.

Additionally to corroborating (or invalidating) this hypothesis, our objective is to:

$\mathscr{O}$ Characterize those situations in which coreference with proper subsets of MWE components does occur.

For the sake of experimental feasibility, we further define the precise scope of our study as follows:

- We focus on nominal coreference, for its much better coverage in the state of the art than event coreference, in terms both of resources and tools. Moreover, non-nominal mentions tend to be verb phrases referring to events and are unlikely to appear as proper subsets of lexicalized components of MWEs.
- We focus on verbal MWEs (VMWEs) since: (i) they occur in syntactic structures where proper subsets of lexicalized components form nominal phrases, i.e. potential nominal mentions (such as *the cart* and *the horse* in ***put the cart before the horse***), (ii) they exhibit a relatively high degree of morpho-syntactic variation, (iii) research on VMWEs has been recently boosted by cross-linguistically unified corpus annotation campaigns and shared tasks on automatic identification of VMWEs (Ramisch *et al.* 2020).
- We focus on French since, for this language, we have access to the resources (corpora annotated manually for VMWEs and nominal coreference) and tools (VMWE identifiers and coreference solvers) needed for the experimental setting.

In sum, this section provides definitions of the basic notions important for this work: a (notably verbal) multiword expression, its lexicalized components and its canonical form; the 4 types of VMWEs relevant to French; a mention, a (trivial and non-trivial) coreference chain and 3 types of nominal coreference. We also formulate our research hypothesis $\mathscr{H}$ and a secondary research objective $\mathscr{O}$. Finally, we define our scope, namely nominal coreference and verbal MWEs in French.

In the following section, we describe the experimental setting designed to address $\mathscr{H}$ and $\mathscr{O}$ within an automated corpus study.

SEARCHING FOR MWE AND COREFERENCE INTERSECTION: METHODOLOGY 4

In brief, the experimental setting includes three French corpora: the first two annotated manually for nominal coreference and VMWEs, respectively, and the third one with no manual annotations at either of these two levels. We apply two NLP tools – a coreference solver and a VMWE identifier – to provide parallel coreference and VMWE annotations in each of the 3 corpora. We automatically search for relevant intersections, i.e. VMWE components occurring in non-trivial coreference chains. We manually validate these intersections so as to identify true positives, for which we then provide quantitative and qualitative analyses. All these steps are described below in more detail.

Corpora 4.1

The corroboration of hypothesis $\mathscr{H}$ requires the corpora to satisfy three conditions:

- The annotations of VMWEs and coreference chains have to be reliable enough for further analysis and comparisons. Therefore, corpora with human annotations are preferred over others and automatic annotation should pass a human check.
- Since coreference chains can spread over several sentences or whole texts, the chosen corpora need to bear some marks of text boundaries. Each text should contain more than one sentence, and should preserve the sentence order and the article structure.

- Since the studied phenomenon is supposed to appear rarely, the chosen corpora should cover various topics and writing styles.

Corresponding to these criteria, the optimal existing resources are: (i) the French ANCOR corpus annotated for coreference (Muzerelle *et al.* 2014), (ii) the French PARSEME corpus annotated for VMWEs (Candito *et al.* 2017). Since they already have human annotation on one side (coreference or VMWEs, respectively), they only need to be annotated automatically and checked manually for the other side, which alleviates the amount of manual work.

The ANCOR corpus consists of transcriptions of oral conversations, including short and long interviews, as well as interactive and phone dialogues. Each conversation is segmented into speech turns. Except for question marks, no punctuation exists in the transcription.

The French PARSEME corpus keeps sentence boundary but not text boundary information and uses mostly disordered sentences. We retain only part of its Sequoia subcorpus (Candito *et al.* 2014), which contains ordered sentences and where the article boundaries are retrievable. It consists of medical reports (emea subcorpus), Wikipedia articles on historical social events (frwiki supcorpus), and articles from the Est Républicain newspaper (annodis.ER subcorpus).

To increase the amount and variety of the data, we also use a raw corpus composed of news articles from the Est Républicain (ER) newspaper,[9] which bears title and text boundaries but no other annotations. The first 100 articles from 2003 with a length of more than 300 words were selected for our experiments. These articles are different from those included in the annodis.ER subcorpus of Sequoia.

Table 1 shows an overview of the corpora.

4.2 *Tools and pipeline*

Coreference resolution is tackled as a two-step task, consisting first in detecting entity mentions, using DeCOFre (Grobol 2019), an end-to-end coreference resolution system, and the only such system de-

[9] `https://hdl.handle.net/11403/est_republicain/v2`

Table 1: Corpora overview

Corpus	Sub-corpora	Number of sentences	Average number of words per text	Total number of words
ANCOR	ESLO_ANCOR, ESLO_CO2, OTG, UBS	32,427	988	449,722
Sequoia	emea, frwiki, annodis.ER	2,538	786	44,818
Est Républicain	first 100 articles of more than 300 words in 2003	2,923	501	50,102
Total		37,888	890	544,642

signed to process full-length documents.[10] In DeCOFre, mention detection is a classification task over text spans, using a deep neural network to extract vector representations of these spans and classify them as mentions (referential pronouns and noun phrases) or non-mentions (both non-constituents and constituents that are not referential). Coreference resolution proper is performed as a classification task over mention pairs by OFCORS,[11] a custom oral French coreference resolution system trained on ANCOR.[12] Its experimentally chosen setting includes: (i) tokenization with splitting of contractions (e.g. *du* → *de le* 'of.the → of the') performed by Stanza (Qi *et al.* 2020), (ii) morpho-syntactic annotation with spaCy (Honnibal and Montani 2017), (iii) restricting candidate pairs to a window of size 8, (iv) pairwise classification, (v) favoring the closest possible antecedent. The DeCOFre/OFCORS suite outputs coreference chains in a JSON file. On

[10] The other existing tool for coreference resolution in French, coFR (Wilkens *et al.* 2020), is trained on both spoken and written data but is limited to a few dozen sentences per document.

[11] `https://gitlab.com/Stanoy/ofcors/`

[12] Training on DEMOCRAT (Landragin 2021) – the only existing coreference corpus of written French – on full-length documents is prone to generate poor models (Grobol 2021).

an extract of the ANCOR corpus, OFCORS showed an overall CoNLL score of 78.2, which is close to the state of the art in French coreference resolution. However, performance varies greatly among coreference types: pronominal, direct, and indirect coreference are solved with F1-measures of 70.9, 67.5, and 28.8, respectively. Human validation of the coreference chains is thus necessary for a reliable corpus study.

The automatic identification of VMWEs is also performed in two steps. First, raw text is tokenized and annotated for lemmas, parts-of-speech, morphology, and syntax with UDPipe.[13] Then, VMWEs are marked with the Seen2Seen system (Pasquer *et al.* 2020), which focuses on accurately identifying variants of VMWEs seen in the training corpus. It is a rule-based system relying on a simple but efficient "extract then filter" approach. In the extraction phase, all VMWEs annotated in the training corpus are extracted and represented as multisets of lemmas, e.g. the VMWE in (fr) ***tu te comporte*** *mal* (lit. 'you yourself contain badly') 'you behave badly' is represented as {comporter, se} '{contain, oneself}'. Then, all co-occurrences of the same multisets of lemmas are identified as VMWE candidates in the test corpus. The filtering phase retains only those candidates which respect certain morpho-syntactic constraints (e.g. all components of the identified candidate must be syntactically connected). A total of 8 filters is defined, each of which can be activated or not. The best combination of active filters is determined in the training phase. Seen2Seen was trained for 14 languages of the PARSEME Shared Task on automatic identification of VMWEs (Ramisch *et al.* 2020). With its very simple architecture and fully interpretable rules, it obtained the second best global score, outperforming several systems based on statistical and deep-learning techniques. For French, the best model has 4 activated filters and obtains the F-score of 0.9 on seen VMWEs, and 0.79 on both seen and unseen ones. Seen2Seen outputs VMWE annotations in the `.cupt` format, native to the PARSEME corpora and shared task.

We applied the DeCOFre/OFCORSE pipeline to the Sequoia corpus, so as to complete the manual annotation of VMWEs with automatic coreference annotation. Conversely, the manual coreference annotations in ANCOR were complemented by automatic VMWE annotations obtained with UDPipe/Seen2Seen. Finally, all 4 tools

[13] `https://ufal.mff.cuni.cz/udpipe/2`

ID	Form	Gloss	…	VMWE	Mention	Chain
2	entama	'started'	…	*	*	*
3	un	'the'	…	*	219	60
4	combat	'fight'	…	*	219	60
…						
11	combat	'fight'	…	1:LVC.full	224	60
12	contre	'against'	…	*	*	*
13	les	'the'	…	*	225	
14	institutions	'institutions'	…	*	225	*
15	,	,	…	*	*	*
16	mené	'carried.on'	…	1	*	*

Figure 1: Merged annotations for VMWEs, mentions and coreference chains. Extract from the Sequoia frwiki corpus

were applied to the Est Républicain corpus. Some tokenization inconsistencies were solved by custom scripts and the joint annotations were converted into an extension of the `.cupt` format, whose simplified extract is given in Figure 1. It is a tabular format with one token per line.[14] The last three columns contain: (i) the VMWE annotation or a '*' if the current token is not part of any VMWE (here, tokens 11 and 16 are components of the first VMWE in the sentence; token 11 additionally carries the VMWE type, i.e. LVC.full), (ii) the identifier of a mention or '∗' if the token does not belong to any mention (here, tokens 3–4 belong to mention 219, token 11 to mention 224 and tokens 13–14 to mention 225), (iii) the identifier of the coreference chain (here, mention 219 with tokens 3–4 and mention 224 with token 11 belong to chain 60).

The last stage of the processing pipeline is an automatic identification of token spans in which a VMWE overlaps with a non-singleton mention. There are 4 possible cases:

1. A VMWE is included in a mention, as in:

(19) ce patient **atteint** d'une **maladie** grave
lit. 'this patient reached by a serious disease'
'this seriously ill patient'
(Sequoia emea)

[14] Columns 1 and 2 contain the token rank in the sentence and the token itself. Column 3 is not part of the format and serves as a gloss of this example only. Columns 4–10 are omitted for brevity.

2. A VMWE covers the same tokens as a mention, as in:

(20) **mise en évidence**
lit. 'putting into evidence' | 'highlighting'
(Sequoia frwiki)

3. A mention is included in a VMWE, as in:

(21) **trouver la mort**
lit. 'find the death' | 'die'
(Sequoia frwiki)

4. A mention and a VMWE overlap partly, as in:

(22) **pris en flagrant délit** de vol
lit. 'taken in flagrant offense of theft'
'caught red-handed while stealing'
(Sequoia frwiki)

All these cases (provided that the mention is not a singleton) were automatically extracted from the files containing aligned coreference and VMWE annotations, as in Figure 1. The resulting 1311 intersections, henceforth simply called *overlaps*, were then validated manually, as explained in the following section.

4.3 *Human validation*

The automatic extraction of overlaps, as described in the previous section, helps us avoid manual analysis of the whole corpus by automatically extracting fragments relevant to hypothesis $\mathscr{H}$ instead. However, due to the limited reliability of the tools (cf. Section 5.1), this automatic procedure calls for manual validation. Thus, for each overlap, we manually checked that:

- The predicted VMWE is correct according to the PARSEME annotation guidelines.
- The span of the predicted mention is correct, and if not, after correcting it, one of cases 1–4 still applies.
- The predicted non-trivial coreference chain is at least partly correct, i.e. it contains at least two correct co-referring mentions, including the one that overlaps with the VMWE.

Any extracted occurrences not respecting these conditions were discarded as *false*, and annotated for the source of the error (*wrong mention, wrong chain, wrong MWE, wrong MWE type,* or *literal MWE occurrence*). The remaining occurrences were marked with one of the 4 labels:

- *true*, if the example is relevant to hypothesis $\mathscr{H}$, i.e. if a proper subset of lexicalized components of a VMWE truly occurred in a non-trivial coreference chain; this implies case 3 or 4 (from the previous section) of a VMWE-mention overlap, as in example (23):

(23) […] l'**ordonnance** de renvoi devant le tribunal […] a été **signée** par le juge […]. Dans son ordonnance, […]
'the **order** of referral to court was **signed** by the judge […]. In his order […]'

(Sequoia frwiki)

- *repeated*, if the example is relevant but coreference occurred "incidentally", as an effect of disfluence in speech (see also Section 6.3), rather than the intended use of a text cohesion device, as in (24):

(24) ça **fait partie** du patrimoine ça aussi je ça **fait partie** du patrimoine oui je trouve
lit. 'this **makes part** of the heritage this also I this **makes part** of the heritage yes I think'
'this belongs to the heritage this also I this belongs to the heritage yes I think'

(ELSO_ANCOR)

- *irrelevant*, when the mention contains the whole VMWE rather than a proper subset of its components (case 1 or 2 from the previous section), which is not relevant to hypothesis $\mathscr{H}$, as in example (25):

(25) De nombreux patients **atteints** d'**ostéoporose** n'ont aucun symptôme, mais ils présentent néanmoins un risque de fracture osseuse

lit. 'many patients reached by osteoporosis do not have any symptoms but they present however a risk of bone fracture'
'many patients with osteoporosis have no symptoms but they still present a risk of bone fracture'

(Sequoia emea)

- *unclear*, if it is hard to decide about the relevance of the example, as in (38), discussed in more detail in Section 6.

As all the extracted samples were manually validated during meetings, so as to achieve a "platinum" standard (discussed and agreed on by all the project members), the validators were not independent. There were between 2 and 6 validators for each example, all with NLP expertise, 3 with linguistic expertise, and 4 native speakers of French. Each example was reviewed by at least one linguist and one native speaker.

In sum, the experimental setting includes three corpora; the first two are manually annotated for one phenomenon in our scope, and the third one is a raw corpus. We pre-processed these corpora using a parser combined with a VMWE identifier on the one hand, and a mention detector combined with a coreference solver on the other. As a result, we obtained partly manual and partly automatic annotations of VMWEs, mentions and coreference chains. We then filtered them so as to retain only the cases in which a VMWE overlaps, at least partly, with a non-singleton mention. These overlaps were then manually annotated with 4 labels describing their relevance to hypothesis $\mathscr{H}$.

5 RESULTS

This section presents quantitative and qualitative results of the corpus study presented in the previous section. There, human validation was performed for 1311 overlaps. Henceforth, we omit two VMWE categories (cf. Section 3) – MVCs and IRVs – since they are beyond the scope of our study. The MVCs are exclusively made up of verbal components, but DeCOFre/OFCORS does not handle verbal coreference.

The IRVs contain verbs with reflexive pronouns, but the latter are not considered mentions in the ANCOR coreference annotation scheme. Omitting MVCs and IRVs reduces the number of manually annotated overlaps to 1307.

Quality of the automatic annotation 5.1

None of the corpora at our disposal is manually annotated for the two phenomena we are interested in (cf. Section 4.1). When automatic annotation is performed for any of them, it is important to estimate the influence of its quality on the outcome of the study. While we know the overall in-domain performances of UDPipe/Seen2Seen and DeCOFre/OFCORS (cf. Section 4.2), we use these tools in a partly out-of-domain setting. However, one of the outcomes of our manual validation (Section 4.3) indicates the source of the errors in the overlaps tagged *false*. Based on these labels, we can estimate the precision of our tools.

The precision of automatic identification of VMWEs by UDPipe/Seen2Seen can be estimated by considering that true positives are all the automatically tagged VMWEs that occur in the 1307 overlaps, except those which have the error source manually tagged as *wrong MWE* or *literal MWE occurrence*.[15]

Table 2 shows the number of overlaps per corpus and VMWE category, and the corresponding precision for the VMWE identification task. The results vary greatly among genres and VMWE categories. In Sequoia, the precision of manual annotation of VMWEs is considered perfect. In ER, whose genre is close to the UDPipe/Seen2Seen training corpus, precision is very high for LVC.full (98%) and reasonable for VID (63%). In ANCOR, which contains spoken language, precision drastically drops to 10% for VIDs and 65% for LVC.full.[16] For LVC.cause, which is overall a relatively infrequent category, the figures are not representative.

[15] The *wrong MWE type* label signals an error of VMWE categorization rather than identification.

[16] This is notably due to missing punctuation in ANCOR, which results in long speech turns, each of which is considered by Seen2Seen as one sentence.

Table 2: Precision of VMWE identification on the manually validated overlaps (OLs)

VMWE category	Sequoia		ER		ANCOR		All corpora	
	Overl.	P_{VMWE}	Overl.	P_{VMWE}	Overl.	P_{VMWE}	Overl.	P_{VMWE}
VID	34	1.00	49	0.63	578	0.10	661	0.18
LVC.full	141	1.00	45	0.98	456	0.65	642	0.75
LVC.cause	2	1.00	1	0.00	1	1.00	4	0.75
All	177	1.00	95	0.79	1035	0.34	1307	0.46

Table 3: Estimation of recall of VMWE identification; (*) signals a non-representative score

VMWE category	Recall		
	Sequoia	ER	ANCOR
VID	1.00	0.78	0.66
LVC.full	1.00	0.60	0.36
LVC.cause	1.00	0.23	0.00 (*)

The manually tagged error sources (Section 4.3) also give some indications about the quality of coreference resolution. In the 1307 overlaps, we find 5 occurrences of the *wrong mention* label, which would amount to an excellent precision of 99.6%. This estimation is, however, much less accurate than for the VMWEs above. Not only is it limited to mentions occurring in overlaps, but a mention is not tagged as wrong if it can be corrected so that an overlap still occurs. Under these circumstances, the *wrong mention* label is very unlikely. As for the quality of the chains, we find 255 occurrences of the *wrong chain* label in the 1307 overlaps. However, it is not assigned to partly correct chains, nor does it signal which mentions are spuriously assigned to a chain. For these reasons, we do not try to transform the *wrong mention* and *wrong chain* counts into standard quality measures for coreference resolution.

The manually tagged error sources (Section 4.3) cannot help estimate the recall of our tools, but we can perform this estimation based on various other factors. Table 3 shows recall estimation for VMWE identification. It is considered perfect in Sequoia, since these annotations are manual. For ER, which has partly the same genre as Sequoia, we can adopt the Seen2Seen recall from the PARSEME shared task (Ramisch *et al.* 2020).[17] For ANCOR, the estimation is harder: since this is an out-of-domain use of Seen2Seen, we have no manual VMWE

[17] `https://multiword.sourceforge.net/sharedtaskresults2020`

annotations in spoken corpora; adding them to all documents would be prohibitively costly for this study. Therefore, to perform this estimation, we selected speech turns from two subcorpora: OTG, 280 turns, 2779 tokens; CO2, 527 turns, 10372 tokens. We manually corrected the errors produced by Seen2Seen in these files. The results show that 150 out of the 259 gold VMWE annotations were correctly predicted by Seen2Seen (114 out of 174 VIDs, 21 out of 58 LVC.fulls, 0 out of 1 LVC.cause, and 15 out of 26 IRVs, neglected here). This gives an overall recall of 0.58 (with a per-category split as detailed in Table 3). Among the 109 missed VMWEs, there are 5 *true* overlaps in LVCs (14%) and none in VIDs.

Recall in coreference resolution is equally hard to estimate, but we conducted an experiment on a sample of the Sequoia corpus, whose genre is the most distant from the training corpus of DeCOFre/OFCORSE. Namely, we selected one VID and one LVC.full expression in which true overlaps are the most frequent in Sequoia: ***porter le nom*** *de* 'to bear the name of' and ***avoir*** *une* ***fracture*** 'to have a fracture'. We then searched manually for all occurrences of these MWEs in Sequoia and checked whether or not they were concerned by true overlaps. We observed that our semi-automatic annotation procedure: (i) had not missed any occurrences or coreference relations concerning the first expression, (ii) had missed 7 out of 10 occurrences of the second expression but none of them was involved in a coreference chain. Although partial, this sample survey suggests that our results should not be significantly biased by silence in terms of coreference resolution.

Corroboration of the hypothesis 5.2

Let us now examine Table 4, which summarizes the general outcomes of the processing chain described in Section 4. In total, 7010 VMWEs (excluding IRVs and MVCs) were (manually or automatically) annotated in the corpora from Table 1.[18] Out of these 7010 occurrences, 1307 were automatically extracted as possibly overlapping, with mentions occurring in non-trivial coreference chains. As a result of the

[18] 8047, if IRVs and MVCs are also considered.

Table 4: Results of the automatic intersection and manual validation

Type	VMWEs	Overlaps	True	%	Repeated	Irrelevant	Unclear
VID	5266	661	29	0.6	23	0	6
LVC.full	1726	642	245	14.2	84	9	2
LVC.cause	18	4	1	5.6	0	0	0
Total	7010	1307	275	3.9	107	9	8

manual validation of the 1307 cases, 908 were qualified as *false*, 275 as *true*, 107 as *repeated*, 9 as *irrelevant*, and 8 as *unclear* (cf. Section 4.3).

The 275 true cases correspond to 3.9% of the initially annotated VMWEs. This roughly corroborates hypothesis $\mathscr{H}$: In 3.9% of VMWEs, proper subsets of lexicalized components occur in non-trivial coreference chains. Several caveats must, however, be mentioned.

First, the frequency of true cases strongly depends on the VMWE category. LVC.full is in sharp contrast with all other categories since 14.2% of its initially annotated instances were validated as true.[19] For LVC.cause, the percentage is lower (5.6%), with only one occurrence validated as true. For VID, the number of examined occurrences is the highest, and only 0.6% of them are tagged true.

Next, the genre of the corpus has to be taken into account. Table 5 shows the breakdown of the two most salient VMWE categories (as per Table 4), VID and LVC.full, within the three source corpora. In Sequoia, where the initial VMWE annotation is manual, only 0.5% of VIDs and 6.5% of LVC.full are validated as true. For ER, where the UDPipe/Seen2Seen precision is reasonable or very good (Table 2), these numbers are even lower (0.0% and 2.5%). In ANCOR, VIDs validated as true still remain below 1%, but for LVC.full this rate reaches 17.4%. This high number is significant, especially given the fact that UDPipe/Seen2Seen results are noisy in ANCOR. It is, however, partly mitigated by the ambiguity and frequency of *ça* 'this', a demonstrative pronoun, as explained in Sections 6.2–6.3. Finally, the quality of automatic annotations has strong but difficult to estimate influence on the results. Let us suppose that the precision and recall estimates in

[19] This count includes 10 VMWEs (tagged as *wrong MWE type*) annotated automatically as VID but whose actual category is LVC.full.

Table 5: Results (corrected for estimated precision and recall) per corpus for the 2 salient VMWE categories: VID and LVC.full

Corpus	VID						LVC.full					
	Annotated		True	Percentage			Annotated		True		Percentage	
Sequoia	204	(204)	1	0.5	(0.5)		340	(340)	22	(22)	6.5	(6.5)
ER	302	(244)	0	0.0	(0.0)		122	(198)	3	(3)	2.5	(1.7)
ANCOR	4760	(721)	28	0.6	(3.9)		1264	(2282)	220	(280)	17.4	(12.3)
All	5266	(1169)	29	0.6	(2.5)		1726	(2821)	245	(305)	14.2	(10.8)

Tables 2 and 3 are representative of VMWE identification in general, i.e. they apply not only to the VMWEs occurring in overlaps but to all VMWEs. Under this (strong) assumption, the annotated VMWEs in Table 5 should be modified as indicated in the parenthesized scores.

True overlaps 5.3

Beyond the sheer numerical results of our corpus study, it is interesting to look at actual examples in which proper subsets of lexicalized components of VMWEs do occur in non-trivial coreference chains. Table 6 lists the VMWEs of types LVC.full and VID whose frequency in true overlaps is the highest.[20] The complete lists of the VMWEs from true overlaps are given in the Appendix.

Sample coreference chains with the two most frequent LVC.full expressions are shown in examples (26) and (27). In the former, the coreference is direct, i.e. all three mentions share the same head, but the head varies in number. In the latter, the coreference is pronominal.

(26) une journée de travail euh ça commence le matin à sept heures […] il y a des coups de téléphone il y a des **études** à **faire** […] vous partez sur des plans vous **faites** une **étude** ce qu'on appelle une étude commerciale

[20] The literal translation is omitted when it is identical to the true meaning.

Table 6: LVCs and VIDs with most frequent true overlaps

LVC.full	True overlaps	VID	True overlaps
faire** des/une **étude(s) (lit. 'do studies/a study') 'study/perform a survey'	50	***avoir le temps*** 'have the time'	16
poser** une **question (lit. 'pose a question') 'ask a question'	25	***poser problème*** 'pose problem'	4
faire grêve (lit. 'do strike') 'go on strike'	19	***prendre le temps*** 'take the time'	2
prendre** des **sanctions (lit. 'take sanctions') 'impose sanctions'	13	***prendre sa place*** 'take one's place'	2
avoir** des **difficultés 'have difficulties'	12	***faire plaisir*** 'make pleasure'	1

'a working day erm it starts at seven a.m. […] there are phone calls to make there are **surveys** to **conduct** […] you start from plans you **conduct** a **survey** what we call a commercial survey' (ELSO_ANCOR)

(27) je vais vous **poser** une **question** […] je vous en prie si je peux y répondre
'I will **ask** you a **question** […] please if I can answer it'
(ELSO_ANCOR)

We found few occurrences of indirect coreference in true overlaps – one example is shown in (28) – and in particular none involving a VID. This cannot be due only to indirect coreference being hard to resolve automatically, since it is also the case in ANCOR, where coreference chains are manually annotated.

(28) j'**ai** une **activité** assez assez intense […] est-ce que vous pourriez parler un peu de votre travail ? […] je fais ce métier-là parce qu'il me plaît
'I **have** a quite quite intense **activity** […] could you talk a bit about your work? […] I do this job because I like it'
(ELSO_ANCOR)

When VIDs involved in true overlaps are considered, we notice that, even if they do pass the PARSEME VID tests, they often resemble LVCs in that their lexicalized nouns bear their literal sense, and they are abstract and/or predicative (*temps* 'time', *problème* 'problem', *place* 'place', *plaisir* 'pleasure'). Sample true overlaps involving VIDs are shown in examples (29)–(32).

(29) est-ce que vous **avez le temps** de faire des mots-croisés ? le temps ou la condition ?
'do you **have time** to do crosswords? time or conditions?'
(ELSO_CO2)

(30) la femme a **une place** à **prendre** [...] on n'est pas du tout préparé à **prendre notre place**
'a woman has a place to take [...] we are not at all prepared to **take our place**'
(ELSO_ANCOR)

(31) il lui faut du temps pour comprendre [...] on verra on **a le temps**
'he will need some time to understand [...] we'll see we **have the time**'
(ELSO_ANCOR)

(32) la télévision ça me **fait** bien **plaisir** [...] après la guerre [...] j'ai pris du plaisir
'TV **gives** me much **pleasure** [...] after the war [...] I took pleasure'
(ELSO_ANCOR)

In some cases, the coreference may be seen as somewhat coincidental. For instance, while in (29) the two mentions of *le temps* clearly refer to the same time (needed to do crosswords), in (31) *le temps* 'the time' is more generic and abstract and it could be argued that coreference is barely present. Example (32) is even more questionable. There, the second mention of *plaisir* refers to a pleasure occurring chronologically before that in the first mention. It is hard to decide whether these two pleasures have different referents, or whether pleasure in general is concerned. Thus, this example clearly belongs to the gray zone of coreference resolution.

In sum, in this section, we first estimated the quality of our tools based on several factors: (i) manual annotations of error sources found in overlaps, (ii) previous results of the VMWE identifier in an in-domain setting, (iii) manual correction of out-of-domain VMWE annotation in a corpus extract. The manually validated overlaps, both in the raw counts and in the counts corrected for precision and recall, seem to corroborate hypothesis $\mathscr{H}$, but these counts vary greatly among VMWE categories and text genres. The study of true overlaps reveals that they often involve direct or pronominal coreference in LVCs, but abstract or general concepts (such as time or pleasure) in VIDs.

6 DISCUSSION AND PERSPECTIVES

Given the quantitative and qualitative outcomes of our study presented in the previous section, we can follow several directions towards more fine-grained observations and conclusions.

6.1 *Semantic properties of true overlaps*

The true overlaps illustrated in Section 5.3 might be considered in terms of the semantic properties of MWEs addressed in the state of the art (Section 2).

First, almost all the examples from Tables 6 to 11 contain nouns used literally rather than metaphorically. Thus, their contribution to the semantics of the whole expression is considerable, which implies a high degree of semantic compositionality.

Next, the question of decomposability is somewhat trivial. There is no need to assign non-standard meanings to the nouns, while the verbs are semantically bleached, i.e. they are assigned a non-standard meaning that is simply (close to) void.

Finally, figuration and transparency have relatively little relevance here, since it is difficult to define literal readings of these expressions that are different from their idiomatic readings. The reason is, again, because the nouns already appear here in their literal meanings, i.e. with no figuration. Exceptions (that remain questionable) include: *photo* in ***prendre une photo*** 'take a photo', *place* in ***prendre sa place*** 'take one's place', and *impression* in ***donner l'impression*** 'give the impression'. Those might indeed respectively be understood as literally grasping a printed photograph, taking possession of one's seat, or handing a printout to someone. With such interpretations, both the literal image and its motivation for the MWE are easy to capture i.e. the expressions are figurative and transparent.

In the light of these observations, we can argue that the possibility for MWE components to occur in non-trivial coreference chains correlates with the semantic properties of these MWEs in the same spirit as their lexical and morpho-syntactic flexibility, discussed in previous works (Section 2). When an MWE is strongly semantically non-compositional, non-decomposable, non-figurative, and/or non-transparent, its components do not corefer with other mentions – or at least we found no examples of such cases in our corpus study.

Note, however, that the analyses offered in this section are informal. We did not follow a rigorous experimental design that would have allowed us to measure the degree of compositionality, decomposability, figuration, and transparency in the true overlaps. We leave such quantification for future work.

Pronominal coreference with LVCs 6.2

A considerable number of LVC.fulls have true overlaps with coreference chains containing pronouns, as in example (33).

(33) je m'excuse de vous **poser** toutes ces **questions** ça ça a l'air très indiscret
'I apologize for **asking** you all these **questions** that that sounds very indiscreet'
(ELSO_ANCOR)

One might argue that the pronoun *ça* 'this' corefers not only with the questions but with the act of asking them, which would imply event coreference rather than nominal coreference (cf. Section 3). Note that this ambiguity is inherent to LVC.fulls, defined in the PARSEME guidelines as verb-(preposition)-noun combinations in which the noun is predicative, i.e. expresses an event or a state, while the verb is semantically light. One of the tests for LVC.full in the guidelines is checking for verb reduction, i.e. checking if an NP without the verb refers to the same event/state. Here, *toutes ces questions* 'all these questions' refers to the same event as *je vous pose toutes ces questions* 'I ask you all these questions'. Obviously, then, the pronoun *ça* 'that', which refers to the same event, corefers both with the whole expression and the nominal group itself.

6.3 *Coreference in spontaneous conversational speech*

Example (33) above is representative of spontaneous speech. In as many as 25% of the true overlaps in the ANCOR corpus, the coreference chains contain the *ça* 'that' mention. This partly mitigates the relatively high rate of LVCs with true overlaps in ANCOR in Table 5.

In Table 4, a considerable number of overlaps is classified as *repeated*. They result from peculiar features of speech such as frequent rewording and disfluencies. In example (34), the second and third occurrences of the mention *importance* are due to the reuse of the whole VMWE ***avoir** de l'**importance*** 'have importance' by the second speaker, and to a verification of the answer by the first speaker.

(34) - vous regrettez que la langue française se dégrade ou bien que ça **a** pas beaucoup d'**importance** ?
'Do you regret that the French language is deteriorating or does that not **have** much **importance**?'
- oh si moi je trouve que ça **a** de l'**importance** ah oui
'oh, yes me I find that that **has** some **importance**, oh yes'
- importance oui ?
'importance yes?'
(ELSO_ANCOR)

In example (35) the speaker rephrases the sentence in order to find the most appropriate formulation. More precisely, the nominal group is reused in a different context.

(35) j'ai toujours du temps je **prends** toujours **le temps**
'I always have the time I always **take the time**'
(ELSO_ANCOR)

Whether such examples should be considered as true cases of coreference is questionable. We believe that the answer depends on the distance between the two mentions and their contextual similarity. These issues should be addressed in more in-depth studies in the future.

Expletive clitics as mentions 6.4

Expletive clitics are pronouns that are syntactically compulsory but cannot be mapped on the semantic arguments of their verbs. In VMWEs, expletives occur systematically in IRVs and occasionally in VIDs. Section 5 mentioned that IRVs are omitted from our results since they are not covered by the ANCOR annotation scheme. The only IRV occurrence tagged true in the validation procedure from Section 4.3 has a reflexive pronoun spuriously annotated as a mention, example (36). The IRV as a whole means 'to go', so the reflexive clitic is truly expletive. However, a coreference chain with two homographic pronouns *vous* 'you', one personal and one reflexive, arguably does occur here, notably due to the compulsory agreement between the reflexive and the agent of the verb. This example shows that it might be interesting to reconsider the ANCOR principle that reflexive pronouns should not be annotated as mentions.

(36) Lorsque vous êtes à l'hôpital […] **dirigez** **vous** immédiatement […]
lit. 'When you are in the hospital […] **direct** **yourself** immediately […]'
'when you are in the hospital […] go directly […]'
(Sequoia emea)

Example (37) shows a VID with a clitic-verb construction (typical for Romance languages) in which the clitic is semantically void. Other examples include ***en valoir la peine*** 'to be worth it', ***en venir*** 'end with', ***en vouloir*** 'blame', etc. Here, the coreference annotator judged the clitic still sufficiently transparent to corefer with a referent introduced by a nominal group.

(37) j'**en** **reviens** toujours à cette question
lit. 'I **of-it** **return** always to this question'
'I always go back to this question'
(ELSO_CO2)

Considering these VMWE examples jointly with coreference allows us to put forward the hypothesis that expletiveness, like semantic compositionality, might be a matter of scale rather than a binary feature.

6.5 A mention as referent

Example (38) raises interesting questions concerning the nature of coreference.

(38) [l'initiateur d'un[système de défense qui **porte** [son **nom**]$_3$]$_2$]$_1$ […] [le prix [André-Maginot]$_5$]$_4$ […]
'initiator of the defense system that **bears** his **name** […] the André-Maginot award'
(Est Républicain)

Arguably, this example contains the 5 mentions (marked here with indexed brackets for readability, rather than underlined). A harder question is how many distinct referents we have in the picture. At least 3 are easy to identify: the statesman André Maginot (referent $r1$), the defense system initiated by him ($r2$), and the award ($r3$). The names of these 3 referents happen to be closely related: *André Maginot*, *ligne Maginot* 'Maginot line' and *prix André-Maginot* 'André-Maginot award'. But the VID ***porte** son **nom*** 'bears his name' contains a mention which introduces a new referent ($r4$): $r1$'s name. Now the questions is: do mentions 3 and 5 corefer? Mention 3 clearly refers to $r4$. But mention 5 could be seen as referring either to $r1$ or to $r4$.

The difficulty with this interpretation lies in the fact that *André Maginot* acts both as a mention (a naming expression) referring to $r1$ and as a referent to which mention 3 refers. This shows the fuzziness of the border between the referents (items of the discourse world) and mentions (items of the language). As a result, we annotated this example as *unclear*.

6.6 Coreference in non-verbal MWEs

Due to the limitations of our corpora and tools, we could consider hypothesis $\mathscr{H}$ with respect to verbal MWEs only. A future study should also cover non-verbal MWEs, including adverbial, prepositional, and conjunctive MWEs containing nouns and pronouns, such as ***en plein air*** (lit. 'at full air') 'outdoors', or ***dans le cadre** de* (lit. 'in the frame of') 'in the framework of'. We might expect sporadic cases of coreference, notably due to the generality or abstractness of concepts referred to by component nouns, as in the fabricated example (39).

(39) le cours a eu lieu **en plein air** [...] L' air était frais
the lesson has had place in full air [...] The air was fresh
[...] C' était bien de le respirer (fr)
[...] It was good to it breathe
'The lesson took place outdoors [...] The air was fresh [...] It was good to breathe it'

In this section, we offered a review of interesting phenomena encountered in the true overlaps between VMWE components and mentions. They provide new evidence that the properties of linguistic objects (here: reference, coreference, and expletiveness) are often a matter of scale rather than binary features. NLP-based methodology like ours, which assumes the existence of clear-cut categories and features, does not offer a perfect modeling for such phenomena. Therefore, its numerical results must be interpreted with care.

CONCLUSIONS 7

In this paper, we explore the crossroads between two linguistic phenomena: multiword expressions and coreference – an area which has rarely been investigated, especially with quantitative methods. Our initial hypothesis is that, due to the semantic non-compositionality of MWEs, their internal components should not be easily accessible to coreference. In other words – as expressed in the title of this paper – coreference is likely to ***shut its eyes to*** 'ignore' MWE components.

Our experimental setup was designed to quantify how far this hypothesis holds. Due to the restricted availability of corpora and tools, we limited our scope to nominal coreference and to verbal MWEs in French only, reducing the relevant MWE types mainly to verbal idioms and light-verb constructions (with the LVC.full type being dominant, and LVC.cause negligible). We set up a processing pipeline in which the available manually annotated corpora were combined with outcomes of fully-automatic tools for coreference resolution and VMWE identification. Overlaps between VMWEs and coreference chains were automatically extracted and manually validated. This allowed us to calculate true overlap frequencies, which we then corrected for precision and recall, based on estimating the quality of the automatic tools and on manual correction of an extract of the corpus.

As an outcome of this methodology, we found that the frequency of non-trivial coreference chains containing proper subsets of lexicalized components of MWEs depends on both MWE type and text genre. For VIDs in newspaper and Wikipedia texts, true overlaps occur very rarely, i.e. in no more than 0.5% of all VID occurrences, whether in raw or precision-corrected counts. In speech, this percentage is similar in raw counts but higher (close to 3.9%) in corrected counts. The picture is different for LVCs. In newspaper and Wikipedia texts, the frequency of true overlaps can reach 6.5%, in both raw and corrected counts, but in speech it can be as high as 17.4% for raw and 12.3% for corrected counts. This shows that the original hypothesis holds mostly for VIDs and partly for LVCs. This is not surprising since LVCs lie in the gray zone between idiomatic and productive constructions. Moreover, the hypothesis is corroborated more clearly by newspaper and Wikipedia texts than by speech.

By examining concrete examples of LVCs and VIDs for which true overlaps do occur in the corpus, we notice that they tend to contain nominal objects that are abstract and predicative (express events or states), and that occur in the VMWEs in their literal rather than figurative sense. This suggests that the probability of true overlaps is positively correlated with the degree of semantic compositionality of VMWEs. This is consistent with previous studies showing correlations between the morpho-syntactic variability of MWEs and their semantic properties such as compositionality, decomposability, transparency, and figuration. Future work might exploit methods for quantifying the semantic compositionality of MWEs (Cordeiro *et al.* 2019), so as to assess its correlation with the MWE/coreference overlap.

Our corpus study also brings a better understanding of the nature of coreference. First, we found that true overlaps between MWEs and non-trivial coreference chains occur mostly with direct and pronominal coreference but rarely with indirect coreference. This might again be related to semantic (non-)compositionality, since indirect coreference requires the reformulation of a component, which is easier if this component retains its literal reading. Next, the peculiarities of speech often result in somewhat coincidental cases of coreference due to disfluencies (repetition, verification, reuse) rather than to intentional use of coreference as a text cohesion device. The percentage of such cases is significant compared to the true overlaps. We also gained new understanding of expletive clitics, which should in principle be non-referential but do occasionally occur in coreference chains. Finally, our study brings to light some intricacies of reference in natural language, such as the fuzzy border between the status of mention and that of referent.

Future work will seek to extend the scope of this study to non-verbal types of MWEs and to other, notably typologically distant, languages.

REFERENCES

Marie Candito, Mathieu Constant, Carlos Ramisch, Agata Savary, Yannick Parmentier, Caroline Pasquer, and Jean-Yves Antoine (2017), Annotation d'expressions polylexicales verbales en français, in *24e conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, pp. 1–9, Orléans, France.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie (2014), Deep syntax annotation of the Sequoia French treebank, in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 2298–2305, European Language Resources Association (ELRA), Reykjavik, Iceland, http://www.lrec-conf.org/proceedings/lrec2014/pdf/494_Paper.pdf.

Silvio Cordeiro, Aline Villavicencio, Marco Idiart, and Carlos Ramisch (2019), Unsupervised compositionality prediction of nominal compounds, *Computational Linguistics*, 45(1):1–57, doi:10.1162/coli_a_00341.

Raymond W. Gibbs and Nandini P. Nayak (1989), Psycholinguistic studies on the syntactic behavior of idioms, *Cognitive Psychology*, 21:100–138.

Loïc Grobol (2019), Neural coreference resolution with limited lexical context and explicit mention detection for oral French, in *Proceedings of the Second Workshop on Computational Models of Reference, Anaphora and Coreference*, pp. 8–14, Minneapolis, Minnesota, USA, https://www.aclweb.org/anthology/papers/W/W19/W19-2802/.

Loïc Grobol (2021), Exploitation du corpus democrat par apprentissage artificiel, *Langages*, 224(4):129–145, doi:10.3917/lang.224.0129, `https://www.cairn.info/revue-langages-2021-4-page-129.htm`.

Gaston Gross (1988), Degré de figement des noms composés, *Langages*, 90:57–71.

Maurice Gross (1994), The lexicon-grammar of a language: Application to French, in Ashley R. E., editor, *The Encyclopedia of Language and Linguistics*, pp. 2195–2205, Oxford/NewYork/Seoul/Tokyo: Pergamon, Oxford.

Matthew Honnibal and Ines Montani (2017), spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, to appear.

Frédéric Landragin (2021), Le corpus DEMOCRAT et son exploitation. Présentation, *Langages*, 224(4):11–24, doi:10.3917/lang.224.0011, `https://www.cairn.info/revue-langages-2021-4-page-11.htm`.

Éric Laporte (2018), Choosing features for classifying multiword expressions, in Manfred Sailer and Stella Markantonatou, editors, *Multiword expressions: Insights from a multi-lingual perspective*, pp. 143–186, Language Science Press, Berlin.

Rosamund Moon (1998), *Fixed expressions and idiomas in English*, Oxford University Press, Cambridge.

Judith Muzerelle, Anaïs Lefeuvre, Emmanuel Schang, Jean-Yves Antoine, Aurore Pelletier, Denis Maurel, Iris Eshkol, and Jeanne Villaneau (2014), ANCOR_Centre, a large free spoken French coreference corpus: description of the resource and reliability measures, in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 843–847, European Language Resources Association (ELRA), Reykjavik, Iceland, `http://www.lrec-conf.org/proceedings/lrec2014/pdf/150_Paper.pdf`.

Geoffrey Nunberg (1978), *The pragmatics of reference*, Ph.D. thesis, City University of New York.

Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow (1994), Idioms, *Language*, 70(3):491–538.

Caroline Pasquer, Agata Savary, Carlos Ramisch, and Jean-Yves Antoine (2020), Verbal multiword expression identification: Do we need a sledgehammer to crack a nut?, in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3333–3345, International Committee on Computational Linguistics, Barcelona, Spain (Online), doi:10.18653/v1/2020.coling-main.296, `https://aclanthology.org/2020.coling-main.296`.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning (2020), Stanza: A Python natural language processing toolkit for many human languages, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 101–108.

Carlos Ramisch, Agata Savary, Bruno Guillaume, Jakub Waszczuk, Marie Candito, Ashwini Vaidya, Verginica Barbu Mititelu, Archna Bhatia, Uxoa Iñurrieta, Voula Giouli, Tunga Güngör, Menghan Jiang, Timm Lichte, Chaya Liebeskind, Johanna Monti, Renata Ramisch, Sara Stymne, Abigail Walsh, and Hongzhi Xu (2020), Edition 1.2 of the PARSEME shared task on semi-supervised identification of verbal multiword expressions, in *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pp. 107–118, Association for Computational Linguistics, online, `https://aclanthology.org/2020.mwe-1.14`.

Agata Savary, Marie Candito, Verginica Barbu Mititelu, Eduard Bejček, Fabienne Cap, Slavomír Čéplö, Silvio Ricardo Cordeiro, Gülşen Eryiğit, Voula Giouli, Maarten van Gompel, Yaakov HaCohen-Kerner, Jolanta Kovalevskaitė, Simon Krek, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Lonneke van der Plas, Behrang QasemiZadeh, Carlos Ramisch, Federico Sangati, Ivelina Stoyanova, and Veronika Vincze (2018), PARSEME multilingual corpus of verbal multiword expressions, in Stella Markantonatou, Carlos Ramisch, Agata Savary, and Veronika Vincze, editors, *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, pp. 87–147, Language Science Press, Berlin, Germany.

Livnat Herzig Sheinfux, Tali Arad Greshler, Nurit Melnik, and Shuly Wintner (2019), Verbal multiword expressions: Idiomaticity and flexibility, in Yannick Parmentier and Jakub Waszczuk, editors, *Representation and parsing of multiword expressions: Current trends*, pp. 35–68, Language Science Press, Berlin.

Rodrigo Wilkens, Bruno Oberle, Frédéric Landragin, and Amalia Todirascu (2020), French coreference for spoken and written language, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 80–89, European Language Resources Association, Marseille, France, `https://aclanthology.org/2020.lrec-1.10`.

A APPENDIX

Table 7: LVC.full in true overlaps with frequencies greater than 1

Expressions	Literal translation	True meaning	True overlaps
faire** des/une **étude(s)	*do studies/a study*	*study/perform a survey*	50
poser** une **question	*pose a question*	*ask a question*	24
faire grêve	*do strike*	*go on strike*	19
prendre** des **sanctions	*take sanctions*	*impose sanctions*	13
avoir** une **difficulté	*have a difficulty*	*have a difficulty*	12
avoir** un **problème	*have a problem*	*have a problem*	6
avoir** un **contact	*have contact*	*have contact*	5
avoir** l'**habitude	*have the habit*	*have the habit*	4
avoir** une **question	*have a question*	*have a question*	4
avoir** un **rapport	*have a relation*	*have a relation*	4
faire** un **essai	*do a test*	*try*	4
passer** des **vacances	*pass holidays*	*spend holidays*	4
avoir** une **fracture	*have a fracture*	*have a fracture*	3
avoir** une **idée	*have an idea*	*have an idea*	3
faire confiance	*do trust*	*trust*	3
faire** un **travail	*do a work*	*do work*	3
avoir** une **activité	*have an activity*	*have an activity*	2
avoir besoin	*have need*	*need*	2
avoir** une **conséquence	*have a consequence*	*have a consequence*	2
avoir** de l'**importance	*have importance*	*have importance*	2
avoir** l'**impression	*have the impression*	*feel like*	2
avoir** une **opinion	*have an opinion*	*have an opinion*	2
avoir** un **projet	*have a project*	*have a project*	2
donner** un **enseignement	*give a teaching*	*teach a lesson*	2
donner** une **réponse	*give an answer*	*give an answer*	2
exercer** un **contrôle	*exercise a control*	*control*	2
faire classe	*do classes*	*give classes*	2
faire** des **courses	*do shopping*	*do shopping*	2
atteint** d'**insuffisance	*attained by insufficiency*	*affected by insufficiency*	2
mener** une **action	*conduct an action*	*conduct an action*	2
mener** une **étude	*conduct a study*	*conduct a study*	2
prendre** une **décision	*take a decision*	*make a decision*	2
prendre** une **photo	*take a photo*	*take a photo*	2
subir** un **traitement	*endure a treatment*	*undergo a treatment*	2

Table 8: LVC.full in true overlaps with frequency 1

Expressions	Literal translation	True meaning	True overlaps
accomplir un travail	*complete a work*	*accomplish work*	1
atteint *de* ***maladie***	*attained by a disease*	*affected by a disease*	1
atteint *de* ***SCA***	*attained by ACS*	*affected by ACS*	1
avoir *la* ***capacité***	*have the ability*	*have the ability*	1
avoir connaissance	*have knowledge*	*know*	1
avoir *une* ***formation***	*have a training*	*have a background*	1
avoir *une* ***influence***	*have an influence*	*have an influence*	1
avoir *l'****intention***	*have the intention*	*to intend*	1
avoir *un* ***intérêt***	*have an interest*	*be interested*	1
avoir *une* ***religion***	*have a religion*	*be religious*	1
avoir *une* ***relation***	*have a relation*	*have a relationship*	1
avoir *un* ***rendement***	*have a return*	*have a yield*	1
avoir *une* ***responsabilité***	*have a responsability*	*be in charge*	1
avoir *un* ***rôle***	*have a role*	*play a role*	1
avoir vocation	*have a vocation*	*have a vocation*	1
commettre *un* ***crime***	*commit a crime*	*commit a crime*	1
comporter *un* ***risque***	*involve a risk*	*pose a risk*	1
dispenser *un* ***enseignement***	*dispense teaching*	*teach*	1
donner *un* ***concert***	*give a concert*	*give a concert*	1
donner *un* ***conseil***	*give an advice*	*give an advice*	1
donner *un* ***cours***	*give a course*	*give a course*	1
donner *un* ***ordre***	*give an order*	*give an order*	1
entreprendre *une* ***action***	*undertake an action*	*take an action*	1
exercer *une* ***activité***	*exercise an activity*	*carry on business*	1
faire *une* ***demande***	*make a request*	*submit a request*	1
faire *un* ***effort***	*make an effort*	*make an effort*	1
faire *une* ***fête***	*make a party*	*have a party*	1
faire *une* ***guerre***	*make a war*	*wage war*	1
faire *une* ***recherche***	*do research*	*make a search*	1
faire *un* ***service***	*do a service*	*do a service*	1
garder *un* ***souvenir***	*keep a memory*	*remember*	1
mener *un* ***combat***	*conduct a fight*	*wage a battle*	1
prendre *un* ***cours***	*take a course*	*take a course*	1
prendre *une* ***position***	*take a position*	*take a stand*	1
produire *un* ***résultat***	*produce a result*	*produce a result*	1
présenter *des* ***saignements***	*present bleedings*	*bleed*	1
présenter *un* ***symptôme***	*present a symptom*	*show a symptom*	1

continued on next page

Table 8: LVC.full in true overlaps with frequency 1 (*continued from previous page*)

Expressions	Literal translation	True meaning	True overlaps
réaliser *une* ***étude***	*realize a study*	*conduct a study*	1
recevoir *une* ***perfusion***	*receive an infusion*	*receive an infusion*	1
recevoir *une* ***éducation***	*receive an education*	*be educated*	1
signer *une* **ordonnance**	*sign a prescription*	*sign a prescription*	1
souffrir *de* ***maladie***	*suffer from a disease*	*suffer from a disease*	1
souffrir *de* ***syndrôme***	*suffer from a syndrome*	*suffer from a syndrome*	1
subir *une* ***angioplastie***	*endure an angioplasty*	*undergo an angioplasty*	1
subir *un* ***pontage***	*endure a bypass surgery*	*undergo a bypass surgery*	1
suivre *un* **cours**	*follow a course*	*take a course*	1
avoir *la* ***perception***	*have the perception*	*perceive*	1
avoir *la* ***possibilité***	*have the possibility*	*have the opportunity*	1

Table 9: VID in true overlaps

Expressions	Literal translation	True meaning	True overlaps
avoir le temps	*have the time*	*have the time*	16
poser problème	*pose problem*	*pose problem*	4
prendre le temps	*take the time*	*take the time*	2
prendre *sa* ***place***	*take one's place*	*take one's place*	2
il est question	*it is question*	*it is about*	1
porter *un* ***nom***	*bear a name*	*bear a name*	1
en revenir	*return of it*	*go back to something*	1
faire plaisir	*make pleasure*	*give pleasure*	1
en savoir	*know of it*	*know*	1

Table 10: LVC.cause in true overlaps

Expressions	Literal translation	True meaning	True overlaps
donner l'impression	*give the impression*	*give the impression*	1

Table 11: IRV in true overlaps

Expressions	Literal translation	True meaning	True overlaps
se diriger	*direct oneself*	*go, proceed*	1

Agata Savary
0000-0002-6473-6477

Paris-Saclay University, CNRS, LISN, France

Jianying Liu
0009-0004-8939-8023

Inalco, Paris, France

Anaëlle Pierredon
0009-0008-5093-0384

Inalco, Paris, France

Jean-Yves Antoine
0000-0002-6028-1663

University of Tours, LIFAT, France

Loïc Grobol
0000-0002-4619-7836

Paris-Nanterre University, MoDyCo, CNRS, France

Agata Savary, Jianying Liu, Anaëlle Pierredon, Jean-Yves Antoine, and Loïc Grobol (2023), *We thought the eyes of coreference were shut to multiword expressions and they mostly are*, Journal of Language Modelling, 11(1):147–187
https://dx.doi.org/10.15398/jlm.v11i1.328