



Journal of Language Modelling

VOLUME 7 ISSUE 2
DECEMBER 2019



*Institute of Computer Science
Polish Academy of Sciences
Warsaw*

Journal of Language Modelling

VOLUME 7 ISSUE 2
DECEMBER 2019

Editorials

Finite-state methods in natural language processing
and mathematics of language.

Introduction to the special issue 1

Frank Drewes, Makoto Kanazawa

Articles

Monotonicity as an effective theory of morphosyntactic variation 3

Thomas Graf

Finite-state Optimality Theory: non-rationality of Harmonic Serialism 49

Yiding Hao

Learning cross-lingual phonological and orthographic adaptations:
a case study in improving neural machine translation

between low-resource languages 101

Saurav Jha, Akhilesh Sudhakar, Anil Kumar Singh

Extracting Subregular constraints from Regular stringsets 143

James Rogers, Dakotah Lambert

How to embed noncrossing trees
in Universal Dependencies treebanks
in a low-complexity regular language 177

Anssi Yli-Jyrä



JOURNAL OF
LANGUAGE MODELLING

ISSN 2299-8470 (electronic version)

ISSN 2299-856X (printed version)

<http://jlm.ipipan.waw.pl/>

MANAGING EDITOR

Adam Przepiórkowski IPI PAN

GUEST EDITORS OF THIS SPECIAL ISSUE

Frank Drewes Umeå University, Umeå, Sweden

Makoto Kanazawa Hosei University, Tokyo, Japan

SECTION EDITORS

Elżbieta Hajnicz IPI PAN

Agnieszka Mykowiecka IPI PAN

Marcin Woliński IPI PAN

STATISTICS EDITOR

Łukasz Dębowski IPI PAN



Published by IPI PAN

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

Circulation: 100 + print on demand

Layout designed by Adam Twardoch.

Typeset in X_YL^AT_EX using the typefaces: *Playfair Display*
by Claus Eggers Sørensen, *Charis SIL* by SIL International,
JLM monogram by Łukasz Dziedzic.

*All content is licensed under
the Creative Commons Attribution 3.0 Unported License.*
<http://creativecommons.org/licenses/by/3.0/>



EDITORIAL BOARD

Steven Abney University of Michigan, USA

Ash Asudeh Carleton University, CANADA;
University of Oxford, UNITED KINGDOM

Chris Biemann Technische Universität Darmstadt, GERMANY

Igor Boguslavsky Technical University of Madrid, SPAIN;
Institute for Information Transmission Problems,
Russian Academy of Sciences, Moscow, RUSSIA

António Branco University of Lisbon, PORTUGAL

David Chiang University of Southern California, Los Angeles, USA

Greville Corbett University of Surrey, UNITED KINGDOM

Dan Cristea University of Iași, ROMANIA

Jan Daciuk Gdańsk University of Technology, POLAND

Mary Dalrymple University of Oxford, UNITED KINGDOM

Darja Fišer University of Ljubljana, SLOVENIA

Anette Frank Universität Heidelberg, GERMANY

Claire Gardent CNRS/LORIA, Nancy, FRANCE

Jonathan Ginzburg Université Paris-Diderot, FRANCE

Stefan Th. Gries University of California, Santa Barbara, USA

Heiki-Jaan Kaalep University of Tartu, ESTONIA

Laura Kallmeyer Heinrich-Heine-Universität Düsseldorf, GERMANY

Jong-Bok Kim Kyung Hee University, Seoul, KOREA

Kimmo Koskenniemi University of Helsinki, FINLAND

Jonas Kuhn Universität Stuttgart, GERMANY

Alessandro Lenci University of Pisa, ITALY

Ján Mačutek Comenius University in Bratislava, SLOVAKIA

Igor Mel'čuk University of Montreal, CANADA

Glyn Morrill Technical University of Catalonia, Barcelona, SPAIN

Stefan Müller Freie Universität Berlin, GERMANY

Mark-Jan Nederhof University of St Andrews, UNITED KINGDOM

Petya Osenova Sofia University, BULGARIA

David Pesetsky Massachusetts Institute of Technology, USA

Maciej Piasecki Wrocław University of Technology, POLAND

Christopher Potts Stanford University, USA

Louisa Sadler University of Essex, UNITED KINGDOM

Agata Savary Université François Rabelais Tours, FRANCE

Sabine Schulte im Walde Universität Stuttgart, GERMANY

Stuart M. Shieber Harvard University, USA

Mark Steedman University of Edinburgh, UNITED KINGDOM

Stan Szpakowicz School of Electrical Engineering
and Computer Science, University of Ottawa, CANADA

Shravan Vasishth Universität Potsdam, GERMANY

Zygmunt Vetulani Adam Mickiewicz University, Poznań, POLAND

Aline Villavicencio Federal University of Rio Grande do Sul,
Porto Alegre, BRAZIL

Veronika Vincze University of Szeged, HUNGARY

Yorick Wilks Florida Institute of Human and Machine Cognition, USA

Shuly Wintner University of Haifa, ISRAEL

Zdeněk Žabokrtský Charles University in Prague, CZECH REPUBLIC

Finite-state methods in natural language processing and mathematics of language. Introduction to the special issue

*Frank Drewes*¹ and *Makoto Kanazawa*²

¹ Department of Computing Science, Umeå University, Umeå, Sweden

² Department of Advanced Sciences, Hosei University, Tokyo, Japan

For more than half a century, finite-state methods and mathematical linguistics have benefitted from a close relation and fruitful interaction. Both research fields aim to achieve a deeper understanding of human language by means of mathematical techniques. For the automated processing of language by computers such a mathematical basis is an indispensable prerequisite. Historically, the goal pursued by mathematical linguists to formalize natural language syntax in a computer-accessible way was one of the strongest driving forces behind the development of finite-state methods. Thus, it is no exaggeration to say that the field of finite-state methods owes its existence to a large extent to mathematical linguistics. In turn, continued research on finite-state methods has resulted in a categorization of various kinds of language classes and language aspects, together with efficient and provably correct algorithms, thus expanding our understanding of the mathematical properties of language.

The two premier conferences in these fields are *Finite-State Methods in Natural Language Processing* (FSMNLP) and *Mathematics of Language* (MoL), organized biannually by their respective ACM Special Interest Groups SIGFSM and SIGMOL. The most recent installments of these conferences were FSMNLP 2017, which took place in Umeå, Sweden, on September 4–6, 2017 and MoL 2017 held on July 13–14, 2017 at Queen Mary University of London, UK. Because of the fruitful interaction between the fields, it was a natural idea to compile a joint special issue that would collect extended versions of a small number

Frank Drewes, Makoto Kanazawa

of selected contributions of both of these conferences. This special issue of the Journal of Language Modelling is the result, containing five articles that substantially extend, and in some cases correct, the corresponding short articles in the conference proceedings of FSMNLP 2017 and MoL 2017. In this, we follow the example of the special issue of FSMNLP 2015 and MoL 2015, which appeared as Vol 5, No 1 of the Journal of Language Modelling almost exactly two years ago.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.
<http://creativecommons.org/licenses/by/3.0/>



Monotonicity as an effective theory of morphosyntactic variation

Thomas Graf¹

Department of Linguistics, Stony Brook University, Stony Brook, USA

ABSTRACT

One of the major goals of linguistics is to delineate the possible range of variation across languages. Recent work has identified a surprising number of typological gaps in a variety of domains. In morphology, this includes stem suppletion, person pronoun syncretism, case syncretism, and noun stem allomorphy. In morphosyntax, only a small number of all conceivable Person Case Constraints and Gender Case Constraints are found. While various proposals have been put forward for each individual domain, few attempts have been made to give a unified explanation of the limited typology across all domains. This paper presents a novel account that deliberately abstracts away from the usual details of grammatical description in order to provide a domain-agnostic explanation of the limits of typological variation. This is achieved by combining prominence hierarchies, e.g. for person and case, with mappings from those hierarchies to the relevant output forms. As the mappings are required to be monotonic, only a fraction of all conceivable patterns can be instantiated.

Keywords:
monotonic
functions,
syncretism,
*typology, *ABA-*
generalization,
Person Case
Constraint,
Gender Case
Constraint

1

INTRODUCTION

In physics, an *effective theory* describes the behavior of a system at a higher level of abstraction that does not necessarily reflect the true causal factors that give rise to the behavior. For example, the physical laws governing the behavior of gases are effective theories of a system whose causal factors reside at the much lower level of atoms and fundamental forces. An effective theory is often easier to understand than

the actual system, and it may furnish generalizations that are harder or impossible to state at a more fine-grained level of description.

The central goal of this paper is to develop such an effective theory for certain areas of morphology and morphosyntax that have attracted a lot of attention in recent years:

- (1) a. **The *ABA generalization in morphology**
 - Stem suppletion in adjectival gradation
 - Syncretism in person pronoun paradigms
 - Syncretism in case paradigms
 - Noun stem allomorphy
- b. **Morphosyntactic constraints on clitic clusters**
 - Person Case Constraint (PCC)
 - Gender Case Constraint (GCC)

In each domain, the goal is to explain why not all logically conceivable patterns are attested. For example, there are 64 logically possible PCC variants, but only a handful have been reported in the literature. Such seemingly arbitrary typological gaps demand a principled explanation, and the explanation should apply across as many domains as possible.

The explanation proposed in this paper consists of two components: a base hierarchy that captures certain prominence relations between the elements in a domain, and a mapping from each base hierarchy to the relevant output forms. Crucially, the mapping must be monotonic. The shape of the base hierarchy and the monotonicity requirement conspire to greatly limit the range of possible patterns.

The approach advocated here is strongly inspired by the mathematical formalism of Graf (2014, 2017) but improves on it in important respects. A broader range of data is considered, including a wide selection of case syncretisms and a new kind of PCC reported by Tyler (2017) for Choctaw. In addition, the analysis in terms of monotonicity greatly simplifies Graf's rather byzantine machinery (I am indebted to an anonymous reviewer of Graf 2017 for pushing me to explore monotonicity as a unifying principle). In contrast to generative accounts such as Anagnostopoulou (2005), Nevins (2007), Caha (2009), Bobaljik (2012), and Zompí (2016), the monotonicity approach provides a unified solution for all the phenomena above, rather than just one or two of them. This is because as an effective theory, my proposal

can focus on describing the general behavior of the system rather than how this behavior arises from the machinery of the grammar.

My proposal is close in spirit to Bobaljik and Sauerland (2018), who seek to derive the *ABA generalization from feature combinatorics without making specific reference to the content or denotation of these features. However, my account is less radical in its quest for content-agnostic explanations as each domain may come with its own base hierarchy. This makes the approach easier to apply to specific phenomena. But as it is still an open question what specific forms the base hierarchies may take, there is also a lot of room for over-generation. The hierarchies proposed in this paper are largely in line with current linguistic thinking, but a tight mathematical characterization of the space of possible hierarchies is still missing. The ideas pursued in Bobaljik and Sauerland (2018) might actually turn out to be equivalent to specific restrictions on base hierarchies. So even though the two approaches differ a fair bit at this point and address slightly different questions, they are at the very least fellow travelers. In particular, both largely abstract away from feature systems and the specifics of the grammar and thus are not tied to any specific grammar formalism.

The paper proceeds as follows. Section 2 defines monotonicity and explains it in intuitive terms. No other mathematical concepts are needed for this paper. Sections 3 and 4 then present the account of the *ABA generalization and the PCC, respectively. I conclude the paper with some brief thoughts on the status of monotonicity in general (Section 5.1), the psychological reality of the monotonicity account and the mechanisms it posits (Section 5.2), and the empirical robustness of the approach in light of an impoverished data sample (Section 5.3).

2

MONOTONICITY:
DEFINITION AND EXPLANATION

Even though monotonicity is a well-known concept of mathematics, I include a detailed explanation here to accommodate as large an audience as possible. Readers who are already familiar with monotonic functions can skip ahead to Section 2.2, where I introduce the notion *feasibly monotonic* as a minor generalization of monotonicity. This generalization step will simplify the discussion of the *ABA generalization

in Section 3. The analysis of PCC effects in Section 4 only needs the standard notion of monotonicity.

2.1 *Monotonic functions*

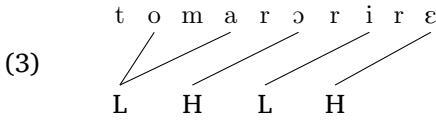
Monotonicity expresses whether a mapping between two objects respects their internal structure. Suppose we are given two structures $\langle A, \leq_A \rangle$ and $\langle B, \leq_B \rangle$ such that A and B are (possibly infinite) sets with respective order relations \leq_A and \leq_B defined over them. For example, $\langle A, \leq_A \rangle$ may be the set of natural numbers ordered by the less-or-equal relation, and $\langle B, \leq_B \rangle$ may be the set of Latin characters in alphabetical order. Then a function f from A to B is *monotonic* iff f preserves the relative order of elements. Sticking with our example of natural numbers and alphabet letters, a monotonic function must not map 10 to H and 100 to E because $10 \leq 100$ but $f(10) = H$ occurs after $f(100) = E$ rather than before it. However, the function may map all numbers between 0 and 10 to E and all other numbers to H, as this does not invert the original order.¹

(2) **Monotonicity**

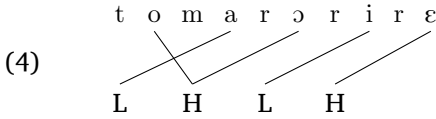
Given two sets A and B , let $\leq_A \subseteq A \times A$ and $\leq_B \subseteq B \times B$. A function $f : A \rightarrow B$ is *monotonic* with respect to \leq_A and \leq_B iff it holds for all x and y in A that $x \leq_A y$ implies $f(x) \leq_B f(y)$.

A linguistic analogy for monotonicity is the ban against crossing branches in autosegmental phonology (Goldsmith 1976). In autosegmental phonology, a phonological representation is not merely a string of segments, but instead consists of multiple tiers whose elements are connected by association lines. Each tier is still linearly ordered, though, and may be regarded as a string on its own. For example, a representation may consist of a string of segments, i.e. the segmental tier, and a string of tones, i.e. the tone tier. This is illustrated below with an example of tone association in Kikuyu.

¹The reader may have noticed that this description of monotonicity only applies to monotonically increasing (or *isotone*) functions. A function can also be monotonically decreasing (or *antitone*). In this case, the order must be inverted: $x \leq_A y$ implies $f(y) \leq_B f(x)$. For the purposes of this paper, the distinction is immaterial because every isotone function from A to B is antitone from A to the dual of B . For instance, an isotone function from numbers to alphabetically ordered letters would be antitone if the letters are instead ordered in reverse.

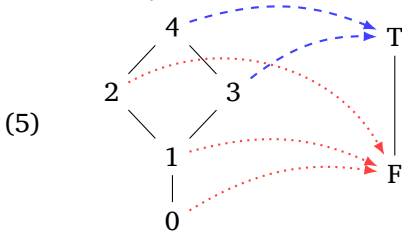


The ban against crossing branches ensures that segmental tier and tonal tier are synchronized to a certain extent. The linear order of tones must reflect the linear order of the segments they are associated to. Whenever this is not the case, some association lines illicitly cross each other as in the representation below.



These are exactly the cases where the mapping from elements on the segmental tier to elements on the tone tier is not monotonic. In the case at hand, the segment *o* linearly precedes *a*, yet *o* is mapped to a high tone H that follows the low tone L that *a* is associated with.

Note that even though the examples above all involve linearly ordered structures, monotonicity is more general and can be evaluated for any arbitrary ordering relation. The example below depicts a mapping from a partially ordered structure **S** on the left to the algebra **2** of truth values on the right. This mapping is monotonic because there are no x and y such that $x \leq_S y$ yet $f(y) <_2 f(x)$. In particular, it is irrelevant for monotonicity that $f(2) <_2 f(3)$ because neither $2 \leq_S 3$ nor $3 \leq_S 2$ hold. If two elements x and y are unordered with respect to each other, the relative order of $f(x)$ and $f(y)$ is immaterial for monotonicity.



However, if the mapping is altered just a bit such that 1 is mapped to True instead of False, monotonicity is lost because then we have $1 <_S 2$ yet $f(2) = F <_2 T = f(1)$.

We will encounter both linearly and partially ordered structures in this paper. Linearly ordered structures are at the center of the *ABA-generalization for adjectival gradation and pronoun syncretism (Section 3.1). Partial orders, on the other hand, are indispensable for broadening the empirical scope to case syncretism (Section 3.3), noun stem allomorphy (Section 3.4), Person Case Constraints (Sections 4.2, 4.3), and the Gender Case Constraint (Section 4.4).

2.2 *Feasibly monotonic functions*

During the discussion of the *ABA generalization in Section 3, there will be some cases where the co-domain B does not have any natural order defined over it. Adjectival gradation, for example, involves two kinds of objects:

1. a set of adjectival degrees, i.e. {positive, comparative, superlative}, and
2. a set of surface realizations, e.g. {slow, slower, slowest}.

Whereas the former can be given a natural order in terms of semantics, the set of surface realizations lacks such an internal structure. There is no obvious ordering relation between these three phonological representations. One could put them in reverse alphabetical order, or line them up according to length or morphological complexity. For our purposes, the important thing is simply that some sufficiently strict order is defined over this domain so that monotonicity can be invoked. We make this requirement explicit via the notion of *feasible monotonicity*.

(6) **Feasible monotonicity**

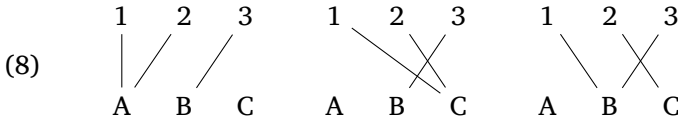
Let A be a set ordered by $\leq_A \subseteq A \times A$, and B some arbitrary set. Then $f : A \rightarrow B$ is *feasibly monotonic* iff there is some linear order $\leq_B \subseteq B \times B$ such that f is monotonic with respect to \leq_A and \leq_B .

(7) **Linear order**

A relation $\leq_B \subseteq B \times B$ is a linear order iff all of the following hold for all $x, y, z \in B$:

- *reflexivity*: $x \leq_B x$
- *antisymmetry*: $x \leq_B y$ and $y \leq_B x$ jointly imply $x = y$
- *transitivity*: $x \leq_B y$ and $y \leq_B z$ jointly imply $x \leq_B z$
- *totality*: $x \leq_B y$ or $y \leq_B x$

The figure below shows three mappings. The leftmost one is monotonic. The one in the middle is not monotonic, but it is feasi- bly monotonic because we can switch the order of B and C and thus obtain a monotonic mapping. The function on the right, on the other hand, is neither monotonic nor feasi- bly monotonic: no matter which order one picks for A, B, and C, two branches will always cross, indi- cating that the mapping is not monotonic.



To sum up, monotonic mappings are order-preserving in the sense that they do not invert existing orderings: $x \leq_A y$ entails $f(x) \leq_B f(y)$. The notion of feasi- bly monotonic mappings extends this to cases where the co-domain lacks internal structure. It does so by consid- ering all possible ways to order the co-domain such that feasible monotonicity holds iff monotonicity holds for at least one of those orders. The next section discusses the *ABA generalization as the first application of (feasible) monotonicity, with the PCC following in Sec- tion 4.

3

*ABA GENERALIZATION

The first part of the empirical analysis is devoted to the *ABA gen- eralization. I initially limit myself to suppletion in adjectival gradation and pronoun systems (Section 3.1). Each domain involves only 3 cells, which simplifies the discussion. Both of them will be explained in terms of a linearly ordered based hierarchy – one for adjectival deg- rees, another one for person. Crucially, the mappings from these hier- archies to surface forms must be feasi- bly monotonic. This requirement severely restricts the range of possible suppletion patterns, providing a close fit for the attested typology and reducing the *ABA generaliza- tion to monotonicity.

I then expand this approach to larger, partially ordered hierar- chies to account for case syncretism (Section 3.3) and noun stem al- lomorphs (Section 3.4). The general idea remains the same, though: once a suitable, linguistically motivated hierarchy has been fixed, the

range of cross-linguistic variation falls out from the limitation to (feasibly) monotonic functions.

As I explain in Section 3.2, the monotonicity program is still in its early stages, and as a result the construction of these hierarchies is primarily guided by empirical concerns. Conceptual or mathematical restrictions on the shape of hierarchies must be postponed until a larger class of hierarchies has been identified. That is not to say, though, that the hierarchies presented in this paper are completely arbitrary. They display many regularities, and are very natural from a linguistic perspective.

3.1 3-cell paradigms: Adjectival gradation and pronoun allomorphy

The *ABA generalization was formulated in Bobaljik (2012) and refers to a particular typological gap in numerous morphological paradigms. Given a morphological subsystem where one may posit an underlying hierarchy $x > y > z$, z cannot pattern with x to the exclusion of y .

The best-known example of the *ABA generalization is suppletion in adjectival gradation, which was analyzed at great depth in Bobaljik (2012). Bobaljik points out that if a language allows for stem suppletion in either comparatives or superlatives, it must allow for both. Data illustrating this generalization is given in Table 1. If one follows the convention to list the three forms in the order *positive*, *comparative*, *superlative* and uses letters to indicate which forms use the same stem, one can decompose the typological gaps into two constraints: *AAB and *ABA. The central puzzle is why these specific constraints should hold for adjectival gradation but not, say, *ABB or *ABC.

In Bobaljik (2012), the ban against ABA patterns is explained via structural mechanisms. Adjectival forms are decomposed into a tree

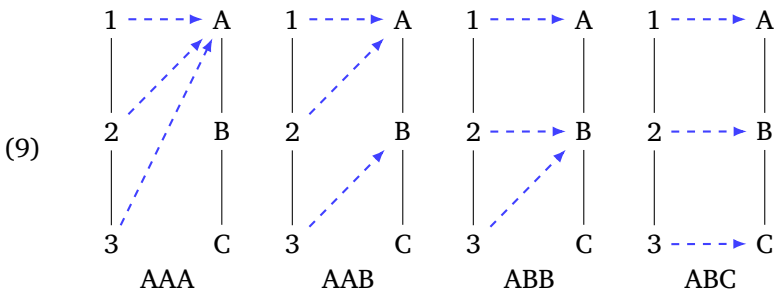
Table 1:
Examples of attested
suppletion patterns from
Smith *et al.* (2018)

Language	Positive	Comparative	Superlative	Pattern
English	smart	smart-er	smart-est	AAA
English	good	bett-er	be-st	ABB
Finnish	hyvä	pare-mpi	parha-in	ABB
Latin	bon-us	mel-ior	opt-imus	ABC
Welsh	da	gwell	gor-au	ABC
unattested	good	bett-er	good-est	*ABA
unattested	good	good-er	be-st	*AAB

template such that comparatives contain the positive base form as a subtree and are in turn themselves subtrees of the corresponding superlative forms. Then *ABA follows from specific assumptions about the rewrite rules that convert these tree structures into morphological surface forms. Bobaljik and Sauerland (2018) provide an alternative explanation grounded in the combinatorics of feature systems. Under both approaches *ABA falls out from the fact that it is impossible for a rewrite rule to target positive and superlative forms to the exclusion of the comparative. Both works also agree that *ABA is the more important constraint of the two – whereas *ABA holds for many morphological paradigms, *AAB seems to be specific to adjectival suppletion and requires additional stipulations.

The increased importance of *ABA relative to *AAB is noteworthy because the former can be explained in terms of monotonicity, but not the latter. Suppose that there is a universal underlying hierarchy of the form *positive* > *comparative* > *superlative*. For the sake of succinctness, I abbreviate this hierarchy as $1 > 2 > 3$. Now let $\{A, B, C\}$ be the set of possible surface forms. Irrespective of how this set is ordered, there can be no monotonic function f with $f(1) = f(3) \neq f(2)$. We already saw this in (8) at the end of Section 2.2. Hence no feasibly monotonic function over $1 > 2 > 3$ can produce a pattern of the form ABA, and consequently the *ABA generalization reduces to a ban against functions that are not feasibly monotonic.

But the other patterns AAA, AAB, ABB, and ABC can be produced by feasibly monotonic functions, as is shown below with an assumed ordering of $A > B > C$.



Some trivial variations are not depicted here, such as a function that maps 1, 2, and 3 to C instead of A. Keep in mind that no further assumptions are made about the exponents of A, B, and C, so it is not the

case that A is the fixed counterpart of a positive form or C the fixed counterpart of a superlative form. Instead, A, B, and C are just abstract variables or bins, and any two forms that are put in the same bin must have the same exponent. Consequently, a function mapping all three forms to A is empirically equivalent to one mapping all three forms to C. In this system, there are only five ways to map three different forms to exponents. But one of them is the illicit (and not monotonic) ABA pattern, so that (9) already exhausts the full range of options.

We see, then, that the typology of adjectival gradation is partially explained by the assumption that

1. there is a universal hierarchy *positive* > *comparative* > *superlative*, and
2. the mapping from this hierarchy to surface forms must be feasibly monotonic.

These two assumptions explain the absence of ABA patterns, but they still allow for AAB patterns, which are unattested cross-linguistically. Just like the previous analyses in Bobaljik (2012) and Bobaljik and Sauerland (2018), monotonicity cannot give a unified explanation of the absence of both ABA and AAB patterns.

However, this is actually a welcome state of affairs because AAB patterns do show up in other empirical domains. Harbour (2015) conducts an extensive survey of pronoun systems. His findings are summarized in Table 2. Putting aside number and the inclusive-exclusive distinction to focus exclusively on person specification, we can infer that all of the languages surveyed by Harbour adopt one of four person systems:

- all persons are the same (AAA),
- first and second person are the same (AAB),

Table 2:
Typology of pronoun systems
according to Harbour (2015, p. 137)

Pronominal contrasts	Language
none	Wichita
1 23	Damin, Elseng/Morwap ²
12 3	Winnebago
1 2 3	Jarawa, Kiowa
1EX 1IN 2 3	Imonda, Matses, Waris
1 23 × SG PL	Sanapaná

- second and third person are the same (ABB),
- all persons are different (ABC).

Again the ABA pattern is missing, and this fact is expected if all languages use an underlying person hierarchy of $1 > 2 > 3$ and the mapping to surface forms must be feasibly monotonic. At the same time, the four attested patterns AAA, AAB, ABB, and ABC are completely expected from this perspective. I thus conclude that monotonicity can successfully limit the possible range of variation in these morphological domains, although certain options such as AAB may still be unattested due to unrelated factors.²

3.2

Motivating the hierarchies

The advantage of the monotonicity approach lies in its ability to account for the *ABA generalization across seemingly unrelated domains. Person and adjectival gradation have nothing in common semantically, and it also seems unlikely that they are syntactically related. More specifically, I am not aware of any proposals where first person is structurally contained by second person which in turn is contained by third person, mirroring Bobaljik's treatment of positive, comparative, and superlative. But from the abstract high-level perspective advocated here, person and adjectival degrees are exactly parallel because their respective hierarchies are isomorphic. Each one is of the form $1 > 2 > 3$, and the only difference is what each element of the hierarchy denotes.

²An anonymous reviewer wonders how the person hierarchy can be extended to the exclusive-inclusive distinction without losing its explanatory force. The reviewer notes that it would be very natural to analyze 1IN as the combination of 1 and 2, such that one gets the ordering relations $1IN > 1 > 3$ and $1IN > 2 > 3$. But with this hierarchy, 1 and 3 can now be syncretic to the exclusion of 2, undermining the results of this section.

This case shows that the choice of hierarchies must be carefully guided by data. An empirically more adequate hierarchy might preserve $1 > 2 > 3$ while also adding $1IN > 1$ and $1IN > 2$. This would incorporate the natural idea that 1IN combines first and second person without loosening the relative order of 1 and 2. At this early stage of the monotonicity enterprise, though, the choice of hierarchy is primarily driven by empirical data, and without a careful analysis of this data all claims about the shape of hierarchies are highly speculative.

This raises the question, though, whether there is any independent motivation for these hierarchies beyond their central role in accounting for the data. There certainly is, but before discussing this in detail I would like to point out that every account of the *ABA generalization has to assume some base hierarchy for the domain in question, at least at a descriptive level. Otherwise, the *ABA generalization in its current form cannot be stated. Suppose that adjectival gradation patterns were by default listed in the order *comparative-positive-superlative*. An attested pattern like *good-better-best* would then be regarded as *better-good-best*, which is an ABA pattern. The discussion of the *ABA generalization thus presupposes an agreed-upon base order for every domain under investigation.³ The monotonicity account simply takes this base order at face value and describes how the expected typology is narrowed down by restricting our attention to feasibly monotonic mappings from base hierarchies to surface forms.

Crucially, though, the hierarchies posited so far are highly plausible from a cognitive perspective. The hierarchy *positive > comparative > superlative* directly reflects the semantics of each form. The person hierarchy $1 > 2 > 3$, on the other hand, has already been argued for by Zwicky (1977) for entirely different reasons. This hierarchy is also implicit in feature-based systems such as that of Nevins (2007), where first person is [+author, +participant], second person is [−author, +participant], and third person is [−author, −participant]. Suppose we represent these specifications in privative terms as {author, participant}, {participant}, and {}, respectively. If one orders these sets by the superset relation, the very same ordering emerges as with Zwicky's person hierarchy. Admittedly, there are other well-known feature systems that give rise to a different ordering, e.g. Harley and Ritter (2002). The posited person hierarchy is also missing the crucial contrast between inclusive and exclusive. The current person hierarchy thus might present an overly simplified picture. But future refinements would only make the hierarchy an even closer

³This holds even for the account of Bobaljik and Sauerland (2018). While their mathematical analysis is order-independent and eliminates ABA patterns based on the shape of the feature system, the application to empirical data requires picking a suitable feature system. This is tantamount to positing a base hierarchy for the empirical domain under investigation.

match for current linguistic thinking. Overall, then, the hierarchies for person and adjectival gradation are both on linguistically solid ground.

Ultimately, the monotonicity approach must furnish a restrictive theory of linguistic hierarchies lest it devolve to a purely descriptive enterprise where hierarchies are tuned and tweaked until monotonicity yields the desired result. But these restrictions cannot be put in place *a priori*. They must be inferred by defining empirically adequate hierarchies for a wide range of phenomena and by isolating properties that separate these hierarchies from conceivable alternatives that produce undesirable data patterns (cf. fn. 2). This bottom-up strategy is a major methodological difference to Bobaljik and Sauerland (2018), who start out with abstract yet linguistically natural restrictions on feature systems and use those to derive the absence of ABA patterns. Such a top-down strategy could also be applied to the monotonicity approach, but I believe that a largely data-driven approach will prove more fruitful for a nascent enterprise like this. Without a rich body of well-established hierarchies, the best option is to craft restrictive hierarchies to fit the data and evaluate their linguistic plausibility. As the number of hierarchies grows, their shared properties will become more apparent and serve to constrain the shape of newly posited hierarchies.

It is also of interest in this connection how the hierarchies relate to linguistic assumptions about feature systems or structural projections. A clearer understanding of this link would make it easier to convert assumptions about linguistic feature systems into constraints on hierarchies. I have already hinted at such a connection between feature systems and hierarchies in my brief discussion of Nevins (2007) and its strong correspondence to the person hierarchy of Zwicky (1977). In this particular case, the relation is easy to discern thanks to the simple nature of both the person hierarchy and the feature system. But as we will see in the remainder of this paper, other empirical domains require much more elaborate hierarchies whose connections to features or projections from the linguistic literature is much less clear. The very next phenomenon, case syncretism, is already a striking example of the complexity of hierarchies.

3.3 *Moving beyond 3 cells: Case syncretism*

Even though adjectival gradation and pronoun allomorphy pertain to vastly different morphological domains, they are both similar in that their paradigms distinguish only three cells: positive-comparative-superlative for the former, first person-second person-third person for the latter. Many paradigms, however, involve more than three cells. Case is a prime example of this, with many languages distinguishing at least four different cases. It will be interesting to see if monotonicity still holds in these larger paradigms, and if so, what shape the relevant hierarchies have.

Caha (2009, 2013) provides a detailed study of case syncretism, i.e. which cases in a noun inflection paradigm may systematically display the same surface form. Such syncretisms are common across languages with multiple morphologically realized cases, e.g. Russian (Caha 2009, p.12).

	window (sg)	teacher (pl)	100
(10) Nom	okn-o	učitel-ja	st-o
Acc	okn-o	učitel-ej	st-o
Gen	okn-a	učitel-ej	st-a
Loc	okn-e	učitel-jax	st-a
Dat	okn-u	učitel-am	st-a
Inst	okn-om	učitel-am-i	st-a

The first column shows syncretism of nominative and accusative. In the second column, accusative and genitive are syncretic. The third column displays two syncretisms, nominative-accusative on the one hand and genitive-locative-dative-instrumental on the other. With six cases, there are 203 logically conceivable patterns, but only a fraction of those are attested. As it is unlikely that all these typological gaps are purely accidental, a more principled explanation of this limited typology is needed.

Even though Caha's primary concern is to accommodate the typological facts in the framework of nano-syntax, he first formulates a purely descriptive universal. His *strong case contiguity hypothesis* limits case syncretism to contiguous areas of Blake's case hierarchy (Blake 2001):

(11) Blake’s case hierarchy (strict version)

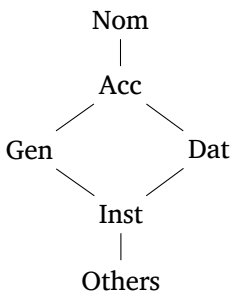
Nom > Acc > Gen > Dat > Inst > others

This means that a language may mark, say, accusative, genitive, dative and instrumental the same, but not accusative and instrumental to the exclusion of dative and genitive. In other words, the strong case contiguity hypothesis extends the *ABA generalization beyond systems with three-way contrasts.

The strong case contiguity hypothesis is yet another instance of monotonicity. Any feasibly monotonic function can only map continuous parts of Blake’s case hierarchy to the same exponent. If such a function mapped, say, accusative and dative to A but genitive to B, then we would have both $f(\text{acc}) \leq f(\text{gen}) \leq f(\text{dat})$ and $f(\text{acc}) = f(\text{dat}) \neq f(\text{gen})$, which is impossible. Hence (feasible) monotonicity over Blake’s case hierarchy rules out case syncretisms of the ABA-variety.

Curiously, though, such ABA-style case syncretisms do exist. Hardarson (2016) points out that accusative and dative are frequently syncretic to the exclusion of the genitive in Germanic languages. For the monotonicity account, the only way of incorporating this fact is to change the case hierarchy. By relaxing Blake’s hierarchy such that genitive and dative are unordered with respect to each other, we can keep all the syncretisms of the original hierarchy while also allowing for accusative-dative syncretism.

(12) Blake’s case hierarchy (relaxed version)



This is the first instance where I have to posit a hierarchy that is only partially ordered. But since monotonicity is not limited to linear orders (Section 2.1), the step to partial orders for base hierarchies is a natural one. With the hierarchy in (12), dative and genitive can still

be syncretic because they are unordered with respect to each other. Recall from Section 2.1 that monotonicity only limits the possible values for ordered elements, which entails that syncretism of these two unordered cases cannot violate monotonicity. At the same time, accusative still cannot be syncretic with instrumental to the exclusion of dative or genitive as this would violate the ordering relations established by the hierarchy.

However, such accusative-instrumental syncretism are actually attested, which means that the current hierarchy is still too restrictive. In fact, the range of attested case syncretisms goes far beyond what our relaxed hierarchy allows for. In a painstaking literature survey, Zompí (2016) has compiled an extensive list of case syncretisms across numerous typologically diverse languages, including languages with an ergative-absolutive system and even nominative-ergative-absolutive systems. His findings are summarized in Table 3 and include many syncretisms that are unexpected even with the relaxed version of Blake's hierarchy in (12).⁴

Zompí (2016) argues for a radically simplified case hierarchy to account for the permissive typology. Cases come in three types: unmarked core case (Nom, Abs), marked core case (Acc, Erg), and oblique case (Gen, Dat, Loc, Inst, Prep, possibly others). The case hierarchy is then simplified to *unmarked* < *marked* < *oblique*, and every syncretism must be continuous over this ordering of classes. For example, nominative-accusative syncretism is licensed because it covers two adjacent classes, unmarked and marked. Similarly, nominative-accusative-dative-instrumental syncretism involves only adjacent classes and thus is permitted. An unattested syncretism of absolutive and genitive, on the other hand, is correctly ruled out because it would involve an unmarked case and an oblique case to the exclusion of all marked cases.

⁴Zompí (2016) classifies some cases like ablative and allative as instances of other cases. The table faithfully reproduces his terminology to the best of my abilities. Note that Prep is short for *prepositional case*. Zompí also discusses what he calls a *prepositional locative*, and it is unclear whether this should be subsumed under Loc like a standard locative or under Prep. I decided to list it as Loc, but treating it as an instance of Prep would work equally well for the purposes of this paper.

Monotonicity in morphosyntax

Syncretism	Page(s)
Nom-Acc	6, 18, 35, 36
Nom-Acc + Dat-Inst	6, 28
Nom-Acc + Gen-Loc	28
Nom-Acc + Gen-Loc + Dat-Inst	18
Nom-Acc + Erg-Inst	37
Nom-Acc + Erg-Prep	39
Nom-Acc-Prep + Erg-Gen	39
Nom-Acc-Dat	19, 22
Nom-Acc-Dat-Inst	23, 24
Nom-Gen	83
Nom-Gen + Acc-Dat	87
Nom-Erg	33, 35, 36, 37
Acc-Gen	18
Acc-Gen + Loc-Dat	28
Acc-Dat	19, 21, 22
Acc-Inst	24
Acc-Prep	39
Acc-Loc-Dat	27
Acc-Loc + Gen-Dat-Inst	26
Gen-Dat	23
Gen-Loc-Dat	26
Loc-Dat	27, 28
Dat-Inst	23
Abs-Erg	32
Erg-Gen	32
Erg-Inst	33

Table 3:
Case syncretism patterns
from Zompí (2016)

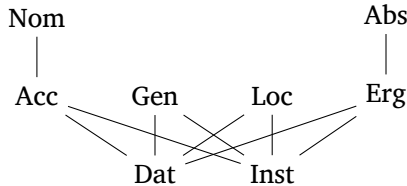
While Zompí's approach represents a marked improvement, it still falls short as it is both too permissive and too restrictive. For one thing, the hierarchy allows syncretisms such as nominative-ergative-dative, which do not seem to occur. When a syncretism involves nominative and an oblique case, the marked case is always accusative and never ergative. Admittedly this might just be a statistical confound: languages with a three way contrast between nominative, ergative, and absolutive are exceedingly rare, and so are syncretisms that involve all three of Zompí's case types. Hence, one is very unlikely to

come across a language that could conceivably display syncretism of nominative, ergative, and an oblique. Still, a more principled explanation that curbs overgeneration and provides a tighter fit for this typological gap would be welcome.

The more severe problem, as Zompí (2016, p. 88) readily admits, is that his solution still undergenerates because it cannot account for the robustly attested syncretism of nominative and genitive. This is an instance of an unmarked case being syncretic with an oblique case to the exclusion of all marked cases, directly contradicting Zompí’s central claim that syncretism must be contiguous across case classes. Zompí (2016, p. 87f) concludes that genitive must enjoy some special status, but does not offer a detailed account of how genitive is supposed to work in his system.

The monotonicity approach can remedy these shortcomings by building on Zompí’s idea of three distinct case classes and combining them with the insight that hierarchies need not be linearly ordered. The result is a hierarchy that roughly breaks down into three “case layers”, but also grants special status to genitive and locative. (To minimize clutter, the hierarchy below omits Zompí’s case Prep; like other oblique cases such as allative, ablative, and so on, it would reside in the third layer.)

(13) **Case layer hierarchy**



If one were to consolidate the individual cases into classes, this hierarchy would directly follow Zompí in positing *unmarked* < *marked* < *oblique*, except that one also has *special* < *oblique* for genitive and locative.

Let us look at several syncretism patterns from Table 3 and why they can be regarded as feasibly monotonic maps over this hierarchy. We start with nominative-accusative-dative as the only syncretism in the paradigm. Suppose that all three cases are mapped to some exponent C. Then genitive, locative, and ergative must be mapped to some

$B_g, B_l, B_e < C$, respectively; absolutive is mapped to some $A \leq B_e$; and instrumental has some $D > C$ as its exponent. Any set that furnishes a sufficient number of case exponents can be ordered in this way, so the mapping is feasibly monotonic.

Next, consider a system with three distinct syncretisms: nominative-accusative, genitive-locative, and dative-instrumental. Suppose nominative and accusative are mapped to some A . Then genitive and locative must be realized as some B , but it does not matter whether $A < B$ or $B < A$ since nominative and accusative are both unordered with respect to genitive and locative. Dative and instrumental must have an exponent C with both $A < C$ and $B < C$, and the remaining cases can be handled as before. Again, it is possible to produce such an ordering of exponents, and consequently we are dealing with yet another feasibly monotonic mapping over the case hierarchy.

Three more examples will prove instructive. First, note that syncretism of nominative and genitive to the exclusion of accusative is readily available in this system because genitive is unordered with respect to nominative and accusative. Hence, the explanation for nominative-genitive syncretism is exactly parallel to our previous account for genitive-dative syncretism in the relaxed version of Blake's hierarchy.

Second, the hierarchy captures the fact that nominative-accusative-dative syncretism is attested, but not nominative-ergative-dative. Since accusative occurs between nominative and all oblique cases (except genitive and locative), any syncretism involving nominative and one of these cases must also include accusative due to monotonicity. Yet it is also possible for nominative and ergative to be syncretic to the exclusion of accusative, which is also an attested pattern.

Third, locative must not be in the same case layer as other oblique cases because of paradigms where accusative and locative are syncretic while genitive, dative, and instrumental display a different syncretism. If we had *genitive* < *locative*, with A and B as the respective exponents, then monotonicity would require $A \leq B$. Since locative and genitive are not syncretic in this specific case, $A < B$ must hold. As accusative and dative are not syncretic, either, and we have *accusative* < *dative*, we also have $C < D$. But accusative and locative are syncretic, so $B = C$. Similarly, syncretism of genitive and dative

implies $A = D$. Put together, these equations entail both $A < B$ and $B < A$. It is clearly impossible for a linear order to obey both $A < B$ and $B < A$. Consequently, the described syncretism pattern cannot be feasibly monotonic unless locative and genitive are unordered, which requires locative to assume a special place in the hierarchy alongside genitive.

The last point highlights an important generalization of the monotonicity account.

(14) **Ban against multiple cross-level case syncretisms**

No case paradigm may display two distinct syncretism patterns $A-X$ and $B-Y$ such that A and B belong to the second case layer and X and Y to the third.

Hence the hierarchy in (13), albeit permissive, still puts a fair number of principled restrictions on case syncretism.⁵

Overall, then, the monotonicity approach does a decent job at accommodating the wide range of attested case syncretisms while still enforcing some testable restrictions on the typology. Its main advantage over competing approaches such as the nano-syntax analysis of

⁵The empirical impact of the generalization as presented here hinges on how one interprets language-specific data. No language instantiates all the cases listed in the hierarchy, which makes the status of unrealized cases an important issue. One option is to treat unrealized cases as if they were not part of the hierarchy at all. This is equivalent to positing a language-specific hierarchy that omits all unrealized cases. But instead one may treat case absence as case syncretism. For example, if a language lacks a distinct genitive but can use a dative for this purpose, one might analyze this as dative and genitive being syncretic across all paradigms. These two approaches are not empirically equivalent.

Consider a language that has both genitive and dative, but the two are sometimes syncretic. Suppose furthermore that the language also has an instrumental, which can also serve the role of a locative. However, instrumental and locative never have distinct forms. If unrealized cases are ignored, such a language is expected to exist since a monotonic function can map both genitive and dative to some exponent A while mapping instrumental to some other exponent B . If, on the other hand, missing cases are analyzed as an instance of complete syncretism, then both locative and instrumental have to be mapped to B . But then the ban against multiple cross-level syncretisms makes it impossible for both genitive and dative to be mapped to A .

I conjecture that the second approach is not empirically feasible, wherefore unrealized cases must be excised from the hierarchy.

Caha (2009) or the case type hierarchy of Zompí (2016) is the flexibility that comes with partially ordered hierarchies. This is difficult to achieve in a syntactic approach, where case hierarchies are replicated in terms of constituency containment (similar to how Bobaljik 2012 analyzes the superlative as containing the comparative, which in turn contains the base form). At the same time, though, the flexibility of the monotonicity approach also risks depriving it of any explanatory power – if just about any hierarchy will do, one can always fit the data as needed.

This is indeed a problem, but I maintain that all the hierarchies so far reflect common linguistic intuitions. As already explained in Section 3.2, the hierarchy for adjectival gradation is directly grounded in semantics, and the person hierarchy is both compatible with contemporary assumptions about person features and can be traced all the way back to Zwicky (1977). The case hierarchy looks more bewildering, but it, too, is built on linguistic ideas. The core of the hierarchy is the stratification into three types of cases, as argued for in Zompí (2016). The distinction between core cases and oblique cases is well-established, and within the core cases it is also standard to single out accusative and ergative as dependent cases that have, in a certain sense, lesser status than nominative and absolutive. The split between nominative and accusative on the one hand, and ergative and absolutive, is typologically well-supported, with most languages adopting one of the two but not both. This only leaves the special position of genitive and locative in need of an explanation.

However, our approach is not an exception in granting these cases privileged status. Caha (2009, p. 130) posits multiple distinct locatives, some of which occur very high in his linear hierarchy:

(15) Refined case hierarchy of Caha (2009)

Nom > Acc > Loc₁ > Gen > Loc₂ > Dat > Loc₃ > Inst

Zompí (2016, p. 87f), on the other hand, argues that genitive exhibits special behavior because it can be an unmarked (= default) case or an inherent case in syntax. Depending on its syntactic status, it may occur higher or lower in the case hierarchy. Instead of distinguishing multiple types of genitive and locative, the hierarchy in (13) instead assigns them a position that makes them more prominent than obliques but not directly comparable to the core cases.

At this point, then, we have a unified explanation of syncretism across three vastly different domains: adjectival gradation, pronouns, and case. In all three areas, typological gaps are accounted for by limiting the range of possible systems to those that can be produced by feasibly monotonic maps from some base hierarchy. Each base hierarchy is motivated by linguistic considerations. In the case of pronouns, the account provides a perfect fit for the typology, whereas it only carves out a superset of the attested patterns for adjectival gradation and case syncretism. For adjectival gradation, only the absence of the pattern AAB remains unexplained and requires a different account, e.g. in terms of syntactic containment. For case syncretisms, the amount of overgeneration is hard to assess because the number of possible combinations is so large that any gaps may just be due to insufficient data rather than principled exceptions. I put off discussion of this methodological issue until Section 5 – for now, I take the hierarchy to present a reasonable approximation of the typology of case syncretism.

I conclude this section on morphological syncretism with a short observation on noun stem allomorphy before moving on to a completely different phenomenon, the Person Case Constraint in morphosyntax.

3.4 *Case in noun stem allomorphy*

So far, I have only considered strict case syncretism, i.e. whether two forms that differ in case may have exactly the same surface realization. Hence the focus is on total identity for case. Instead, one can also look at partial identity, in particular whether two distinct cases attach to identical noun stems. This does not always occur. In Latin, for example, the nominative of ‘man’ is *hom-o*, whereas the accusative is *homin-em*. Nominative and accusative thus are formed with different stems of the same noun. This is known as *noun stem allomorphy*. As it turns out, the kinds of allomorphy observed with singular stems can also be captured in terms of monotonicity (I ignore plural stem allomorphy here because I am unaware of any detailed studies in this area).

McFadden (2018) proposes that all languages obey a strict condition on stem allomorphy.

(16) **Nominative stem-allomorphy generalization**

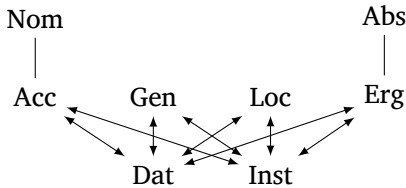
If noun stem allomorphy is conditioned by case, it distinguishes the nominative from all other cases.

In other words, noun stem allomorphy always displays an AB^n pattern, where A and B may be identical. Hence only the nominative may pick out a different stem. For a language with three cases, McFadden’s generalization permits only AAA and ABB while excluding AAB, ABA, and ABC.

We already know from our discussion in Section 3.1 that AAA and ABB can be produced from a linear hierarchy by feasibly monotonic maps. Generalizing from this, it does not take much effort to verify that both A^{n+1} and AB^n are licit patterns given the case layer hierarchy in (13). The puzzle of noun stem allomorphy, hence, is not why the attested patterns are possible. Once the case layer hierarchy is in place for case syncretism, the attested noun stem allomorphy patterns are also readily available. Instead, the question is why the majority of conceivable allomorphy patterns are unattested.

Although I cannot envision a convincing reason as to why noun stem allomorphy is much more restricted than case syncretism, it is worth noting that the observed restrictions can be given a natural account in terms of the case layer hierarchy in (13). In this hierarchy, all cases in the second layer are more prominent than the cases in the third layer. But suppose that the hierarchy contains a cycle such that all third-layer cases are also more prominent than the second-layer cases. This is illustrated below, with arrows reflecting prominence.

(17) **Conflated case layer hierarchy**



Such cycles naturally enforce identity of exponents. Let f be a feasibly monotonic map and suppose that $x \leq y$ and $y \leq x$. By monotonicity, this implies $f(x) \leq f(y)$ and $f(y) \leq f(x)$. But since f is feasibly monotonic, the exponents are linearly ordered. Therefore $f(x) \leq f(y)$ and $f(y) \leq f(x)$ both hold if and only if $f(x) = f(y)$. So if all second-layer and third-layer cases are part of one large cycle, any two cases x and y that are part of this cycle stand in the relation $x \leq y$ and $y \leq x$, and consequently they must be mapped to the same exponent.

From this it also follows that if nominative is syncretic with any case, it is syncretic with all of them. The one exception to this is absolutive, but since the data in McFadden (2018) does not include any languages with both a nominative and an absolutive, it remains to be seen whether this exception is undesirable.⁶ For McFadden’s sample, the solution in (17) works as desired. It allows for A^{n+1} and AB^n , and nothing else.

As I admitted earlier on, it is unclear at this point why noun stem allomorphy should use the conflated hierarchy in (17) instead of the more stratified one in (13). Like the missing AAB pattern in adjectival gradation, this issue is beyond the scope of this research project. It is one of the cases where abstracting away from the concrete mechanisms of morphology and syntax comes at the cost of leaving some aspects entirely unexplained. But as we will see next, this is exactly what makes it possible to relate morphological paradigms for adjectival gradation, person, and case to morphosyntactic well-formedness constraints.

4 PERSON CASE CONSTRAINT

Our exploration of monotonicity in language now transitions from morphology proper to a widely studied phenomenon of morphosyntax, the Person Case Constraint (PCC). The PCC is a restriction on clitic clusters. A direct object (DO) and indirect object (IO) clitic may co-occur in a cluster only if their person specifications are compatible. The “case” in PCCs thus refers to the DO-IO distinction rather than morphological case.

(18) **PCC (Spanish)** (Ormazabal and Romero 2007, p. 316f)

- a. Pedro {me, te} lo envía.
Pedro {1SG.DAT, 2SG.DAT} 3SG.M.ACC send.PRES.3SG
‘Pedro sends it to {me, you}.’
- b. * Pedro le {me, te} envía.
Pedro 3SG.M.DAT {1SG.ACC, 2SG.ACC} send.PRES.3SG
‘Pedro sends {me, you} to him.’

⁶ An anonymous reviewer, citing observations in Bobaljik (2008), points out that every known split-ergative language with nominative and absolutive case has identical stem forms for the two. If this generalization is indeed exceptionless, then (17) must also contain a loop between those two cases.

The PCC is very different from the phenomena we have considered so far. It is morphosyntactic in nature, not morphological. It does not regulate the range of available exponents, but the well-formedness of clitic combinations. But just like the phenomena from Section 3 surrounding the *ABA generalization, the PCC presents an interesting puzzle where only a fraction of all conceivable options are typologically attested. And just like for the previous phenomena, the typological gaps can be readily explained in terms of monotonic mappings from a base hierarchy to some domain of values.

In the following, I first describe the PCC typology in detail (Section 4.1) and explain how PCCs can be conceptualized as monotonic maps (Section 4.2). Section 4.3 then uses this perspective to explain why only a few PCC variants seem to occur in natural languages. In contrast to previous approaches (Anagnostopoulou 2005; Adger and Harbour 2007; Nevins 2007, a.o.), the monotonicity approach correctly predicts the existence of a recently described PCC variant in Choctaw. As shown in Section 4.4, it also generalizes straightforwardly to the Gender Case Constraint (Foley *et al.* 2017). Monotonicity thus manages to provide a unified perspective on a wide range of seemingly unrelated phenomena.

4.1 *The PCC typology*

The PCC was first observed in Perlmutter (1971), but it is only in recent years that it has attracted significant attention (see e.g. Bonet 1994; Anagnostopoulou 2005; Adger and Harbour 2007; Nevins 2007, 2011; Rezac 2007; Walkow 2012; Graf 2014, 2017). The current literature distinguishes four different types of PCC.

- (19) a. **S(trong)-PCC**
DO must be 3. (Bonet 1994)
- b. **U(ltrastrong)-PCC**
DO is less prominent than IO, where 3 is less prominent than 2, and 2 is less prominent than 1. (Nevins 2007)
- c. **W(eak)-PCC**
3IO combines only with 3DO. (Bonet 1994)
- d. **M(e first)-PCC**
If IO is 2 or 3, then DO is not 1. (Nevins 2007)

Very recent work argues for the existence of additional PCCs (Stegovec 2016; Tyler 2017), but for the sake of exposition I will limit the discussion to these four PCCs for now and return to the others later on.

Even with just four PCCs under discussion, it is hard to deny that the class of PCCs as defined above seems rather bewildering. However, a clearer picture emerges if one simply represents the licit and illicit combinations in tabular form as in Table 4. The tables make it readily apparent that each PCC represents a specific way for grammaticality to spread from the top right corner – the 1-3 combination – towards the bottom left, where 3-1 resides.

Table 4: Attested variants of the PCC

IO↓/DO→	1	2	3
1	NA	*	✓
2	*	NA	✓
3	*	*	NA

(a) S-PCC

IO↓/DO→	1	2	3
1	NA	✓	✓
2	*	NA	✓
3	*	*	NA

(b) U-PCC

IO↓/DO→	1	2	3
1	NA	✓	✓
2	✓	NA	✓
3	*	*	NA

(c) W-PCC

IO↓/DO→	1	2	3
1	NA	✓	✓
2	*	NA	✓
3	*	✓	NA

(d) M-PCC

Before we proceed, an important disclaimer is in order regarding the diagonal of each table. The cells where IO and DO have the same person specification are marked as NA, which is short for “not applicable”. The reason for this value is not a lack of available data, but rather how this data should be analyzed. These cases are known to exhibit special behavior, such as *3-3 effects* (Perlmutter 1971, p. 132).

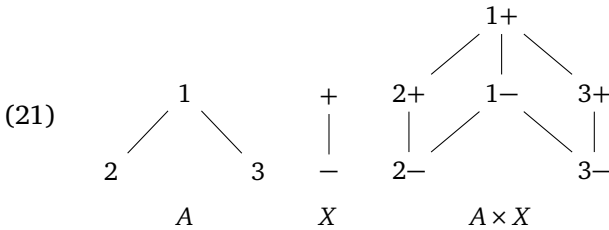
- (20) * Le lo recommandé.
 3SG.DAT 3SG.M.ACC recommend.PAST.1SG
 ‘I recommended it to him.’

Although (20) looks like a PCC-effect, it is generally assumed that the source of 3-3 effects is morphological in nature (see Foley *et al.* 2017 for a full discussion). Since PCCs are taken to be syntactic in nature, morphologically conditioned phenomena do not fall under their purview and should be treated differently. I follow this common line of reasoning and exclude all cells along the diagonal from the PCCs.

Even with the diagonal excluded, there are $2^6 = 64$ conceivable PCC variants. Only a small fraction of those 64 are actually attested. Every account of the PCC thus has to explain why the range of variation is severely limited. As we will see soon, the monotonicity approach does so with little effort.

4.2 PCCs as monotonic maps

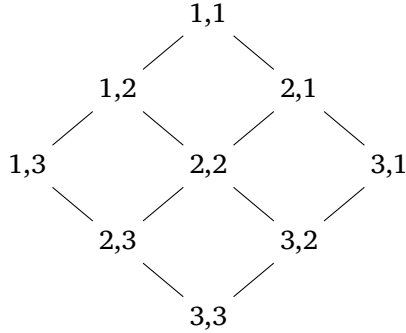
We now turn our attention to the four patterns in Table 4 and why each one of them can be regarded as the result of a monotonic map. First, we have to define an appropriate structure to represent IO/DO-combinations. We do this by constructing a specific crossproduct based on the person hierarchy $P := 1 > 2 > 3$ of Zwicky (1977), which we already encountered during the discussion of person syncretisms in Section 3.1. Given two hierarchies A and X , their crossproduct is the structure $\langle A \times X, \leq \rangle$ such that $\langle a, x \rangle \leq \langle b, y \rangle$ iff $a \leq_A b$ and $x \leq_X y$. An example is shown below.



With the right crossproduct, all the PCCs in Table 4 will turn out to be captured by monotonic maps from this crossproduct to the algebra $\mathbf{2}$ of truth values.

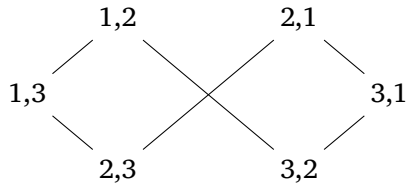
It is very tempting to go with the intuitively pleasing option to construct the crossproduct $P \times P$. This hierarchy, which is shown below, combines the person hierarchy with itself so that every node of the hierarchy represents a specific IO-DO combination.

(22) **Full person-person hierarchy** $P \times P$



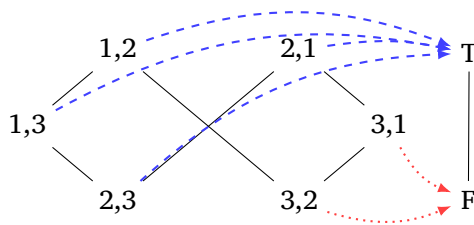
Since we ignore the diagonal, we remove all nodes of the form x, x , for $x \in \{1, 2, 3\}$. This leaves us with the reduced hierarchy \mathcal{R} below.

(23) **Reduced person-person hierarchy** \mathcal{R}



However, this is not the correct hierarchy for our purposes. The W-PCC does indeed correspond to a monotonic mapping from this hierarchy to the algebra $\mathbf{2}$, where T indicates a well-formed combination and F an ill-formed one. But the same does not hold for the U-PCC. Let us look at this in detail, starting with the W-PCC:

(24) **W-PCC as a monomorphic map**

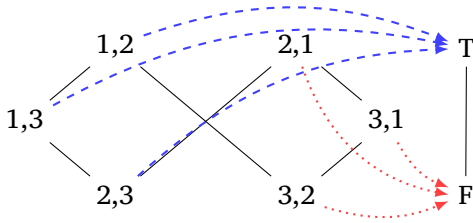


The relation between a map like in (24) and the PCC tables is as follows: if a node of the form x, y is mapped to T, then the cell in row x and column y has a checkmark. In other words, a combination of an IO with person x and a DO with person y is well-formed. The reader is

invited to verify for themselves that the mapping in (24) does indeed define the pattern of the W-PCC.

Note that (24) contains no x and y such that $x \leq_{\mathcal{R}} y$ yet $f(y) \leq_2 f(x)$, which establishes that the W-PCC mapping is monotonic. But the mapping for the U-PCC is not monotonic.

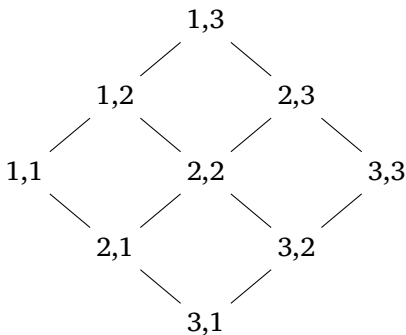
(25) **U-PCC mapping**



Here, we have $2,3 \leq_{\mathcal{R}} 2,1$ yet $f(2,1) = F \leq_2 T = f(2,3)$. Switching the order of T and F does not help, for then the problematic pair would be 1,2 and 3,2. So the U-PCC mapping isn't even feasibly monotonic for \mathcal{R} , which entails that it cannot be monotonic. If the typology of PCCs is to be analyzed as yet another instance of monotonicity, a different hierarchy is needed.

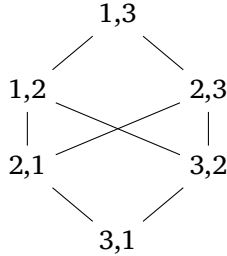
Recall from the initial discussion of the patterns in Table 4 that well-formedness seems to be growing out from the top-right corner, which corresponds to the combination 1,3. This suggests that we should construct a hierarchy where the top element is 1,3 rather than 1,1. One candidate is the crossproduct of the person hierarchy $P := 1 > 2 > 3$ with its dual $P^{-1} := 3 > 2 > 1$. The result is shown below.

(26) **Dual Person Hierarchy** $P \times P^{-1}$



Once again all elements of the form x, x are removed, which results in a new person hierarchy \mathcal{P} .

(27) **Reduced dual person hierarchy \mathcal{P} (final)**

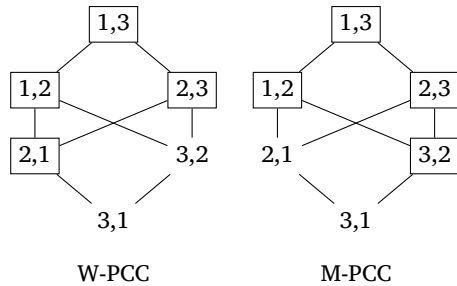
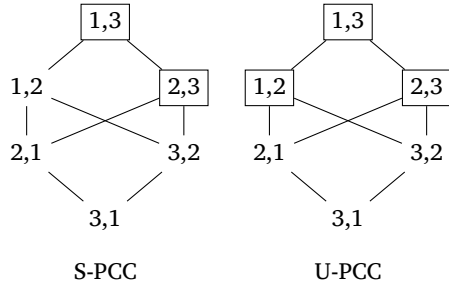


Over \mathcal{P} , each one of the four attested PCCs corresponds to a monotonic map to $\mathbf{2}$. These are depicted in Table 5. For the sake of simplicity, I omit $\mathbf{2}$ and instead put a box around a node iff it is mapped to T.

In conclusion, the attested PCC variants can all be regarded as monotonic functions from the hierarchy \mathcal{P} to the algebra $\mathbf{2}$ of truth values.

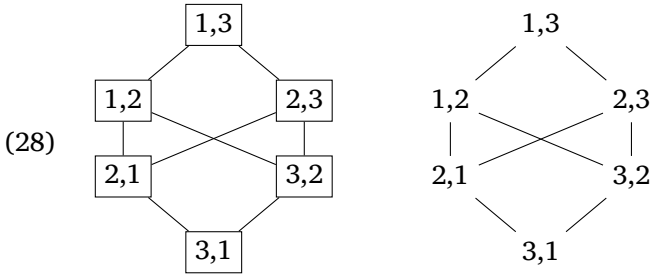
Table 5:

The four PCC variants as monotonic maps from the person hierarchy to the algebra of truth values; boxed nodes are mapped to true, all others to false



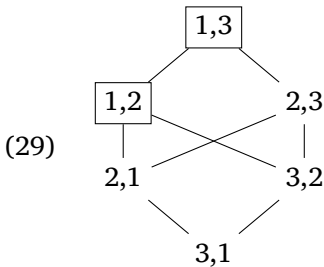
The four mappings depicted in Table 5 do not exhaust the range of available monotonic functions. So we have succeeded in correlating each attested PCC with monotonicity, but we have not given a monotonicity-based characterization of the PCC typology. But as I argue next, this too is fairly simple.

Only five other maps from \mathcal{P} to $\mathbf{2}$ are monotonic. The first two map everything to T or everything to F.



This corresponds to PCCs where either all combinations are allowed or all options are forbidden. Even though such patterns are usually not considered PCCs in the literature, they do exist. Many languages allow clitics to be freely combined, e.g. German. And at least in Cairene Arabic, clitics may never be combined, irrespective of their person specification (Shlonsky 1997; Walkow p.c.). We may consider these to be instances of a F[ree]-PCC and an I[ndiscriminate]-PCC, respectively. So these two monotonic maps do have attested counterparts in the typology.

The next mapping is a mirror image of the S-PCC, where the only licit combinations are 1,3 and 1,2 instead of 1,3 and 2,3.



For the longest time, this pattern has been believed not to exist. But Tyler (2017, p. 10) reports that Choctaw has a rather unusual PCC of the following form:

(30) **PCC in Choctaw** (as reported)

IO↓/DO→	1	2	3
1	NA	✓	✓
2	*	NA	*
3	✓	✓	NA

This pattern is clearly not monotonic over any of the hierarchies entertained so far: (22), (23), (26), and (27). However, Tyler (p.c.) states that the combinations 3, 1 and 3, 2 never surface in the data. He argues that they are blocked for reasons that are unrelated to the PCC – similar to how the diagonal often exhibits special behavior. The value ✓ in the corresponding cells thus does not indicate an empirically attested combination, but rather the theoretical claim that these patterns would be well-formed if it were not for these independent factors. We might also entertain the scenario, then, that these combinations are also illicit with respect to the PCC, which yields a very different table.

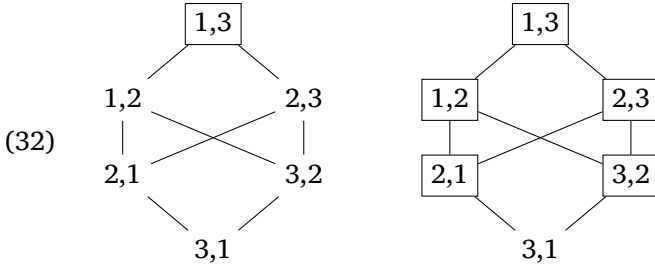
(31) **PCC in Choctaw** (reanalyzed)

IO↓/DO→	1	2	3
1	NA	✓	✓
2	*	NA	*
3	*	*	NA

Let us call this pattern the C[hoctaw]-PCC. The C-PCC corresponds exactly to the monotonic mapping in (29) where only 1,3 and 1,2 are mapped to T.

So far then, three monotonic maps beyond the initial four have been successfully related to some attested data pattern. This leaves only two more monotonic maps to consider. In one 1,3 is the only licit combination, in the other one 3,1 is the only illicit combination.

Monotonicity in morphosyntax



To the best of my knowledge, no language exhibits a pattern of this kind. So just as in the case of adjectival gradation, monotonicity is a bit too loose as a characterization of the typology.

That said, it is easy to distinguish these two unattested patterns from the rest. They are the only ones that define a class with only a single member. Either the set of well-formed combinations is a singleton, or the class of ill-formed combinations is. There may be independent reasons that drive languages towards patterns that do not put a single combination in opposition to the rest of the paradigm. Or perhaps this is yet another case where the answer can only be found at a less abstracted level of description.

Be that as it may, monotonicity in combination with the hierarchy \mathcal{P} from (27) provides a very tight fit for the attested PCC typology, with only minimal overgeneration. This establishes a direct connection to the morphological syncretism phenomena discussed in Section 3. But as we will see next, it can also be extended to other aspects of morphosyntax, such as the recently reported Gender Case Constraint.

4.4 *Hierarchical reversal in the Gender Case Constraint*

Besides furnishing two unattested patterns, the monotonicity account also seems stipulative in that it fails for the intuitively most pleasing hierarchy $P \times P$ and instead has to use (a reduced version of) $P \times P^{-1}$, where the order in the second component is the reverse of that in the first component. Why should natural language operate with such a peculiar hierarchy? I have no insightful answer to this puzzle, but I would like to point out that the puzzle is not limited to the PCC.

Foley *et al.* (2017) report that Zapotec displays a restriction on subject-object clitic clusters that is driven by gender rather than person, a *Gender Case Constraint* (GCC). This constraint only acts on third

person clitics. Zapotec distinguishes four genders: elder human, non-elder human, animal, and inanimate. For the sake of simplicity, I refer to these as 1, 2, 3, and 4. The Zapotec GCC then produces the following pattern:

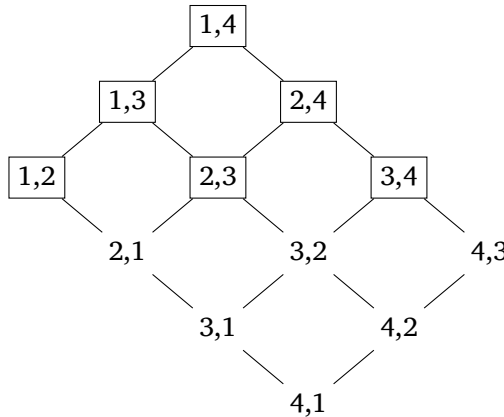
(33) **Zapotec GCC** (Foley *et al.* 2017, p. 6)

S↓/O→	1	2	3	4
1	NA	✓	✓	✓
2	*	NA	✓	✓
3	*	*	NA	✓
4	*	*	*	NA

Mirroring standard practice for the PCC, Foley *et al.* (2017) argue that the diagonal can be subject to a separate constraint against identical combinations. For this reason, I have given those cells the value NA here, but nothing in the subsequent discussion hinges on that.

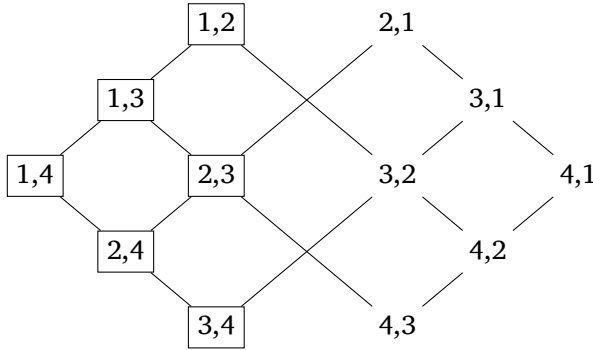
After our extensive discussion of monotonicity in the PCC, the reader should be able to see immediately that the pattern in (33) is monotonic if one starts with a hierarchy $1 > 2 > 3 > 4$ and combines it with its inverse $4 > 3 > 2 > 1$.

(34) **Gender hierarchy with the second component reversed**



If, on the other hand, the hierarchy were simply built by combining $1 > 2 > 3 > 4$ with itself, the Zapotec GCC would not be monotonic by virtue of not being feasiably monotonic.

(35) Gender hierarchy with identical components



Here we have $3,4 \leq 3,2 \leq 1,2$ and thus $f(3,4) = T \leq f(3,2) = F \leq f(1,2) = T$ – but $T \leq F \leq T$ cannot be satisfied irrespective of how one orders T and F. It seems, then, that it is a general fact of language that clitic combination patterns “grow” from the top-right corner of the paradigm, which is formally captured by reversing the second hierarchy. I doubt that the reason for this can be discerned at the level of abstraction at which the monotonicity approach operates. This question requires the more fine-grained and detailed assumptions of syntactic and/or morphological formalisms.

4.5 *Other PCCs*

There is one more point regarding the typology of PCCs that merits discussion. Stegovec (2016) argues based on data from Slovenian that the established PCCs also have counterparts where the relevant contrast is not that between IO and DO, but rather which clitic occurs linearly first. For example, Slovenian has a counterpart of the S-PCC that looks exactly the same as the one in Table 4 except that the x and y -axis do not correspond to DO and IO, but rather to the linearly second and the linearly first clitic.

Stegovec’s findings present a major challenge for syntactic approaches such as Anagnostopoulou (2005), which derive the limited PCC typology from structural asymmetries between IO and DO. Without this structural asymmetry, syntactic accounts lose all their force. In particular, it becomes mysterious why the range of possible PCC patterns remains the same when the conditioning factor is linear order instead of the IO-DO asymmetry.

The monotonicity approach, on the other hand, can easily accommodate these findings. All the work is done by the hierarchy in (27), which is agnostic about what each component of a node encodes. The standard interpretation of, say, 3,1 is that IO is third person and DO is first person. But we might just as well interpret it as saying that the linearly first clitic is third person and the second one is first person. The monotonicity approach is not concerned with identifying potential triggers or the exact linking from syntactic configurations to the hierarchies it operates over. What it does is provide an abstract characterization of the range of variation once the appropriate triggers have been identified. As we have seen throughout this paper, this high-level approach has great unifying power, but it comes at the expense of leaving certain issues entirely unaddressed. Stegovec's findings show that this kind of agnosticism, albeit occasionally unsatisfying, increases the robustness of the approach.

5

METHODOLOGICAL REMARKS

Before I turn to the summary of this paper's key findings, there still are several methodological issues that deserve a careful exploration. These concern the status of monotonicity as a desirable property of mappings (Section 5.1), the cognitive status of the analysis advanced in this paper (Section 5.2), and the risks of building a formal model on a limited range of data (Section 5.3).

5.1

Monotonicity in language

Monotonicity plays a central role in this paper. It is the key ingredient that narrows down the range of variation once a suitable base hierarchy has been defined. One may wonder, then, why monotonicity should be a desirable property for language. This is a very deep question that cannot be answered in a few lines. That said, the role of monotonicity seems to extend beyond the phenomena discussed in the preceding sections.

In Section 2.1, I used the *No Crossing Branches* constraint from autosegmental phonology to illustrate the concept of monotonicity. But monotonicity in phonology goes beyond this constraint. To give but one example, there seems to be no phonological process that

targets high and low vowels to the exclusion of mid vowels – a kind of phonological *ABA constraint, and arguably an instance of monotonicity.

Monotonicity can also be found in syntax. The Accessibility hierarchy of Keenan and Comrie (1977) classifies different kinds of NP by their relative prominence as $SU > DO > IO > OBL > GEN > OCOMP$ and states that if a language allows for NPs of type x to be relativized, it also allows this for every NP type $y > x$. This implicational universal amounts to the requirement that mappings from the linear hierarchy of NP-types to the algebra $\mathbf{2}$ must be monotonic (being mapped to T means that relativization is allowed).

Another example comes from the ordering of operations in syntax. A phrase can undergo three types of operations: selection, A-move, and A'-move. Once a phrase has undergone A-movement, it can no longer be selected or select arguments of its own. Furthermore, A-movement is impossible once the phrase has undergone A'-movement. This, too, can be viewed as an instance of monotonicity. For any given phrase, consider the linear sequence of operations it undergoes during the syntactic derivation. For example, an arbitrary DP's record may read *Select-A-A'-A'*. Now suppose furthermore that operation types are linearly ordered such that *Select* < *A-move* < *A'-move*. Then the inability to select after A-moving or to A-move after A'-moving follows from the requirement that the mapping from a phrase's derivational record to the hierarchy of operation types must be monotonic.

A more complex example from syntax is the analysis of adjunct island effects in Graf (2013). There, the fact that extraction from an adjunct is ungrammatical is reinterpreted as a monotonicity requirement over a specific kind of algebra. This monotonicity entailment can produce situations where a syntactic structure is illicit even though it does not violate any syntactic constraints. Hence, there is no such thing as an Adjunct Island Constraint, the observed effects are a direct consequence of monotonicity.

Monotonicity also surfaces in semantics. The denotations of determiners, for example, are always monotonic (Keenan and Westerståhl 1996; Peters and Westerståhl 2006). Even in the realm of lexical semantics, it has been argued that word meanings tend to be convex (Gärdenfors 2000; Jäger 2007, 2010), a notion that is closely related to monotonicity.

Overall, then, there is plenty of evidence for monotonicity in language, although the motivations for that are still largely unclear. This paper just adds a few more entries to a long list of phenomena that involve monotonicity.

5.2 *Cognitive commitment*

The previous section suggests that monotonicity plays a fundamental role in language. This seems to be at odds with the initially stated aim for an effective theory, i.e. an account that correctly characterizes the system under investigation but does not necessarily encompass the causal factors that give rise to this system. Upon reflection, though, these two positions are perfectly compatible.

It is true that the monotonicity account deliberately abstracts away from those factors that linguists consider the nuts and bolts of mental grammars: features bundles, feature checking, structure-building operations, and so on. As a consequence, the concepts I rely on may not have direct counterparts in the grammar. For example, the posited person hierarchy of $1 > 2 > 3$ is entirely agnostic about the long-standing issue whether third person is a feature or the absence of person features. At the same time, this does not entail that the person hierarchy is merely a descriptive device without any cognitive reality. Rather, the claim is that the mechanisms of the grammar, whatever they may be, are such that they give rise to this kind of hierarchy for the surface forms we observe in the data. In a sense, this is no different from syntacticians asserting the cognitive reality of their grammar formalism while leaving the neural substrate of the grammar unspecified. The monotonicity approach employs the same strategy, characterizing the high-level behavior of language while abstracting away from the grammar substrate.

As the reader has seen throughout the paper, this has several advantages. One can now generalize across domains that arguably do not look very similar and behave very differently at the usual level of grammatical description. By abstracting away from technical details, the account remains remarkably simple on a formal level. It also is largely framework agnostic and is directly compatible with Minimalism (Chomsky 1995), Distributed Morphology (Halle and Marantz 1993; Embick and Marantz 2008), GPSG (Gazdar *et al.* 1985), HPSG (Pollard and Sag 1994), LFG (Bresnan 1982), and TAG (Joshi 1985),

among others. By not directly linking into the tools and concepts of existing grammar formalisms, the account also enjoys a greater amount of freedom and can easily be adjusted to fit new data – the discovery of new PCC types in Stegovec (2016), for example, poses a major problem for syntactic accounts, but not for the monotonicity approach.

The obvious downside of abstraction is that some typological gaps cannot be adequately explained. Without the connection to the grammar substrate, domain-specific limitations such as the absence of AAB patterns in adjectival gradation remain mysterious. There is a risk that this abstractness, combined with the malleability of the approach, will ultimately lead to blind descriptivism, with the hierarchies constantly being tweaked and refined until they fit the data. As I argue next, though, the goals of the enterprise make this an unappealing option and hence an unlikely outcome.

5.3

The risk of overfitting

Typological accounts always face the danger of overfitting their theory to an unrepresentative data sample. Even large-scale studies rarely contain data from more than 150 languages. Since many phenomena such as the PCC are exceedingly rare, the sample of languages that display the phenomenon in question is even smaller. At the same time, combinatorial explosion leads to large numbers of logically possible systems in certain domains. For instance, there are 203 distinct case syncretism patterns for a system with 6 cases. In the face of such numbers, it is doubtful that our current data exhausts the full range of variation. In addition, the analysis of existing data is fraught with difficulties. Syncretism, for example, has to be distinguished from merely accidental homophony, which leaves plenty of wiggle room in how the data is interpreted (but see Sauerland and Bobaljik 2013 for a more rigorous approach to accidental homophony). Even in cases where sufficient data is available, then, it might have been misanalyzed.

While all these points are certainly correct, they are unavoidable given the realities of doing empirical work in linguistics. All competence data is heavily theory-laden, and since we do not know the full extent of universal grammar, even 6000 languages may only provide a small sample of the full space of grammars. Instead of stopping dead in their tracks, linguists accept that the empirical landscape may change

from one day to the next and try to formulate the most insightful theories given the currently available data.

Things are no different for the monotonicity approach, but it has several properties that mitigate the data problem. First and foremost, the account is about identifying principles that hold across many different domains. Hence, a data shortage in one domain is less of an issue because it can be offset by insights from another domain. Person syncretism and the PCC, for instance, mutually support each other as they both operate over person hierarchies. In addition, the monotonicity approach is robust in the sense that it does not try to perfectly fit the existing data but rather identifies upper bounds on the range of variation. For example, the ABA pattern is ruled out on systematic grounds, whereas the absence of AAB patterns in adjectival gradation has to be stipulated. By focusing on what holds across many domains, the approach avoids overfitting the data for any given domain.

How well this enterprise will work out in practice remains to be seen. But at this point, there is no reason to dismiss it on conceptual or methodological grounds.

The account proposed in this paper derives typological gaps from two components: a fixed underlying hierarchy shared across all languages (a person hierarchy, case hierarchy, and so on), and the requirement that the mappings from these hierarchies to output forms must be monotonic. This simple principle produces close approximations of the range of variation for each domain, with only some requiring further stipulations (e.g. the ban against AAB pattern in adjectival gradation, or the absence of PCC patterns with only one well-formed or ill-formed clitic combination). The relevant hierarchies are summarized in Table 6.

While it must remain an open question for now why monotonicity should play such an important role, it cannot be denied that it surfaces in many other areas of language, including phonology, morphology, syntax, and semantics. Hopefully future work will be able to shed light on this fundamental question.

A lot of analytic work also remains to be done. The literature on typological generalizations is enormous, and only a few could be

Monotonicity in morphosyntax

Phenomenon	Hierarchy
Adjectival gradation	<pre> positive comparative superlative </pre>
Pronoun syncretism	<pre> 1 2 3 </pre>
Case syncretism	<pre> Nom Abs Acc Gen Loc Erg +---+---+---+ Dat Inst </pre>
Noun stem allomorphy	<pre> Nom Abs Acc Gen Loc Erg +---+---+---+ Dat Inst </pre>
PCC	<pre> 1,3 / \ 1,2 2,3 2,1 3,2 \ / 3,1 </pre>
GCC	<pre> 1,4 / \ 1,3 2,4 / \ 1,2 2,3 3,4 \ / \ 2,1 3,2 4,3 \ / \ 3,1 4,2 \ / 4,1 </pre>

Table 6:
Overview of posited
hierarchies for each
phenomenon

explored here. Number was completely ignored, and preliminary work on syncretism patterns in verbal inflection suggests that its behavior is more complicated than that of person in the PCC. Other topics for future research are inverse marking and resolved agreement.

Expanding the range of domains is of vital importance as it may provide additional support for hierarchies posited here. For example, the person hierarchy $1 > 2 > 3$ should surface in every domain that involves person. In addition, hierarchies for entirely new domains will increase our understanding of what hierarchies can (and cannot) look like. This is vital for keeping the approach from devolving into pure stipulation of suitable hierarchies.

ACKNOWLEDGMENTS

During their 2-year gestation period, the ideas discussed in this paper have benefited tremendously from detailed feedback and informal discussions with numerous colleagues. I would like to thank David Adger, Mark Aronoff, Jonathan Bobaljik, Pavel Caha, Alex Clark, Sophie Moradi, Omer Preminger, Uli Sauerland, Matthew Tyler, and Stanislaw Zompí, as well as the anonymous reviewers for JLM and MOL.

REFERENCES

- David ADGER and Daniel HARBOUR (2007), Syntax and Syncretisms of the Person Case Constraint, *Syntax*, 10:2–37, doi:10.1111/j.1467-9612.2007.00095.x.
- Elena ANAGNOSTOPOULOU (2005), Strong and Weak Person Restrictions: A Feature Checking Analysis, in Lorie HEGGIE and Francisco ORDOÑEZ, editors, *Clitics and Affix Combinations: Theoretical Perspectives*, pp. 199–235, John Benjamins, Amsterdam.
- Barry J. BLAKE (2001), *Case*, Cambridge University Press, Cambridge.
- Jonathan D. BOBALJIK (2008), Where's ϕ ? Agreement as a Post-Syntactic Operation, in David ADGER, Daniel HARBOUR, and Susana BEJAR, editors, *Phi Theory: Phi-Features Across Modules and Interfaces*, pp. 295–328, Oxford University Press, Oxford.
- Jonathan D. BOBALJIK (2012), *Universals in Comparative Morphology: Suppletion, Superlatives, and the Structure of Words*, MIT Press, Cambridge, MA.

Monotonicity in morphosyntax

- Jonathan D. BOBALJIK and Uli SAUERLAND (2018), ABA and the Combinatorics of Morphological Features, *Glossa: A Journal of General Linguistics*, 3, doi:10.5334/gjgl.345, article 15.
- Eulàlia BONET (1994), The Person-Case Constraint: A Morphological Approach, in *The Morphology-Syntax Connection*, number 22 in MIT Working Papers in Linguistics, pp. 33–52.
- Joan BRESNAN (1982), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA.
- Pavel CAHA (2009), *The Nanosyntax of Case*, Ph.D. thesis, University of Tromsø.
- Pavel CAHA (2013), Explaining the Structure of Case Paradigms by the Mechanisms of Nanosyntax: The Classical Armenian Nominal Declension, *Natural Language and Linguistic Theory*, 31:1015–1066, doi:10.1007/s11049-013-9206-8.
- Noam CHOMSKY (1995), *The Minimalist Program*, MIT Press, Cambridge, MA, doi:10.7551/mitpress/9780262527347.003.0003.
- David EMBICK and Alec MARANTZ (2008), Architecture and Blocking, *Linguistic Inquiry*, 39:1–53.
- Steven FOLEY, Nick KALIVODA, and Maziar TOOSARVANDANI (2017), Forbidden Clitic Clusters in Zapotec: Implications for the Person-Case Constraint, handout of a talk presented at CLS, May 25 2017, University of Chicago.
- Gerald GAZDAR, Ewan KLEIN, Geoffrey K. PULLUM, and Ivan A. SAG (1985), *Generalized Phrase Structure Grammar*, Blackwell, Oxford.
- John GOLDSMITH (1976), *Autosegmental Phonology*, Ph.D. thesis, MIT.
- Thomas GRAF (2013), The Syntactic Algebra of Adjuncts, in *Proceedings of CLS 49*, <http://thomasgraf.net/doc/papers/Graf13CLS.pdf>, to appear.
- Thomas GRAF (2014), Feature Geometry and the Person Case Constraint: An Algebraic Link, in *Proceedings of CLS 50*, <http://thomasgraf.net/doc/papers/Graf14CLS.pdf>, to appear.
- Thomas GRAF (2017), Graph Transductions and Typological Gaps in Morphological Paradigms, in *Proceedings of the 15th Meeting on the Mathematics of Language (MOL 2017)*, pp. 114–126, <http://www.aclweb.org/anthology/W17-3411>.
- Peter GÄRDENFORS (2000), *Conceptual Spaces: The Geometry of Thought*, MIT Press, Cambridge, MA.
- Morris HALLE and Alec MARANTZ (1993), Distributed Morphology and the Pieces of Inflection, in Ken HALE and Samuel J. KEYSER, editors, *The view from building 20*, pp. 111–176, MIT Press, Cambridge, MA.

- Daniel HARBOUR (2015), Poor Pronoun Systems and What They Teach us, *Nordlyd*, 41(1), doi:10.7557/12.3314.
- Heidi HARLEY and Elizabeth RITTER (2002), Person and Number in Pronouns: A Feature-Geometric Analysis, *Language*, 78:482–526, doi:10.1353/lan.2002.0158.
- Gísli Rúnar HARÐARSON (2016), A Case for a Weak Case Contiguity Hypothesis — A Reply to Caha, *Natural Language and Linguistic Theory*, 34:1329–1343, doi:10.1007/s11049-016-9328-x.
- Aravind JOSHI (1985), Tree-Adjoining Grammars: How Much Context Sensitivity is Required to Provide Reasonable Structural Descriptions?, in David DOWTY, Lauri KARTTUNEN, and Arnold ZWICKY, editors, *Natural Language Parsing*, pp. 206–250, Cambridge University Press, Cambridge.
- Gerhard JÄGER (2007), The Evolution of Convex Categories, *Linguistics and Philosophy*, 30:551–564.
- Gerhard JÄGER (2010), Natural Color Categories Are Convex Sets, in Maria ALONI, Harald BASTIAANSE, Tikitu DE JAGER, and Katrin SCHULZ, editors, *Logic, Language and Meaning*, pp. 11–20, Springer, Berlin, Heidelberg, doi:10.1007/978-3-642-14287-1_2.
- Edward KEENAN and Dag WESTERSTÅHL (1996), Generalized Quantifiers in Linguistics and Logic, in Johan VAN BENTHEM and Alice TER MEULEN, editors, *Handbook of Logic and Language*, pp. 837–894, North-Holland, Amsterdam.
- Edward L. KEENAN and Bernard COMRIE (1977), Noun Phrase Accessibility and Universal Grammar, *Linguistic Inquiry*, 8:63–99.
- Thomas MCFADDEN (2018), *ABA in Stem-Allomorphy and the Emptiness of the Nominative, *Glossa: A Journal of General Linguistics*, 3, article 8.
- Andrew NEVINS (2007), The Representation of Third Person and Its Consequences for Person-Case Effects, *Natural Language and Linguistic Theory*, 25:273–313, doi:10.1007/s11049-006-9017-2.
- Andrew NEVINS (2011), Multiple Agree with Clitics: Person Complementarity vs. Omnivorous Number, *Natural Language and Linguistic Theory*, 28:939–971, doi:10.1007/s11049-006-9017-2.
- Javier ORMAZABAL and Juan ROMERO (2007), The Object Agreement Constraint, *Natural Language and Linguistic Theory*, 25:315–347.
- David M. PERLMUTTER (1971), *Deep and Surface Structure Constraints in Syntax*, Holt, Rinehart and Winston, New York.
- Stanley PETERS and Dag WESTERSTÅHL (2006), *Quantifiers in Language and Logic*, Oxford University Press.
- Carl POLLARD and Ivan SAG (1994), *Head-Driven Phrase Structure Grammar*, CSLI and The University of Chicago Press, Stanford and Chicago.

Monotonicity in morphosyntax

Milan REZAC (2007), Escaping the Person Case Constraint: Reference-Set Computation in the ϕ -system, *Linguistic Variation Yearbook*, 6:97–138.

Uli SAUERLAND and Jonathan D. BOBALJIK (2013), Syncretism Distribution Modeling: Accidental Homophony as a Random Event, in Nobo GOTO, Koichi OTAKI, Atsusi SATO, and Kensuke TAKITA, editors, *Proceedings of GLOW in Asia 9*, pp. 31–53.

Ur SHLONSKY (1997), *Clause Structure and Word Order in Hebrew and Arabic*, Oxford University Press, Oxford.

Peter W. SMITH, Beata MOSKAL, Ting XU, Jungmin KANG, and Jonathan D. BOBALJIK (2018), Case and Number Suppletion in Pronouns, *Natural Language & Linguistic Theory*, doi:10.1007/s11049-018-9425-0.

Adrian STEGOVEC (2016), Personality Disorders and Missing Persons: Deriving the Person-Case Constraint without Case, <http://ling.auf.net/lingbuzz/002632>, ms., University of Connecticut.

Matthew TYLER (2017), Choctaw PCC Repair: Basque-Style PCC Repair in a Language with no Dative, handout of a talk presented at CLS, May 25 2017, University of Chicago.

Martin WALKOW (2012), *Goals, Big and Small*, Ph.D. thesis, University of Massachusetts Amherst.

Stanislao ZOMPÍ (2016), *Case Decomposition Meets Dependent-Case Theories. A Study of the Syntax-Morphology Interface*, Master's thesis, University of Pisa.

Arnold ZWICKY (1977), Hierarchies of Person, in *Chicago Linguistic Society*, volume 13, pp. 714–733.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



Finite-state Optimality Theory: non-rationality of Harmonic Serialism

Yiding Hao

Department of Linguistics
Department of Computer Science
Yale University
New Haven, Connecticut, USA

ABSTRACT

This paper analyzes the language-theoretic complexity of *Harmonic Serialism* (HS), a derivational variant of Optimality Theory. I show that HS can generate non-rational relations using *strictly local* markedness constraints, proving the “result” of Hao (2017), that HS is rational under those assumptions, to be incorrect. This is possible because deletions performed in a particular order have the ability to enforce nesting dependencies over long distances. I argue that coordinated deletions form a canonical characterization of non-rational relations definable in HS.

Keywords:
optimality theory,
harmonic serial-
ism, phonology,
finite-state,
strictly local,
subregular

1

INTRODUCTION

A classical question in mathematical linguistics concerns whether or not patterns describable by grammar formalisms resemble natural language dependencies (Chomsky 1959; Berwick 1984). An ideal grammar formalism should be expressive enough to generate all attested languages, but also restrictive enough to exclude patterns thought to be impossible in natural language.

In phonology, the seminal work of Johnson (1970, 1972) and Kaplan and Kay (1994) showed that the Sound Pattern of English (SPE) formalism of Chomsky and Halle (1968) is equivalent in generative

power to *finite-state transducers* (FSTs). Henceforth, it has been generally accepted that the mapping from underlying phonological representations (URs) to surface phonetic representations (SRs) in a given language can be computed using an FST. Empirical evidence in favor of this hypothesis can be found in the SPE phonology literature, as well as in the success of hidden-Markov-model-based approaches to automatic speech recognition (Baker 1975; Lowerre 1976; Jelinek 1976).

If it is empirically true that phonological mappings are finite-state, then theoretical frameworks for phonology should ideally describe only mappings that are finite-state. Frank and Satta (1998) carry out an analysis of Optimality Theory (OT, Prince and Smolensky 1993, 2004), showing that the full OT framework can describe non-rational relations, and is therefore too powerful according to the criterion of finite-stateness. They do this by following Ellison (1994) in thinking of OT constraints as FSTs that read input–output pairs and emit violation marks. However, Frank and Satta also find that OT can be made equivalent to FSTs by assuming that for each constraint there is an upper bound on the number of violation marks that the constraint may assign to any given input–output pair. Accordingly, Karttunen (1998) has developed a finite-state calculus for implementing this *violation-bounded* version of OT.

This paper presents an analysis of *Harmonic Serialism* (HS), a variant of OT in which surface forms are computed by recursively applying incremental changes to underlying forms. These incremental changes are chosen from a collection of basic operations based on an OT-style constraint ranking system. Previous work on the expressive power of HS includes Lamont (2018a,b), who shows that HS can implement bounded alphabetical sorting if constraints are *strictly piecewise* and if the basic operations include metathesis. Hao (2017), on the other hand, arrives at the main conclusion that if markedness constraints are strictly local and if the basic operations only include insertion, deletion, and substitution of a single symbol, then HS only produces rational relation. The present paper disproves the latter “result” by constructing an HS grammar that produces a non-rational relation while fulfilling Hao’s assumptions.

The structure of this paper is as follows. Section 2 defines terminology and notation used in the rest of this paper. Section 3 introduces

the formalism of OT and reviews existing literature in finite-state OT. Section 4 defines HS along with various kinds of markedness constraints. In Section 5, I show that the model from Section 4 can produce a non-rational relation by relying on carefully coordinated deletions. Section 6 argues that all non-rational mappings in HS are implemented in a manner similar to the non-rational mapping from Section 5. Section 7 concludes.

2

PRELIMINARIES

Let us adopt the following standard notations and definitions. \mathbb{Z} is the set of integers, and $\mathbb{N} \subsetneq \mathbb{Z}$ is the set of non-negative integers. Unless otherwise specified, the letters Σ , Δ , and Γ denote finite alphabets not including the special symbols \bowtie and \bowtie . When used, these special symbols represent the left and right boundaries of a string, respectively.

The length of a string x is denoted by $|x|$, and λ denotes the empty string, the unique string of length 0. Alphabet symbols are identified with strings of length 1, Σ^k denotes the set of strings of length k , $\Sigma^{\leq k}$ denotes the set of strings of length at most k , Σ^* denotes the set of all strings over Σ , and Σ^+ denotes the set $\Sigma^* \setminus \{\lambda\}$. For strings a and b , ab denotes the concatenation of a and b . This notation is extended to sets of strings A and B in the usual way. We say that a is a *substring* of b if there exist strings l and r such that $b = lar$.

For sets A and B , $A \times B$ is the Cartesian product of A and B . An *n -ary relation over A_1, A_2, \dots, A_n* is a subset $R \subseteq A_1 \times A_2 \times \dots \times A_n$. If $\langle a_1, a_2, \dots, a_n \rangle \in R$, then we may write $R(a_1, a_2, \dots, a_n)$. If $n = 2$, then we may also write $a_1 R a_2$. For any sets A , B , and C , the *composition* of a relation $R \subseteq B \times C$ with a relation $S \subseteq A \times B$ is the unique binary relation $R \circ S \subseteq A \times C$ such that $a R \circ S c$ if and only if there exists b such that $a S b$ and $b R c$. The *transitive closure* of a relation R is defined as

$$R^+ := \bigcup_{i=1}^{\infty} \underbrace{R \circ R \circ \dots \circ R}_{i\text{-many times}}$$

An *equivalence relation over A* is a relation $R \subseteq A \times A$ satisfying the following properties.

- For all $a \in A$, $a R a$ (i.e., R is *reflexive*).
- For all $a, b \in A$, if $a R b$, then $b R a$ (i.e., R is *symmetric*).

- For all $a, b, c \in A$, if $a R b$ and $b R c$, then $a R c$ (i.e., R is *transitive*).

For each $a \in A$, the *equivalence class of a with respect to R* is the set $[a]_R := \{b \mid b \in A, b R a\}$. The *quotient of A under R* is the set $A/R := \{[a]_R \mid a \in A\}$. Each element of A/R is called an *equivalence class*.

A *finite-state transducer* (FST) is a 6-tuple $T = \langle Q, \Sigma, \Gamma, I, F, \rightarrow \rangle$, where

- Q is a finite set of *states*;
- Σ is an alphabet called the *input alphabet*;
- Γ is an alphabet called the *output alphabet*;
- $I \subseteq Q$ is the set of *initial states*;
- $F \subseteq Q$ is the set of *final states*; and
- $\rightarrow \subseteq Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q$ is the *transition relation*.

We assume without loss of generality that if $\rightarrow(q, x, y, r)$, then $xy \neq \lambda$. The *extended transition relation* is denoted by \rightarrow_* . The notations $q \xrightarrow{x:y} r$ and $q \xrightarrow{x:y}_* r$ denote $\rightarrow(q, x, y, r)$ and $\rightarrow_*(q, x, y, r)$, respectively. The *behavior* of an FST T is the relation $[T]$ defined by $x [T] y$ if and only if T has an initial state q and a final state r such that $q \xrightarrow{x:y}_* r$. A relation is *rational* if it is the behavior of an FST.

A *finite-state automaton* (FSA) is a 5-tuple $A = \langle Q, \Sigma, I, F, \rightarrow \rangle$, where

- Q is a finite set of *states*;
- Σ is an alphabet called the *input alphabet*;
- $I \subseteq Q$ is the set of *initial states*;
- $F \subseteq Q$ is the set of *final states*; and
- $\rightarrow \subseteq Q \times (\Sigma \cup \{\lambda\}) \times Q$ is the *transition relation*.

The *extended transition relation* is denoted by \rightarrow_* . The notations $q \xrightarrow{x} r$ and $q \xrightarrow{x}_* r$ denote $\rightarrow(q, x, r)$ and $\rightarrow_*(q, x, r)$, respectively. We say that A *accepts* a string w if and only if A has an initial state q and a final state r such that $q \xrightarrow{w}_* r$. The *language recognized by A* is the set of all strings accepted by A . A language is *regular* if it is recognized by an FSA.

Optimality Theory (OT, Prince and Smolensky 1993, 2004) is a formalism that defines mappings between URs and SRs using ranked, violable

constraints. An OT grammar is standardly considered to be given by three components. Firstly, the function GEN takes an input and returns a set of *candidates*. Then, the function EVAL chooses one or more of these candidates to be the output of the grammar. The SR of a word is assumed to be the output of the grammar when given the UR as input. The computation performed by EVAL is parameterized by the input and by CON, a set of *constraints* that must be satisfied by input–output pairs. These constraints typically contradict one another, so EVAL specifies a *ranking* over CON that determines which constraints are prioritized over others.¹ The relationships among these three components are shown visually in Figure 1.

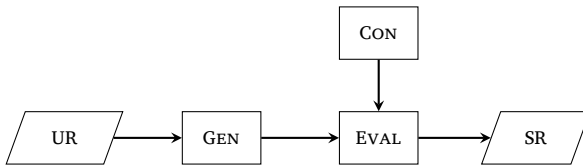


Figure 1:
The three components
of standard OT

To illustrate, let us consider a concrete example. (1) shows three words from the Māori language. If a UR ends with a consonant, the SR is produced by deleting this consonant.

- (1) Coda Deletion in Māori (Hohepa 1967; Hale 1973)
- a. /hopuk/ \rightsquigarrow [hopu] “catch”
 - b. /arum/ \rightsquigarrow [aru] “follow”
 - c. /maur/ \rightsquigarrow [mau] “carry”

A typical OT implementation of this mapping is as follows. GEN takes a UR as input, and produces as candidates all possible strings that may be obtained by inserting symbols to the input, deleting symbols from the input, or changing symbols from the input to other symbols. From among these possibilities, EVAL chooses an output based on the following constraints from CON. This output is taken to be the SR.

¹ Other formalisms with violable constraints may feature other methods for adjudicating between constraints. For example, *Harmonic Grammar* (Pater 2009; Potts *et al.* 2010) features *weighted constraints* that contribute additively to the computation of SRs from URs. Ranked constraints give rise to explanations of linguistic universals based on *factorial typology* (see Prince and Smolensky 1993, 2004), while weighted constraints account for *gang effects* (see Pater 2009).

- (2)
- a. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.
 - b. DEP: Assign one violation for each symbol inserted into the input.
 - c. NOCODA: Assign one violation if the input ends with a consonant.²
 - d. MAX: Assign one violation for each symbol deleted from the input.

Each constraint assigns to each candidate a score, expressed in units called *violations*, that measures the degree to which the candidate violates the constraint. While several possible rankings of the four constraints above may yield the mapping shown in (1), for simplicity let us assume that the four constraints are ranked in the order shown, with higher-ranking constraints taking priority over lower-ranking ones. We denote the constraint ranking by

ID \gg DEP \gg NOCODA \gg MAX.

Now, let us consider an input ending with a consonant. Based on this constraint ranking, the output cannot be produced by performing substitutions or insertions, as candidates produced in this manner violate ID and DEP. The output also cannot end with a consonant, lest it violate NOCODA. Deleting the final consonant violates MAX, but this is tolerated because MAX is the lowest-ranked constraint. All candidates produced by GEN without deletion either end with a consonant, thus violating NOCODA, or avoid a NOCODA violation by inserting or changing symbols, thus violating ID or DEP. Constraints may be violated to varying degrees; thus, deleting additional symbols beyond the final consonant is not possible, because such candidates violate MAX more severely than the candidate that only deletes the final consonant.

The computation of an output is shown in a table called a *tableau*.³ An example of a tableau is shown in (3). The columns represent the

²Typically, NOCODA assigns a violation for each *syllable* ending with a consonant (Prince and Smolensky 1993, 2004). For simplicity, syllabification is not discussed here, hence this alternate statement of NOCODA.

³For simplicity, only *violation tableaux* are used in this paper. *Comparative tableaux* (Prince 2002, 2003) are also commonly used in OT phonology.

constraints, from highest-ranked to lowest-ranked. The rows represent a selection of candidates produced by GEN. Typically, the candidates shown in a tableau are those used to illustrate or justify claims about the behavior of the grammar, such as the effect of changing the constraint ranking. Each cell shows the number of violations the constraint assigns to the candidate.⁴ The candidate that is identical to the input is known as the *faithful* candidate. The candidate that is chosen as the output, marked with the symbol ☞ , is known as the *winning* candidate, or *winner*. For a non-winning candidate, the cell associated with the highest-ranking constraint that distinguishes the candidate from the winner is annotated with the symbol !.

(3) /hopuk/ \rightsquigarrow [hopu]

hopuk	ID	DEP	NOCODA	MAX
a. hopuk	0	0	1!	0
☞ b. hopu	0	0	0	1
c. hopuku	0	1!	0	0
d. hopuu	1!	0	0	0
e. ho	0	0	0	3!

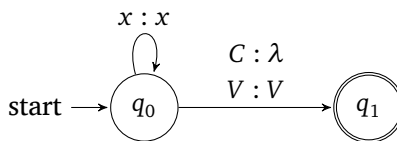
In (3), we consider the tableau of the grammar described by (2) for the UR /hopuk/. The faithful candidate violates NOCODA. Candidates c and d satisfy NOCODA, but violate DEP and ID, respectively. Candidates b and e, obtained by deleting the final consonant, violate only the lowest-ranked constraint MAX. However, because candidate e violates MAX thrice while candidate b violates MAX only once, candidate b is the winner.

3.1 Finite-State Optimality Theory

Māori coda deletion, as described here, is implemented by the FST in Figure 2. Recall that an FST is a device that reads an input string from left to right and produces an output while doing so. Throughout the course of its computation, the FST can be in one of a finite collection of *states*, serving as a limited form of memory. The diagram in Figure 2

⁴ Numbers of violations are typically represented in unary notation over the symbol *. Arabic numerals are used here because some parts of this paper represent numbers of violations using algebraic expressions.

Figure 2:
An FST that deletes codas



is interpreted as follows. The FST begins in state q_0 . After reading each symbol, the FST may choose to add that symbol to the output while remaining in state q_0 ($x : x$), or *transition* to state q_1 . Should the machine choose the latter, it must either read a vowel and add it to the output ($V : V$), or read a consonant and omit it from the output ($C : \lambda$). The two circles around q_1 indicate that the computation is allowed to end there; the FST crashes if its computation ends at q_0 . Since the FST in Figure 2 has no permissible actions once it has entered state q_1 , on any given input it must remain in q_0 , outputting a copy of its input, and then transition to q_1 while reading the last input symbol. If the last input symbol is a consonant, that consonant is deleted. We say that Māori coda deletion is *rational*, since it can be implemented by an FST.


In general, OT can define mappings that cannot be computed using an FST. To see how this is possible, let us consider an example of a grammar defining a non-rational relation. In this grammar, given by Gerdemann and Hulden (2012), GEN once again produces candidates by inserting symbols into the UR, deleting symbols from the UR, and changing symbols of the UR into other symbols. CON contains the following four constraints, shown in order from highest-ranking to lowest-ranking.⁵

- (4)
- a. DEP: Assign one violation for each symbol inserted into the input.
 - b. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.
 - c. AGR: Assign one violation for each occurrence of the substring ab or ba in the output.
 - d. MAX: Assign one violation for each symbol deleted from the input.

⁵ Again, this is not the only ranking with the intended behavior; for example, DEP and ID may be switched without consequence.

Suppose that URs are strings over the alphabet $\Sigma = \{a, b\}$ and SRs are strings over the alphabet $\Delta = \{a, b, c\}$. Consider an input of the form a^*b^* , such as *aaabb*. A tableau for *aaabb* is shown in (5).

(5) *aaabb* \rightsquigarrow *aaa*

<i>aaabb</i>	DEP	ID	AGR	MAX
a. <i>aaabb</i>	0	0	1!	0
b. <i>aaacbb</i>	1!	0	0	0
c. <i>aaaaa</i>	0	2!	0	0
d. <i>bbbbb</i>	0	3!	0	0
 e. <i>aaa</i>	0	0	0	2
f. <i>bb</i>	0	0	0	3!

The faithful candidate a violates AGR, since it contains the substring *ab*. Candidates satisfying AGR cannot be chosen as the winner if they are formed by inserting or changing symbols, since such candidates violate the higher-ranking constraints DEP and ID, respectively. Thus, candidates b, c, and d cannot be the winner. An AGR-obeying candidate may be obtained by deleting all the *as* or the *bs*, resulting in candidates f and e, respectively. Because $AGR \gg MAX$, violation of MAX in order to satisfy AGR is warranted. Between candidates f and e, the candidate that involves less deletion, namely candidate e, violates MAX to a lesser degree, and is therefore chosen as the winner.

In general, when presented with a UR of the form a^*b^* , this grammar deletes all instances of either *a* or *b*, whichever symbol occurs less frequently. This kind of mapping, in which the SR depends on the frequency of each symbol in the UR, is known as a *majority-rules* mapping (Baković 1999, 2000). Since counting the number of *as* and *bs* in a string requires infinite-state memory, finite-state transducers cannot compute majority-rules mappings.

Examining tableau (5), we see that the adjudication between candidates e and f is done by counting the number of symbols deleted from the UR. Thus, OT is endowed with the ability to count by the fact that constraints may be violated to varying degrees. Because of this insight, existing approaches to finite-state OT have sought to strip constraints of counting power by imposing restrictions on how constraints may assign violation marks, or how two candidates may be compared with

one another. Using the nomenclature of Eisner (2002), the current proposals for restricted constraints are listed below.

- (6) a. *n*-Bounded Approximation (Frank and Satta 1998; Karttunen 1998): Each constraint may assign at most *n*-many violations.
- b. Matching (Gerdemann and van Noord 2000; Gerdemann and Hulden 2012): Each constraint is computed by an FST that reads candidates and emits violation marks, and candidate *y* is considered worse than candidate *z* if the set of positions where violations are assigned to *y* is a strict superset of those for *z*.
- c. Directional Evaluation (Eisner 2000): Each constraint is computed by an FST. Candidates are compared to each other by scanning them left-to-right or right-to-left in parallel, and a candidate is eliminated as soon as it receives a violation that at least one other candidate does not receive.

While these approaches do not reflect the version of OT used in phonology, each of them has a finite-state implementation, and therefore none of them can generate majority-rules mappings. Beyond these approaches, Riggle (2004) proposes an algorithm called the *Optimality Transducer Construction Algorithm* (OTCA) that takes an OT grammar and attempts to produce an FST computing the mapping defined by the grammar. However, this algorithm is not guaranteed to terminate.

Harmonic Serialism (HS) is a variation of OT in which SRs are produced by making incremental changes to URs. In HS, for any given input, GEN is assumed only to produce candidates that may be obtained by applying to the input one of a small collection of basic operations. Most existing HS analyses assume that these operations may insert, delete, or change at most one symbol of the input, so that candidates differ from the input by an edit distance of at most 1 (McCarthy 2007). Other proposals for basic operations include applying multiple instances of the same one-symbol change (McCarthy 2008; Walker 2008, 2010), creating and adjoining syllables (Elfner 2009,

2016), creating feet and assigning stress (Pruitt 2008, 2012), and inserting or deleting autosegmental association lines (McCarthy 2009). In order to effect more dramatic changes to URs, HS stipulates that recursive calls to the grammar are made until a fixed point is reached. In other words, suppose y is the winning candidate chosen by EVAL for the UR x . If $y = x$, then y is the SR for x . If not, then the SR for x is the SR for y . This process is illustrated in Figure 3.

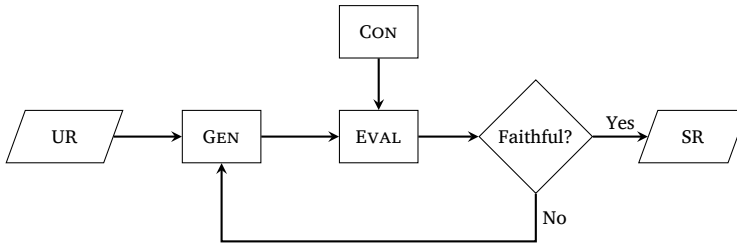


Figure 3:
Harmonic
Serialism

To see how HS works, let us consider an example due to McCarthy (2010). In Classical Arabic, the two symbols [ʔi] are appended to the beginning of the SR if the UR begins with more than one consonant. For example, the SR for the UR /fʕal/ “do!” is [ʔiffal]. One possible constraint ranking deriving the correct SR is shown below.

- (7)
- a. *CO: Assign one violation if the word begins with more than one consonant.
 - b. MAX: Assign one violation for each symbol deleted from the input.
 - c. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.
 - d. ONSET: Assign one violation if the word does not begin with a consonant.
 - e. DEP: Assign one violation for each symbol inserted into the input.

Since the constraint *CO outranks all faithfulness constraints, any violations of *CO occurring in a UR must be repaired in the SR. The fact that DEP is the lowest-ranking faithfulness constraint means that *CO will be repaired via insertion. Let us assume that low-ranking markedness constraints not shown above ensure that any inserted vowel is i,

and any inserted consonant is ʔ.⁶ Accordingly, on input fʔal, we see that the winner repairs the *CO violation by inserting a vowel.

(8) Step 1: fʔal \rightsquigarrow iffal

fʔal	*CO	MAX	ID	ONSET	DEP
a. fʔal	1!	0	0	0	0
b. ʔal	0	1!	0	0	0
c. iffal	0	0	1!	1	0
☞ d. iffal	0	0	0	1	1

The tableau above follows McCarthy (2007) in assuming that GEN can only change a single symbol. Thus, the final SR [ʔiffal] is not available among the candidates shown. Since the winner in (8) is not the faithful candidate, the grammar is called a second time.

(9) Step 2: iffal \rightsquigarrow ʔiffal

iffal	*CO	MAX	ID	ONSET	DEP
a. iffal	0	0	0	1!	0
b. fʔal	1!	1	0	0	0
c. ʔiffal	1!	0	1	0	0
☞ d. ʔiffal	0	0	0	0	1

This time, the input violates ONSET, since it begins with a vowel. Since DEP is still the lowest-ranking faithfulness constraint, the winning candidate is the one that repairs the ONSET violation by inserting a consonant. Since the winner is not the faithful candidate, the grammar is called for a third time.

(10) Step 3: Convergence

ʔiffal	*CO	MAX	ID	ONSET	DEP
☞ a. ʔiffal	0	0	0	0	0
b. iffal	0	1!	0	1	0
c. iiffal	0	0	1!	1	0
d. iʔiffal	0	0	0	1!	1

This time, the input does not violate any of the markedness constraints. The unfaithful candidates introduce gratuitous violations of faithful-

⁶This is known as *the emergence of the unmarked* (McCarthy and Prince 1994).

ness constraints, so they are all eliminated. Since the faithful candidate is chosen as the winner, the grammar terminates here. We say that the grammar has *converged* to the output [ʔiffal], so it is chosen as the final SR for the UR /ʔfal/.

Applications of HS in OT phonology are typically motivated by phonological phenomena that are most elegantly explained by decomposing the UR–SR mapping into several derivational steps. Such phenomena famously include examples of *opacity*. Elfner (2009, 2016), for example, studies opaque interactions between vowel insertion and stress assignment, and adopts an HS analysis that implements the two processes separately, allowing them to interfere with one another. Other arguments in favor of HS note that phonological processes seem to be composed of small, incremental operations. Pruitt (2008), for example, argues that locality effects in foot parsing are best explained by the gradual nature of GEN in HS. A brief survey of phonological research in HS can be found in McCarthy (2010).

The remainder of this section declares the assumptions about HS that I make in order to construct the non-rational HS grammar in Section 5. Subsection 4.1 formally defines the version of HS studied in this paper. Subsection 4.2 defines the basic operations of GEN this paper utilizes, as well as a restrictive class of markedness constraints that includes the constraints appearing in Section 5.

4.1 *Formalization of Harmonic Serialism*

For completeness, this section presents a formal definition of HS. The formalization here roughly follows Ellison’s (1994) formalization of standard OT, which forms the basis of other formalizations appearing in finite-state OT. There, GEN is taken to be an FST producing candidates. Each constraint is modelled by an FST that reads candidates and emits numbers in unary notation. The behavior of EVAL is described by an ordering relation on candidates induced by the constraint ranking mechanism.

Definition 11. A *constraint over* Σ is a rational function $c : (\Sigma^{\leq 1} \times \Sigma^{\leq 1})^* \rightarrow \mathbb{N}$. A *constraint ranking over* Σ is a sequence $\langle c_1, c_2, \dots, c_n \rangle$, where each c_i is a constraint over Σ . For each i, j , we say that c_i *outranks* c_j and write $c_i \gg c_j$ if $i < j$.

Definition 12. An *HS Grammar* is an ordered triple $\langle \Sigma, \text{GEN}, \text{CON} \rangle$, where

- Σ is an alphabet;
- $\text{GEN} : \Sigma^* \rightarrow (\Sigma^{\leq 1} \times \Sigma^{\leq 1})^*$ is a rational relation; and
- CON is a constraint ranking over Σ .

The above definition departs from Ellison (1994) in that candidates are represented as strings of pairs rather than strings of alphabet symbols. This kind of representation allows Definition 12 to model faithfulness constraints (Chen-Main and Frank 2003), which depend on both the input and the potential output represented by a candidate. The pair-string representation is also standardly used in OT analyses following *Correspondence Theory* (McCarthy and Prince 1995). I follow Riggle (2004) in modelling constraints as FSTs that read pair strings and emit violations.

Definition 13. A *candidate over* Σ is a string in $(\Sigma^{\leq 1} \times \Sigma^{\leq 1})^*$. We may sometimes denote the candidate $\langle x_1, y_1 \rangle \langle x_2, y_2 \rangle \dots \langle x_n, y_n \rangle$ using the notation $x_1 x_2 \dots x_n \mapsto y_1 y_2 \dots y_n$.

EVAL was formalized by Samek-Lodovici and Prince (1999, 2002), who noted that the behavior of an HS grammar towards a candidate $x \mapsto y$ is completely dependent on the number of violations assigned to $x \mapsto y$ by each constraint. To that end, Samek-Lodovici and Prince identify candidates with their *violation profiles*, or *costs*.

Definition 14. Let $C = \langle c_1, c_2, \dots, c_n \rangle$ be a constraint ranking over Σ . For $x \mapsto y \in (\Sigma^{\leq 1} \times \Sigma^{\leq 1})^*$, the *cost of* $x \mapsto y$ *with respect to* C is the vector

$$c_C(x \mapsto y) := \langle c_1(x \mapsto y), c_2(x \mapsto y), \dots, c_n(x \mapsto y) \rangle \in \mathbb{N}^n.$$

Candidates violating lower-ranked constraints are preferred over candidates violating higher-ranked constraints, and candidates incurring few violations of a particular constraint are preferred over candidates incurring many violations of that constraint. This preference relation is represented by an ordering over cost vectors.

Definition 15. Let $a = \langle a_1, a_2, \dots, a_n \rangle$ and $b = \langle b_1, b_2, \dots, b_n \rangle$ be vectors in \mathbb{Z}^n . We say that a is *more harmonic than* b , and write $a \succ_H b$, if there exists $j \in \{1, 2, \dots, n\}$ such that $a_j < b_j$ and for all $i < j$, $a_i \leq b_i$. We write $a \succeq_H b$ if $a \succ_H b$ or $a = b$.⁷

⁷ Note that \succ_H is the lexicographic ordering on \mathbb{Z}^n .

Under Definition 15, candidates with more harmonic cost vectors are preferred over candidates with less harmonic cost vectors. Therefore, among a set of candidates, EVAL chooses the candidate with the most harmonic cost vector as the winner.

Definition 16. Let C be a constraint ranking over Σ . The function $\text{EVAL}_C : \mathcal{P}((\Sigma^{\leq 1} \times \Sigma^{\leq 1})^*) \rightarrow \mathcal{P}(\Sigma^*)$ is defined by

$$\text{EVAL}_C(K) := \{y \mid x \mapsto y \in K, c_C(x \mapsto y) = \max_{\alpha \mapsto \beta \in K} c_C(\alpha \mapsto \beta)\},$$

where max is taken with respect to \succeq_H .⁸

Finally, let us conclude this subsection by defining the UR–SR mapping \mathcal{H}_G^+ generated by an HS grammar G .

Definition 17. Let $G = \langle \Sigma, \text{GEN}, \text{CON} \rangle$ be an HS grammar. The relation \mathcal{H}_G is defined as follows: $x \mathcal{H}_G y$ if and only if $y \in \text{EVAL}_{\text{CON}}(\text{GEN}(x))$. The relation \mathcal{H}_G^+ is defined as follows: $x \mathcal{H}_G^+ y$ if and only if $y \mathcal{H}_G x$ and there exist x_1, x_2, \dots, x_n such that $x_1 = x$, $x_n = y$, and for each i , $x_i \mathcal{H}_G x_{i+1}$. If $x \mathcal{H}_G y$, then we say that y is an *output of G on input x* . If $x \mathcal{H}_G^+ y$, then we say that G *converges to y on input x* .

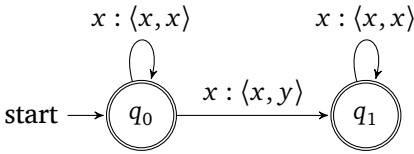
Example 18. Fix $\Sigma = \{a, b\}$. Let us construct a simple HS grammar in order to illustrate how GEN and CON may be implemented using FSTs. Suppose GEN inserts, deletes, or substitutes a single symbol from its input, and suppose CON consists of a single constraint, shown below.

(19) *CC: Assign one violation for each instance of bb in the output.

Intuitively, *CC declares a dispreference for outputs containing consonant clusters.

GEN is implemented by the FST in (20).

(20) FST for GEN ($x, y \in \Sigma^{\leq 1}$; $x \neq y$)

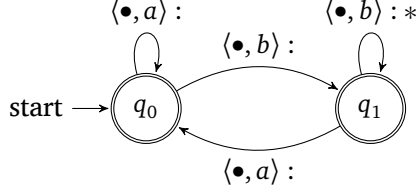


*CC is implemented by the FST in (21). Here, \bullet represents any symbol from $\Sigma^{\leq 1}$, so that an arc from state q to state r labelled with

⁸Note that despite the notation $\max_{x' \mapsto y' \in K} c_C(x' \mapsto y')$, EVAL_C is not choosing the candidate with the “maximum cost,” since what is maximal is the *harmonicity* of the cost vector.

$\langle \bullet, y \rangle : z$ means that $q \xrightarrow{\langle a, y \rangle : z} r$, $q \xrightarrow{\langle b, y \rangle : z} r$, and $q \xrightarrow{\langle \lambda, y \rangle : z} r$ are all possible transitions of the FST.

(21) FST for *CC ($x \in \Sigma \cup \{\lambda\}$)



On input $abbba$, GEN produces candidates such as $abbba \mapsto abbba$, $abbba \mapsto ababba$, and $abbba \mapsto abba$.⁹ For the candidate $abbba \mapsto abbba$, the FST for *CC outputs **; for $abbba \mapsto ababba$ and $abbba \mapsto abba$, it outputs *. Thus, $*CC(abbba \mapsto abbba) = 2$, while $*CC(abbba \mapsto ababba) = *CC(abbba \mapsto abba) = 1$. Since *CC is the only constraint, we have $c_{\text{CON}}(abbba \mapsto abbba) = \langle 2 \rangle$ and $c_{\text{CON}}(abbba \mapsto ababba) = c_{\text{CON}}(abbba \mapsto abba) = \langle 1 \rangle$. Observe that $\langle 1 \rangle \succ_H \langle 2 \rangle$.

4.2

Assumptions about HS

The previous subsection defined an HS grammar as a tuple $\langle \Sigma, \text{GEN}, \text{CON} \rangle$, but did not address the question of what kinds of FSTs may implement GEN or constraints of CON. This subsection presents the following weak assumptions about HS grammars, which suffice to construct the non-rational HS grammar in Section 5.

- GEN can insert a single symbol, delete a single symbol, or change a single symbol to another symbol.
- Markedness constraints are *strictly local*.
- Each faithfulness constraint is defined by a set of banned operations, assigning one violation to any candidate produced by applying a banned operation.

These assumptions are made explicit in Subsections 4.2.1, 4.2.2, and 4.2.3, respectively.

⁹Technically, the notation $abbba \mapsto abba$ does not specify which of the bs is deleted. This is not consequential for the rest of the paper.

4.2.1

Basic operations of GEN

I assume that GEN performs at least the following basic operations.

Definition 22. Let Σ be an alphabet. An *operation over Σ* is an ordered pair $\langle a, b \rangle$, denoted $a \mapsto b$, where $a, b \in \Sigma^{\leq 1}$ and $ab \neq \lambda$. We refer to $a \mapsto b$ simply as an *operation* when the alphabet Σ is clear from context. Additionally, we say that $a \mapsto b$ is

- an *insertion* if $a = \lambda$,
- a *deletion* if $b = \lambda$,
- a *substitution* if $\lambda \neq a \neq b \neq \lambda$, and
- an *identity* if $a = b$.

Definition 23. Let Σ be an alphabet, and define the string of pairs

$$\omega := \langle x_1, y_1 \rangle \langle x_2, y_2 \rangle \dots \langle x_n, y_n \rangle \in (\Sigma^{\leq 1} \times \Sigma^{\leq 1})^* .$$

For any operation $a \mapsto b$ over Σ , we say that ω is an *application of $a \mapsto b$* if there exists $i \in \{1, 2, \dots, n\}$ such that $a \mapsto b = x_i \mapsto y_i$ and $x_j \mapsto y_j$ is an identity as long as $j \neq i$. When ω is an application of an operation $a \mapsto b$, we may denote ω by $\omega = x_1 x_2 \dots x_n \mapsto y_1 y_2 \dots y_n$. Without explicitly specifying the operation $a \mapsto b$, we may refer to ω as a *change*. If $a \mapsto b$ is an identity, then we say that ω is *faithful*; otherwise, ω is *unfaithful*.

A change, as defined above, is an insertion, a deletion, or a substitution of a single input symbol. In practice, HS grammars considered in OT phonology rely on much richer operations. For example, the syllable-building operations of Elfner (2009, 2016) and the foot-building operations of Pruitt (2008, 2012) may change multiple symbols of the input, depending on how syllables and feet are represented. However, the availability of richer operations does not change the result of Section 5 as long as the basic operations above are available.

4.2.2

Markedness constraints

I assume that markedness constraints are *strictly local* in the following sense.

Definition 24. A constraint c is a *markedness constraint* if for every $x \mapsto z$ and $y \mapsto z$, $c(x \mapsto z) = c(y \mapsto z)$. Otherwise, c is a *faithfulness constraint*.

Definition 25. A markedness constraint c over Σ is *strictly k -local* (k -SL) if there exists a set $S \subseteq (\Sigma \cup \{\times, \times\})^k$ such that for any candidate $x \mapsto y$, $c(x \mapsto y)$ is the number of unique decompositions of the string $\times^{k-1}y\times^{k-1}$ into substrings l, s, r such that $\times^{k-1}y\times^{k-1} = lsr$ and $s \in S$. We say that c *bans* s if $s \in S$. A markedness constraint is *strictly local* (SL) if it is k -SL for some k .

SL constraints are constraints of the form “Assign one violation for each instance of ...” Thus, the constraint *CC from Example 18 is a 2-SL constraint that bans the substring bb . Other markedness constraints may be SL even if they are stated differently. For example, NOCODA from (2) is a 2-SL constraint banning substrings of the form $x\times$, where x is a consonant. Similarly, *CO from (7) is a 3-SL constraint banning substrings of the form $\times xy$, where x and y are both consonants. Observe also that the constraint AGR from the non-rational standard OT grammar of (4) is a 2-SL constraint banning the substrings ab and ba . Therefore, non-rational mappings in OT may be implemented using only markedness constraints that are SL.

The definition of SL constraints above is based on the *strictly local languages* of McNaughton and Papert (1971), a small subclass of the regular languages belonging to the *subregular hierarchy*. Intuitively, a language L is *k -strictly local* (k -SL) if there exists a k -SL markedness constraint c such that $c(x \mapsto x) = 0$ for every $x \in L$. A related subregular class of languages is the *tier-based strictly local* (TSL) *languages*, a generalization of the SL languages in which banned substrings may be interrupted by symbols from a designated subset of the alphabet. The class of TSL languages was defined by Heinz *et al.* (2011), who propose it as a formal characterization of the kinds of phonotactic dependencies that may occur in natural language phonology. Various forms of justification have been given for this hypothesis. Local phonotactic restrictions on what kinds of phonemes can occur adjacent to one another are clearly SL, and therefore TSL. Additionally, Heinz (2007), Edlefsen *et al.* (2008), Heinz (2009), and Heinz (2014) study the typology of stress patterns across languages, showing that they are usually TSL, while McMullin (2016) and McMullin and Hanson (2016) show that long-distance consonant interactions are tier-based strictly 2-local. Experimentally, Rogers and Pullum (2011), Lai (2012), Lai (2015), and McMullin (2016) investigate the ability of humans and non-human animals to learn linguistic and non-linguistic

patterns, and find that patterns that are learned successfully can be modelled by TSL languages.

4.2.3 Faithfulness constraints

Finally, I assume that each faithfulness constraint c is associated with a set O of operations such that $c(x \mapsto y) = 1$ if $x \mapsto y$ is an application of an operation in O , and $c(x \mapsto y) = 0$ otherwise. For each $o \in O$, we say that c bans o .

The faithfulness constraints DEP, MAX, and ID are all of this form, since they ban all insertions, all deletions, and all substitutions, respectively. Section 5 will make use of faithfulness constraints that ban deletions of a particular alphabet symbol.

5 NON-RATIONAL MAPPINGS IN HS

Define the function $f : (a^2b^2)^+ a^2ca^2 (b^2a^2)^+ \rightarrow \Sigma^*$ as follows.

$$f \left((a^2b^2)^{m+1} a^2ca^2 (b^2a^2)^{n+1} \right) := \begin{cases} (a^2b^2)^{m-n} a^2ca^2, & m \geq n \\ a^2ca^2 (b^2a^2)^{n-m}, & m \leq n \end{cases}$$

This function is defined on inputs consisting of a c preceded by a string in $(a^2b^2)^+ a^2$ and followed by a string in $a^2 (b^2a^2)^+$. The behavior of f is to delete an integer number of b^2a^2 units to the left of the c , along with the *same* number of a^2b^2 units to the right of the c . The function deletes as many b^2a^2 and a^2b^2 units as possible, so that the total number of units deleted depends on the length of the material to the left of the c or the material on the right of the c , whichever is shorter. Since the number of b^2a^2 s deleted must match the number of a^2b^2 s deleted, I refer to f as the *matching deletion function*.

The goal of this section is to show that matching deletion is not finite-state, and that HS can implement it under the weak assumptions from Subsection 4.2. Intuitively, computing f requires comparing the lengths of two substrings of its input, separated by the special character c . Since FSTs are not capable of this kind of computation, f is non-rational. I begin this section by making this argument rigorous in Subsection 5.1. In Subsection 5.2, I construct an HS grammar with 6-SL markedness constraints that maps an underlying form x to the

surface form $f(x)$ as long as x is within the domain of f . Since rational relations are closed under domain restriction to regular subsets, the relation described by this HS grammar is not rational.

5.1 Non-rationality of matching deletion

The non-rationality of matching deletion can be proven using a straightforward application of the *Pumping Lemma*, a standard tool in formal language theory.¹⁰

Lemma 26 (Pumping Lemma). *Let L be a regular language. There exists an integer $p \geq 1$ such that every string $\xi \in L$ of length at least p may be decomposed into three substrings $\xi = \alpha\beta\gamma$ such that $|\beta| > 1$, $|\alpha\beta| < p$, and $\alpha\beta^+\gamma \subseteq L$. The number p is called the pumping length of L .*

Intuitively, the Pumping Lemma describes the behavior of FSAs when reading long strings. Since an FSA A only has finitely many states, when ξ is sufficiently long, there must be some state q that A enters at least twice when reading ξ . The substring β is the substring that is read during the two occurrences of q ; i.e., $q \xrightarrow{\beta} q$. Since $q \xrightarrow{\beta} q \xrightarrow{\beta} q$, the substring β may be repeated arbitrarily many times without producing a string not accepted by A .

To adapt the Pumping Lemma to rational relations, observe that an FST over input alphabet Σ and output alphabet Γ may be thought of as an FSA over the alphabet $\Sigma^{\leq 1} \times \Gamma^{\leq 1}$ by replacing each FST transition $q \xrightarrow{x:y} r$ with an FSA transition $q \xrightarrow{\langle x,y \rangle} r$ (Kaplan and Kay 1994). Thus, the matching deletion function f may be thought of as a language by encoding each pair $\langle x, f(x) \rangle$ as a string of pairs. Unlike in Subsection 4.2.1, here we cannot make any assumptions regarding which symbols of x are aligned with which symbols of $f(x)$ in the pair-string representation. Instead, since the representation of f as a language is not unique, we must show that *no* language representing f is regular.

The proof will proceed as follows. We will consider an input x with a long left-hand side and an even longer right-hand side, so that the left-hand side is completely deleted by f . We will show that the substring β represents some number of a^2b^2 units from the input and some number of b^2a^2 units from the output. Thus, repeating β results in increasing the number of a^2b^2 s in x and b^2a^2 s in $f(x)$. However,

¹⁰A complete treatment of the Pumping Lemma may be found in formal language theory textbooks such as Sipser (2013).

since the left-hand side is much shorter than the right-hand side, the number of b^2a^2 s on the right-hand side of $f(x)$ must decrease when the number of a^2b^2 s on the left-hand side of x increases, whence a contradiction.

Theorem 27. *The matching deletion function f is not rational.*

Proof. Suppose that f were computed by an FST T . Thinking of T as an FSA, let p be the pumping length of T . Let

$$x := (a^2b^2)^{2p} a^2ca^2 (b^2a^2)^{4p}, \text{ so that}$$

$$f(x) = a^2ca^2 (b^2a^2)^{2p},$$

and let ξ be the string over $\Sigma^{\leq 1} \times \Sigma^{\leq 1}$ corresponding to the pair $\langle x, f(x) \rangle \in [T]$.

By the Pumping Lemma, since $|\xi| > p$, there exist α, β, γ such that $\xi = \alpha\beta\gamma$, $|\beta| > 1$, $|\alpha\beta| < p$, and $\alpha\beta^+\gamma \subseteq [T]$. Writing $\beta = \langle y_1, z_1 \rangle \langle y_2, z_2 \rangle \dots \langle y_n, z_n \rangle$, let $y := y_1y_2 \dots y_n$ and $z := z_1z_2 \dots z_n$, so that β represents T reading the substring y and emitting the substring z as output.

Now, observe that the domain of f is $(a^2b^2)^+ a^2ca^2 (b^2a^2)^+$, and the range of f is $(a^2b^2)^+ a^2ca^2 \cup a^2ca^2 (b^2a^2)^+$. Any string in either the domain or the range of f contains an integer number of a^2b^2 s or b^2a^2 s surrounding an a^2ca^2 in the middle. Therefore, since $\alpha\beta^+\gamma \subseteq [T]$, and since y must be a substring of the first p -many symbols of x , there must exist i and j such that the pair-string $\alpha\beta\beta\gamma$ represents the pair $\langle x', f(x') \rangle$, where

$$x' = (a^2b^2)^{2p+i} a^2ca^2 (b^2a^2)^{4p} \text{ and}$$

$$f(x') = a^2ca^2 (b^2a^2)^{2p+j}.$$

Since $|\alpha\beta| < p$, we must have $i < p$, so $2p + i < 4p$. According to the definition of f ,

$$f(x') = a^2ca^2 (b^2a^2)^{4p-(2p+i)} = a^2ca^2 (b^2a^2)^{2p-i},$$

so $j = -i$. Since a string cannot have a negative length, we must have $j = i = 0$. However, this implies that $|\beta| = 0$, contradicting the Pumping Lemma, so we conclude that f cannot be computed by an FST. \square

Having shown that f is not rational, let us now implement f in HS. We shall do this by constructing a grammar that, given a UR of the form $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$, behaves as follows.

- First, delete the a immediately to the right of the c :

$$(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1} \rightsquigarrow (a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}.$$

- Next, delete the a immediately to the right of the c :

$$(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1} \rightsquigarrow (a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}.$$

- Next, delete the a immediately to the left of the c :

$$(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1} \rightsquigarrow (a^2b^2)^{m+1}ac(b^2a^2)^{n+1}.$$

- Next, delete the a immediately to the left of the c :

$$(a^2b^2)^{m+1}ac(b^2a^2)^{n+1} \rightsquigarrow (a^2b^2)^{m+1}c(b^2a^2)^{n+1}.$$

- Next, do the same with the bs adjacent to the c ; delete the two bs immediately to the right, and then the two bs immediately to the left:

$$(a^2b^2)^{m+1}c(b^2a^2)^{n+1} \rightsquigarrow \dots \rightsquigarrow (a^2b^2)^m a^2ca^2(b^2a^2)^n.$$

The effect of these eight derivational steps is to delete a full b^2a^2 from the right and a full a^2b^2 from the left. This process continues until either the left side has no more a^2b^2 s or the right side has no more b^2a^2 s. If $m \leq n$, then the total number of b^2a^2 s and a^2b^2 s deleted is $m + 1$, so the SR is $a^2ca^2(b^2a^2)^{n-m}$. If $m \geq n$, then the total number of b^2a^2 s and a^2b^2 s deleted is $n + 1$, so the SR is $(a^2b^2)^{m-n}a^2ca^2$.

To obtain this behavior, the grammar we construct must fulfill three criteria. Firstly, the grammar needs to contain markedness constraints against substrings occurring near the c . This is because HS grammars can only perform unfaithful operations in order to repair violations of markedness constraints, so the deletions performed by the grammar must destroy banned substrings. Secondly, the grammar must contain a mechanism for ensuring that the symbols adjacent to the c are deleted in the correct order – first to the right of

the c , and then to the left of the c . If the grammar were allowed to delete symbols adjacent to the c in an arbitrary order, then there would be no non-rational dependency between the number of a^2b^2 s deleted to the left of the c and the number of b^2a^2 s deleted to the right of the c . Finally, the grammar must ensure that no more deletions occur when all the material either to the left or to the right of the a^2ca^2 center marker has been deleted. Otherwise, the grammar would associate each UR with the SR a^2ca^2 , resulting in a rational map.

These three components of the construction are presented in Subsections 5.2.1, 5.2.2, and 5.2.3, respectively. Subsection 5.2.4 shows how the three components are combined together to form an HS grammar implementing f .

5.2.1 Motivating deletion

Let us assume that faithfulness constraints are ranked so that the only possible actions of the grammar are to delete an a , to delete a b , or to do nothing. The behavior we wish to implement is for the grammar to delete symbols adjacent to the c . These deletions are driven by two markedness constraints, ranked in the order shown below.

- (28) a. $*ab$: Assign one violation for each occurrence of ab or ba .
 b. $*caa$: Assign one violation for each occurrence of caa , aac , cbb , or bbc .

Consider a string $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$ in f 's domain. Every symbol of this string other than those comprising the aca in the middle forms part of an ab or ba sequence banned by $*ab$. Since each a^2b^2 segment and each b^2a^2 segment introduces one ab substring and one ba substring, the total number of violations assigned by $*ab$ is $2(m+1) + 2(n+1) = 2(m+n) + 4$. The a^2ca^2 segment in the middle consists of two overlapping instances of substrings banned by $*caa$, namely a^2c and ca^2 . Thus, $*caa$ assigns 2 violations in total.

Since every a is adjacent to another a and every b is adjacent to another b , deleting a single a or a single b cannot repair a violation of $*ab$. This is seen in candidate d of tableau (29a): there, an a is deleted from the last a^2b^2 segment to the left of the c , but the number of violations of $*ab$ does not change. On the other hand, deleting one of the as adjacent to the c results in the destruction of either the ca^2

segment or the a^2c segment, so the number of violations assigned by $*caa$ is reduced by 1. Thus, the first step of the derivation is to delete an a adjacent to the c .

The result of the first step is a string in which the c is flanked by two a s on one side and a single a on the other. These are shown in candidates b and c of (29a).¹¹ Since the single a is no longer adjacent to another a , it now forms an ab segment with the neighboring b that is vulnerable to deletion. This is shown in candidate b of (29b). While it is still possible to repair a violation of $*caa$ by deleting an a on the other side of the c , as in candidate c of (29b), repairing $*caa$ is dispreferred to repairing $*ab$ because the latter constraint outranks the former. Thus, the second step of the grammar's derivation is to delete the single a that occurs between the c and a b .

(29) a. First repair $*caa...$

$(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	$*ab$	$*caa$
a. $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	$2!$
☞ b. $(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$2(m+n)+4$	1
☞ c. $(a^2b^2)^{m+1}aca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	1
d. $(a^2b^2)^mab^2a^2ca^2(b^2a^2)^{n+1}$	$2(m+n)+4$	$2!$

b. ...then repair $*ab$

$(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$*ab$	$*caa$
a. $(a^2b^2)^{m+1}a^2ca(b^2a^2)^{n+1}$	$2(m+n)+4!$	1
☞ b. $(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$2(m+n)+3$	2
c. $(a^2b^2)^{m+1}aca(b^2a^2)^{n+1}$	$2(m+n)+4!$	0

Observe that while destroying the ab segment in (29b), the grammar has created a cb^2 segment, which is banned by $*caa$. This is permissible because $*ab$ outranks $*caa$, so introducing a new violation of the latter is justified by removing a violation of the former. The result of the two derivational steps is again a string in which $*caa$ is violated twice and $*ab$ cannot be repaired. In general, the effect of the two constraints of (28) is to ensure that deletions occur two at a time:

¹¹ Both candidates have been marked as winning candidates, but in the next subsection candidate c will be eliminated as a possible winner.

first an a or a b to the left or to the right of the c , and then the other a or b on the same side of the c .

5.2.2 Enforcing directionality

The constraints of (28) create a mechanism by which the grammar is now required to delete as and bs adjacent to the c two at a time. The next step is to ensure that these pairs of deletions occur in the correct order: first the as on the right, then the as on the left, then the bs on the right, then the bs on the left. This particular ordering of the deletions ensures that every eight derivational steps, exactly one complete a^2b^2 segment to the left of the c and one complete b^2a^2 segment to the right of the c are deleted. Therefore, no partial a^2b^2 or b^2a^2 segment may remain at the end of the derivation, and the number of a^2b^2 s deleted must always match the number of b^2a^2 s deleted.

In (29b), we see that a deletion repairing $*ab$ must always occur on the same side of the c as the previous deletion repairing $*caa$. However, in (29a) we see that the constraints of (28) allow for a choice between two possible actions: destroying a ca^2 or cb^2 segment by deleting to the right of the c , or destroying an a^2c or b^2c segment by deleting to the left. The goal of this subsection is to remove this choice by imposing a preference for correct deletions over incorrect ones.

To that end, consider the possible strings that may be obtained from $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$ by deleting symbols adjacent to the c in the manner described in Subsection 5.2.1. Any such string containing two violations of $*caa$ and no repairable violations of $*ab$ must adhere to one of the following patterns.

- (30) a. $\Sigma^*a^2ca^2\Sigma^*$
 b. $\Sigma^*a^2cb^2\Sigma^*$
 c. $\Sigma^*b^2cb^2\Sigma^*$
 d. $\Sigma^*b^2ca^2\Sigma^*$

After a $*caa$ violation is repaired, the resulting string contains either an a sandwiched between the c and a b or a b sandwiched between the c and an a . The location of the single a or b reflects the location of the $*caa$ -repairing deletion: either both occur to the left of the c or both occur to the right of the c . (31) shows that eight unique configurations can be reached from one of the patterns in (30) by deleting either on the left or on the right.

- (31) a. $\Sigma^* a^2 c a^2 \Sigma^* \rightsquigarrow \Sigma^* b a c a^2 \Sigma^*$ (left) / $\Sigma^* a^2 c a b \Sigma^*$ (right)
 b. $\Sigma^* a^2 c b^2 \Sigma^* \rightsquigarrow \Sigma^* b a c b^2 \Sigma^*$ (left) / $\Sigma^* a^2 c b a \Sigma^*$ (right)
 c. $\Sigma^* b^2 c b^2 \Sigma^* \rightsquigarrow \Sigma^* a b c b^2 \Sigma^*$ (left) / $\Sigma^* b^2 c b a \Sigma^*$ (right)
 d. $\Sigma^* b^2 c a^2 \Sigma^* \rightsquigarrow \Sigma^* a b c a^2 \Sigma^*$ (left) / $\Sigma^* b^2 c a b \Sigma^*$ (right)

Since the eight configurations are unique, from each configuration on the right-hand side of an \rightsquigarrow above it is possible to recover both the pattern from (30) on the left-hand side and whether the configuration was obtained by deleting on the left or the right.

When the deletions are performed in the correct order, the intermediate strings encountered in the derivation of the grammar cycle through the four patterns of (30), in the order shown. Because of this, whether the correct deletion occurs to the left or the right of the c is completely determined by whether the input matches pattern (30a), (30b), (30c), or (30d). Four of the eight configurations in (31) reflect the result of performing the correct action based on the pattern from (30). The remaining four configurations are obtained when incorrect actions are performed. Therefore, the correct order of the deletions can be enforced using a markedness constraint against the four configurations reflecting incorrect deletions.

- (32) $*baca$: Assign one violation for each occurrence of $baca$, $acba$, $abcb$, or $bcab$.

To illustrate, consider again the tableau of (29a), but this time including the constraint $*baca$. For considerations of space, let $M := 2(m+n)+4$. While both candidates b and c were marked as winners in (29a), in (33) candidates b and c are distinguished by the latter's violation of $*baca$. Thus, only candidate b , in which deletion occurs to the right of the c , is a winner in (33).

- (33) $*baca$ distinguishes between candidates b and c

$(a^2 b^2)^{m+1} a^2 c a^2 (b^2 a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2 b^2)^{m+1} a^2 c a^2 (b^2 a^2)^{n+1}$	M	$2!$	0
b. $(a^2 b^2)^{m+1} a^2 c a (b^2 a^2)^{n+1}$	M	1	0
c. $(a^2 b^2)^{m+1} a c a^2 (b^2 a^2)^{n+1}$	M	1	$1!$
d. $(a^2 b^2)^m a b^2 a^2 c a^2 (b^2 a^2)^{n+1}$	M	$2!$	0

Since the purpose of $*baca$ is to distinguish between candidates that are treated identically by the other markedness constraints, the ranking of $*baca$ relative to $*ab$ and $*caa$ is inconsequential. This is indicated by the dashed line in tableau (33) separating the $*baca$ column from the other columns. For convenience, tableaux in the remainder of this section will show the three markedness constraints in the order presented in (33).

5.2.3 Stopping condition

We have now seen that (28) and (32) together create the eight-step effect of deleting a full b^2a^2 to the right of the c and a full a^2b^2 to the left of the c . The final step of the construction is to force the grammar to converge to a final SR when one of the two sides runs out of a^2b^2 s or b^2c^2 s. This has occurred when the string matches one of the following patterns.

- (34) a. $\Sigma^*ba^2ca^2$
 b. $a^2ca^2b\Sigma^*$
 c. a^2ca^2

Pattern (34c) occurs when the original UR $(a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$ contains the same number of a^2b^2 s to the left of the c as b^2a^2 s to the right of the c ; i.e., when $m = n$. (34a) occurs when $m > n$, and therefore the right side runs out of b^2a^2 s before the left side runs out of a^2b^2 s. (34b) occurs when $m < n$.

As in Subsection 5.2.2, we can consider the configurations that result when a symbol adjacent to the c is deleted from a string matching one of the patterns in (34).

- (35) a. $\Sigma^*ba^2ca^2 \rightsquigarrow \Sigma^*baca^2$ (left)/ Σ^*ca (right)
 b. $a^2ca^2b\Sigma^* \rightsquigarrow ac\Sigma^*$ (left)/ $a^2cab\Sigma^*$ (right)
 c. $a^2ca^2 \rightsquigarrow aca^2 \in ac\Sigma^*$ (left)/ $a^2ca \in \Sigma^*ca$ (right)

Since no further deletions should occur when one of the patterns in (34) is reached, all the configurations on the right-hand side of an \rightsquigarrow above reflect incorrect deletions, so all the configurations must be banned by a markedness constraint.

- (36) STOP: Assign one violation for each occurrence of $baca^2 \times$, $ca \times^4$, \times^4ac , or $\times a^2cab$.

Each of the deletions shown in (35) repairs a violation of $*caa$, so STOP must rank above $*caa$.

To confirm that STOP behaves as intended, let us consider a string in which the left side contains no a^2b^2 segments, but the right side contains at least one b^2a^2 segment.

(37) Converging to a final UR

$a^2ca^2(b^2a^2)^{n+1}$	STOP	$*ab$	$*caa$	F
☞ a. $a^2ca^2(b^2a^2)^{n+1}$	0	$2n+2$	2	0
b. $aca^2(b^2a^2)^{n+1}$	1!	$2n+2$	1	1
c. $a^2ca(b^2a^2)^{n+1}$	1!	$2n+2$	1	1
d. $a^2ca^2ba^2(b^2a^2)^n$	0	$2n+2$	2	1!

Since no violation of $*baca$ can be introduced by deleting a single a or b from $a^2ca^2(b^2a^2)^{n+1}$, $*baca$ is not shown in the tableau above. Candidate b, obtained by deleting an a to the left of the c , contains the banned substring \times^4ac , so it violates STOP. Candidate c, obtained by deleting an a to the right of the c , contains $\times a^2cab$, so it also violates STOP. These violations of STOP eliminate b and c as potential winners. Observe that none of the markedness constraints distinguishes between candidate a, the faithful candidate, and candidate d, obtained by deleting a b not adjacent to the c . Instead, candidates like d are eliminated by low-ranking faithfulness constraints against deleting as and bs , represented collectively in the column labelled “F.”

5.2.4 Constraint ranking

To complete the construction, all that remains is to arrange the markedness constraints of (28), (32), and (36) and relevant faithfulness constraints to form a ranking that produces the desired behavior. Let us briefly summarize the ranking requirements identified in the previous subsections.

- $*ab \gg *caa$.
- The ranking of $*baca$ is inconsequential.
- $STOP \gg *caa$.

The faithfulness constraints need to ensure that the only permissible actions, other than doing nothing, are deleting an a or a b . The constraints ID and DEP can be used to ban substitutions and insertions, re-

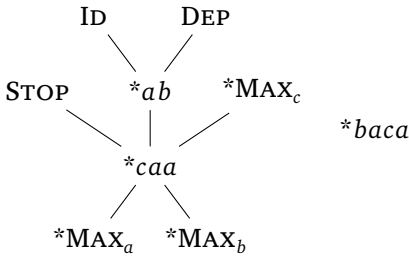


Figure 4:
Required ranking relations
among constraints

spectively. The constraint MAX will be replaced with three constraints, each of which bans deletion of a particular alphabet symbol.

- (38) a. ID: Assign one violation for each symbol from the input that is changed to a different symbol in the output.
 b. DEP: Assign one violation for each symbol inserted into the input.
 c. MAX_a: Assign one violation for each *a* deleted from the input.
 d. MAX_b: Assign one violation for each *b* deleted from the input.
 e. MAX_c: Assign one violation for each *c* deleted from the input.

To allow deletions to occur, MAX_a and MAX_b must rank below *caa, and to ban substitutions and insertions, ID and DEP must rank above *ab. Since deleting a *c* cannot repair a violation of *ab, MAX_c needs to rank above *caa, but not necessarily above *ab. These ranking requirements, along with the requirements for markedness constraints, are shown visually in Figure 4. Any ranking compatible with Figure 4 produces the desired behavior, so let us simply assume the ranking shown below.

ID ≫ DEP ≫ MAX_c ≫ STOP ≫ *ab ≫ *caa ≫ *baca ≫ MAX_a ≫ MAX_b

Theorem 39. *There exists an HS grammar with strictly local markedness constraints that generates a non-rational mapping between underlying forms and surface forms.*

Proof. Let us confirm that the HS grammar we have constructed behaves as expected. Given a UR $x = (a^2b^2)^{m+1}a^2ca^2(b^2a^2)^{n+1}$, the grammar should produce $f(x)$ as the SR. As mentioned before, since the

restriction of a rational function to a regular subset of its domain is still rational, any grammar that computes f on its domain implements a non-rational function in general.

I will proceed by first showing that the grammar implements the eight-step derivational process that deletes exactly one a^2b^2 to the left of the c and one b^2a^2 to the right of the c . Then, I show that STOP correctly implements the stopping condition that terminates the derivation.

Subsections 5.2.1 and 5.2.2 showed that the first two steps of the eight-step process behave as expected. The following tableaux show the remaining six steps. As before, let us define $M := 2(m + n) + 4$.

(40) a. Deleting an a on the left

$(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}a^2c(b^2a^2)^{n+1}$	$M - 1$	$2!$	0
☞ b. $(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$M - 1$	1	0
c. $(a^2b^2)^{m+1}a^2cba^2(b^2a^2)^n$	$M - 1$	1	$1!$

b. Deleting another a on the left

$(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}ac(b^2a^2)^{n+1}$	$M - 1!$	1	0
☞ b. $(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$M - 2$	2	0

c. Deleting a b on the right

$(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}c(b^2a^2)^{n+1}$	$M - 2$	$2!$	0
☞ b. $(a^2b^2)^{m+1}cba^2(b^2a^2)^n$	$M - 2$	1	0
c. $(a^2b^2)^m a^2bc(b^2a^2)^{n+1}$	$M - 2$	1	$1!$

d. Deleting another b on the right

$(a^2b^2)^{m+1}cba^2(b^2a^2)^n$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}cba^2(b^2a^2)^n$	$M - 2!$	1	0
☞ b. $(a^2b^2)^{m+1}ca^2(b^2a^2)^n$	$M - 3$	2	0

e. Deleting a b on the left

$(a^2b^2)^{m+1}ca^2(b^2a^2)^n$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^{m+1}ca^2(b^2a^2)^n$	$M-3$	$2!$	0
☞ b. $(a^2b^2)^m a^2 b c a^2 (b^2 a^2)^n$	$M-3$	1	0
c. $(a^2b^2)^{m+1}ca(b^2a^2)^n$	$M-3$	1	$1!$

f. Deleting another b on the left

$(a^2b^2)^m a^2 b c a^2 (b^2 a^2)^n$	$*ab$	$*caa$	$*baca$
a. $(a^2b^2)^m a^2 b c a^2 (b^2 a^2)^n$	$M-3!$	1	0
☞ b. $(a^2b^2)^m a^2 c a^2 (b^2 a^2)^n$	$M-4$	2	0

Tableaux (40a), (40c), and (40e) are analogous to tableaux (29a) and (33): $*caa$ is violated twice, $*ab$ violations cannot be repaired, and $*baca$ causes the grammar to choose the correct deletion to repair $*caa$ over the incorrect one. Tableaux (40b), (40d), and (40f), like (29b), repair a violation of $*ab$ while introducing a violation of $*caa$. The winner in (40f), after the eighth step of the derivation, is $(a^2b^2)^m a^2 c a^2 (b^2 a^2)^n$ – the result of deleting exactly one $a^2 b^2$ and one $b^2 a^2$ from x .

The final part of the proof is to show that the stopping condition behaves as expected. (37) showed that STOP correctly terminates the derivation when $m < n$. The following tableaux show that the derivation terminates correctly when $m > n$ and when $m = n$.

(41) a. Converging when $m > n$

$(a^2b^2)^{m+1}a^2ca^2$	STOP	$*ab$	$*caa$	F
☞ a. $(a^2b^2)^{m+1}a^2ca^2$	0	$2m+2$	2	0
b. $(a^2b^2)^{m+1}a^2ca$	$1!$	$2m+2$	1	1
c. $(a^2b^2)^{m+1}aca^2$	$1!$	$2m+2$	1	1
d. $(a^2b^2)^m a^2 b a^2 c a^2$	0	$2m+2$	2	$1!$

b. Converging when $m = n$

a^2ca^2	STOP	$*ab$	$*caa$	F
☞ a. a^2ca^2	0	0	2	0
b. aca^2	1!	0	1	1
c. a^2ca	1!	0	1	1

Candidate b of (41a) is eliminated by STOP because it contains the offending substring $ca \times^4$. Similarly, candidate c contains $baca^2 \times$. As in (37), candidate d is eliminated by low-ranking faithfulness constraints. In (41b), b and c are the only candidates that may be obtained by deleting an a. These candidates contain \times^4ac and $ca \times^4$, respectively, so they are eliminated by STOP. \square

6 CANONICAL NON-RATIONAL MAPPINGS

We have now seen that in general, HS grammars with SL markedness constraints can produce non-rational relations. Computing the matching deletion function requires the ability to enforce dependencies between arbitrarily many nesting pairs of a^2b^2 and b^2a^2 units separated by arbitrarily long distances. While the markedness constraints $*caa$ and $*ab$ are only sensitive to the material immediately adjacent to the c, deletion was able to extend the reach of these constraints by moving a^2b^2 and b^2a^2 units close to the c. The carefully choreographed manner in which the deletions were carried out allowed the grammar to maintain nesting dependencies reminiscent of context-free grammars.

The result of the previous section raises the question of what kinds of non-rational mappings HS can generate. The goal of this section is to argue that all non-rational relations generated by HS involve coordinated deletions that occur on opposite sides of some center marker. Thus, the matching deletion function is a canonical example of a non-rational mapping in HS, just as majority-rules mappings may be seen as canonical examples of non-rational mappings in standard OT.

Hao (2017) attempted to construct a finite-state model of HS by first showing that \mathcal{H}_G is rational for any HS grammar G , and then applying an algorithm due to Abdulla *et al.* (2002, 2003) that takes an

FST as input and attempts to construct an FST computing its transitive closure. Subsection 6.1 reviews this algorithm in detail, including sufficient conditions for its termination. Subsection 6.2 argues that HS grammars violate these conditions exactly when performing coordinated deletions. In particular, I will show that HS grammars with SL markedness constraints are rational if they cannot perform deletion.

6.1 *Transducer iteration*

Computing the transitive closure of a relation is a difficult problem. Since transitions between Turing machine configurations can be modelled by rational relations, the halting problem can be reduced to finding the transitive closure of a rational relation. Nonetheless, the problem of computing transitive closures, or approximations thereof, is of substantial practical interest. The field of *model checking*, for example, is concerned with determining what states of a program or other computational system can be reached from an initial configuration, and in particular whether any of these reachable states indicate undesirable behavior. Several studies in this area have explored the possibility of performing reachability analysis using FSTs by modelling state transitions as rational relations. To that end, partial algorithms have been developed that attempt to produce FSTs computing the transitive closures of rational relations. One approach, developed by Bouajjani *et al.* (2000), Jonsson and Nilsson (2000), Abdulla *et al.* (2002), and Abdulla *et al.* (2003), considers infinite-state transducers computing transitive closures and defines a behavior-preserving equivalence relation under which the state set has a finite quotient. Another approach, due to Dams *et al.* (2001a,b, 2002), also attempts to produce a finite quotient of an infinite state set, but the equivalence relation is constructed algorithmically while computing increasingly large compositions of an FST with itself.

The transducer iteration algorithm used in Hao (2017) is the quotient-based algorithm of Abdulla *et al.* (2002) and Abdulla *et al.* (2003). Since this algorithm is only compatible with FSTs that perform substitutions, Hao's construction simulates insertions and deletions by thinking of them as substitutions involving a designated alphabet symbol representing λ . Abdulla *et al.*'s technique relies on the observation

that for any FST T , the composition of $[T]$ with itself n times for some fixed n is a rational relation. An infinite-state transducer for $[T]^+$ is produced using a construction for the n -fold composition of T by taking n to infinity. The size of the state set is reduced by identifying states with the same behavior and merging them. The algorithm terminates if finitely many states remain after merging is complete. This termination condition gives us a sufficient condition for the rationality of $[T]^+$.

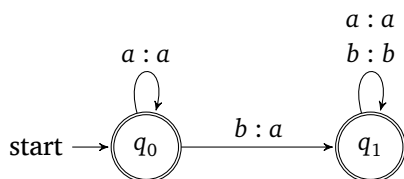
Subsection 6.1.1 describes the construction for the infinite-state transducer computing $[T]^+$. Subsection 6.1.2 presents the equivalence relation used to collapse the infinite state set into a possibly finite one.

6.1.1 Column transducers

To understand the construction for the n -fold composition of an FST, let us consider a concrete example.

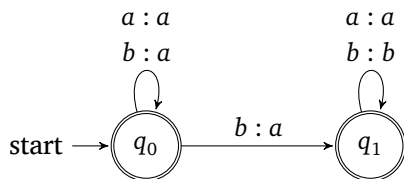
Example 42. Consider the following FST over the alphabet $\Sigma = \{a, b\}$.

(43) An FST that changes the first b to an a



The behavior of FST (43) is to change the first b it encounters to an a . The transitive closure of this relation would change the first n instances of b to as , where the value of n is chosen nondeterministically. This is clearly finite-state, since it can be computed by the FST shown below in (44).

(44) An FST that changes the first n -many bs to as



The intuition behind Abdulla *et al.*'s technique is as follows. We can understand the behavior of an FST on a particular input by inspecting its *run* – the sequence of states entered into during the course

of the computation. For example, (45) shows the runs of (43) when it is applied twice to the input $aabb$, producing first $aaab$ and then $aaaa$.

(45) Runs of FST (43) on input $aabb$

Time:	0		1		2		3		4
Input 1:		a		a		b		b	
Run 1:	q_0		q_0		q_0		q_1		q_1
Input 2:		a		a		a		b	
Run 2:	q_0		q_0		q_0		q_0		q_1
Input 3:		a		a		a		a	

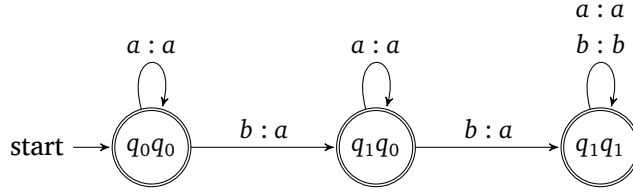
During the first run of (43), the FST is in state q_0 for three time steps, and then state q_1 for two time steps. During the second run, (43) is in state q_0 for four time steps, and then state q_1 for one time step. The behavior of (43) when it is applied twice to some input can be described by inspecting the *columns* of table (45). These columns describe, for each time step, the state of the FST during each of its iterations. Based on this, we may construct an FST that simulates two iterations of (43) by taking each state to represent a column of (45). The run of such a transducer, shown in (46), resembles table (45), except that the “Input 2” row is omitted and the “Run 1” and “Run 2” rows are merged.

(46) Combining the two runs of (45)

Time:	0		1		2		3		4
Input:		a		a		b		b	
Run:	q_0		q_0		q_0		q_1		q_1
	q_0		q_0		q_0		q_0		q_1
Output:		a		a		a		a	

Whenever the transitions $p \xrightarrow{x:y} q$ and $r \xrightarrow{y:z} s$ occur in (43), an FST whose behavior is described by (46) has the transition $pr \xrightarrow{x:z} qs$. For example, between time steps 2 and 3, (45) shows that (43) undergoes the transition $q_0 \xrightarrow{b:a} q_1$ during its first run and $q_0 \xrightarrow{a:a} q_0$ during its second run. Accordingly, (46) shows that the 2-fold iteration of (43) undergoes the transition $q_0q_0 \xrightarrow{b:a} q_1q_0$ from time step 2 to time step 3. By combining all possible transitions of (43) in this way, we obtain the FST shown below.

(47) The composition of (43) with itself



The three states of (47) represent the three possible values that might appear in a column of table (45). The FST is in state q_0q_0 when it has not seen any bs . The FST enters state q_1q_0 after seeing the first b . This represents the fact that (43) enters state q_1 on upon seeing b in its first iteration. However, because the b is changed to an a , (43) does not enter state q_1 in its second iteration until the second b of the original input is seen. When this happens, (47) enters state q_1q_1 . It is clear that the behavior of (47) is to change the first two bs of its input to as .

Note that the state q_0q_1 does not appear in (47). This is because (43) can only be in state q_0 at time t if it has not emitted any bs , so the output of the first run cannot contain any bs before time t . However, (43) can only be in state q_1 if it has seen at least one b . Since the input to the second run, which is the output of the first run, does not contain any bs before time t , (43) cannot enter state q_1 during its second run until after time t .

The ideas discussed in Example 42 are formalized as follows.

Definition 48. An FST $\langle Q, \Sigma, \Gamma, I, F, \rightarrow \rangle$ is *same-length* if for every $q, r \in Q$, $q \xrightarrow{a:b} r$ implies that $|a| = |b| = 1$.

Definition 49. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. For each $n > 1$, the *n -fold column transducer for T* is defined as the transducer $T^n := \langle Q^n, \Sigma, \Sigma, I^n, F^n, \rightarrow_n \rangle$, where for each $q_1q_2 \dots q_n, r_1r_2 \dots r_n \in Q^n$,

$$q_1q_2 \dots q_n \xrightarrow{a:b}_n r_1r_2 \dots r_n$$

holds if and only if there exist $a_0, a_1, \dots, a_n \in \Sigma$ such that $a_0 = a$, $a_n = b$, and for each $i \in \{1, 2, \dots, n\}$, $q_i \xrightarrow{a_{i-1}:a_i} r_i$.

For each n , $[T_n]$ is the n -fold composition of $[T]$ with itself. A transducer for $[T]^+$ is created by taking the union of all the T_n s.

Definition 50. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. The *column transducer for T* is defined as the transducer $T^+ := \langle Q^+, \Sigma, \Sigma, I^+, F^+, \rightarrow_+ \rangle$, where

$$\rightarrow_+ = \rightarrow \cup \left(\bigcup_{n=2}^{\infty} \rightarrow_n \right).$$

Since T^+ has infinitely many states, it is not an FST. The next subsection shows how we can attempt to reduce the size of the state set by taking its quotient under an equivalence relation. An FST equivalent to T^+ is obtained if the quotient is finite.

6.1.2 Quotient transducers

The technique for merging states is based on eliminating repetitions of *copying states* – states whose behavior is to copy the input.

Definition 51. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. A state q is *left-copying* if for every start state $q_0 \in I$, $q_0 \xrightarrow{a:b}_* q$ implies $a = b$. A state q is *right-copying* if for every accept state $q_f \in F$, $q \xrightarrow{a:b}_* q_f$ implies $a = b$. A state q is *non-copying* if it is neither left-copying nor right-copying. A state q is *copying* if it is not non-copying.

For a state q to be left-copying means that the only way to reach q is for the transducer to copy its input. For q to be right-copying means that once q has been reached, the only possible action of the transducer is to copy its input. The equivalence relation on the states of T^+ is defined by merging any two columns that are identical except for repetitions of copying states of T .

Definition 52. Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. Define the equivalence relation \simeq_T over Q^+ as follows. We say that $q \simeq_T r$ if and only if we can write

$$\begin{aligned} q &= q_1^{p_1} q_2^{p_2} \dots q_m^{p_m} \\ r &= q_1^{r_1} q_2^{r_2} \dots q_m^{r_m}, \end{aligned}$$

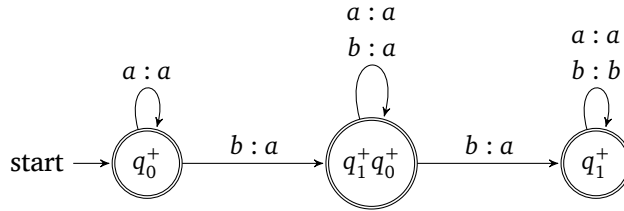
where $p_i = r_i$ whenever q_i is non-copying and for each j , $p_j > 0$ and $r_j > 0$. The *quotient transducer for T* is defined as the transducer $T_{\simeq} := \langle Q^+ / \simeq, \Sigma, \Sigma, I / \simeq, F / \simeq, \Rightarrow \rangle$, where $[q]_{\simeq_T} \xrightarrow{a:b} [r]_{\simeq_T}$ if and only if $q \xrightarrow{a:b}_+ r$.

Since \simeq does not distinguish between multiple repetitions of the same copying state, a canonical notation for equivalence classes of \simeq can be defined by replacing $q_i^{p_i}$ and $q_i^{r_i}$ with q_i^+ whenever q_i is a

copying state. For example, if q is copying but p and r are not, then the column $p^8q^5r^9$ belongs to the equivalence class $p^8q^+r^9$.

Example 53. Consider again the FST (43). The state q_0 is left-copying, since the only transition reaching q_0 copies an a to the output stream. The state q_1 is right-copying, since the unique transition from q_1 copies an a or a b to the output stream. In (47), we saw that the columns of length 2 for the column transducer of (43) are q_0q_0 , q_1q_0 , and q_1q_1 . In general, columns obtained by iterating (43) are of the form $q_1^*q_0^*$. Since q_1 is right-copying and q_0 is left-copying, the quotient transducer for (43), shown below, contains three states: q_0^+ , $q_1^+q_0^+$, and q_1^+ .

(54) Quotient transducer for (43)



It is easy to see that the transducer above has the same behavior as (44), the transducer computing the transitive closure of (43).

Abdulla *et al.* (2002) prove that merging states in this way does not change the behavior of T^+ under the condition that no reachable column in Q^+ contains a substring of the form pq , where $p \neq q$ and p and q are both left-copying or both right-copying. They ensure that this condition is fulfilled by requiring that T be deterministic. Abdulla *et al.* (2003) relaxes the assumption of determinism by introducing an algorithm that makes the portion of T containing only the left-copying states deterministic, and the portion of T containing only the right-copying states reverse-deterministic. FSTs preprocessed in this manner are called *bideterministic*, and any same-length FST can be bideterminized. Once an FST has been bideterminized, it can be safely used to construct a quotient transducer without changing the behavior of the column transducer.

Theorem 55 (Abdulla *et al.* 2002, 2003). *If T is same-length and bideterministic, then $[T_{\simeq}] = [T^+] = [T]^+$.*

The algorithm for constructing T_{\simeq} from an FST T is as follows. First, T is made bideterministic. Then, the algorithm constructs T^n for n increasingly large, while taking the quotient of the state set

after each iteration. The algorithm terminates when $T^n = T^{n+1}$ after columns have been merged based on \simeq .

```

1: procedure CLOSURE( $T$ )
2:   Initialize  $T_{\simeq} \leftarrow T$ , but with each state  $q$  replaced with  $[q]_{\simeq}$ .
3:   Make  $T_{\simeq}$  bideterministic.
4:   do
5:     Set  $T'_{\simeq} \leftarrow T_{\simeq}$ .
6:     for each transition  $[q]_{\simeq} \xrightarrow{a:b} [r]_{\simeq}$  of  $T_{\simeq}$  and  $s \xrightarrow{b:c} t$  of  $T$  do
7:       Add the transition  $[qs]_{\simeq} \xrightarrow{a:c} [rt]_{\simeq}$  to  $T_{\simeq}$ .
8:     Remove unreachable states from  $T_{\simeq}$ .
9:   while  $T_{\simeq} \neq T'_{\simeq}$ 
10:  return  $T_{\simeq}$ .

```

Algorithm 56:
Constructing T_{\simeq}
from T , Abdulla
et al. (2002,
2003)

Each iteration of the algorithm discovers equivalence classes of T_{\simeq} with increasingly longer canonical names. If the length of the canonical name of a reachable state in T_{\simeq} is bounded by some $n \in \mathbb{N}$, then after the n th iteration all reachable states of T_{\simeq} will have been discovered, and the exit condition for the while-loop on line 9 will be satisfied. This gives us the termination condition for the algorithm.

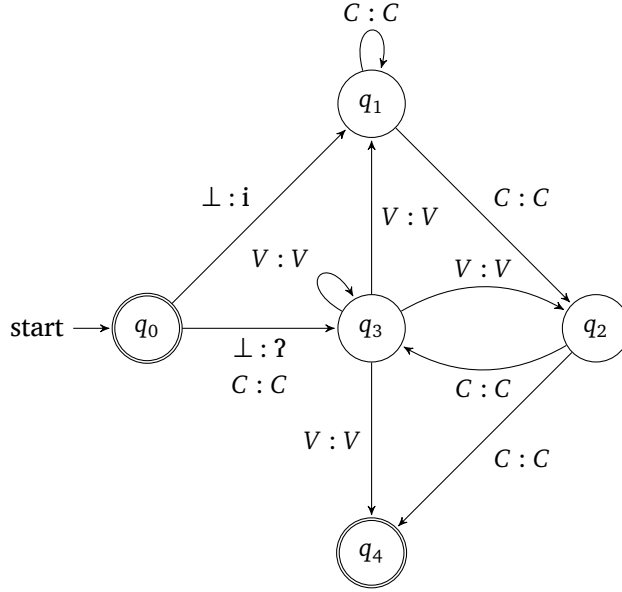
Theorem 57 (Abdulla *et al.* 2002, 2003). *Let $T = \langle Q, \Sigma, \Sigma, I, F, \rightarrow \rangle$ be a same-length FST. If the following conditions are met, then $[T]^+$ is rational.*

- *There is a bound on the number of non-copying states from Q appearing in the reachable states of T^+ .*
- *There is a bound on the number of alternations between left-copying and right-copying states from Q appearing in the reachable states of T^+ .*

Example 58. Let G be the HS grammar described at the beginning of Section 4. Recall that this grammar describes a process of Classical Arabic wherein an SR receives the prefix [ʔi] if the UR begins with a consonant cluster. The behavior of \mathcal{H}_G is as follows. If the input begins with a consonant cluster, then \mathcal{H}_G adds an i to the beginning. If the input begins with a vowel, then \mathcal{H}_G adds a ʔ to the beginning. For example, the SR [ʔifʔal] for the UR /fʔal/ “do!” is derived as follows: fʔal \mathcal{H}_G ifʔal \mathcal{H}_G ʔifʔal \mathcal{H}_G ʔifʔal. An FST T with this behavior is shown below. To ensure that T is same-length, the symbol \perp

represents λ , and insertions are represented as substitutions of \perp for other symbols. Let us assume that for each state q , T has the transition $q \xrightarrow{\perp:\perp} q$.

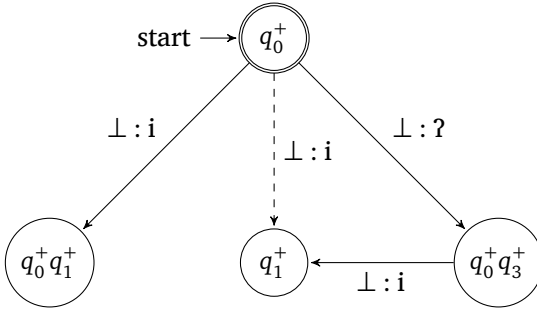
(59) An FST for \mathcal{H}_G (C is a consonant, V is a vowel)



The start state q_0 is a left-copying state, while $q_1, q_2, q_3,$ and q_4 are right-copying states. Observe that the sub-automaton containing only state q_0 is deterministic, while the sub-automaton excluding state q_0 is reverse-deterministic. Therefore, T is already bideterministic.

Let us apply Abdulla *et al.*'s algorithm to T . The equivalence class q_0^+ is the start state of T_{\sim} . Bideterminism guarantees that no equivalence class of T_{\sim} has a canonical name containing pq where $p \neq q$ and p and q are both left-copying or both right-copying. Therefore, after the first iteration, the new states added to T_{\sim} are $q_0^+q_1^+, q_0^+q_2^+, q_0^+q_3^+, q_0^+q_4^+, q_1^+q_0^+, q_2^+q_0^+, q_3^+q_0^+,$ and $q_4^+q_0^+$. Starting from the start state q_0^+ , the new transitions $q_0^+ \xrightarrow{\perp:i} q_0^+q_1^+$ and $q_0^+ \xrightarrow{\perp:c} q_0^+q_3^+$, with $c = ?$, are added to T_{\sim} based on the condition in line 6. From $q_0^+q_3^+$, the transition $q_0^+q_3^+ \xrightarrow{\perp:i} q_1^+$ is added. No other new transitions begin at one of the currently reachable states, so the first iteration terminates here. The new transitions are shown below.

(60) Transitions added during the first iteration (the dashed arrow represents an existing transition)



In the second iteration, the new states added are $q_0^+q_3^+q_0^+$ and $q_0^+q_1^+q_0^+$. Note that (60) has no transitions originating from $q_0^+q_1^+$, and the sole transition originating from $q_0^+q_3^+$ emits a vowel as output. However, the no transition T from q_0 reads a vowel as input, so no new transitions are added. Since T_{\approx} has not changed during the second iteration, the algorithm terminates, returning an FST with the transitions in both (59) and (60). Observe that the behavior of this FST is to either simulate (59) or to add ?i before a consonant cluster – exactly the behavior of $[T]^+$.

6.2 Matching deletion and copying-state alternations

Theorem 57 identifies two conditions under which the transitive closure $[T]^+$ of a same-length rational relation $[T]$ is rational. This subsection applies Theorem 57 to FSTs computing \mathcal{H}_G , developing some intuition on when the transitive closure of \mathcal{H}_G is rational. Observe first that Theorem 57 is simplified in the case of FSTs implementing a single change because the first condition is automatically satisfied.

Proposition 61. *Suppose T is a same-length FST such that $x [T] y$ if and only if $x \mapsto y$ is a single change. Then, every reachable state of T is left-copying or right-copying.*

Proof. Suppose q is a reachable non-copying state of T . Then, there exist a starting state q_0 and an accepting state q_f such that

- $q_0 \xrightarrow{a:b}_* q$ with $a \neq b$ and
- $q \xrightarrow{c:d}_* q_f$ with $c \neq d$.

Thus, $ac [T] bd$. However, since $a \neq b$ and $c \neq d$, at least two symbols of ac must differ from their counterparts in bd , so $ac \mapsto bd$ cannot be a change. Thus, T cannot have any reachable non-copying states. \square

The remaining condition of Theorem 57 is a bound on the number of alternations between left-copying and right-copying states in the reachable columns of T^+ . Suppose T is a same-length FST implementing a single change. If T enters a left-copying state at time t , then it has copied its input between time steps 0 and t , so the single change that T implements occurs after time t . Similarly, if T enters a right-copying state at time t , then the single change occurs before time t , since T can only copy its input after time t . This means that a bound on the number of alternations between left-copying and right-copying states corresponds to a bound on the number of times T can make a change before time t followed by a change after time t and *vice versa*, for each time step t .

Example 62. Define the *matching substitution* function g as follows.

$$g \left((a^2b^2)^{m+1} a^2ca^2 (b^2a^2)^{n+1} \right) \\ := \begin{cases} (a^2b^2)^{m-n} (\perp^4)^{n+1} a^2ca^2 (\perp^4)^{n+1}, & m \geq n \\ (\perp^4)^{m+1} a^2ca^2 (\perp^4)^{m+1} (b^2a^2)^{n-m}, & m \leq n \end{cases}$$

This function is like the matching deletion function, except that instead of deleting symbols adjacent to the c , g replaces them with \perp . Let G be the HS grammar for the matching deletion function constructed in Section 5, and let T be a same-length FST that implements \mathcal{H}_G , except that deleted symbols are replaced with \perp .

On input $(a^2b^2)^{m+1} a^2ca^2 (b^2a^2)^{n+1}$, T reads the symbol c between time steps $t = 4(m+1) + 2 = 4m + 6$ and $t + 1 = 4m + 7$. For each a^2b^2 unit and b^2a^2 unit that is changed to \perp^4 , T makes two substitutions after time t and two substitutions before time t . Thus, at time t , T_{\sim} is in the state $(q_L^+ q_R^+)^k$, where q_L is left-copying, q_R is right-copying, and k is the total number of a^2b^2 and b^2a^2 units deleted. Since k can be arbitrarily large for large values of m and n , there is no bound on the number of alternations between right-copying and left-copying states in the reachable columns of T^+ , so T does not fulfill the termination conditions for Abdulla *et al.*'s (2002) algorithm.

As Example 62 shows, the kind of single-change behavior that is incompatible with Theorem 57 – the kind that alternates between making changes before and after a certain position in the string – is exactly the kind of behavior for \mathcal{H}_G that is used to implement the matching deletion function. This justifies the claim that non-rational mappings in HS with strictly local markedness constraints consist of coordinated changes occurring on opposite sides of some center marker. The following example illustrates how deletion makes such behavior possible.

Example 63. Recall that, in Section 5, the markedness constraint $*baca$ was used to ensure that deletions occurred in the correct order. By specifying a set of banned substrings containing the c , $*baca$ is able to enforce a dependency between the choice of which deletion is carried out and the material that exists on each side of the c .

Now, consider the matching substitution function. It is straightforward to modify the grammar from Section 5 for the matching deletion function so that substitutions of a or b for \perp are performed in place of deletions. However, as more and more symbols are changed to \perp , the a^2b^2 and b^2a^2 units closest to the c become arbitrarily far away from each other. This suggests that no finite set of banned substrings can enforce a dependency between the two sides of the c , since for long inputs the number of \perp s adjacent to the c will exceed the maximum length of a banned substring attempting to enforce the dependency. In other words, deletion makes coordinated changes possible by making the dependency between the deletions *local*.

In the preceding sections, we have seen that HS can generate a non-rational mapping by performing deletions that occur on opposite, alternating sides of some center marker. These deletions create context-free nesting dependencies between various portions of the deleted material. Although SL constraints by definition can only enforce dependencies across a bounded distance, we saw that deletion was able to extend the reach of SL constraints by moving far-away material into their domains of influence.

The matching deletion function defined in Section 5 differs qualitatively from the majority-rules deletion mappings presented in

Section 3. Majority-rules deletion reflects the ability of standard OT to perform *global* optimization. Because standard OT does not assume GEN to be limited to one change at a time, faithfulness constraints have the ability to measure the degree to which SRs differ from URs. Thus, in the example from Section 3, MAX is responsible for determining which symbol should be deleted from the input string. This determination itself is not rational, since FSTs cannot distinguish between large numbers of *as* and *bs*. It is not obvious whether or not HS can implement majority-rules deletion using SL constraints because the limited nature of GEN strips faithfulness constraints of the ability to count the number of symbols deleted across the string, so that both markedness constraints and faithfulness constraints are limited to a local domain of influence. However, as Lamont (2018a,b) shows, majority-rules mappings are possible in HS if markedness constraints are given global scope, compensating for the limited power of faithfulness constraints.

While the majority-rules mapping reflects the global nature of optimization in standard OT, the matching deletion function reflects the derivational nature of HS and the propensity of HS derivations to converge to a fixed point.¹² The constraints **baca* and STOP are able to control the deletion process powered by **ab* and **caa* by exploiting the fact that intermediate strings can encode the previous action of the grammar. Thus, despite the limited power of constraints in HS, complex computations are still made possible if state is encoded into the intermediate strings produced in a derivation. This kind of technique has been used in the HS phonology literature to derive certain complex patterns. For example, McCarthy (2008) gives an HS derivation of *rhythmic syncope*, a mapping in which every second vowel is deleted, by first organizing phonemes into two-syllable feet, then assigning stress to the first syllable in each foot, and then deleting the unstressed vowels. This cascade of processes serves to mark up the UR with syllable boundaries, foot boundaries, and stress markers, so that it is possible for an SL constraint to determine by inspection whether or not a vowel belongs to an even-numbered position within the string. While McCarthy presents this analysis of rhythmic syncope as an ad-

¹²Moreton (1999, 2004) shows that derivations driven by OT-style ranked constraint systems must *always* converge to a fixed point.

vantage of HS over standard OT, the construction of Section 5 shows that encoding state in intermediate strings can enable computations that are too complex for phonology.

While this paper has shown that HS is not rational, I have left open the question of finding a suitable limitation of HS that would eliminate the possibility of generating non-rational mappings. Using Abdulla *et al.*'s algorithm on Hao's (2017) model would provide a method to construct an FST for an HS grammar, although there is no guarantee that the algorithm would terminate. Thus, the approach to finite-state OT studied here is similar to Riggle's (2004) OTCA, which does not impose any *a priori* restrictions on standard OT, but also does not have a guarantee of termination. One possibility for a finite-state restriction might be to replace recursive calls to the grammar with a bounded cascade of distinct phonological processes, in the style of McCarthy's implementation of rhythmic syncope. I leave the development of such ideas for future work.

ACKNOWLEDGEMENTS

I would like to thank Ryan Bennett and Robert Frank for suggesting the line of inquiry that led to this paper and for their extensive discussions on this topic. Andrew Lamont, Dustin Bowers, Nicholas Hathaway, and the audience at FSMNLP provided additional useful feedback. Finally, I would like to thank Frank Drewes and the anonymous reviewers for their rigorous evaluation of this work and their guidance in its development.

REFERENCES

- Parosh Aziz ABDULLA, Bengt JONSSON, Marcus NILSSON, and Julien D'ORSO (2002), Regular Model Checking Made Simple and Efficient, in Luboš BRIM, Mojmír KŘETÍNSKÝ, Antonín KUČERA, and Petr JANČAR, editors, *CONCUR 2002—Concurrency Theory: 13th International Conference, Brno, Czech Republic, August 20–23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pp. 116–131, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45694-0, doi:10.1007/3-540-45694-5_9.
- Parosh Aziz ABDULLA, Bengt JONSSON, Marcus NILSSON, and Julien D'ORSO (2003), Algorithmic Improvements in Regular Model Checking, in Warren A. HUNT and Fabio SOMENZI, editors, *Computer Aided Verification: 15th*

International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003. Proceedings, volume 2725 of *Lecture Notes in Computer Science*, pp. 236–248, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45069-6, doi:10.1007/978-3-540-45069-6_25.

James K. BAKER (1975), The DRAGON System—An Overview, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, ISSN 0096-3518, doi:10.1109/TASSP.1975.1162650.

Eric BAKOVIĆ (1999), Assimilation to the Unmarked, in Jim ALEXANDER, Alexis DIMITRIADIS, Na-Rae HAN, Elsi KAISER, Michelle Minnick FOX, Christine MOISSET, and Alexander WILLIAMS, editors, *Proceedings of the 23rd Annual Penn Linguistics Colloquium*, volume 6.1 of *University of Pennsylvania Working Papers in Linguistics*, pp. 1–16, Penn Graduate Linguistics Society, Philadelphia, PA, USA.

Eric BAKOVIĆ (2000), *Harmony, Dominance and Control*, PhD dissertation, Rutgers University, New Brunswick, NJ, USA.

Robert C. BERWICK (1984), Strong Generative Capacity, Weak Generative Capacity, and Modern Linguistic Theories, *Computational Linguistics*, 10(3-4):189–202, ISSN 0891-2017.

Ahmed BOUAJJANI, Bengt JONSSON, Marcus NILSSON, and Tayssir TOULI (2000), Regular Model Checking, in E. Allen EMERSON and Aravinda Prasad SISTLA, editors, *Computer Aided Verification: 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pp. 403–418, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-45047-4, doi:10.1007/10722167_31.

Joan CHEN-MAIN and Robert FRANK (2003), Implementing Faithfulness Constraints in a Finite State Model of Optimality Theory, in Pádraig CUNNINGHAM, Tim FERNANDO, and Carl VOGEL, editors, *Proceedings of the 14th Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 28–33, Trinity College Dublin, Dublin, Ireland.

Noam CHOMSKY (1959), On Certain Formal Properties of Grammars, *Information and Control*, 2(2):137–167, ISSN 0019-9958, doi:10.1016/S0019-9958(59)90362-6.

Noam CHOMSKY and Morris HALLE (1968), *The Sound Pattern of English*, Harper & Row, New York, NY, USA, 1 edition, ISBN 978-0-06-041276-0.

Dennis DAMS, Yassine LAKHNECH, and Martin STEFFEN (2001a), Iterating Transducers, Technical Report TR-ST-01-03, University of Kiel Institut für Informatik und praktische Mathematik, Kiel, Germany.

Dennis DAMS, Yassine LAKHNECH, and Martin STEFFEN (2001b), Iterating Transducers, in Gérard BERRY, Hubert COMON, and Alain FINKEL, editors, *13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*,

Non-rationality of Harmonic Serialism

volume 2102 of *Lecture Notes in Computer Science*, pp. 286–297, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-44585-2, doi:10.1007/3-540-44585-4_27.

Dennis DAMS, Yassine LAKHNECH, and Martin STEFFEN (2002), Iterating Transducers, *The Journal of Logic and Algebraic Programming*, 52–53:109–127, ISSN 1567-8326, doi:10.1016/S1567-8326(02)00025-5.

Matt EDLEFSEN, Dylan LEEMAN, Nathan MYERS, Nathaniel SMITH, Molly VISSCHER, and David WELLCOME (2008), Deciding Strictly Local (SL) Languages, in Jon BREITENBUCHER, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pp. 66–73, College of Wooster, Wooster, OH, USA.

Jason EISNER (2000), Directional Constraint Evaluation in Optimality Theory, in *Proceedings of the 18th Conference on Computational Linguistics*, volume 1, pp. 257–263, Association for Computational Linguistics, Saarbrücken, Germany, ISBN 1-55860-717-X, doi:10.3115/990820.990858.

Jason EISNER (2002), Comprehension and Compilation in Optimality Theory, in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 56–63, Association for Computational Linguistics, Philadelphia, PA, USA, doi:10.3115/1073083.1073095.

Emily ELFNER (2009), Syllabification and Stress-Epenthesis Interactions in Harmonic Serialism, *Rutgers Optimality Archive*, ROA-1047.

Emily ELFNER (2016), Stress-Epenthesis Interactions in Harmonic Serialism, in John J. MCCARTHY, Joe PATER, Vieri SAMEK-LODOVICI, and Armin MESTER, editors, *Harmonic Grammar and Harmonic Serialism*, Advances in Optimality Theory, pp. 261–300, Equinox Publishing, Sheffield, United Kingdom, ISBN 978-1-84553-149-2.

T. Mark ELLISON (1994), Phonological Derivation in Optimality Theory, in *Proceedings of the 15th Conference on Computational Linguistics*, volume 2, pp. 1007–1013, Association for Computational Linguistics, Kyoto, Japan, doi:10.3115/991250.991312.

Robert FRANK and Giorgio SATTA (1998), Optimality Theory and the Generative Complexity of Constraint Violability, *Computational Linguistics*, 24(2):307–315, ISSN 0891-2017.

Dale GERDEMANN and Mans HULDEN (2012), Practical Finite State Optimality Theory, in *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pp. 10–19, Association for Computational Linguistics, San Sebastián, Spain.

Dale GERDEMANN and Gertjan VAN NOORD (2000), Approximation and Exactness in Finite State Optimality Theory, in Jason EISNER, Lauri KARTTUNEN, and Alain THÉRIAULT, editors, *Finite-State Phonology: Proceedings*

of the *Fifth Workshop on the ACL Special Interest Group in Computational Phonology*, pp. 34–45, Association for Computational Linguistics, Luxembourg City, Luxembourg.

Kenneth HALE (1973), Deep-Surface Canonical Disparities in Relation to Analysis and Change: An Australian Example, in Thomas A. SEBEOK, editor, *Current Trends in Linguistics*, volume 8: Linguistics in Oceania, pp. 401–458, Mouton, The Hague, Netherlands.

Yiding HAO (2017), Harmonic Serialism and Finite-State Optimality Theory, in Frank DREWES, editor, *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing*, pp. 20–29, Association for Computational Linguistics, Umeå, Sweden, doi:10.18653/v1/W17-4003.

Jeffrey HEINZ (2009), On the Role of Locality in Learning Stress Patterns, *Phonology*, 26(2):303–351, ISSN 0952-6757, doi:10.1017/S0952675709990145.

Jeffrey HEINZ (2014), Culminativity Times Harmony Equals Unbounded Stress, in Harry VAN DER HULST, editor, *Word Stress: Theoretical and Typological Issues*, pp. 255–275, Cambridge University Press, Cambridge, United Kingdom, ISBN 978-1-107-03951-3, doi:10.1017/CBO9781139600408.012.

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, Association for Computational Linguistics, Portland, OR, USA.

Jeffrey Nicholas HEINZ (2007), *Inductive Learning of Phonotactic Patterns*, PhD dissertation, University of California, Los Angeles, Los Angeles, CA, USA.

Patrick W. HOHEPA (1967), *A Profile Generative Grammar of Maori*, number 20 in Indiana University Publications in Anthropology and Linguistics, Waverly Press, Baltimore, MD, USA.

Frederick JELINEK (1976), Continuous Speech Recognition by Statistical Methods, *Proceedings of the IEEE*, 64(4):532–556, ISSN 0018-9219, doi:10.1109/PROC.1976.10159.

C. Douglas JOHNSON (1970), *Formal Aspects of Phonological Description*, PhD dissertation, University of California, Berkeley, Berkeley, CA, USA.

C. Douglas JOHNSON (1972), *Formal Aspects of Phonological Description*, Mouton, The Hague, Netherlands.

Bengt JONSSON and Marcus NILSSON (2000), Transitive Closures of Regular Relations for Verifying Infinite-State Systems, in Susanne GRAF and Michael SCHWARTZBACH, editors, *6th International Conference, TACAS 2000, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25–April 2, 2000, Proceedings*, volume 1785 of *Lecture Notes in Computer Science*, pp. 220–235, Springer Berlin Heidelberg, Berlin, Germany, ISBN 978-3-540-46419-8, doi:10.1007/3-540-46419-0_16.

Non-rationality of Harmonic Serialism

- Ronald M. KAPLAN and Martin KAY (1994), Regular Models of Phonological Rule Systems, *Computational Linguistics*, 20(3):331–378, ISSN 0891-2017.
- Lauri KARTTUNEN (1998), The Proper Treatment of Optimality in Computational Phonology, in *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pp. 1–12, Association for Computational Linguistics, Ankara, Turkey.
- Regine LAI (2015), Learnable vs. Unlearnable Harmony Patterns, *Linguistic Inquiry*, 46(3):425–451, ISSN 0024-3892, doi:10.1162/LING_a_00188.
- Yeeking Regine LAI (2012), *Domain Specificity in Learning Phonology*, PhD dissertation, University of Delaware, Newark, DE, USA.
- Andrew LAMONT (2018a), Precedence Is Pathological, conference presentation at Phonology in the Northeast, Cambridge, MA, USA.
- Andrew LAMONT (2018b), Precedence Is Pathological: The Problem of Alphabetical Sorting, conference presentation at the West Coast Conference on Formal Linguistics, Los Angeles, CA, USA.
- Bruce T. LOWERRE (1976), *The HARPY Speech Recognition System*, PhD dissertation, Carnegie-Mellon University, Pittsburgh, PA, USA.
- John J. MCCARTHY (2007), *Hidden Generalizations: Phonological Opacity in Optimality Theory*, Advances in Optimality Theory, Equinox Publishing, Sheffield, United Kingdom, ISBN 978-1-84553-051-8.
- John J. MCCARTHY (2008), The Serial Interaction of Stress and Syncope, *Natural Language & Linguistic Theory*, 26(3):499–546, ISSN 1573-0859, doi:10.1007/s11049-008-9051-3.
- John J. MCCARTHY (2009), Harmony in Harmonic Serialism, *Rutgers Optimality Archive*, ROA-1009.
- John J. MCCARTHY (2010), An Introduction to Harmonic Serialism, *Language and Linguistics Compass*, 4(10):1001–1018, ISSN 1749-818X, doi:10.1111/j.1749-818X.2010.00240.x.
- John J. MCCARTHY and Alan PRINCE (1994), The Emergence of the Unmarked: Optimality in Prosodic Morphology, in Mercè GONZÁLEZ, editor, *Proceedings of the North East Linguistics Society 24*, pp. 333–379, GLSA Publications, Amherst, MA, USA.
- John J. MCCARTHY and Alan PRINCE (1995), Faithfulness and Reduplicative Identity, *University of Massachusetts Occasional Papers in Linguistics*, 18: Papers in Optimality Theory:249–384.
- Kevin MCMULLIN and Gunnar Ólafur HANSSON (2016), Long-Distance Phonotactics as Tier-Based Strictly 2-Local Languages, in *Proceedings of the 2014 Annual Meeting on Phonology*, Proceedings of the Annual Meetings on Phonology, pp. 13–24, Linguistic Society of America, Cambridge, MA, USA, doi:10.3765/amp.v2i0.3750.

- Kevin James MCMULLIN (2016), *Tier-Based Locality in Long-Distance Phonotactics: Learnability and Typology*, Ph.D. thesis, University of British Columbia, Vancouver, Canada.
- Robert MCNAUGHTON and Seymour A. PAPERT (1971), *Counter-Free Automata*, number 65 in Research Monograph, MIT Press, Cambridge, MA, USA, ISBN 978-0-262-13076-9.
- Elliott MORETON (1999), Non-Computable Functions in Optimality Theory, *Rutgers Optimality Archive*, ROA-364.
- Elliott MORETON (2004), Non-Computable Functions in Optimality Theory, in John J. MCCARTHY, editor, *Optimality Theory in Phonology: A Reader*, pp. 141–164, Blackwell Publishing, Oxford, United Kingdom, ISBN 978-0-470-75617-1, doi:10.1002/9780470756171.ch6.
- Joe PATER (2009), Weighted Constraints in Generative Linguistics, *Cognitive Science*, 33(6):999–1035, ISSN 1551-6709, doi:10.1111/j.1551-6709.2009.01047.x.
- Christopher POTTS, Joe PATER, Karen JESNEY, Rajesh BHATT, and Michael BECKER (2010), Harmonic Grammar with Linear Programming: From Linear Systems to Linguistic Typology, *Phonology*, 27(1):77–117, ISSN 1469-8188, doi:10.1017/S0952675710000047.
- Alan PRINCE (2002), Arguing Optimality, *Rutgers Optimality Archive*, ROA-562.
- Alan PRINCE (2003), Arguing Optimality, *University of Massachusetts Occasional Papers in Linguistics*, 26.
- Alan PRINCE and Paul SMOLENSKY (1993), Optimality Theory: Constraint Interaction in Generative Grammar, Technical Report 2, Rutgers University, New Brunswick, NJ, USA.
- Alan PRINCE and Paul SMOLENSKY (2004), *Optimality Theory: Constraint Interaction in Generative Grammar*, Blackwell Publishing, Malden, MA, USA, ISBN 978-1-4051-1932-0.
- Kathryn PRUITT (2008), Iterative Foot Optimization and Locality in Stress Systems, *Rutgers Optimality Archive*, ROA-999.
- Kathryn Ringle PRUITT (2012), *Stress in Harmonic Serialism*, PhD dissertation, University of Massachusetts Amherst, Amherst, MA, USA.
- Jason Alan RIGGLE (2004), *Generation, Recognition, and Learning in Finite State Optimality Theory*, PhD dissertation, University of California, Los Angeles, Los Angeles, CA, USA.
- James ROGERS and Geoffrey K. PULLUM (2011), Aural Pattern Recognition Experiments and the Subregular Hierarchy, *Journal of Logic, Language and Information*, 20(3):329–342, ISSN 1572-9583, doi:10.1007/s10849-011-9140-2.
- Vieri SAMEK-LODOVICI and Alan PRINCE (1999), Optima, *Rutgers Optimality Archive*, ROA-363.

Non-rationality of Harmonic Serialism

Vieri SAMEK-LODOVICI and Alan PRINCE (2002), *The Fundamental Properties of Harmonic Bounding*, Technical Report TR-71, Rutgers Center for Cognitive Science, Piscataway, NJ, USA.

Michael SIPSER (2013), *Introduction to the Theory of Computation*, Cengage Learning, Boston, MA, USA, 3 edition, ISBN 978-1-133-18781-3.

Rachel WALKER (2008), *Gradualness and Fell-Swoop Derivations*, conference presentation at the UCSC Graduate Alumni Conference, Santa Cruz, CA, USA.

Rachel WALKER (2010), *Nonmyopic Harmony and the Nature of Derivations*, *Linguistic Inquiry*, 41(1):169–179, ISSN 00243892, doi:10.1162/ling.2010.41.1.169.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



Learning cross-lingual phonological and orthographic adaptations: a case study in improving neural machine translation between low-resource languages

Saurav Jha¹, Akhilesh Sudhakar², and Anil Kumar Singh²

¹ MNNIT Allahabad, Prayagraj, India

² IIT (BHU), Varanasi, India

ABSTRACT

Out-of-vocabulary (OOV) words can pose serious challenges for machine translation (MT) tasks, and in particular, for low-resource language (LRL) pairs, i.e., language pairs for which few or no parallel corpora exist. Our work adapts variants of seq2seq models to perform *transduction* of such words from Hindi to Bhojpuri (an LRL instance), learning from a set of cognate pairs built from a bilingual dictionary of Hindi-Bhojpuri words. We demonstrate that our models can be effectively used for language pairs that have limited parallel corpora; our models work at the character level to grasp phonetic and orthographic similarities across multiple types of word adaptations, whether synchronic or diachronic, loan words or cognates. We describe the training aspects of several character level NMT systems that we adapted to this task and characterize their typical errors. Our method improves BLEU score by 6.3 on the Hindi-to-Bhojpuri translation task. Further, we show that such transductions can generalize well to other languages by applying it successfully to Hindi-Bangla cognate pairs. Our work can be seen as an important step in the process of: (i) resolving the OOV words problem arising in MT tasks; (ii) creating effective parallel corpora for resource constrained languages; and (iii) leveraging the enhanced semantic knowledge captured by word-level embeddings to perform character-level tasks.

Keywords: neural machine translation, Hindi, Bhojpuri, word transduction, low resource language, attention model

With recent advances in the field of machine translation (MT) – and in neural machine translation (NMT) in particular – there has been an increasing need to shift focus to out-of-vocabulary (OOV) words (Wu *et al.* 2016; Sennrich *et al.* 2016). In the case of low resource languages (LRLs), which lack linguistic resources such as parallel corpora, most words are OOV words; this is problematic. Current data-intensive translation systems work poorly with OOV words for such languages, purely because of a severe lack of resources. Hence, for such languages, it becomes necessary to deal with OOV words in specific ways, outside the ambit of general-purpose NMT systems. Even in the case of resource-rich languages, methods to deal with OOV words are still being actively researched (Pham *et al.* 2018; Luong and Manning 2016).

Many approaches (as elaborated upon in Section 2) have been investigated to deal with the OOV problem. In this paper, we use the method of ‘transduction’, learned from a dictionary of cognate word pairs from Hindi and Bhojpuri. The fundamental guiding principle of our approach is the fact that Bhojpuri and Hindi are closely related languages, and hence have a good amount of vocabulary overlap, while sharing orthographic and phonetic traits. These two languages have common ancestors, and both of them employ the orthographically shallow alpha-syllabic Devanagari script. Tracing the origin of a considerable portion of the modern Bhojpuri vocabulary could weakly suggest that many of Hindi words got adapted to the local Bhojpuri phonology with the passage of time. This is a well-known phenomenon and it can be observed in other pairs of closely related languages (Macedonian – Bulgarian; Spanish – Catalan; Turkish – Crimean Tatar; Czech – Slovak; Irish – Scottish Gaelic) which share a close ancestor within the language family they belong to.

The Indian linguistic space, as reported by Grierson (1928), has 179 independent languages and 544 dialects. Similarly, the survey of Mahapatra *et al.* (1989) demonstrates that there are at least 50 Indian languages in which writing and publishing are done in substantial quantity. However, a majority of these languages lack proper linguistic resources. Hindi – being the *lingua franca* of the ‘Hindi belt’ (most parts of the north) of India – is a commonly spoken language in more than

10 states and has (according to one of the views¹) seven major closely related languages, often called ‘dialects’ or ‘sub-languages’, (namely, *Awadhi, Bagheli, Bhojpuri, Bundeli, Haryanvi, Kanauji and Khari Boli*) (Mishra and Bali 2011). Despite it having 33 million native speakers in India and over 6 million native speakers outside India,² Bhojpuri still suffers from the lack of language resources. So far, very little work has been done towards developing language resources (Singh and Jha 2015), resulting in scarcity of resources such as a Bhojpuri lexical database or parallel corpus that could have made state-of-the-art machine translation (MT) systems accessible to this language.

Due to the lack of such resources, neither traditional phrase based machine translation (PBMT) (Chiang 2005) nor NMT (Bahdanau *et al.* 2014) are feasible for Bhojpuri, as such approaches require large parallel corpora. In their recent work, Sharma and Singh (2017) introduce a ‘word transduction’ approach to deal with the presence of unknown (or out of vocabulary) words for MT systems involving such resource-scarce languages. The concept of word transduction is somewhat similar to Hajič (2000), where the author suggests that the use of word-for-word translation for very closely related languages provides a good solution for MT of such language pairs.

Furthermore, for the task of language translation, it is necessary to take into account the fact that not all languages possess the same morphological features. For example, Finnish has more than 2000 possible inflected noun forms (Ekman and Järvelin 2007); Hindi and Bhojpuri have more than 40 inflectional forms (Singh and Sarma 2010); while English has a mere 7–8 (these numbers indicate the different possible valid combinations of morphological tags that nouns can possess). Therefore, a good MT system designed for such morphologically rich languages must be intricate enough to deal with their diverse inflectional morphology. In order to address this issue, we adapt character-level NMT systems to our task in order to ex-

¹ There is no consensus about the meaning of the word ‘Hindi’ and so different scholars have different views. For example, some other sub-languages like Rajasthani, Maithili and Magahi are also often included in the Hindi spectrum. However, the usual meaning of the word ‘Hindi’ in literature refers to standard Hindi, whose base is Khari Boli and which is an official language of India.

² http://www.censusindia.gov.in/Census_Data_2001/Census_Data_Online/Language/data_on_language.aspx

exploit morphological information encoded in inter-character interactions and intra-word patterns. As observed by (Nakov and Tiedemann 2012): “character-level models combine the generality of character-by-character transliteration and lexical mappings of larger units that could possibly refer to morphemes, words or phrases, as well as to various combinations thereof” (Nakov and Tiedemann 2012). We also introduce a novel pre-trained character-level embedding (Bojanowski *et al.* 2017) for Devanagari alphabets derived from the 300 dimensional Hindi fastText embeddings.³

As regards the phonetic considerations of transduction, we make use of the fact that Hindi and Bhojpuri have a phonetic writing system, meaning there is an almost one-to-one mapping between phonemes (pronunciation) and graphemes (transcription). This is due to the fact that they both derive from common ancestor languages such as Prakrit and then Apabhramsha (Choudhury 2003). Hence, it suffices to work in either one of the spaces – orthographic or phonetic, and we choose the orthographic space since it does away with the need to convert the graphemes of text to and from phonemes.

Although Bhojpuri phonology is close to that of Hindi, it is not the same. There are notable differences between the two. While Hindi has a symmetrical ten vowel system, Bhojpuri has six vowel phonemes and ten vocoids. Similarly, Hindi has 37 consonants (including those inherited from earlier Indo-Aryan and those from loan words), whereas Bhojpuri has 31 consonant phonemes and 34 contoids. As is usual with any pair of languages, there are many phoneme sequences which are allowed in Hindi, but not found in Bhojpuri and vice-versa. This will be evident in examples given in the paper later. More details about the Bhojpuri phonology are available in the article by Trammell (1971).

1.1 *A note on Roman representations and English glosses of Hindi*

Throughout this paper, we have used the WX notation (Gupta *et al.* 2010) to represent (in a transliteration-like fashion) Hindi and Bhojpuri characters in English, for the benefit of readers who are not familiar with the Devanagari script. A ready reference table of the WX notation can be found in its Wikipedia page.⁴ Every non-English word

³<https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md>

⁴https://en.wikipedia.org/wiki/WX_notation

used in this paper is followed by its WX notation in square brackets ([]) and its italicized English gloss, in parenthesis ().

1.2 *Transduction and translation*

Our usage of the word ‘transduction’ distinguishes it from translation, in that transduction is a task which is trained exclusively on cognates, and in that sense, the dataset it uses is a subset of the dataset that a translation system would use. Cognates are word pairs that not only have similar meaning but are also phonetically (and, in our case, orthographically) similar. The underlying observation that guides the usage of our proposed method of ‘transduction’ of OOV words as a possible substitute for their translation is as follows:

As stated earlier, Bhojpuri is a language closely related to Hindi. In the case of an OOV Hindi word (or any Hindi word for that matter), there is a good chance that the Bhojpuri translation of the word is a cognate of the Hindi word adapted to the phonological and orthographic space of Bhojpuri due to the presence of borrowed words, common origins, geographic proximity, socio-linguistic proximity, etc. A phonemic study of Hindi and Bundeli (Acharya 2015), mainly focusing on the prosodic features and the syllabic patterns of these two languages, (unsurprisingly) concluded that the borrowing of words from Hindi to Bundeli generally follows certain (phonological) rules. For instance if a word in Hindi begins with the character य [ya], it is replaced by character ज [ja] in its Bundeli equivalent as यजमान [yajamaan] (*host*) becomes जजमान [jjamaan], यमुना [yamunaa (*name of a river*)] becomes जमुना [jamunaa], etc. We observe that a similar process happens for Hindi-Bhojpuri. This category of word pairs is our main motivation behind the work described in this paper.

Our model is agnostic to what sort of words are considered to be OOV (based on their unigram probabilities, their parts-of-speech (POS), or whether named entities, etc.) because the above assumptions hold uniformly across the language pair. Section 2 specifies some of the metrics that have been used earlier to identify OOV words in related work.

Further, the above assumption (of transduction improving overall translation performance) has been demonstrated to be valid in the case of many closely related language pairs, in a number of previous works. For instance, Kondrak *et al.* (2003) extracted a parallel list of

cognate word pairs and re-appended them to the parallel list of all word translations, thereby increasing the training weights of cognate words. Giving added importance to these cognate words, “resulted in a 10% reduction of the word alignment error rate, from 17.6% to 15.8%, and a corresponding improvement in both precision and recall.” (Kon-drak *et al.* 2003). Mann and Yarowsky (2001) used cognates to expand translation lexicons, Simard *et al.* (1993) to align sentences in a parallel translation corpora, and Al-Onaizan *et al.* (1999) used cognate information to improve statistical machine translation.

Finally, transduction induces less sparsity in the model as compared to translation, because the hypothesis space is restricted to only functions that map words to their possible cognates. For closely related languages, the added reduction in sparsity also comes from the fact that there are consistent variations between how a source word transduces to its cognate target word. Hence, transduction is a task that performs better with a small training set than translation would when using a similarly complex model. This reduced sparsity enables transduction models to perform well on OOV words.

The ensuing section provides background about NMT systems and the manner in which we have adapted them to our task.

1.3 *Neural machine translation*

Neural machine translation (NMT) has delivered promising results in large-scale translation tasks such as Chinese-to-English (Tu *et al.* 2017) and English-to-French (Chen and Wu 2017). Initially, NMTs were used as a sub-component of the PBMT system such as for generating the conditional probabilities of phrase pairs (Cho *et al.* 2014), for generating (machine learning) features for the PBMT, or for re-ranking the *n*-best hypotheses produced by the system (Kalchbrenner and Blunsom 2013; Sutskever *et al.* 2014a). Such combined systems produced state-of-the-art results. One most appealing feature of NMTs is that they are largely memory efficient. Unlike PBMT systems, an NMT system does not require keeping track of phrase pairs or language models. Additionally, the work of Bentivogli *et al.* (2016) pointed out that NMTs offer a range of other superior attributes including:

- generation of outputs that require considerably lower post-edit efforts

- better translation quality in terms of BLEU score, Translation Edit Rate, and good performance on longer sentences
- fewer and/or less-severe errors in terms of morphology and word order

Most NMT systems today make use of the encoder-decoder based approach (e.g. Forcada and Āeco 1997; Cho *et al.* 2014; Kalchbrenner and Blunsom 2013; Sutskever *et al.* 2014a), which consists of two recurrent neural networks (or their variants). The first encodes a variable-length source token x into a fixed length vector and the second decodes the vector into a variable-length target token y . NMT approaches were initially designed to work at the word-level and translate sentences. However, noting the encouraging results of adapting NMTs to character-level translation by Vilar *et al.* (2007), we adapt NMTs to our character-level transduction. Figure 1 shows the architecture of such an encoder-decoder based NMT system performing character-level transduction. The model is trained over a parallel corpus to learn the maximum conditional probability of y given x , i.e., $\arg \max_y p(y | x)$. Once trained, the model can then be used to generate the corresponding transduction for a given source word.

However, the performance of NMT degrades largely in the case of longer length sequences (words, in our case) due to the vanishing

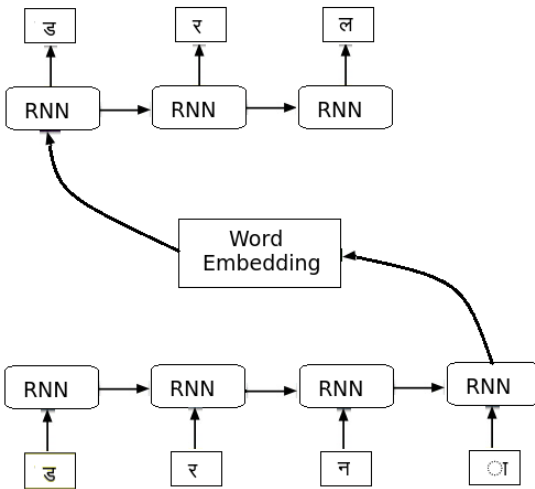


Figure 1:
Encoder-decoder network
architecture transducing the Hindi
word डरना [darnA] (*to be scared*)
to Bhojpuri डरल [darala]

gradient problem (Bengio *et al.* 1994) arising during the training of the underlying RNN. So far, the use of an attention mechanism, as stated by Bahdanau *et al.* (2014), Luong *et al.* (2015a), Vinyals *et al.* (2015) and Yang *et al.* (2016) has been the most plausible solution to the aforementioned problem for RNNs and its variants. The concept of ‘attention’ takes into account the fact that in the task of translation, different tokens in a sequence are differentially informative, with the information carried by them being highly context dependent. Thus, for predicting each corresponding token, the model looks at the current context of the source token that is relevant to predicting the target token.

1.4

Summary

We adapt NMT models to perform ‘transduction’ of a Hindi word to a Bhojpuri word. These word-transduction models work with characters as the fundamental units. They are trained on Hindi-Bhojpuri cognate pairs. This task is important because it helps to solve the OOV problem in larger downstream tasks, the most prominent example of which is machine translation for low resource languages. To improve machine translation of Hindi to Bhojpuri, we first identify OOV words in Hindi texts and then use our model to transduce them to their Bhojpuri counterparts. All other words are translated, and not transduced. Using such separate treatment of OOV words, we obtain improvements in BLEU score with respect to the originally translated texts. The section on Related Work (Section 2) elaborates previous approaches to transduction and the OOV problem.

2

RELATED WORK

A number of methods have been proposed to handle the OOV problems in machine translation of unknown or rare words. Luong *et al.* (2015a) simply use a shortlist of 30,000 most frequent words and map all other less frequent words to an UNK (unknown) token. Sutskever *et al.* (2014b) use a vocabulary of 80,000 words and achieve better performance. However, any UNK-based approach is problematic because in larger sentences, UNK tokens heavily degrade performance (Cho *et al.* 2014; Jean *et al.* 2015) make model specific improvements, using a smaller batch for normalization and including only frequent

words in the denominator of this normalization. They fall back to other translation and alignment models to replace UNK tokens.

Other approaches to handle OOV words include using a back-off dictionary look-up (Jean *et al.* 2015; Luong *et al.* 2015b) but as observed by Sennrich *et al.* (2016), these techniques make impractical assumptions. One such assumption is a one-to-one source – target word correspondence, which is unwarranted. Further, some of these methods assume the existence of a parallel corpus of source-target word pairs, which is not always available in the case of low resource languages. Sennrich *et al.* (2016), in turn, use a Byte-Pair Encoding for transduction, which is very similar to character-level encoding of sequences as strings of characters.

We also borrow ideas from previous approaches that have used cognates. Simard *et al.* (1993) use cognates to align sentences in a parallel corpus and report 97.6% accuracy on the alignments obtained, when compared to reference alignments. Mann and Yarowsky (2001) use cognates extracted based on edit distances for inducing translation lexicons based on transduction models. Scannell (2006) presents a detailed study on translation of a closely related language pair, Irish-Scottish Gaelic. They learn transfer rules based on alignment of cognate pairs, and use these rules to generate transductions on new words. They use a fine-grained cognate extraction method, by first editing Scottish words to ‘seem like’ Gaelic words, and then using edit string similarity on the new word pairs and choosing only close words with the additional constraint that both words in the pair should share a common English translation. However since we use linguistic experts to extract cognates from our dataset, we do not need to encode string similarity measures explicitly to extract cognates.

We borrow insights from character-level machine transliteration and translation models that have been proposed in the past, as transduction can be viewed as a variation of transliteration (which is, in turn, viewed as character-level translation in many works), albeit working within the same script. Alternatively, it can also be thought of simply as a translation of ‘true friend’ cognates.

Vilar *et al.* (2007) work on transliteration at the character level (and translation at the word level) to build a combined system that shows increasing gains over just the word-level system, as the corpus size grows smaller. This is because the character-level transliter-

ation takes into account the added morphological information such as base forms and affixes. Tiedemann (2012) experimented with different types of alignment methods and learning models, and showed that in each type of method, there exists at least one character-level model that performs better than word-level models (in the case of closely related language pairs). Denoual and Lepage (2006) also show merits of using characters as appropriate translations, and highlight issues with making assumptions about words being natural units for the task. Finch and Sumita (2009) view transliteration as a character-level machine translation, and use Phrase-Based SMT for bidirectionally encoding source sequences. They observe the lack of necessity to model phonetics of source or target language, due to the use of direct transformations. One point of difference between some of the related work on cognates and ours is that we do not perform context-sensitive transduction simply due to lack of annotated data that is context-sensitive.

3

METHODOLOGY

We run experiments on four different benchmark encoder-decoder networks, namely:

- a simple sequence-to-sequence model (Cho *et al.* 2014) – abbreviated as ‘seq2seq’.
- the alignment method (Bahdanau *et al.* 2014) incorporated with the seq2seq model – abbreviated as ‘AM’.
- the Hierarchical Attention Network (Yang *et al.* 2016) incorporated with the seq2seq model – abbreviated as ‘HAN’.
- the Transformer Network (Vaswani *et al.* 2017) solely based on attention – abbreviated as ‘TN’.

In the basic seq2seq RNN encoder-decoder model (Cho *et al.* 2014) we incorporate a ‘peek’ at the context vector at every time step. However, the model performed poorly on this translation task, with the validation accuracy plateauing at a low value early on in the training process. Section 3.1, Section 3.2 and Section 3.3 describe the models – AM, HAN and TN, respectively – in detail.

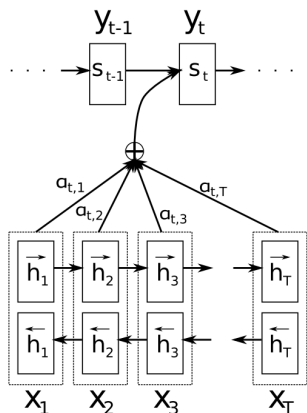


Figure 2:
Schematic of the alignment model
adapted from Bahdanau *et al.* (2014)

3.1 Alignment model (AM)

As shown in Figure 2, the alignment model (AM) proposed by Bahdanau *et al.* (2014) facilitates searching through the source sequence during the decoding phase using a unique context vector for each token. Specifically, given a translation y_i and the source sequence \mathbf{x} , the decoder decomposes the conditional probability over all the previously predicted tokens (y_1, \dots, y_{i-1}) as:

$$(1) \quad p(y_i | y_1, \dots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i),$$

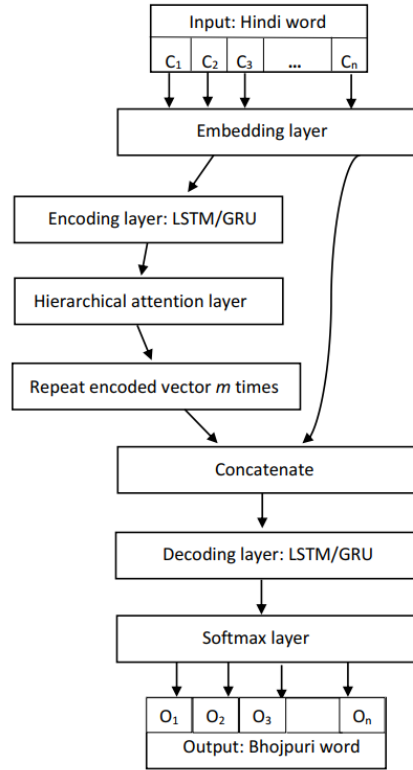
where s_i is the hidden state of the decoder model computed for time i , c_i is the distinct context vector for each target token y_i and g is a non-linear function that outputs the probability of y_i being the correct translation at time i .

In addition to the use of a unique context vector for each decoding time step, all hidden states computed so far contribute to the context vector c_i with weight α .

$$(2) \quad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

α thus serves as a normalized importance weight, measuring the degree of importance of the context tokens around position j in predicting the translation of the current source token at the output position i (Bahdanau *et al.* 2014). Figure 2 shows the architecture of the encoder-decoder network incorporating the alignment-based attention decoder.

Figure 3:
Encoder–decoder network architecture with HAN: n and m denote the number of characters in input and output words respectively



3.2 Hierarchical attention network (HAN)

Proposed by Yang *et al.* (2016), Hierarchical Attention Networks (HAN) exploit the hierarchical nature of documents (i.e., characters form words, words form sentences and sentences form a document) and are comprised of two levels of attention mechanisms (Bahdanau *et al.* 2014; Lai *et al.* 2015) – the first at the word level while the other at the sentence level. In our case, the former attention can be thought of as being effective at the character level, while the latter at the word level, thus allowing the model (Figures 3 and 4) to discover the amount of attention required to be paid to the individual characters and words to form a character-level transduction.

3.3 Self-attentional transformer network (TN)

The Transformer Network (Vaswani *et al.* 2017) consists of an encoder made up of a stack of six identical layers. Each layer further con-

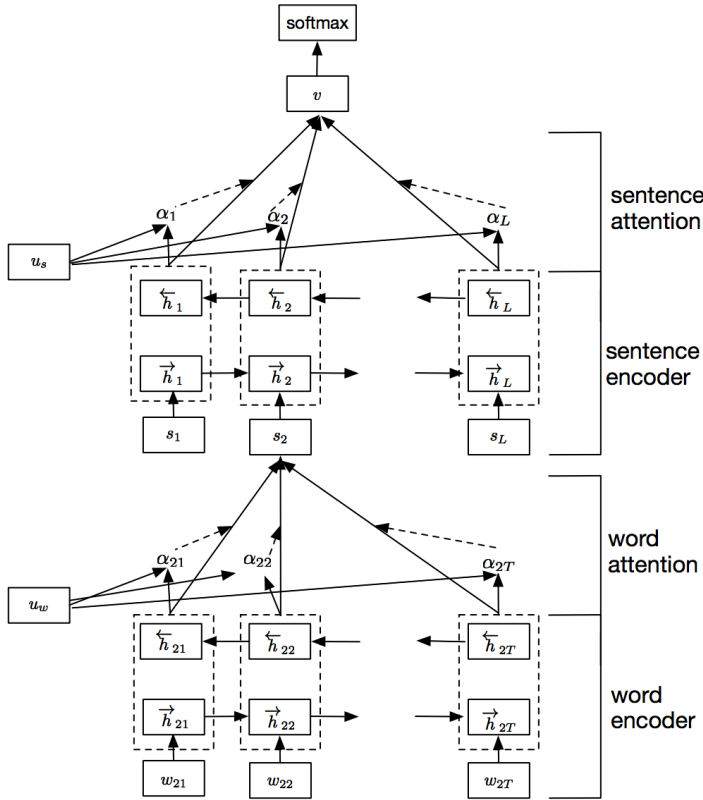
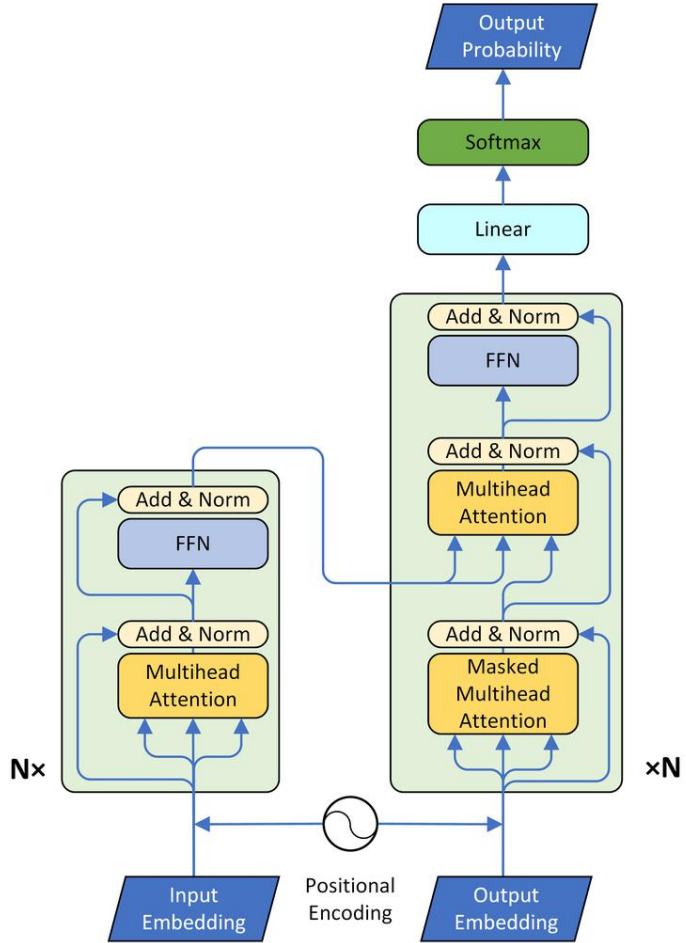


Figure 4:
The HAN framework
adapted from Yang *et al.*
(2016)

sists of two sub-layers: a *multi-head self-attention* and a simple *position-wise fully connected feed-forward network (FFN)*. The decoder too has a similar architecture except for an additional sub-layer performing multi-head attention over the output of the encoder stack. Both the encoder and the decoder unit employ a residual connection (He *et al.* 2016) in between their respective sub-layers, followed by layer normalization (Ba *et al.* 2016).

As shown in Figure 5, the *positional embeddings* serve to make the representation at time step i independent from the other time steps. The *multi-head attention* layer serves to replace recurrent dependencies by repeatedly applying self-attention over the same inputs using separate parameters (attention heads) followed by combining the results. This combination acts as an alternative to applying a single pass of attention with more parameters so that the model can easily learn

Figure 5:
The transformer
architecture as described
in Vaswani *et al.* (2017)
and adapted from Li *et al.*
(2018)



to attend to different types of relevant information in parallel with each head. In other words, the decoder can now use multiple encoder-attention mechanisms in each of its layers resulting in a significantly faster training than architectures based on recurrent or convolutional layers. Inspired by the success of the Transformer in sequence generation tasks (such as achieving state-of-the-art results on both WMT2014 English-German and WMT2014 English-French translation tasks⁵), we use the Transformer Network in our task.

⁵<http://www.statmt.org/wmt14/>

EXPERIMENTS

For the AM and HAN models, we consider various parameters while training, such as LSTM/GRU as encoding/decoding units, sequence chunking and batch sizes, optimization methods, regularization, and we report that there is a huge variance in the transduction performance depending on the combinations of the parameters used (for details of these experiments, the interested reader may refer to the Appendix section in the pre-print of this work⁶). As regards the TN model, due to computational and implementation limitations, we could not perform such an extensive hyperparameter search. Nevertheless, we take care of the basic parameter settings (as suggested in Popel and Bojar 2018) by limiting our model to a single-GPU base, setting batch size to 512, maximum sequence length to the length of the longest token in the parallel corpus followed by a final averaging of the last 6 training checkpoints while leaving the learning rate and the number of warm up steps at their default values. Finally, the extracted character embeddings are used to train the AM and HAN models while the TN model is left void of these mainly because of its inherent dependency on segmenting the training tokens into semantically useful sub-tokens which are hard to be reproduced in varying experiments (Popel and Bojar 2018), and thus cannot be easily assigned such embeddings. We compare the results of the TN model with the best results (among various hyperparameter settings) of the AM and HAN models.

The implementation of the Alignment model (AM) and the Hierarchical attention network (HAN) is based on Keras-2.0.6 (Chollet *et al.* 2015); that of the Transformer Network (TN) is based on tensorflow-1.4.1 (Abadi *et al.* 2015) and tensor2tensor-1.4.3.⁷ Experiments were run on x86_64 GNU/Linux with 8G memory, using one NVIDIA GeForce 840M with CUDA v8.0.61, and Python 3.5.2+.

4.1

Dataset

In order to be able to compare our results with the state of the art (SOTA) – described in Section 4.3 – we use the same dataset as Sharma and Singh (2017), which was the state-of-the-art method at the time of writing. This dataset consists of 4220 Hindi-Bhojpuri word

⁶<https://arxiv.org/ftp/arxiv/papers/1811/1811.08816.pdf>

⁷<https://github.com/tensorflow/tensor2tensor>

cognate pairs chosen from a pre-compiled lexicon of Hindi-Bhojpuri word translations. This dataset was compiled by three linguistic experts (who are native speakers of both Bhojpuri and Hindi) who came to consensus on the annotations. Cognate pairs were identified from this set using domain expertise by linguistic experts. This method of cognate extraction is in contrast to possibly sub-optimal rule-based and similarity-based approximations, such as those used by Mann and Yarowsky (2001). In summary, the training data has Hindi-Bhojpuri word pairs, each of which comprises a Hindi and a Bhojpuri word that have the same meaning, as well as similar pronunciations. We split the dataset into 3:1 ratio for training and testing our models. A validation split of 0.1 was further made on the train set comprising of 3165 word pairs. This test is held-out and is used for reporting only the final results. All hyperparameter tuning is done on the validation set. The validation set is not fixed and the training is cross-validated, with a new random validation-set being used at each iteration of tuning. Table 1 consolidates these statistics. We perform a random shuffle of the train and validation set prior to each training epoch.

Table 1:
Transduction
corpus statistics

	Training set	Validation set	Test set
Total number of words	2849	316	1055
% of words of full dataset	67.5%	7.5%	25%

4.1.1 Semantic ambiguity when selecting cognates

It is worthwhile to mention that not all Hindi words in the training dataset have just one possible corresponding Bhojpuri cognate. Around 3.85% of Hindi words in the Hindi-Bhojpuri cognate pairs have more than one corresponding Bhojpuri cognate. The same applies the other way too, i.e., not all Bhojpuri words have just one possible corresponding Hindi cognate. Around 3.37% of Bhojpuri words in the Hindi-Bhojpuri cognate pairs have more than one corresponding Hindi cognate. When either a Hindi word or a Bhojpuri has multiple corresponding cognates, only two possible cases arise, as described next.

1. The multiple possible cognates have the same semantic sense and are different in only surface forms (spellings). For instance, the Bhojpuri word हलुवा [haluVA] (*a sweet dish*) has two Hindi cognates – हलवा [halVA] and हलवा [halaV] – both meaning the same, but differing in surface forms due to the presence of the diacritic in one and absence in the other. The diacritic is a character explained in Section 6.1.1.
2. They are different inflected forms of the same root word. For instance, the Bhojpuri word राखल [rAKala] (*kept, or to keep*) has two Hindi cognates – रखा [raKA] (*kept*) and रखना [raKanA] (*to keep*) – both of which are different inflected forms of the root word रख [raKa] (*keep*).

Since multi-cognate words together form only a small percentage of the dataset, they are not accorded any special treatment, and the model learns from the multiple forms.

We plan to work more on the deeper change in ambiguity from source to target word in the case of ‘true friend’ cognates, particularly for the Hindi-Bhojpuri language pair.

4.2 Evaluation measures

We report the accuracy of each experiment using the BLEU score and Levenshtein distance-based string similarity (SS) measure, as in Equation 3.⁸ After obtaining the optimum hyperparameter set for AM and HAN, we compare the word accuracy (WA, Equation 4) report defined by the percentage of correctly translated words for all the models including the SOTA. SS, WA and BLEU score formulae for two arbitrary strings ‘s1’ and ‘s2’ are given below. The averaged score across the validation/test set are reported in the tables in ensuing sections. We employ the character n -grams version of BLEU score (as used in Denoual and Lepage 2005) as our work is at the word-level, instead of document-level. Using this metric also alleviates the comparison with other state-of-art transduction methods since other popular metrics such as CHRF and TER correlate well with the character n -gram version of BLEU score (as observed by Popović 2015).

⁸<https://pypi.python.org/pypi/python-Levenshtein/0.12.0>

$$(3) \quad SS(s1,s2) = \left(1 - \frac{\text{Levenshtein Edit Distance}(s1,s2)}{\text{len}(s1) + \text{len}(s2)}\right) * 100$$

$$(4) \quad WA(s1,s2) = \{1 \text{ if } s1 == s2 ; 0 \text{ otherwise}\}$$

4.3 *State of the art*

We consider the results of Sharma and Singh (2017) to be state of the art, and to the best of our knowledge, the only relevant one on Hindi-Bhojpuri transduction or even any form of OOV word handling technique for this language pair. While their work builds upon traditional PBMT approaches, they first convert lexical word representations into phoneme strings followed by the alignment of phonemes in these strings. The word is then segmented into phoneme chunks which thus facilitates the extraction of weighted rewrite rules for these chunks.

Since the work of Sharma and Singh (2017) extensively compares and contrasts their own work to the related work in transduction, we refrain from such an elaborate comparison, and suggest their work to the reader for more comparisons to other techniques. We show improvements in performance over their results.

4.4 *Common aspects across models*

Our adaptations of each of the models (i.e., seq2seq, AM, HAN and TN) use Bidirectional LSTMs (BLSTM) as encoder-decoder units unless specified otherwise. A detailed analysis of hyperparameter tuning and training aspect for each model can be found in the Appendix section in the pre-print of this work.⁹ The analyses document the experiments motivating our decisions on using LSTM vs GRU, sizes of encoding and decoding layers, number of layers, batching, optimization methods, regularization methods and pre-training embeddings for each of the three models. We hope that these results may be useful for future work in morphology-related tasks.

4.5 *Character embeddings*

Since the atomic token in our models is the character (and not the word), we explore two pre-training strategies – ‘LM-LSTM-Embed’ and

⁹<https://arxiv.org/ftp/arxiv/papers/1811/1811.08816.pdf>

‘fT-Avg-Embed’ – to represent characters as dense vector embeddings. These embeddings are used only as pre-trainings, and the transduction models are allowed to update these weights during training, i.e., they are not frozen. The following sections describe methods used to create these embeddings.

4.5.1 Creating embeddings using LSTM

Taking cues from Sundermeyer *et al.* (2012), we use a simple one-layered Bi-LSTM of hidden dimension 75 and a dropout of 0.5 to train a character-level language model for Hindi. Working at the character-level, we formulate the problem as a character prediction task – given a sequence of 29 characters, predict the next one, i.e., the 30th character. We found 30 to be the optimum window size after varying window sizes from 5 to 40, keeping in mind memory requirements as well as perplexity. We randomly sample 3000 Hindi Wikipedia articles as a train set (with a validation split of 0.2) and another random 600 articles as a held-out test set. With a train set size of over 19M characters, a vocabulary of 397 distinct characters (including special characters and code-mixed characters¹⁰), and having trained for 31 epochs monitored upon validation loss convergence with a patience of 7 epochs, the model resulted in a perplexity score of 5.18 over the test set. The weights of the character embedding layer were then extracted to be used as pre-trained yet trainable character embeddings for our transduction models. We refer to this method of pre-training as ‘LM-LSTM-Embed’. The embedding weights are allowed to be trained along with the hyperparameters of the models as freezing these lead to a decrease in performance – a plausible reason being the orthographic and grammatical distinction of Hindi (on which the character embeddings are based) from Bhojpuri (the target language whose orthography and grammar must be reflected by modifying the pre-trained embedding weights).

4.5.2 Creating embeddings by averaging fastText embeddings

While using the LM-LSTM-Embed embeddings gives the best performance, we also introduce a novel pre-trained character-level embeddings set for the Devanagari script; these are derived from the 300-D

¹⁰ Characters borrowed from other languages.

Hindi fastText embeddings (Bojanowski *et al.* 2017). We choose the fastText embeddings over other benchmark embeddings because of their inherent sub-word information preserving property that arises from representing each word as a bag of character n -grams¹¹ and not as an atomic token itself. As has been observed by Bojanowski *et al.* (2017), such subword-level representation of words is highly useful in capturing morphological structure of words. We start with the existing word embeddings for Hindi.¹² These fastText embeddings have been constructed for each word by averaging the embeddings of the character n -grams that make up the word. We derive the pre-trained embedding of a character by averaging over all word vectors of words in which the character occurs, weighted by the number of times it occurs in each word. We refer to this method of pre-training as ‘ft-Avg-Embed’. We outline the reason to support this method.

For character n -grams, in the case when $n = 1$ (and loosely extending for $n > 1$), each existing word embedding would have been the average of all its characters’ embeddings (or when $n > 1$, groups of characters’ embeddings). An approximation of ‘re-obtaining’ the character embeddings from the word embeddings would then be to average over all word vectors of words in which the character occurs. This linear transformation would preserve the ‘component’ of the embedding of the particular character, as the components of all the other characters would cancel each other out. This justification also takes into account the fact that the components of all characters that are not ‘close’ (in the sense of co-occurring in the same context, i.e., word) to the particular character would cancel out each other since their vectors can be thought of as being spatially randomly distributed with respect to the vector of the character in question. This also implies that the resulting character embedding will have contributions from the components of characters that are ‘close’ to it. We propose this method of computing character embeddings as follows.

¹¹ Consequently, we observe that using the fastText derived embeddings provides highly consistent results in comparison to using GloVe (Pennington *et al.* 2014) or word2vec (Mikolov *et al.* 2013) character adaptations over monolingual Hindi corpora.

¹²<https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md>

Initialization methods	AM			HAN		
	BLEU	SS	ep	BLEU	SS	ep
ft-Avg-Embed	87.32	85.56	59	83.14	80.89	21
LM-LSTM-Embed	89.71	88.03	22	85.94	84.05	17
Random	73.51	71.68	16	64.11	61.33	12
Zero	68.44	63.10	14	60.83	59.20	13

Table 2:
Effect of pre-trained character embeddings: *ep* indicates the number of epochs until convergence

- (a) It allows us to circumvent the need for extensive lexical and computational resources required for training character embeddings on a large Hindi monolingual corpus from scratch (while making small compromises on overall downstream accuracy).
- (b) It uses pre-existing vectors that have been successfully tested upon a range of tasks (Chaudhary *et al.* 2018), lending the method of deriving character embeddings (from word embeddings) in this manner to any pre-trained word vectors that have been themselves composed from sub-word representations.
- (c) It provides scope to study the notion of character embeddings, vis-a-vis word embeddings, since the semantic notion of a word and its embedding is well understood but the notion of a dense representation of a character is not fully understood yet.

It is important to note that pre-training using the LM-LSTM-Embed method results in the best transduction performance, and also that using the ft-Avg-Embed pre-trainings does significantly better than random initializations, while incurring lesser computation costs than LM-LSTM-Embed. These differences are shown in Table 2 for Hindi-Bhojpuri and in Table 3 for Hindi-Bangla (additional experiments on this language pair will be described in Section 5.3). In both these tables, we observe that the performance improvements across the metrics alongside the increase in training epochs before convergence of the models remain consistent, i.e., LM-LSTM-Embed is a better pre-training strategy than ft-Avg-Embed.

5.1 Comparing our models to the state-of-the-art model

Table 4 compares the performances of the best AM and HAN models (after the hyperparameter search) with those of the standard encoder-

Table 3:
Effect of pre-trained
character embeddings for
Hindi-Bangla cognate pairs

Initialization methods	AM			HAN		
	BLEU	SS	ep	BLEU	SS	ep
ft-Avg-Embed	78.49	77.10	56	73.65	70.22	26
LM-LSTM-Embed	81.03	79.72	33	75.28	73.15	24
Random	72.66	72.25	19	60.07	58.46	10
Zero	70.39	69.88	19	59.21	55.33	12

Table 4:
Comparison of evaluation
metrics among
encoder-decoder models

Metric (%)	seq2seq	AM	HAN	TN	SOTA
BLEU	52.89	89.71	85.94	90.89	79.82
SS	57.22	88.03	84.05	90.23	-
WA	16.32	67.22	59.77	75.71	64.41

decoder model, the TN model and the current state-of-the-art (SOTA) model (Sharma and Singh 2017). The BLEU scores of AM, HAN and TN are higher than that of the phoneme-chunk based model used by Sharma and Singh (2017). While TN outperforms the SOTA model in terms of word accuracy (WA), AM and HAN lag behind. The simple seq2seq model performs the worst among all our models, and fails to match up to SOTA.

TN performs the best due to two main reasons:

1. Residual connections that connect the input character embeddings to the final decoder of the output word: The transduction of a Hindi word can be thought of as making character level edits upon the Hindi word, in the same orthographic space. The residual connections, hence, help in learning these edits by conditioning the final decoder not only on the attention-based representation of the input word but also directly the input word itself.
2. The multi-head attention mechanism of TN helps to model the dependencies of characters in the input and output words, regardless of their distances from each other. This would otherwise have to be learnt from a restricted fixed-sized representation, which is usually an LSTM.

The simple seq2seq model performs the poorest because, perhaps, the generic architecture is not adapted to the task in any manner, based on knowledge about the linguistic properties between cognate-pairs. This simple architecture fails to capture long-range dependencies for 1) longer words, and 2) words in which the dependencies be-

tween orthographic segments in the source and target word are not very obviously aligned.

The performance improvements of HAN and AM could be attributed to their attention mechanisms that facilitate better capturing of the intricacies of phonetic and orthographic dependencies of cognates. It is interesting to see that HAN performs worse than the AM, and this might be because of the over-parametrization of HAN. HAN is over-parameterized since it was originally proposed for building a hierarchy over documents where a layer of attention is effected across words in sentences of the document, and another layer is effected across characters in words of sentences. In our case we only deal with individual words, and not documents. This, combined with the fact that we have a small training set, perhaps causes HAN to overfit to the training data and perform poorly on the test set. We observed that the HAN transductions commonly had erroneous repeated characters at the ends of words; we thus employ a post-processing step to remove all but the first recurring character appended to these words. This will be referred to in later sections as the ‘post-processing’ step.

An elaborate account of errors made on different word-pair types by each of these models is presented in Section 6. These differences in errors occur due to the differences in the models as expounded above.

5.2 *Gains over the state of the art*

Table 4 depicts the performance of the aforementioned models using both the pre-trained character embeddings obtained using averaging (AVG) and that built from the language model (LM). It is evident that TN performs best across all accuracy metrics. This is our best model and using this model achieves gains over SOTA of 11.07 BLEU score points (a percentage gain of 13.9%) and a Word Accuracy gain of 11.3% (a percentage gain of 17.5%). The SOTA paper did not provide information of SS scores, and hence we have not made this comparison.

5.3 *Additional experiments on Hindi - Bangla cognate pairs*

We extend our experiments to transducing from Hindi to Bangla by training on a corpus of Hindi-Bangla cognate pairs comprised of 3220 word pairs. We carried out the same 3:1 split upon this corpus to hold out the test set while making a further split of 0.1 upon the train set

Table 5:
Comparison of evaluation metrics
between seq2seq, AM, HAN, and TN
for Hindi-Bangla cognate pairs

Metric (%)	seq2seq	AM	HAN	TN
BLEU	41.88	78.49	73.65	83.76
SS	55.12	77.10	70.22	82.59
WA	9.87	59.27	47.19	71.11

(2415 word pairs) to obtain a validation set. The results of the Hindi – Bangla experiments are presented in Table 5.

The decline in the scores of all four models across the metrics in Table 5 (compared to Table 4) could be boiled down to two major reasons:

1. *Word formation methods*: Bangla has its roots in the Prakrit or middle Indo-Aryan language, which in turn descended from the old Indo-Aryan language, of which Sanskrit is a standardized form. Hindi also shares roots with Sanskrit. Bangla is therefore neither a dialect nor an immediate sibling of Hindi. This is unlike Bhojpuri. This also means that Bangla, which has its own Brahmi-derived script (namely, the Bangla script) has possible word formation rules that are quite different from Hindi. One such instance is the Bangla consonant clustering mechanism. For example, the name Vishnu, written as विष्णु [viRNu] in Hindi, has the consonant cluster $\text{ṣa} + \text{ṇa}$ [RN]. While the Hindi consonant cluster ($\text{ष} + \text{ण}$) can easily be decomposed into its constituent letters, the Bangla cluster can form a new character in itself.
2. *Smaller size of the training corpora*: The training corpora for Hindi-Bangla cognate pairs comprises 2415 word pairs, i.e., 1000 instances less than that of Hindi-Bhojpuri train set. The aforementioned grammatical restrictions shared by Hindi and Bangla make it very demanding to discover more such cognate pairs between the two languages, thus constraining the training corpora for our experiments.

We analyse the outputs of each model to study a pattern in the most common errors made by each of them. We identify six types of orthographic and lexical errors, and four types of errors related to overall

translation quality for a word. While orthographic errors are motivated with respect to the types of graphemes generated by character patterns, quality-related errors focus on overall aspects of the transduction being close to the correct translation. Further, we have been able to make some preliminary correlations between the model architectures (AM, HAN, TN) and the errors they make. The main weakness of the AM model is that since it is predominantly bidirectional LSTM-based and only weakly attention-based as compared to HAN and TN, it tends to bias character predictions towards either the early characters or the later characters in the input sequence, sometimes giving poor results towards the mid-sections. Since it processes input in a sequential manner, it also tends to lose out some orthographic information in the process. The TN model's weakness lies in the fact that it is infamously bad at performing copy mechanisms (Dehghani *et al.* 2018), and hence it fails in places where characters/character-groups have to be preserved in the transduction. The HAN model's trade-off between the LSTM's influence and the attention weights' influence lies between the AM and the TN, and the behaviour it shows with respect to errors it leads to, reflects this in certain ways. However, these are only approximate inferences that we make retrospectively, with the actual behaviour varying on a case-by-case basis.

6.1 Orthographic and lexical errors

6.1.1 Halant

Halants refer to diacritics used to signify the lack of an inherent vowel in written Devanagari scripts. In Devanagari, the halant is represented by a diacritic below the consonant it applies on (e.g., ढ्), while it is represented by not using any vowel after the consonant it applies on in the WX notation (e.g., ढ् is represented as simply 'x'). In most of the cases, halants are preserved during translation of Hindi to Bhojpuri words, except a few (e.g. प्रथा [praWA] (*tradition*) becomes परथा [paraWA]). We study the capability of each model to handle the translation of halants (see Table 6). While HAN and TN perform reasonably well at translating halants, AM tends to replace the character possessing halant with some ligature combined with a neighbouring character, e.g. ल्क [wka] in पल्कारनन [pawkAranana] and ल्ल [wwa] in पल्लिता [paviwwA]. The possible reason for this could be the better attention capabilities of the HAN and the TN, which helps these models force the

Table 6: Handling halant

Hindi	Bhojpuri (correct)	AM	HAN	TN
प्रथा [praWA] (tradition)	परथा [paraWA]	पररा [pararA]	परथा [paraWA]	परथा [paraWA]
पवित्रता [paviwrawA] (purity)	पवित्रता [paviwrawA]	पवित्ता [paviwwA]	पवित्रत [paviwraw]	पवित्रता [paviwrawA]
पत्रकारों [pawrakAroM] (journalists)	पत्रकारन [pawrakArana]	पत्कारनन [pawkAranana]	पत्रा [pawrA]	पत्रकारन [pawrakArana]

halant to be appended in the right place, thus ensuring that the immediate neighbouring character of the halant in Hindi is joined with the previous character in the Bhojpuri transduction. The AM, having only limited attention influence in comparison, skips the immediate neighbouring character and combines the halant with a later character as the LSTM layer in the AM has seen the later character more recently.

6.1.2 Handling vowels

Vowels play an important role in the translation of Hindi words to their closely related languages. Our investigations show that TN performs the worst in recognising appropriate vowel translations for vowels used in Hindi words. While the outputs of HAN are the most reasonable ones after the post-processing step (described in Section 5.1) of removing repeating characters at the end of the word, the AM model performs moderately well in learning correct vowel translations. We attribute this to the fact that a Hindi word’s vowels are mostly retained in its Bhojpuri cognate. The TN perhaps performs the worst at retaining vowels as it is infamous for being bad at copy mechanisms (Dehghani *et al.* 2018). These examples are shown in Table 7.

6.1.3 Handling anusvāra

An anusvāra is a diacritic used in a variety of written Indic scripts to denote a nasal sound, either a nasalized vowel or a nasal consonant that is not followed by a vowel. Anusvāra is used often in Hindi, but is absent in Bhojpuri writing. In Devanagari, the anusvāra is represented

Table 7: Handling vowels

Hindi	Bhojpuri (correct)	AM	HAN	TN
आना [AnA] (to come)	आइला [AilA]	अयना [ayanA]	आना [AnA]	अयन [ayana]
घुमाना [GumAnA] (to stir)	घुमावल [GumaAvala]	घुहावल [GuhAvala]	घुमावल [GumaAvala]	घोमावल [GomaAvala]
मौसी [mOsI] (maternal aunt)	मउसी [mausI]	मउसी [mausI]	मउसी [mausI]	मउसी [mausI]

by a dot (◌̣) above a character, while it is represented by the letter ‘M’ in the WX notation. The two general patterns for translation of anusvāra are:

1. replacing them by adding an extra ‘न [na]’ in front of the consonant, e.g. लेखों [leKoM] (*writings*)-> लेखन [leKana],
2. completely removing them, e.g. भेंट [BeMta] (*offering*)-> भेट [Beta].

Our study shows that all three models get confused at choosing the correct rule and generally end up choosing the former one. Table 8 shows a few examples of this. Comparatively, the AM based model performs better than the other two.

6.1.4 Handling conjuncts for क्ष [kRa] and ज्ञ [jFa]

Conjuncts are formed when successive consonants, lacking a vowel in between them, physically join together. The conjuncts for क्ष [kRa] and ज्ञ [jFa] are special cases in that they are not clearly derived from the letters making up their components, i.e., the conjunct for क्ष [kRa] is क् [k] + ष [Ra] and for ज्ञ [jFa] it is ज् [j] + ञ [Fa]. The rules for translation of such conjuncts from Hindi to Bhojpuri are difficult to model (e.g. in the translation वृक्ष [vqkRa] (*plant*) to बिरिछ [biriCa], क्ष [kRa] becomes छ [Ca], while it remains as क्ष [kRa] in other cases); and therefore we explore the capability of the models in learning such translations. Our study shows that while the translation for क्ष [kRa] is easily learned by the models in most cases, the translation for ज्ञ [jFa]

Table 8: Handling anusvāra

Hindi	Bhojpuri (correct)	AM	HAN	TN
कठिनाइयों [kaTinAyioM] (hardships)	कठिनाइयन [kaTinAyiana]	कठिनाइयन [kaTinAyiana]	कठिनाइयन [kaTinAyiana]	कठिनाइन [kaTinAyiana]
लेखों [leKoM] (writings)	लेखन [leKana]	लेखन [leKana]	लेखन [leKana]	लेखन [leKana]
भेंट [BeMta] (offering)	भेट [Beta]	भेन्ट [Benta]	भेन् [Ben]	भेन्ट [Benta]

Table 9: Handling conjuncts for क्ष [kRa] and ज्ञ [jFa]

Hindi	Bhojpuri (correct)	AM	HAN	TN
बरैक्षा [barEkRA] (village's name)	बरइक्षा [baraikRA]	बरबक्धा [barabakDa]	बरइक्षा [baraikRA]	बरइक्षा [baraikRA]
अवैज्ञानिक [avEjFAnika] (unscientific)	अबैज्ञानिक [abEjFAnika]	अवैज्जानिक [avEjajAnika]	अबैज्ानिक [abEjianika]	अवैज्ञानिक [avEjFAnika]
वृक्ष [vqkRa] (plant)	बिरिछ [biriCa]	बक्छ [bakC]	बिक्ष [bikRiRa]	बिरिछ [biriCa]

often results in ambiguity. Overall, TN performs the best in learning such translations while AM performs the worst. This, and many of the other errors mentioned in this section, are due to the nature of the writing system used. Table 9 shows a few examples of how different models handle conjuncts.

6.1.5 Handling the diacritic ‘reph’ for र् [r]: र्व [rva], र्वा [rvA], र्ष [rspa]

र् [r] in certain contexts is written as a special diacritic that takes the form of a curved upward dash above the preceding consonants. While translating the Hindi words containing such diacritics to their Bhojpuri counterparts, the diacritic is either replaced completely by र or simply

Table 10: Handling the diacritic for र [a]

Hindi	Bhojpuri (correct)	AM	HAN	TN
अहिंसापूर्वक [ahiMsApUrvaka] (nonviolently)	अहिन्सापूर्वक	अहिन्सापूरन्	अहिन्सपूरसक	अहिन्साप्रवक
नाचपार्टी [nAcapArtI] (dance party)	नाचपारटी	नाचपार्ट	नाचपारप	नाचपारी
फर्नीचर [ParnIcara] (furniture)	फरनीचर	फर्ीचर	फर्ि	फरनीचर

kept unchanged. Our results show that, in most cases, the AM model learns to preserve the diacritic as is; whereas the HAN and the TN models generally replace it by a complete र [ra]. Table 10 shows a few examples of how different models handle these diacritics.

6.1.6 Handling hrsva and deergha varna (short and long vowels)

The Bhojpuri transliterations of most Hindi words show the same long and short vowels appearing after the corresponding consonants (words such as दूसरा [xUsarA] (*second*) can be treated as exceptions). We observe that for words preserving the nature of vowels (long or short), the AM and HAN based models perform better than that of the TN model while all the models fail to learn the cases where the nature of vowels (long or short) is switched upon translation. Interestingly, the TN model actually recognizes words in which the long vowel has to be switched to the short vowel and vice versa (we deduce this because it does not preserve the long/short nature in these cases) but it does not perform the switch correctly, and instead shows ambiguous behaviour on predictions. Table 11 shows a few examples of how different models handle short and long vowels.

6.2 Transduction-based properties

6.2.1 Performance on long words

We consider words exceeding six characters in length to be long words. We found that while the translation quality of each model degrades as

Table 11: Handling hrasva (short) and deergh (long) varna (letter)

Hindi	Bhojpuri (correct)	AM	HAN	TN
हिम [hima] (<i>snow</i>)	हिम [hima]	हिम [hima]	हिम [hima]	हीम [hIma]
दूसरा [dUsrA] (<i>second</i>)	दुसर [dusar]	दूसर [dUsar]	दूसर [dUsar]	दोसर [dosar]
घी [GI] (<i>ghee</i>)	घीव [GIva]	घी [GI]	घी [GI]	घि [GI]

Table 12: Performance on long words

Hindi	Bhojpuri (correct)	AM	HAN	TN
विद्यार्थियों [vidyArWiyōM] (<i>students</i>)	विद्यारथियन [vidyAraWiyana]	बिद्यार्थयन [vidyArWayana]	बिद्यारसियन [vidyArasiiyana]	बिद्यारियन [vidyAriyana]
सुरक्षाकर्मीयों [surakRAkarmIyōM] (<i>guards</i>)	सोरक्षाकरमियन [sorakRAkaramiyana]	सुरक्जाकररनन [surakFAkararanana]	सुरक्षाकरीयन [surakRAkairIyana]	सुरक्षारन [surakRARana]
हथियारबन्द [haWiyArabanda] (<i>armed</i>)	हथियारबन्द [haWiyAraband]	हथियारबन्द [haWiyArabanda]	हथियारबन्द [haWiyArabanda]	हथियारब्द [haWiyArabda]

word length increases, the AM based architecture is able to maintain the most sensible outputs, followed by the TN and the HAN based models. Table 12 shows examples of the behaviour of different models when they encounter long words.

6.2.2 Identical transduction

For words in Hindi that are written identically in Bhojpuri, we observe that the HAN based model gives the best results after post-processing (described in Section 5.1). The TN model fails in cases of longer words while the performance of the AM based model deteriorates for shorter words as well as vowels. Examples of behaviour in cases of identical transductions are presented in Table 13.

6.2.3 Sensible yet erroneous translations

We individually study the translations made by each model which sound legitimate when compared to the translations of other similar words but are actually wrong. For example, for diphthongs, while the

Table 13: Performance on identical translations

Hindi	Bhojpuri (correct)	AM	HAN	TN
मौलवी [mOlavI] (Muslim doctor)	मौलवी [mOlavI]	मउबि [maubi]	मउलवी [maulavI]	मउलवी [maulavI]
झट [Jata] (instant)	झट [Jata]	झछट [JaCata]	झट [Jatatata], [Jata]	झट [Jata]
हथियारबन्द [haWiyArabanda] (armed)	हथियारबन्द [haWiyArabanda]	हथियारबन्द [haWiyArabanda]	हथियारबन्द [haWiyArabanda]	हथियारब्द [haWiyArabda]

Bhojpuri translation for भैया [BEyA] is भइया [BaiyA] (*elder brother*) (भै [BE] replaced by भइ [Bai]), the translation for कैमरा [kEmarA] (*camera*) does not follow such approach, whereby the character कै [kE] remains preserved. We observe that each model has its own types of erroneous translations due to such ambiguities as shown in Table 14.

Table 14: Some sensible yet erroneous translations

Model	Hindi	Bhojpuri (correct)	Predicted
AM	तन्मयता [wanmayawA] (concentration)	तन्मयता [wanmayawA]	तनमयता [wanamayawA]
AM	कैमेरा [kEmarA] (camera)	कैमेरा [kEmarA]	कइमेरा [kaimerA]
HAN	तरबूजा [warabUjA] (watermelon)	तरबूजा [warabUjA]	तरबूज [warabUja]
HAN	यमुना [yamunA] (yamuna)	यमुना [yamunA]	जमुन [jamun]
TN	खिलाड़ी [KilAdI] (player)	खेलारी [KelAdI]	खेलाड़ी [KelAdI]
TN	उपकरण [upkaraNa] (equipment)	उपकरन [upakaraNa]	ओपकरन [opkaraNa]

6.2.4 Phonetically/orthographically invalid translations

These are predicted transductions which do not follow the necessary phonetic rules in order to be pronounced. We study the type of such words individually for each model. Our findings suggest that the TN performs excellently in learning such rules since we did not notice any such instance of unpronounceable words present in the outputs of TN. Overall, HAN produced the most unpronounceable translations as can be seen from Table 15.

Table 15: Phonetically/orthographically invalid translations: for invalid predictions, a closest approximation of the WX notation is given

Model	Hindi	Bhojpuri (correct)	Predicted
AM	धर्मान्तरण [XarmanwaraNa] (religious conversion)	धरमान्तरन [Xaramanwarana]	धर्मान्ि [Xarmaani]
AM	सुरक्षाकर्मीयों [surakRAkarmIyoM] (security guards)	सोरक्षाकरमियन [surakRAkaramIyana]	सुरक्षाकुं्रियन [surakRAkairIyana]
AM	वरजित [varjiwa] (contraband)	वरजित [varajiwa]	बरामि [baraimi]
HAN	कुंजी [kuMjI] (key)	कुन्जी [kunjI]	कुंी [kuMI]
HAN	अवैज्ञानिक [avEjFAnika] (unscientific)	अबैज्ञानिक [abEjFAnika]	अवैजज्ानिक [avEjajAnika]

6.3

Limitations of the method

While training the system on cognate pairs ensures that the model does not require exhaustive resources such as a parallel corpora, the fact remains that cognate-property is inherently confined to the case of closely-related languages. As described in Section 1, Bhojpuri is closely related to Hindi; it is a dialect/immediate sibling and shares the same script.¹³ By contrast, Bangla (Section 5.3) is a direct descendant of

¹³ Although Bhojpuri was historically written in Kaithi scripts, those have become obsolete; Devanagari is now the primary script.

Sanskrit-family or Pali¹⁴ to Prakrit and Apabhramsha (Chatterji 1926; Bali 2016) but is only loosely related to Hindi: it is neither a dialect nor an immediate sibling of Hindi; it has its own script; and it has word formation rules that are sometimes contrary to Hindi. The effects of such increased ‘language distance’ are apparent from Table 4 (Hindi-Bhojpuri) and Table 5 (Hindi-Bangla) where Hindi-Bhojpuri cognate pairs show far better transduction quality than Hindi-Bangla.

Further, with the increased distance between the source and the target language in a language pair, it is safe to say that the set of cognates shared by these dwindles (Beel and Felder 2014), since cognates, by definition, are word pairs that not only have similar meaning and phonetics but also reflect an allied linguistic derivation. We observed this even in our own cognate datasets: the Hindi-Bangla dataset had 25% fewer cognates than the Hindi-Bhojpuri dataset (see Section 5.3). Having fewer real-world cognate pairs means that even if a nominally complete dataset of all cognates were to be compiled, there would still be comparatively less data to train models on, hence exhibiting another limitation in extending the method to language pairs of arbitrary distance.

7 IMPROVEMENTS ON HINDI – BHOJPURI MACHINE TRANSLATION

This section talks of the improvements we make on machine translation from Hindi to Bhojpuri. The authors would like to mention two key points here. First, this section stands distinguished from the rest of the paper in that the methods and results discussed thus far hold for word-to-word transduction. In contrast, we now depict how such transductions can be used to improve the accuracy of a machine translation system. Second, we must mention that to the best of our knowledge, no prior machine translation systems have been trained on a Hindi-Bhojpuri parallel corpus of a trainable size, simply because such a corpus does not exist.

¹⁴Hindi descended from Prakrit, Apabhramsha and Perso-Arabic while Bangla and Bhojpuri inherited the regional dialects of Apabhramsha. Consequently, Bangla and Bhojpuri, sharing several regional boundaries of Northern India, Bangladesh and Nepal, can be thought of having similar ancestral languages to Hindi.

For our purposes, we build an artificial parallel corpus in the following manner. We first scrape four Bhojpuri blogs for Bhojpuri sentences: Anjoria,¹⁵ TatkaKhabar,¹⁶ Bhojpuri Manthan,¹⁷ and Bhojpuri Sahitya Sarita.¹⁸ From these, we collect a set of approximately 40,000 Bhojpuri sentences. We then follow a two-step process for obtaining Hindi translations of these sentences using the Google Translate API. Since the API does not offer Bhojpuri as a source language, we first use the Hindi-English API for translating the Bhojpuri sentences to English (as an intermediate language). We then translate these English sentences to Hindi using the English-Hindi API. Although this process gives us noisy translations between Bhojpuri and Hindi, this method is based upon the hypotheses that:

1. in the absence of an actual reported machine translation system between these languages, back-translation can be used as the only available substitute for showing improvement due to word transduction;
2. that this is the best available solution in the absence of Hindi-Bhojpuri parallel corpora, let alone a pre-trained Hindi-Bhojpuri translation system; and
3. that Bhojpuri is sufficiently similar to Hindi that its linguistic properties remain preserved satisfactorily throughout the intermediate processing.

Also, the choice of English as an intermediate language is based on the fact that it gives the best BLEU scores when Hindi is fixed as the other language in the translation pair. At the time of writing this work, there is no Bhojpuri-Hindi API (or publicly reported Hindi-Bhojpuri machine translation system), while trivially translating using the Hindi-Hindi API does not seem to work as the API simply copies the inputs to outputs. For reporting the machine translation results, we use the standard document-level BLEU score as suggested by Papineni *et al.* (2002).

For the held-out test set, we curated 1000 sentences for which ground truth translations were manually obtained from experts, and

¹⁵<https://www.anjoria.com/>

¹⁶<http://khabar.anjoria.com/>

¹⁷<http://bhojpurimanthan.com/>

¹⁸<http://www.bhojpurisahityasarita.com/>

	Training set	Test set
Number of sentences	40,000	1,000
Total number of tokens	812,070	19,689
Number of unique tokens	20,551	620

Table 16:
Machine translation corpus statistics

not artificially generated. Statistics on training and test sentences can be found in Table 16.

We train a Bi-LSTM based encoder-decoder network (500 units) with Luong attention, as described in Luong *et al.* (2015a) (we use OpenNMT’s global attention implementation) on the training set for Hindi-Bhojpuri translation. The model thus trained resulted in a BLEU score of around 7.1 on the test set, which is not surprising given the method of training. With this as the baseline, we make corrections to this model’s output. We first align each Hindi sentence and its Bhojpuri translation, by aligning pairs of source-target words based on their pairwise weights in the attention matrix. Following this, we identify OOV Hindi words using a simple dictionary-based approach as has been suggested by Bahdanau *et al.* (2014). We use a shortlist of 15,000 most common words in Hindi, obtained from the Hindi Wikipedia monolingual corpus, and treat all other words as OOV. We experiment with 5000, 10,000, 15,000, and 20,000 most common words, and find the 15,000 word shortlist produced the best BLEU score improvements. We then replace the translation (as obtained by the alignments using the attention matrix) of each OOV word with its corresponding transduction generated by our word transduction model. Replacing the translation of OOV words with that of their transductions leads to an improvement of 6.3 points in the BLEU score, which is substantial considering that we are translating to a low-resource language. We obtain a BLEU score of 13.4 with such a basic translation set-up followed by simple correction of OOV translations using transductions.

More importantly, the improvement in the MT BLEU score shows that even though the task of transduction is a focused one, it generalizes well to OOV Hindi words that are not part of a Hindi-Bhojpuri cognate pair. This stands as an important aspect of our work. Our transduction model, despite being trained on a dataset that is constrained to cognate pairs, lends reasonable improvements to the highly

generalized machine translation task by exploiting the closeness of Hindi and Bhojpuri.

8

CONCLUSION

We propose a character-level transduction of OOV words between a pair of closely related languages, out of which at least one is a low-resource language. Word transduction aims to predict the orthographic form of the word in the target language, given the word in the source language. We restrict the training space to a set of cognates, since in the case of closely-related languages, a cognate can be a good approximation to a translation, if not the translation itself. We present three different models for the same, each of which performs well on handling certain types of grapheme transformations, while performing sub-optimally on others. While all our models¹⁹ outperform the current state of art for Hindi-Bhojpuri transduction, the TN model gives the overall best performance. We suggest a two-step procedure to improve a low-resource NMT system by:

1. identifying the need to handle OOV words separately; and
2. transducing them to their target equivalents, instead of translating.

In the process, we also propose a primitive yet useful MT method using Google Translate APIs for a pair of languages which has no known MT system. In the future, we would like to test our models on more closely related language pairs. Further, we would like to build a state-of-the-art MT pipeline for low resource languages, which incorporates our method to handle OOV words. We would also like to compare the effect of transduction as a solution to the OOV problem faced by word-level MT systems by comparing it to a character-level MT system.

REFERENCES

Martín ABADI, Ashish AGARWAL, Paul BARHAM, Eugene BREVDO, Zhifeng CHEN, Craig CITRO, Greg S. CORRADO, Andy DAVIS, Jeffrey DEAN, Matthieu DEVIN, Sanjay GHEMAWAT, Ian GOODFELLOW, Andrew HARP, Geoffrey

¹⁹We open source the code and data curated for this work in our github repo: <https://github.com/Saurav0074/nmt-based-word-transduction>.

IRVING, Michael ISARD, Yangqing JIA, Rafal JOZEFOWICZ, Lukasz KAISER, Manjunath KUDLUR, Josh LEVENBERG, Dan MANÉ, Rajat MONGA, Sherry MOORE, Derek MURRAY, Chris OLAH, Mike SCHUSTER, Jonathon SHLENS, Benoit STEINER, Ilya SUTSKEVER, Kunal TALWAR, Paul TUCKER, Vincent VANHOUCKE, Vijay VASUDEVAN, Fernanda VIÉGAS, Oriol VINYALS, Pete WARDEN, Martin WATTENBERG, Martin WICKE, Yuan YU, and Xiaoqiang ZHENG (2015), TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, <https://www.tensorflow.org/>, software available from tensorflow.org.

Ankita ACHARYA (2015), Contrastive Study of Bundeli and Hindi Pronunciation, in *regICON-2015: Regional Symposium on Natural Language Processing*, Varanasi, India.

Yaser AL-ONAIZAN, Jan CURIN, Michael JAHR, Kevin KNIGHT, John LAFFERTY, Dan MELAMED, Franz-Josef OCH, David PURDY, Noah A. SMITH, and David YAROWSKY (1999), Statistical Machine Translation, in *Final Report, JHU Summer Workshop*, volume 30.

Jimmy BA, Ryan KIROS, and Geoffrey E. HINTON (2016), Layer Normalization, *Computing Research Repository (CoRR)*, abs/1607.06450.

Dzmitry BAHDANAU, Kyunghyun CHO, and Yoshua BENGIO (2014), Neural Machine Translation by Jointly Learning to Align and Translate, *Computing Research Repository (CoRR)*, abs/1409.0473.

Purnima BALI (2016), Evolution of Bengali Literature: An Overview, *International Journal of English Language, Literature and Translation Studies*, 3:325–332.

Rachel BEEL and Jennifer N. FELDER (2014), Phonological Adaptations of English Loanwords in Turkish, in *Proceedings of the Big South Undergraduate Research Symposium (BigSURS)*, High Point, NC, USA.

Yoshua BENGIO, Patrice Y. SIMARD, and Paolo FRASCONI (1994), Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks*, 5(2):157–66.

Luisa BENTIVOGLI, Arianna BISAZZA, Mauro CETTOLO, and Marcello FEDERICO (2016), Neural versus Phrase-Based Machine Translation Quality: a Case Study, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 257–267, Austin, TX, USA.

Piotr BOJANOWSKI, Edouard GRAVE, Armand JOULIN, and Tomas MIKOLOV (2017), Enriching Word Vectors with Subword Information, *Transactions of the Association for Computational Linguistics (ACL)*, 5:135–146.

Suniti Kumar CHATTERJI (1926), *The Origin and Development of the Bengali Language*, Calcutta University Press.

- Aditi CHAUDHARY, Chunting ZHOU, Lori S. LEVIN, Graham NEUBIG, David R. MORTENSEN, and Jaime G. CARBONELL (2018), Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations, in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 3285–3295, Brussels, Belgium.
- Qiming CHEN and Ren WU (2017), CNN Is All You Need, *Computing Research Repository (CoRR)*, abs/1712.09662.
- David CHIANG (2005), A Hierarchical Phrase-based Model for Statistical Machine Translation, in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 263–270, Association for Computational Linguistics, Ann Arbor, Michigan.
- Kyunghyun CHO, Bart VAN MERRIËNBOER, Caglar GULCEHRE, Dzmitry BAHDANAU, Fethi BOUGARES, Holger SCHWENK, and Yoshua BENGIO (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar.
- François CHOLLET *et al.* (2015), Keras, <https://github.com/keras-team/keras>.
- Monojit CHOUDHURY (2003), Rule-based grapheme to phoneme mapping for Hindi speech synthesis, in *90th Indian Science Congress of the International Speech Communication Association (ISCA)*, Bangalore, India.
- Mostafa DEGHANI, Stephan GOUWS, Oriol VINYALS, Jakob USZKOREIT, and Lukasz KAISER (2018), Universal Transformers, *Computing Research Repository (CoRR)*, abs/1807.03819.
- Etienne DENOVAL and Yves LEPAGE (2005), BLEU in characters: towards automatic MT evaluation in languages without word delimiters, in *Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing*, pp. 81–86.
- Etienne DENOVAL and Yves LEPAGE (2006), The character as an appropriate unit of processing for non-segmenting languages, in *Speech Processing Society 12th Annual Meeting (NLP2006)*, pp. 731–734, Hiroshima University, Japan.
- Inger EKMAN and Kalervo JÄRVELIN (2007), Spoken Document Retrieval in a Highly Inflectional Language, in *Proceedings of the NODALIDA Conference*, pp. 44–50.
- Andrew FINCH and Eiichiro SUMITA (2009), Transliteration by bidirectional statistical machine translation, in *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 52–56, Association for Computational Linguistics (ACL), Singapore.

- Mikel L. FORCADA and Ramón P. ÑECO (1997), Recursive Hetero-associative Memories for Translation, in *IWANN*.
- Sir George GRIERSON (1928), Linguistic Survey of India, *The Journal of the Royal Asiatic Society of Great Britain and Ireland*, 3:711–718.
- Rohit GUPTA, Pulkit GOYAL, and Sapan DIWAKAR (2010), Transliteration among Indian Languages using WX Notation, in *Proceedings of KONVENS*, pp. 147–150, Saarbrücken, Germany.
- Jan HAJIČ (2000), Machine Translation of Very Close Languages, in *Proceedings of the Applied Natural Language Processing Conference (ANLP)*, pp. 7–12, Seattle, Washington, USA.
- Kaiming HE, Xiangyu ZHANG, Shaoqing REN, and Jian SUN (2016), Deep Residual Learning for Image Recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.
- Sébastien JEAN, Kyunghyun CHO, Roland MEMISEVIC, and Yoshua BENGIO (2015), On Using Very Large Target Vocabulary for Neural Machine Translation, in *Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 1–10, Beijing, China.
- Nal KALCHBRENNER and Phil BLUNSOM (2013), Recurrent Continuous Translation Models, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1700–1709, Seattle, USA.
- Grzegorz KONDRAK, Daniel MARCU, and Kevin KNIGHT (2003), Cognates can improve statistical translation models, in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers–Volume 2*, pp. 46–48, Association for Computational Linguistics, Edmonton, Canada.
- Siwei LAI, Liheng XU, Kang LIU, and Jun ZHAO (2015), Recurrent Convolutional Neural Networks for Text Classification, in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, Austin, TX, USA.
- Naihan LI, Shujie LIU, Yanqing LIU, Sheng ZHAO, Ming LIU, and Ming ZHOU (2018), Close to human quality TTS with transformer, *arXiv preprint arXiv:1809.08895*.
- Minh-Thang LUONG and Christopher D. MANNING (2016), Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models, *Computing Research Repository (CoRR)*, abs/1604.00788.
- Thang LUONG, Hieu PHAM, and Christopher D. MANNING (2015a), Effective Approaches to Attention-based Neural Machine Translation, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421, Lisbon, Portugal.

Thang LUONG, Ilya SUTSKEVER, Quoc V. LE, Oriol VINYALS, and Wojciech ZAREMBA (2015b), Addressing the Rare Word Problem in Neural Machine Translation, in *Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 11–19, Beijing, China.

B.P. MAHAPATRA, P. PADMANABHA, G.D. MCCONNELL, and V.S. VERMA (1989), *The Written Languages of the World: A Survey of the Degree and Modes of Use: Volume 2: India: Book 1 Constitutional Languages*, volume 1, Quebec : Presses de l'Université Laval; Forest Grove, Oregon.

Gideon S MANN and David YAROWSKY (2001), Multipath translation lexicon induction via bridge languages, in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pp. 1–8, Association for Computational Linguistics, Pittsburgh, PA, USA.

Tomas MIKOLOV, Ilya SUTSKEVER, Kai CHEN, Greg S. CORRADO, and Jeff DEAN (2013), Distributed representations of words and phrases and their compositionality, in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3111–3119.

Diwakar MISHRA and Kalika BALI (2011), A Comparative Phonological Study of the Dialects of Hindi, in *Proceedings of the International Congress of Phonetic Sciences (ICPhS)*, pp. 1390–1393, Hong Kong.

Preslav NAKOV and Jörg TIEDEMANN (2012), Combining word-level and character-level models for machine translation between closely-related languages, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 301–305, Association for Computational Linguistics (ACL), Jeju Island, Korea.

Kishore PAPINENI, Salim ROUKOS, Todd WARD, and Wei-Jing ZHU (2002), BLEU: a method for automatic evaluation of machine translation, in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 311–318, Association for Computational Linguistics, Pennsylvania, PA, USA.

Jeffrey PENNINGTON, Richard SOCHER, and Christopher MANNING (2014), Glove: Global vectors for word representation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar.

Ngoc-Quan PHAM, Jan NIEHUES, and Alexander H. WAIBEL (2018), Towards one-shot learning for rare-word translation with external experts, in *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, 56th Annual Meeting of the Association for Computational Linguistics*, pp. 100–109, Melbourne, Australia.

- Martin POPEL and Ondřej BOJAR (2018), Training Tips for the Transformer Model, *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.
- Maja POPOVIĆ (2015), chrF: character n-gram F-score for automatic MT evaluation, in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal.
- Kevin P. SCANNELL (2006), Machine translation for closely related language pairs, in *Proceedings of the Workshop Strategies for Developing Machine Translation for Minority Languages*, pp. 103–109, Citeseer, Genoa, Italy.
- Rico SENNRICH, Barry HADDOW, and Alexandra BIRCH (2016), Neural Machine Translation of Rare Words with Subword Units, *Computing Research Repository (CoRR)*, abs/1508.07909.
- Shashikant SHARMA and Anil Kumar SINGH (2017), Word Transduction for Addressing the OOV Problem in Machine Translation for Similar Resource-Scarce Languages, in *Proceedings of the International Conference on Finite State Methods and Natural Language Processing (FSMNL)*, pp. 56–63, Umeå, Sweden.
- Michel SIMARD, George F. FOSTER, and Pierre ISABELLE (1993), Using cognates to align sentences in bilingual corpora, in *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research: Distributed Computing – Volume 2*, pp. 1071–1082, IBM Press.
- Smriti SINGH and Vijayanthi M. SARMA (2010), Hindi noun inflection and Distributed Morphology, in *Proceedings of the International Conference on Head-Driven Phrase Structure Grammar*, pp. 307–321.
- Srishti SINGH and Girish Nath JHA (2015), Statistical Tagger for Bhojpuri (employing Support Vector Machine), in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1524–1529, Kochi, India.
- Martin SUNDERMEYER, Ralf SCHLÜTER, and Hermann NEY (2012), LSTM neural networks for language modeling, in *Proceedings of Thirteenth Annual Conference of the International Speech Communication Association*, pp. 194–197, Portland, OR, USA.
- Ilya SUTSKEVER, Oriol VINYALS, and Quoc V. LE (2014a), Sequence to Sequence Learning with Neural Networks, in *Advances in neural information processing systems (NIPS)*.
- Ilya SUTSKEVER, Oriol VINYALS, and Quoc V. LE (2014b), Sequence to sequence learning with neural networks, in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3104–3112.
- Jörg TIEDEMANN (2012), Character-based pivot translation for under-resourced languages and domains, in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 141–151, Association for Computational Linguistics, Avignon, France.

Robert L. TRAMMELL (1971), The Phonology of the Northern Standard Dialect of Bhojpuri, *Anthropological Linguistics*, 13(4):126–141.

Zhaopeng TU, Yang LIU, Lifeng SHANG, Xiaohua LIU, and Hang LI (2017), Neural Machine Translation with Reconstruction, in *Association for the Advancement of Artificial Intelligence (AAAI)*, San Francisco, CA, USA.

Ashish VASWANI, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Lukasz KAISER, and Illia POLOSUKHIN (2017), Attention is All you Need, in *Advances in Neural Information Processing Systems (NIPS)*.

David VILAR, Jan-T. PETER, and Hermann NEY (2007), Can We Translate Letters?, in *Proceedings of the Second Workshop on Statistical Machine Translation, StatMT '07*, pp. 33–39, Association for Computational Linguistics, Stroudsburg, PA, USA, <http://dl.acm.org/citation.cfm?id=1626355.1626360>.

Oriol VINYALS, ŁUKASZ KAISER, Terry KOO, Slav PETROV, Ilya SUTSKEVER, and Geoffrey E. HINTON (2015), Grammar as a Foreign Language, in *Advances in neural information processing systems (NIPS)*.

Yonghui WU, Mike SCHUSTER, Zhifeng CHEN, Quoc V LE, Mohammad NOROUZI, Wolfgang MACHEREY, Maxim KRIKUN, Yuan CAO, Qin GAO, Klaus MACHEREY, *et al.* (2016), Google’s Neural machine translation system: Bridging the gap between human and machine translation, *Computing Research Repository (CoRR)*, abs/1609.08144.

Zichao YANG, Diyi YANG, Chris DYER, Xiaodong HE, Alexander J. SMOLA, and Eduard H. HOVY (2016), Hierarchical Attention Networks for Document Classification, in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pp. 1480–1489, San Diego, CA, USA.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



Extracting Subregular constraints from Regular stringsets

James Rogers and Dakotah Lambert

Dept. of Computer Science, Earlham College, Richmond, IN, USA

ABSTRACT

We introduce algorithms that, given a finite-state automaton (FSA), compute a minimal set of forbidden local factors that define a Strictly Local (SL) tight approximation of the stringset recognised by the FSA and the set of forbidden piecewise factors that define a Strictly Piecewise (SP) tight approximation of that stringset, as well as a set of co-SL factors that, together with the SL and SP factors, provide a set of purely conjunctive literal constraints defining a minimal superset of the stringset recognised by the automaton.

Keywords: regular languages, finite-state automata, local languages, piecewise languages

Using these, we have built computational tools that have allowed us to reproduce, by nearly purely computational means, the work of Rogers and his co-workers (Rogers *et al.* 2012) in which, using a mix of computational and analytical techniques, they completely characterised, with respect to the Local and Piecewise Subregular hierarchies, the constraints on the distribution of stress in human languages that are documented in the StressTyp2 database.

Our focus, in this paper, is on the algorithms and the method of their application. The phonology of stress patterns is a particularly good domain of application since, as we show here, they generally fall at the very lowest levels of complexity. We discuss these phonological results here, but do not consider their consequences in depth.

1

INTRODUCTION

That phonology is finite-state – characterised by patterns and functions that can be recognised by finite-state automata of varying types –

is uncontroversial. Over the last several years, a growing body of work has emerged characterising the complexity of phonological phenomena within a more finely resolved hierarchy of Subregular stringsets¹ and functions, focusing, in particular, on the lowest levels of the hierarchy. (For a comprehensive survey, see Heinz 2018.)

In this paper we present two main results. One is a set of algorithms that are capable of automatically analysing finite-state acceptors in terms of these classes. We have incorporated these into a computational workbench for exploring systems of Subregular constraints both automatically and interactively.

Using that, we have completed a longterm program of characterising the complexity of suprasegmental stress patterns in human languages. This yields our second result, which strengthens previous characterisations and places these patterns almost exclusively at the very bottom levels of the hierarchy: the Strictly Local, Locally Testable, Strictly Piecewise and Piecewise Testable stringsets. These classes are significant cognitively because they depend only on information that is explicit in the string itself without requiring inference of additional structure.²

A stringset L is *Strictly k -Local* (SL_k) if and only if (iff) it is strictly determined by its local k -factors: the *substrings* of length at most k that

¹Following Sampson (1975), we distinguish between stringsets and languages when our topic extends to both formal and natural languages. In these situations, the formal languages serve as models for aspects of natural languages. To fail to distinguish the stringsets (or others sets of structures) that are the models from the phenomena they are modelling is falling into one of the most basic fallacies of mathematical modelling. Conclusions that are valid in the realm of the models are only valid in the realm of the phenomena to the extent that the models are faithful. That their faithfulness is limited is the very essence of the modelling process.

This is not just a pedantic issue: the history of formal linguistics is peppered with examples in which the failure to distinguish the two has led to erroneous conclusions. For discussion of specific examples see, *inter alia*, Pullum and Scholz (2001) or Rogers (1996). As the scope of this journal is, specifically, language modelling, it seems appropriate to us to be careful in maintaining the distinction.

²Regular stringsets, for example, from a purely declarative perspective, require a mechanism to infer a sequence of states (abstract categories) in parallel with a string, corresponding to a run of an automaton, and then classify the string based on that run.

occur in strings $\bowtie \cdot w \cdot \bowtie$ for $w \in L$. (The ‘ \bowtie ’ and ‘ \bowtie ’ are endmarkers.) By “strictly determined” we mean that L contains all and only the strings that are generated by the (inverse) substring relation from that set of k -factors.³ A string is in the set as long as its k -factors are a subset of the generating set. The class of stringsets that are Strictly k -local for some k is known as SL.

A stringset is *k-Locally Testable* (LT_k) iff it is a Boolean combination of SL_k stringsets. While they are also determined by the set of local k -factors that occur in the strings in the set, the stringset is not generated by the substring relation from a single set of factors. Rather, there are multiple SL_k stringsets that interact in complex ways. Nevertheless, constraints of this form depend only on the information that is explicit in the string, in this case the particular subset of k -factors that occurs; various combinations of factors can be forbidden, permitted or required.

A stringset L is *Strictly k-Piecewise* (SP_k) iff it is strictly determined by its k -pieces: the *subsequences* of length at most k that occur in strings w for $w \in L$, where v is a subsequence of w iff the symbols in v occur in w in order, but not necessarily adjacently. That is to say, L contains all and only the strings that are generated by the (inverse) subsequence relation from that set of k -pieces. The class of stringsets that are Strictly k -Piecewise for some k is known as SP.

The class of *k-Piecewise Testable* (PT_k) stringsets is analogous to the class of LT_k stringsets, but based on subsequences rather than substrings.

These classes are at the bottom of the local and piecewise sides of a collection of classes of stringsets, all strict subclasses of the class of Regular stringsets, which are hierarchically related and are characterised by finite sets of either substrings (the Local Hierarchy) or subsequences (the Piecewise Hierarchy) or by combinations of the two. The Local hierarchy was established primarily by the work of McNaughton and Papert (1971) and Thomas (1982) (with many others); the Piecewise Hierarchy was established primarily by the work of Simon (1975), Lothaire (1983), Rogers *et al.* (2010) and Fu *et al.* (2011) (and many others). Rogers *et al.* (2012) argue that these

³Specifically, all and only those strings that include no substrings other than those in the given set.

hierarchies provide a robust notion of cognitive complexity for constraints on strings.

Our work here was inspired by the work carried out by the Theory Group at Earlham College in which they characterised all of the stress patterns collected in Goedemans *et al.* (2015) – a wide-coverage database of stress patterns occurring in human languages – with respect to this hierarchy. In Edlefsen *et al.* (2008), they established that roughly 75% of these patterns are SL_k for $k \leq 6$ and that half are SL_k for $k \leq 3$. Subsequently, they derived a set of “primitive” constraints sufficient to define all of the non-SL patterns by co-occurrence and classified them into abstract categories (Fero *et al.* 2014). Most of these constraints were, in fact, SL, and their main result was that all of the patterns could be defined by co-occurrence of constraints at the bottom two levels of the hierarchies – the Strict and Testable levels described above. Recent work by Heinz and his co-workers (Heinz 2018, 2010; Chandlee 2014; Jardine 2016) suggests that much of phonology may be characterisable by correspondingly simple sets of structures or functions.

The work on primitive constraints, however, did not provide the factors of the SL stringsets because the algorithm for determining if a given finite-state automaton recognises an SL stringset, and determining k if it does, does not yield the set of k -factors that define the stringset. We resolve that problem in this work. Moreover, the work on non-SL constraints was largely based on the English glosses of the constraints included in the database and were tailored specifically to capturing those specific non-SL lects. This potentially misses SP constraints that may not be explicit in these glosses.

In this paper we specifically address the piecewise classes as well as the class of co-SL constraints, complements of SL constraints. Combinations of SL and co-SL constraints define stringsets in terms of both forbidden and required local factors, but remain a weak fragment of LT. Working with this range of constraints we can sharpen the earlier result, which established $LT + SP$ as an upper-bound for all but two lects in the database: 98 of the 106 lects are $SL + co-SL + SP$, another six require combinations of three properly LT constraints (rather than the nine identified in the earlier work) and the two properly Regular lects share a single Regular constraint, which entails counting modulo two.

While our working domain in developing these algorithms has been phonotactics, and stress patterns in particular, the algorithms are applicable to any Regular stringset. On the other hand, the algorithms are of relatively high complexity, exponential in the size of the automaton if it recognises a Strictly Local stringset and doubly exponential for the SL approximations of non-strict stringsets; the Piecewise algorithms are singly exponential in either case. But these are optimal for algorithms that return the set of forbidden factors of the stringset. In our corpus all of the automata are of moderate size – the largest has 33 states – and they are quite feasible; running on hardware that is unremarkable for modern desktop computers, without aggressive optimisation, it takes less than one minute to process all of the 106 lects of the StressTyp2 corpus of automata.

2

OVERVIEW OF THIS PAPER

In the next section we introduce our notation and basic formal definitions. In Section 3.1 we introduce the notion of local and piecewise factors and in Section 3.2 we consider stringsets from a model-theoretic perspective, which exposes the underlying relationship between these. From that perspective both the substring and the subsequence relations are just restricted variants of a more general is-a-factor-of relation. Henceforth, we will refer to substrings and subsequences as factors of either the local or piecewise type, except when the type is clear from the context. (In Sections 4 and 5, if the type is not specified, “factor” refers to local factors; in Sections 6 and 7 it refers to piecewise factors.)

In Section 4 we formally define Strictly Local stringsets and discuss their formal properties. In Sections 4.1–4.3 we distinguish five types of forbidden local factors – factors in the complement of the set of factors that generate the stringset – and develop the foundations of our algorithms for extracting those local factors given a finite-state automaton. We give details of these algorithms in the remainder of Section 4.

We adapt these algorithms to work on non-SL stringsets in Section 5. While the sets of factors we extract are, of course, insufficient to generate the stringset, they do generate an SL approximation of the stringset. We then introduce the notion of a *residue set*, the difference

between the original stringset and our approximation. This becomes the basis of our further computational analysis.

In Section 6 we formally define Strictly Piecewise stringsets, discuss their formal properties and develop our algorithms for extracting forbidden piecewise factors given a finite-state automaton that recognises an SP stringset. In Section 7 we adapt these to obtain optimal SP approximations of non-SP Regular sets.

In Section 8 we combine these algorithms in a way that allows us to extract co-SL constraints, which enables us to fully characterise 92% of the lects in StressTyp2 purely computationally.

We close by summarising our results, discussing complexity issues and sketching plans for recoding the factors we collect in a way that is more useful than their current form, a flat enumeration.

3 FORMAL PRELIMINARIES

Let Σ be an alphabet. For strings $v, w \in \Sigma^*$ we say v is a *substring* of w ($v \preceq w$) iff $w = u_1 v u_2$ for $u_1, u_2 \in \Sigma^*$. We say v is a *subsequence* of w ($v \sqsubseteq w$) iff the symbols of v occur in w in order, but not necessarily adjacently:

$$\text{if } v = \sigma_1 \sigma_2 \dots \sigma_n \text{ then } v \sqsubseteq w \stackrel{\text{def}}{\iff} w = u_0 \sigma_1 u_1 \sigma_2 u_2 \dots \sigma_n u_n \mid u_i \in \Sigma^*.$$

We denote the *reversal* of w by w^R . We use this same notation for the reversal of a stringset.

A finite-state automaton (FSA) is an edge-labelled directed graph with distinguished vertices, that we will represent by a five-tuple $\langle \Sigma, Q, \delta, I, F \rangle$ where Σ is the alphabet of the language of the automaton, Q is the set of states, $\delta \subseteq (\Sigma \times Q \times Q)$ is a transition relation where $\langle \sigma, q_1, q_2 \rangle \in \delta$ iff there is an edge labelled σ from q_1 to q_2 , I is the set of initial states, and F is the set of accepting states. Let $\mathcal{A} = \langle \Sigma, Q, \delta, I, F \rangle$.

Let $w = \sigma_1 \sigma_2 \dots \sigma_n \in \Sigma^*$ be a string and let $q_1, q_{n+1} \in Q$. Then there is a path $q_1 \overset{w}{\rightsquigarrow} q_{n+1}$ iff there exists some sequence of edges:

$$\langle \langle \sigma_i, q_i, q_{i+1} \rangle \in \delta \mid 0 < i \leq n, \\ w = \sigma_1 \sigma_2 \dots \sigma_n \rangle.$$

This is an accepting path on w if q_1 is in I and q_{n+1} is in F , else it is a non-accepting path.

The automaton \mathcal{A} is *total* iff for every symbol $\sigma \in \Sigma$ and for every state $q \in Q$, there exists some q' such that $\langle \sigma, q, q' \rangle \in \delta$. It is (partial) *functional* iff δ is functional in its first two places. That is, given a state $q \in Q$ and a symbol $\sigma \in \Sigma$, there is at most one $q' \in Q$ such that $\langle \sigma, q, q' \rangle \in \delta$. An FSA is (fully) *deterministic* (a proper DFA) iff it has exactly one initial state and it is both total and functional.

An automaton is *trim* iff for all states $q \in Q$ there is some accepting path from q . Though trim automata may not be total, we still consider them to be deterministic if they have a single start state and are partial functional.

An automaton is *minimal* iff it is deterministic and no two states are Nerode-equivalent.⁴ Further, it is *normalised* iff it is both minimal and trim.

The *reversal* of \mathcal{A} , that is, an automaton that accepts a string iff the reversal of that string is accepted by \mathcal{A} , is denoted \mathcal{A}^R .

The powerset graph of the automaton \mathcal{A} , $\text{PSG}(\mathcal{A}) = \langle V, E \rangle$, is another edge-labelled directed graph where:

$$\begin{aligned} V &= \mathcal{P}(Q) \quad \text{and} \\ E &= \{ \langle \sigma, S_1, S_2 \rangle \mid \sigma \in \Sigma, \\ &\quad S_2 = \{ q' \in Q \mid (\exists q \in S_1)[\langle \sigma, q, q' \rangle \in \delta] \} \}. \end{aligned}$$

Often we are interested only in the subgraph of this generated from a given set of initial subsets.

Lemma 1. *If \mathcal{A} is deterministic, then the sizes of the sets along any path in $\text{PSG}(\mathcal{A})$ are monotonically non-increasing.*

This is because if \mathcal{A} is deterministic δ maps each state in S_1 to at most one state in S_2 .

Corollary 1. *All sets in any cycle are equal in size.*

Corollary 2. *All in-edges to Q and all out-edges from \emptyset are self-edges.*

3.1 Local and piecewise factors

Let Σ be the alphabet of L and let $\Sigma^k = \{v \in \Sigma^* \mid |v| = k\}$ and $\Sigma^{\leq k} = \bigcup_{1 \leq i \leq k} [\Sigma^i]$.

⁴ States q_1 and q_2 are Nerode-equivalent iff for all strings w , there is an accepting path on w from q_2 iff there is an accepting path on w from q_1 (Hopcroft and Ullman 1979).

Definition 1 (Local k -factors). For any string $w \in \Sigma^*$, the local k -factors of w are:

$$F_k^{\triangleleft}(w) = \begin{cases} \{w\} & \text{if } |w| \leq k, \\ \{v \in \Sigma^k \mid v \preceq w\} & \text{otherwise.} \end{cases}$$

Similarly for $F_{\leq k}^{\triangleleft}(w)$. This lifts to sets of strings in the obvious way.

Definition 2 (Shuffle ideal). Let $v = \sigma_1\sigma_2\dots\sigma_n \in \Sigma^*$. The *shuffle ideal* of v is defined as the set $SI(v) = \{w \in \Sigma^* \mid v \sqsubseteq w\}$.⁵

Definition 3 (Piecewise k -factors). For any string $w \in \Sigma^*$, the piecewise k -factors of w are:

$$F_k^{\leq}(w) = \{v \in \Sigma^* \mid v \sqsubseteq w \wedge |v| = k\}.$$

Similarly for $F_{\leq k}^{\leq}(w)$. Again, this lifts to sets of strings in the obvious way.

3.2

A unifying perspective

There is a fundamental regularity between the Local and Piecewise hierarchies that becomes apparent if one looks at strings as ordinary first-order structures. From this perspective, a string is just a labelled finite discrete linear order:

$$\mathcal{W} = \langle \mathcal{D}, \triangleleft, <, P_\sigma \rangle_{\sigma \in \Sigma}.$$

Where \mathcal{D} is a finite domain, \triangleleft is the successor relation (as well as the relation symbol denoting it), $<$ is the less-than relation (and symbol) and the P_σ are unary relations picking out the subset of the positions in the domain at which the symbol σ appears.

From this perspective a local factor is just a structure generated by a subset of the domain that is connected, in the graph-theoretic sense, by the \triangleleft relation and a piecewise factor is just a similar structure connected by the $<$ relation. The size of the factor is just the size of the subset.

The testable classes of local and piecewise stringsets turn out to be the class of all and only those stringsets that are definable

⁵The term “shuffle ideal” appears to have been coined by J. Sakarovitch and I. Simon in Lothaire (1983).

in a propositional logic in which the atomic propositions are factors (Rogers *et al.* 2012). The strict classes are the class of stringsets definable by conjunctions of negative literals in this same logic.

This model-theoretic machinery extends to factors that incorporate both successor and less-than as well as to any class of relational structures, although here we will consider only local and piecewise formulae along with conjunctions of the two. It is this model-theoretic perspective that leads us to refer to both substrings and subsequences as factors, local factors and piecewise factors, respectively.

4 STRICTLY LOCAL STRINGSETS

An *anchored* string is one that has been augmented with one or both of the endmarkers ‘ \bowtie ’ (left end) and ‘ \bowtie ’ (right end).

A stringset L is *Strictly k -Local* ($L \in \text{SL}_k$) iff it is strictly determined by the local k -factors of its fully anchored strings.

Let $\Sigma_{\bowtie\bowtie}^* = \{\bowtie\} \cdot \Sigma^* \cdot \{\bowtie\}$.

Let $G_{L,k}^\triangleleft \subseteq F_{\leq k}^\triangleleft(\Sigma_{\bowtie\bowtie}^*)$ be the set of local factors that occur in fully anchored strings in L . Then the stringset generated by $G_{L,k}^\triangleleft$ is:

$$L(G_{L,k}^\triangleleft) = \{w \in \Sigma^* \mid F_{\leq k}^\triangleleft(\bowtie \cdot w \cdot \bowtie) \subseteq G_{L,k}^\triangleleft\}.$$

$L(G_{L,k}^\triangleleft)$ is, by definition, SL_k . If L is also SL_k then $L(G_{L,k}^\triangleleft) = L$.

Since Σ is assumed to be finite, $F_{\leq k}^\triangleleft(\Sigma^*)$ is also finite, and an SL_k language can equivalently be defined in terms of its forbidden factors: $G_{L,k}^\triangleleft = F_{\leq k}^\triangleleft(\Sigma_{\bowtie\bowtie}^*) - G_{L,k}^\triangleleft$. This is more natural in many applications, including many linguistic ones (as in “no pair of unstressed syllables occur adjacently”).

A stringset is said to be SL if it is SL_k for any finite k .

The following proposition characterises SL_k .

Proposition 1 (Suffix Substitution Closure). (**SSC**)

$$\begin{aligned} L \in \text{SL}_k \text{ iff} \\ (\forall x \in F_{k-1}^\triangleleft(\Sigma_{\bowtie\bowtie}^*)) [\quad & \text{if } w_1 = u_1 \cdot x \cdot v_1 \in \{\bowtie\} \cdot L \cdot \{\bowtie\} \\ & \text{and } w_2 = u_2 \cdot x \cdot v_2 \in \{\bowtie\} \cdot L \cdot \{\bowtie\} \\ & \text{then } u_1 \cdot x \cdot v_2 \in \{\bowtie\} \cdot L \cdot \{\bowtie\} \quad]. \end{aligned}$$

This is because if a symbol σ can follow x in some string of $L(\mathcal{A})$ then $x \cdot \sigma$ is a permitted local factor and σ can follow x in any string of L .

One consequence of this is that if $L(\mathcal{A}) \in SL_k$ and \mathcal{A} is deterministic, then for each length $k - 1$ string x , all states in the set

$$\{q' \in Q \mid (\exists q \in Q)[q \xrightarrow{x} q']\}$$

are Nerode-equivalent. If \mathcal{A} is minimal as well, then all paths that end with the same $(k - 1)$ -factor lead to the same state. The computations of the automaton synchronise after at most $k - 1$ steps.

This is the basis of the algorithm used by Edlefsen *et al.* (2008) to determine if a given \mathcal{A} recognises an SL stringset and, if it does, to find the parameter k .

Proposition 2. *Suppose \mathcal{A} is a normalised DFA. Then $L(\mathcal{A}) \in SL_k$ iff every path from Q in $PSG(\mathcal{A})$ that is of length $k - 1$ leads to a vertex that is either a singleton subset of Q or empty. If that is the case, then k is one plus the length of the longest path from Q to a singleton (that does not include other singletons). If there is no such longest path (i.e., there is an infinite path) then there is some cycle of non-singleton vertices, in which case $L(\mathcal{A})$ does not satisfy SSC for any k and it is not SL.*

In practice, it is not necessary to build even just the subgraph of $PSG(\mathcal{A})$ generated by Q . All that one needs for a counter-example to SSC is a single pair of strings in which SSC fails. So it suffices to just explore the subgraph of $PSG(\mathcal{A})$ that is generated by doubleton subsets of Q . The size of this subgraph is only $\Theta(\mathbf{card}(Q)^2)$, in contrast to the subgraph generated by Q , which is $\Theta(2^{\mathbf{card}(Q)})$.⁶

The following is an immediate consequence of this proposition.

Corollary 3. *If \mathcal{A} is a normalised DFA and $L(\mathcal{A}) \in SL_k$ then all cycles in $PSG(\mathcal{A})$ are cycles of singletons.*

4.1 *Classes of forbidden local factors*

Local factors may or may not include a left-end marker at the beginning or a right-end marker at the end or both. In the case that a factor contains neither, it can occur anywhere in a string (including, possibly, at the beginning or end) and we say that it is a *free factor* or, if forbidden, *free forbidden factor*. If the length of a free forbidden factor

⁶The pair-graph algorithm appears to have been first published in Caron (2000).

is one, then it has somewhat different status than free forbidden factors of greater length; it is, in essence, a restriction to the alphabet. We will refer to these as *forbidden units*. If the first symbol of a forbidden factor is ‘ \times ’, then it can only occur at the left end of the word; this is an *initial forbidden factor*. If the last symbol is ‘ \times ’, then it can only occur at the right end of the word; it is a *final forbidden factor*. Note that the length of the string that these anchored factors match is $k - 1$. An SL_k definition can restrict prefixes and suffixes of length up to $k - 1$, but not, in general length k prefixes and suffixes.⁷ Finally, if a factor contains both endmarkers it is a *forbidden word*, where the (unanchored) word it forbids is actually of length $k - 2$.

4.2 Free forbidden factors

Suppose \mathcal{A} is a DFA. A factor w is a free forbidden factor of $L(\mathcal{A})$ iff there is no path in the transition graph of \mathcal{A} from q_0 to an accepting state that includes w as a substring. If \mathcal{A} is normalised, this will be the case iff there is no path at all that is labelled w from any state of \mathcal{A} , as all such paths would necessarily lead to the sink state which has been trimmed. Thus, in $PSG(\mathcal{A})$ the path from Q that is labelled w leads to \emptyset . Again, the converse holds.

So the members of the set of all labels of paths from Q to \emptyset in $PSG(\mathcal{A})$ are free forbidden factors of $L(\mathcal{A})$. Moreover, that set includes *all* free forbidden factors of $L(\mathcal{A})$. Since in general $PSG(\mathcal{A})$ may include cycles and even in the case that $L(\mathcal{A})$ is SL it may include cycles of singleton vertices, in general this set of paths will be infinite. (In fact, since $PSG(\mathcal{A})$ invariably includes a trivial cycle on \emptyset for each $\sigma \in \Sigma$, it will *always* be infinite.) The paths including trivial cycles on \emptyset are labelled with strings in $w \cdot \Sigma^*$, where w is a free forbidden factor. We are interested in the set of paths that are minimal in the sense that the label of the path does not include the label of any other such path as a substring.

Note that, by Corollary 2, any such path that includes an in-edge to Q or an out-edge from \emptyset includes another path from Q to \emptyset that is

⁷ In the original definition of SL_k (McNaughton and Papert 1971) prefix and suffix factors and forbidden words could be of length k . But the definition we use is equivalent in all significant aspects and accounts for the information contained in an anchored factor; it has become the prevailing definition in most of the literature.

strictly shorter. Thus none of those paths are minimal free forbidden factors. Note, also, that if $L(\mathcal{A}) \in \text{SL}$, then there are no cycles on Q , although there will always be trivial cycles on \emptyset for each $\sigma \in \Sigma$.

The next two lemmas establish that if $L(\mathcal{A})$ is SL then there is some bound such that all cyclic paths from Q to \emptyset in $\text{PSG}(\mathcal{A})$ with length greater than that bound will be labelled with a string that includes, as a suffix, the label of an acyclic path from Q to \emptyset . Thus the set of minimal free forbidden factors of $L(\mathcal{A})$ is just the set of labels from paths from Q to \emptyset in $\text{PSG}(\mathcal{A})$ that do not include the label of any other such path as a suffix and that do not include self-edges on \emptyset . This allows us to collect forbidden factors with a breadth-first bottom-up traversal of $\text{PSG}(\mathcal{A})$.

Lemma 2. *If v and w label paths from Q to \emptyset in $\text{PSG}(\mathcal{A})$ that do not include loops on \emptyset and $v \preceq w$, then $w = uv$ for some $u \in \Sigma^*$.*

Proof. $v \preceq w$ iff, by definition, $w = uvx$ for some $u, x \in \Sigma^*$. Since $Q \xrightarrow{v} \emptyset$ and all vertices S of $\text{PSG}(\mathcal{A})$ are subsets of Q , for all vertices S , $S \xrightarrow{v} \emptyset$ as well, and, in particular, $Q \xrightarrow{u} S \xrightarrow{v} \emptyset$. Hence x is either ε or the path it labels is a self-loop on \emptyset . ■

Lemma 3. *Let \mathcal{A} be a DFA such that $L(\mathcal{A}) \in \text{SL}$. If a path from Q to \emptyset in $\text{PSG}(\mathcal{A})$ includes a cycle other than a trivial cycle on Q or \emptyset , then there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to \emptyset as a suffix.*

Proof. Since $L(\mathcal{A})$ is SL, any cycle must be a cycle of singletons. Suppose then that there is a path:

$$Q \xrightarrow{u} \{q_0\} \xrightarrow{v} \{q_1\} \xrightarrow{w} \{q_0\} \xrightarrow{x} \emptyset$$

where, possibly, v may be a prefix of x . Since $q_0, q_1 \in Q$ there must be a path:

$$Q \xrightarrow{u} S_0 \xrightarrow{v} S_1 \xrightarrow{w} S_2 \xrightarrow{v} S_3 \dots$$

where $q_0 \in S_{2i}$ and $q_1 \in S_{2i+1}$ for $i \geq 0$. Since there are no cycles of non-singletons, by Lemma 1 the sequence of S_i s must ultimately be decreasing in size. Thus, for some n it resolves to:

$$Q \xrightarrow{v} S_1 \xrightarrow{w} S_2 \xrightarrow{v} S_3 \dots \xrightarrow{w} S_{2n} = \{q_0\} \xrightarrow{x} Q$$

So $(vw)^n x$ labels a path from Q to \emptyset and will be a suffix of all paths Q to \emptyset that take the $\{q_0\} \xrightarrow{w} \{q_1\}$ cycle at least $2n$ times. ■

An example of this lemma, taken from the PSG of the canonical automaton for Cairene Arabic, is shown in Figure 1.

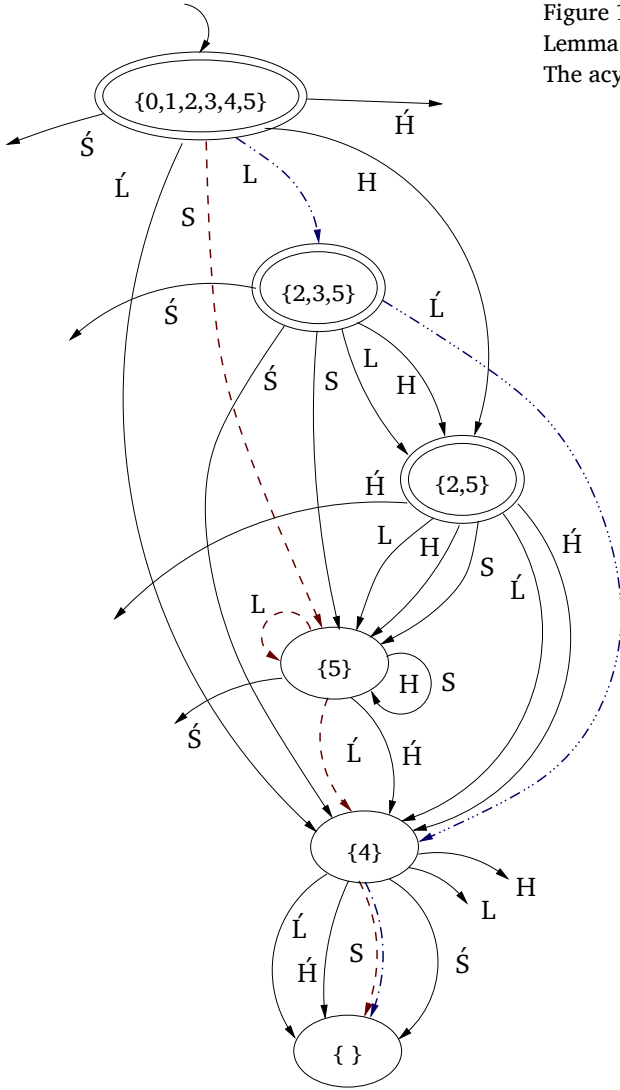


Figure 1:
 Lemma 3: The cyclic path is labelled $S(L)^*L\acute{S}$.
 The acyclic path is labelled $L\acute{L}S$

Theorem 1. *If $L(\mathcal{A}) \in SL$ then a string w is a free forbidden factor of $L(\mathcal{A}) \in SL$ iff it labels a path in $PSG(\mathcal{A})$ from Q to \emptyset . It is minimal if that path does not include any cycles other than cycles of singletons and w does not include the label of any other such path as a suffix.*

Note that if $L(\mathcal{A}) \in SL$ then the only cycles of non-singletons will be trivial cycles on \emptyset . Labels of paths including these will include some free forbidden factor as a prefix and are thus not minimal.

Paths including cycles of singletons are necessary since none of the paths labelled $u(vw)^i x$ as in the proof of Lemma 3 is labelled with a factor of any of the others; they are minimal with respect to each other. It is only the label of the acyclic path that subsumes the labels of further iterations.

4.3 *Final forbidden factors*

Suppose \mathcal{A} is a DFA. A factor w is a final forbidden factor of $L(\mathcal{A})$ iff there is no path from q_0 to an accepting state in the transition graph of \mathcal{A} that includes w as a suffix but there is some path from q_0 to an accepting state that includes w as a proper substring. (If there is no such accepting path, then w is a free forbidden factor.) If \mathcal{A} is normalised then w is a final forbidden factor iff all paths labelled w from any state in Q end at a non-accepting state and there is some such path. This will be the case iff the path from Q in $PSG(\mathcal{A})$ labelled w ends at a non-empty vertex that is disjoint with F .

Proposition 3. *No final forbidden factor of any stringset includes a free forbidden factor of that stringset as a substring.*

This is because no string includes a free forbidden factor anywhere, whereas final forbidden factors are forbidden only as suffixes; to be final but not free, there must be some string that includes the factor as a non-suffix.

In terms of the PSG free forbidden factors label paths from Q to \emptyset , so as long as we only consider paths that lead to non-empty vertices, we don't have check to see if the factor is subsumed by a free forbidden factor. Note, though, that a final forbidden factor may include another as a suffix. (It is irrelevant whether it includes a final forbidden factor as a non-suffix, since final forbidden factors are, by definition, only relevant as suffixes.)

Lemma 4. *If a path from Q to a non-empty vertex disjoint from F in $PSG(\mathcal{A})$, with $L(\mathcal{A}) \in SL$, includes a cycle other than a trivial cycle on Q , then there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to a non-empty vertex disjoint from F as a suffix.*

Theorem 2. *If $L(\mathcal{A}) \in SL$ then a string w is a final forbidden factor of $L(\mathcal{A}) \in SL$ iff it labels a path in $\text{PSG}(\mathcal{A})$ from Q to a non-empty vertex disjoint from F . It is minimal if that path does not include any cycles other than cycles of singletons and w does not include the label of any other such path as a suffix.*

The proofs are essentially the same as the proof of Lemma 3 and Theorem 1.

4.4 Algorithms for extracting forbidden local factors

Theorem 1 guarantees that if we do a breadth-first bottom-up traversal of $\text{PSG}(\mathcal{A})$ then we will discover each minimal forbidden factor before we discover any of its proper suffixes. Expanding the frontier of the search in discrete stages, every (reverse) path from \emptyset to Q found in the k^{th} stage will be a minimal forbidden k -factor.

There may be more than one such path so we do need to avoid gathering more than one instance of the factor. In general, there will be open paths (not reaching Q) that are labelled with the same factor. Extended to Q , they would include the factor as a proper suffix. So we exclude these from the frontier for the next stage.

We structure the bottom-up traversal of $\text{PSG}(\mathcal{A})$ as a top-down traversal of $\text{PSG}^R(\mathcal{A})$, in which each of the edges of $\text{PSG}(\mathcal{A})$ is reversed. For convenience (and convergence) we trim self-edges on \emptyset and Q while reversing the graph. Since we are traversing bottom-up, we actually find w^R of each factor w , but we gather these in a list structure, inserting at the head, which reverses the factor again as we construct it.

For the purposes of the algorithm, a *Path* in an edge-labelled graph $\langle V, E \rangle$, as a computational structure, is a three-tuple $\langle v, S, w \rangle$, where $v \in V$ is the final vertex of the path, $S \subseteq V$ is the (unordered) set of vertices along the path and $w \in \Sigma^*$ is the sequence of labels of the edges in the path, in reverse order. A *Frontier* is a set of paths. Forbidden factors are gathered in stages, with Stage_i expanding Frontier_{i-1} to Frontier_i , gathering the set FF_i of all minimal forbidden i -Factors in the process.

The initial frontier Frontier_0 , when searching for free forbidden factors, includes just the trivial (0-length) path from \emptyset . When searching for final forbidden factors, Frontier_0 includes the trivial path from each vertex that is a subset of $Q - F$.

Theorem 1 guarantees that, if we eliminate paths labelled with a forbidden i -Factor from Frontier_i the search will converge after finitely many iterations, k , with Frontier_k empty. (Note it is an empty set of Paths, not a set including a path ending at \emptyset .) The set of minimal free forbidden factors will be the union of the sets of factors gathered at stages 2 through k , where $L(\mathcal{A}) \in \text{SL}_k$. (Forbidden 1-factors are not included, since they are forbidden units.) The search for final forbidden factors will terminate after $k - 1$ iterations, with the minimal k -final forbidden factors including the right-end marker.

The time complexity of these algorithms is $\Theta(\mathbf{card}(\Sigma)^k)$, which is optimal for algorithms that return sets of k -factors. For an arbitrary automaton that recognises an SL stringset, we know from the pair-graph algorithm that k is no more than $\mathbf{card}(Q)^2$. Thus the complexity is $O(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)^2})$.

4.5 *Initial forbidden factors*

The initial forbidden factors of $L(\mathcal{A})$ are just the final forbidden factors of $L(\mathcal{A})^R$. We identify these by constructing \mathcal{A}^R and applying the algorithm for identifying final forbidden factors. This adds a determination step prior to generating the PSG, so this ends up being doubly exponential in the size of the automaton, $O(\mathbf{card}(\Sigma)^{2^{\mathbf{card}(Q)^2}})$. As a practical matter, this was not an issue in our application, so we did not pursue a direct algorithm, but it would be worthwhile to do so in future work.

4.6 *Forbidden words for SL stringsets*

If $L(\mathcal{A}) \in \text{SL}_k$ and \mathcal{A} is deterministic, then the words it forbids are just the labels of paths of length $k - 2$ (to allow for the endmarkers) from the (single) initial state to a state in $Q - F$. These can be gathered by doing a bounded traversal of \mathcal{A} . The time complexity of this traversal is $\Theta(\mathbf{card}(\Sigma)^k)$, thus $O(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)^2})$.

4.7 *Forbidden units*

If \mathcal{A} is normalised (minimal and trim), the forbidden units of $L(\mathcal{A})$ are just the symbols of Σ that do not label any edge in δ . In $\text{PSG}(\mathcal{A})$ these will label edges Q to \emptyset and will be gathered in Stage_1 while gathering free forbidden factors. But these may not be the only forbidden units of interest. In many applications there will be an alphabet that includes

all symbols that occur in any of a collection of stringsets and the subset of that alphabet that is not included in the alphabet of the FSA will also be significant. This is the case in many linguistic applications, for example (as in “this lect forbids unstressed heavy syllables”).

In those applications we need to include the difference between some default alphabet and the set of symbols that label edges in \mathcal{A} . Since we are building $\text{PSG}(\mathcal{A})$ anyway, the simplest way of doing this is to just take the difference between the default alphabet and the labels of the out-edges from Q . If we union that with the labels of the subset of those edges that lead to \emptyset we get the free forbidden 1-factors as well. We can avoid gathering the latter in both the set of free forbidden factors and the set of forbidden units by not including the forbidden factors gathered in Stage_1 . (Or, in order to simplify the code, by removing them from the set of free forbidden factors.)

Both of these approaches involve constructing the PSG and are consequently exponential in the number of states in the automaton. Alternatively, if all that is needed is the set of forbidden units, the set of permitted units can be gathered from the set of transitions of the automaton in time $\Theta(\text{card}(Q)^2 \cdot \text{card}(\Sigma))$. The forbidden units are then the difference between the default alphabet (plus the alphabet of the automaton, if it is not known to be a subset of the default) and the set of permitted units.

5 APPROXIMATING REGULAR STRINGSETS IN SL

Every stringset can be fully defined by the conjunction of a set of SL constraints (possibly trivial: Σ^* , \emptyset and Σ^+ are SL_1 , SL_1 and SL_2 , respectively) along with a set of properly non-SL constraints. In applications that are exploring constraints across a collection of stringsets – most linguistic applications for instance – these SL constraints are significant. If the stringset is Regular we can factor the constraints so that the non-SL constraints capture just the non-strictly-local aspects of the patterns.

The problem isn’t finding factors that characterise the stringset; the problem is that there are too many of them. $\Sigma^* - L(\mathcal{A})$, augmented with left and right endmarkers, is a set of forbidden factors that characterises $L(\mathcal{A})$ exactly. It is, of course, in general infinite.

The algorithms for SL stringsets are still partially correct for non-SL stringsets. The problem is that if $L(\mathcal{A})$ is non-SL, then there will be

non-singleton cycles (in addition to those on \emptyset) and the traversal will not terminate. These non-singleton cycles actually localise the reason that the stringset is not SL. They capture circumstances under which the automaton fails to synchronise ever; they identify places in which SSC (Proposition 1) fails for $L(\mathcal{A})$.

As with the set of forbidden words, the set of labels of the paths in $\text{PSG}(\mathcal{A})$ from Q to \emptyset that include non-singleton cycles are all legitimate forbidden factors of $L(\mathcal{A})$, but again there are infinitely many of them. The stringset they define is what we would like to isolate as the non-SL fragment of $L(\mathcal{A})$.

It is tempting to try modifying the traversal so it follows only singleton cycles. But, unfortunately, if there are non-singleton cycles the chain of the proof of Lemma 3 may be infinite, so there is no guarantee of termination even when following only singleton cycles.

Even if this were not the case, in general forbidden factors that label paths that take a non-singleton cycle one or more times can be necessary constraints in defining the SL fragment of a stringset. The reason is that it may be the case that there are acyclic paths that eventually subsume, in the sense of being included as a substring (specifically, being a suffix: see Lemma 2), paths with further iterations of the cycle. The subsuming path may be longer than the cyclic one and it may take several iterations of the cycle to form the acyclic path as a suffix.

The issue is termination. How many times must a cycle be taken before it has yielded all of the additional forbidden factors that are not just instances of an unbounded pattern? Note that if the cyclic paths will ultimately be subsumed, there will be at least one path from Q that will eventually be a suffix of that path. Traversing the PSG bottom-up, these paths will share the same initial sequence of labels.

Lemma 5. *Let \mathcal{A} be any DFA. If a path from Q to \emptyset in $\text{PSG}(\mathcal{A})$ includes a cycle other than a trivial cycle on Q or \emptyset , then either:*

- *there is a finite bound on the number of times the cycle can be taken before the label of the path includes the label of an acyclic path from Q to \emptyset as a suffix*
- *or there is a finite bound on the number of times the cycle can be taken before all paths with labels that share the same suffix as this*

one, beginning at the first iteration of the cycle, are instances of a non-SL pattern.

This is simply because if there is no acyclic path with labels that share the same suffix, then the label of the path is an instance of a factor that contains a substring that can be iterated unboundedly many times.

In order to identify the factors that label cyclic paths from Q to \emptyset which will eventually be subsumed by an acyclic path, i.e., that are not simply instances of an unbounded pattern, we modify the bottom-up algorithm to follow all paths in parallel.⁸ Rather than a frontier that is a set of paths, the frontier becomes a set of equivalence classes of paths with each path in a class sharing the same sequence of labels. If a class includes a path that reaches Q , then all paths in the class are subsumed by that path; the class is removed from the frontier of the traversal. On the other hand, if a class includes only cyclic paths, then it will never be subsumed in this way, the paths in the class are instances of unbounded patterns and we drop them; hence this traversal can be limited to acyclic paths and is guaranteed to terminate in time bounded by the size of the PSG, i.e. $\Theta(2^{\text{card}(Q)})$.

The same strategy suffices for initial and final factors with the initial frontier taken to be the single class of trivial paths from subsets of Q that are disjoint with F , all of which are labelled ε . Note that the naïve algorithm for initial forbidden factors is in this case triply exponential in the size of the automaton.

In gathering forbidden words of non-SL stringsets the bound k is not known from building the PSG, but it can be estimated by the maximum of the length of the longest free forbidden factor minus 2, the length of the longest forbidden final or initial factor minus 1 and the length of the longest acyclic path from a start state to a non-accepting state in the automaton. This bound can be found implicitly while gathering the forbidden words. Once all words shorter than the bounds provided by the free, final and initial forbidden factors have been collected, all cyclic paths can be trimmed.

Unlike the case of SL stringsets, there is no smaller bound on the size of the factors in the approximation. The time complexity of the

⁸The reversed PSG is non-deterministic and this is just the same strategy used in determining an NFA using the powerset construction.

traversals of these algorithms is $O(\mathbf{card}(\Sigma)^{(2^{\mathbf{card}(Q)})})$. The number of forbidden factors of an SL approximation of a non-SL Regular stringset is similarly bounded above, although we leave open the question of whether there are DFAs that witness this complexity (i.e., whether it is a big- Ω bound, as well). But relative to this upper bound the algorithm is optimal, and if a smaller bound can be established the traversal will almost certainly be bounded in the same way. The following theorem shows that the approximation is optimal in the sense of producing a stringset of minimal size.

Theorem 3. *If $L(\mathcal{A})$ is non-SL, the free, final and initial forbidden factors gathered by the modified bottom-up traversal of $\text{PSG}(\mathcal{A})$, when combined with the forbidden words up to the bound given above, define the minimal SL superset of $L(\mathcal{A})$ that does not include the effect of arbitrarily many instances of properly non-SL constraints.*

Proof. By Theorems 1 and 2, every path from Q to \emptyset or a vertex disjoint from F is either a free forbidden or final forbidden factor of $L(\mathcal{A})$, respectively. The same is true for the initial forbidden factors and the forbidden words. Thus the approximation does not exclude any string that is not also excluded by $L(\mathcal{A})$.

Those paths that are trimmed in the traversal either include another such path as a suffix or are instances of an unbounded pattern, i.e., include iterations of non-singleton cycles that may be iterated arbitrarily many more times without being subsumed by an acyclic path. These instances of unbounded patterns are the (infinite) set of forbidden factors defined by a properly non-SL constraint. Since the constraints are all negative, these instances can only reduce the stringset defined by the constraints. Including all of them yields $L(\mathcal{A})$; including none of them yields the SL approximation that is minimal in the sense of the theorem. ■

5.1 *Residue automata*

As noted in the proof of Theorem 3, the approximation, if not exact, will overgenerate. We turn now to the problem of characterising the strictly non-SL constraints that it omits.

Most work on approximating stringsets with stringsets in a weaker complexity class has focused on approximating CFLs with Regular stringsets (Nederhof 2000 includes a good survey) or Tree-Adjoining

Stringsets (TALs) with CFLs (Schabes and Waters 1993; Rogers 1994). Whenever the class of stringsets that is being approximated includes CFLs, the (symmetric) difference between the approximation and the target will not, in general, be a decidable set. Consequently, there is little that can be determined about that difference.

We have the advantage that all of our stringsets are Regular, and so the difference is not only decidable but an automaton recognising it is effectively constructible. Moreover, in this case, we know that every string excluded by the approximation is necessarily excluded by the target. The approximation never undergenerates. To isolate the non-SL characteristics of the target, we construct an automaton that recognises exactly the set of strings that are overgenerated by the SL approximation.

Using well-known algorithms for combining automata, an automaton \mathcal{A}_{FF} that recognises the set of strings licensed by the set of forbidden factors can be constructed. One starts with deterministic automata that recognise each of the given factors, complements them and then builds the automaton that recognises the intersection of those complements. It is then straightforward to construct \mathcal{A}_{res} , the residue automaton,⁹ which recognises exactly $L(\mathcal{A}_{\text{FF}}) - L(\mathcal{A})$. This residue automaton captures exactly the non-SL aspects of $L(\mathcal{A})$.

6 STRICTLY PIECEWISE STRINGSETS

A stringset L is *Strictly k -Piecewise* ($L \in \text{SP}_k$) iff it is strictly determined by its k -subsequences.

Let $G_{L,k}^< \stackrel{\text{def}}{=} F_{\leq k}^<(L)$ be the set of piecewise factors that occur in L . Then the stringset generated by $G_{L,k}^<$ is:

$$L(G_{L,k}^<) = \{w \in \Sigma^* \mid F_{\leq k}^<(w) \subseteq G_{L,k}^<\}.$$

If $L \in \text{SP}_k$ then $L(G_{L,k}^<) = L$.

⁹The term “residue” is motivated from the perspective of factoring constraints. These automata should not be confused with the *residual automata* of Denis *et al.* (2002), NFAs in which every state corresponds to the residual stringset w.r.t. some prefix. “Residual” in that context is justified from the perspective of factoring strings.

Let $\overline{G_{L,k}^<} = F_{\leq k}^<(\Sigma^*) - G_{L,k}^<$, the set of forbidden piecewise factors of L . This is more natural in many applications, including many linguistic ones (as in “no syllable with primary stress occurs following another such syllable”). Given $\overline{G_{L,k}^<}$, the stringset L can be described as the intersection of the complements of finitely many shuffle ideals:

$$L = \bigcap_{w \in \overline{G_{L,k}^<}} [\overline{\text{SI}(w)}].$$

A stringset is said to be SP if it is SP_k for any finite k . The following propositions characterise SP and SP_k .

Proposition 4 (Subsequence Closure). *A stringset L is SP iff*

$$(\forall w \in L, v \in \Sigma^*) [v \sqsubseteq w \implies v \in L].$$

This is because if $v \sqsubseteq w$ and w is in L , then v is a permitted factor and thus cannot be excluded from the stringset.

Proposition 5. *An SP stringset L is SP_k iff*

$$(\forall w \notin L) [(\exists v \notin L) [|v| \leq k \wedge v \sqsubseteq w]].$$

In other words, if $L \in \text{SP}_k$ and a string w is forbidden in L , then w contains a subsequence v of length less than or equal to k such that v is also forbidden in L .

The properties of Strictly Piecewise stringsets combine to allow efficient extraction of $\overline{G_{L,k}^<}$ from the automaton representation of such a stringset.

Theorem 4. *If \mathcal{A} is a DFA representing an SP stringset, then a factor w is a forbidden subsequence of $L(\mathcal{A})$ iff there is no path labelled w from the initial state, q_0 , to an accepting state.*

Proof. Let $w \in \Sigma^*$ be a string. We first show that if there is no path labelled w from the initial state q_0 to an accepting state, then w is a forbidden subsequence of $L(\mathcal{A})$. Suppose the hypothesis; i.e., that for all accepting states q_f , there is no path $q_0 \xrightarrow{w} q_f$. Then $w \notin L(\mathcal{A})$, and by the contrapositive of Proposition 4, for all strings $x \in \Sigma^*$, if $w \sqsubseteq x$, then $x \notin L(\mathcal{A})$. Then w is by definition a forbidden subsequence.

Next, we show that if w is a forbidden subsequence of $L(\mathcal{A})$, then there is no path labelled w from the initial state q_0 to an accepting

state q_f . Let w be a forbidden subsequence. Since \mathcal{A} is deterministic, there is exactly one path from q_0 labelled w . Since w is a forbidden subsequence and necessarily a subsequence of itself, $w \notin L(\mathcal{A})$. So the final state of this path must not be accepting. In other words, there is no path labelled w from the initial state, q_0 , to an accepting state. ■

A direct consequence of Theorem 4 is that, if \mathcal{A} is a DFA that represents an SP stringset, then its forbidden factors can be extracted by a simple traversal of the graph, collecting the strings that do not end in an accepting state. One concern is that a graph may have cycles, but in the case of a cyclic path $q_0 \xrightarrow{w} q_1 \xrightarrow{x} q_1 \xrightarrow{y} q_2$ there is also a path $q_0 \xrightarrow{w} q_1 \xrightarrow{y} q_2$ that avoids the cycle, and in this case wxy is a permitted subsequence iff wy is. So an acyclic traversal of a DFA of an SP stringset L is sufficient to extract $G_{L,k}^<$. This extraction algorithm is thus $\Theta(\mathbf{card}(\Sigma)^{\mathbf{card}(Q)})$. Since the maximum length of a piecewise factor of the SP stringset recognized by \mathcal{A} is $\mathbf{card}(Q)$, this is also the number of possible forbidden factors of that stringset and this algorithm is optimal for algorithms that return the set of subsequences. To reduce the result to a minimal set of forbidden factors is quadratic in the same measure, as each extracted factor must be tested for subsequence against each longer one. Given this minimal set of forbidden factors, the necessary factor width k is simply the size of the longest factor.

Note that this traversal will not necessarily yield the expected result on an arbitrary NFA, as an edge could be missing following one path but present following another with the same label.

7 APPROXIMATING REGULAR STRINGSETS IN SP

The minimal SP superset of any stringset L is just the closure of L under subsequence: any such superset must include L , and every SP superset must be closed under subsequence.

In Section 6 we describe an algorithm to extract subsequences from DFAs that represent SP stringsets. To use this to obtain an SP approximation of an arbitrary Regular stringset the DFA must first be closed under subsequence. We can achieve this closure by adding edges on ε in parallel to each existing edge of the automaton; Theorem 5 shows that the set of permitted subsequences of this generated stringset is exactly the same as that of the input.

Theorem 5. *If \mathcal{A} is a DFA and \mathcal{A}' is the automaton produced by adding ε -edges in parallel to each existing edge of \mathcal{A} (applying the subsequence closure algorithm), then the set of permitted factors of $L(\mathcal{A})$ is the same as that of $L(\mathcal{A}')$.*

Proof. In order to show that these sets are equal, we will show that each is a subset of the other.

To show that the set of permitted subsequences of $L(\mathcal{A})$ is a subset of that of $L(\mathcal{A}')$, let $u \in \Sigma^*$ be a permitted subsequence of $L(\mathcal{A})$. Then there exists a word $v \in L(\mathcal{A})$ such that $u \sqsubseteq v$, and since $L(\mathcal{A}) \subseteq L(\mathcal{A}')$ by construction, it follows that $v \in L(\mathcal{A}')$. Thus u is a permitted subsequence in the latter. Since this holds for all such u , the set of permitted subsequences of $L(\mathcal{A})$ is a subset of that of $L(\mathcal{A}')$.

To show that the reverse holds as well, let $w \in \Sigma^*$ be a permitted subsequence of $L(\mathcal{A}')$. Then there exists some string $x \in L(\mathcal{A}')$ such that $w \sqsubseteq x$. Since \mathcal{A}' was formed by allowing a computation to skip edges in \mathcal{A} , there must exist some string $y \in L(\mathcal{A})$ such that $x \sqsubseteq y$. By transitivity, it follows that $w \sqsubseteq y$ and thus w is a permitted subsequence of $L(\mathcal{A})$. Since this holds for all such w , the set of permitted subsequences of $L(\mathcal{A}')$ is a subset of that of $L(\mathcal{A})$.

Since each is a subset of the other, these sets are equal. ■

Since $L(\mathcal{A}')$ is by construction closed under subsequence, it is Strictly Piecewise; Theorem 5 guarantees that its permitted subsequences are the same as those of $L(\mathcal{A})$. If the desired outcome is simply an SP approximation, this algorithm is good: this subsequence closure is the closest SP approximation that contains its input as a subset, and the NFA is produced in linear time in the number of node pairs, $\Theta(\mathbf{card}(Q \times Q))$. If all that is needed is the approximation, this NFA can be determined and is of size $\Theta(2^{\mathbf{card}(Q)})$. The extraction algorithm from Section 6 can then be used, if desired, to extract the set of forbidden factors from this approximation. The time complexity of the whole process is $\Theta(\mathbf{card}(\Sigma)^{2^{\mathbf{card}(Q)}})$.

On the other hand, a modified version of the algorithm is more efficient. Let \mathcal{A} be a trim DFA, and create a map $ec : Q \rightarrow \{Q\}$ such that $ec(q)$ is the set of states reachable from q (the ε -closure of q). As finding $ec(q)$ is a graph traversal that touches each edge exactly once, it is $\Theta(\mathbf{card}(\Sigma \times Q))$. In the worst case, we find this for each state, so finding the ε -closure map is $\Theta(\mathbf{card}(\Sigma \times Q \times Q))$.

To compute the subsequence closure of \mathcal{A} without explicitly including any edges on ε , we must create new edges and determine the set of initial and accepting states. For each edge $\langle \sigma, p, q \rangle$, create new edges $\{\langle \sigma, p, r \rangle \mid r \in ec(q)\}$. Since every state is reachable from the initial state, in the subsequence closure the set of initial states is simply the set of all states. Since \mathcal{A} is trim, if there is any accepting state at all, then every state must be able to reach a final state, and thus similarly in the subsequence closure the set of accepting states is also the set of all states. Since the original automaton \mathcal{A} is deterministic, there are at most $\mathbf{card}(\Sigma \times Q)$ edges, and since $\mathbf{card}(ec(q))$ is at most $\mathbf{card}(Q)$, assuming sublinear lookup of $ec(q)$, the computation of this subsequence closure is then $\Theta(\mathbf{card}(\Sigma \times Q \times Q))$.

Next, add a marked non-accepting sink state $\perp \notin Q$ and complete the automaton, adding edges to \perp from each state q for each symbol not already labelling an out-edge of q . The time complexity of the completion is $\Theta(\mathbf{card}(\Sigma \times Q))$. Because \mathcal{A} was trim, this sink will necessarily be the unique non-accepting sink. Then there will be at most $\mathbf{card}(\Sigma \times Q)$ edges out of each state.

From this point, we traverse the automaton nondeterministically (grouping paths with the same label) from the set of initial states (equal to Q), where the desired paths are those that end in the unique non-accepting sink state \perp and at least one component path is acyclic. If all component paths are cyclic, then there is a strictly shorter forbidden subsequence that subsumes it, thus the paths are bounded in length by that of the longest possible acyclic path: $\mathbf{card}(Q)$. We alleviate the concern that in an NFA a factor may appear to be forbidden along one path but actually be permitted along another through the requirement that all component paths lead to \perp . Then since paths will be at most $\mathbf{card}(Q)$ edges long where each may be any of the $\Theta(\mathbf{card}(\Sigma \times Q))$ edges that start at the current state, we gather $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$ paths. The overall time complexity for this modified algorithm is then the sum of each step, dominated by the gathering of paths. In all, it is $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$.

Again, finding a minimal set requires a quadratic-time filter across the extracted factors, which can either be done during the extraction or as a post-processing step.

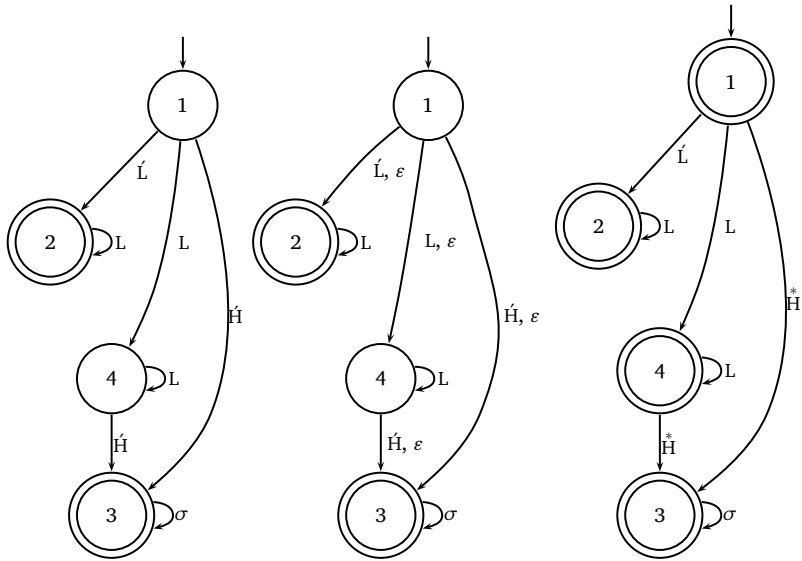
As the longest forbidden factor is of length $\Theta(\mathbf{card}(Q))$, there are $\Theta(\Sigma^{\mathbf{card}(Q)})$ possible minimal forbidden factors and this modified algo-

rithm for extracting piecewise factors from a non-SP stringset is still suboptimal, though it is still of lower asymptotic complexity than determinising and using the algorithm of Section 6.

If all that is needed is the Strictly Piecewise approximation, the NFA of the subsequence closure can be produced in $\Theta(\mathbf{card}(Q \times Q))$ time, or the DFA in $\Theta(2^{\mathbf{card}(Q)})$ time. If the set of forbidden factors is desired, the algorithm described in this section provides this in $\Theta(\mathbf{card}(\Sigma \times Q)^{\mathbf{card}(Q)})$ time. It should be noted that the Strictly Piecewise factors of Regular stringsets are derived from the approximation, in contrast to the Strictly Local factors and approximations where this relationship is inverted. This saves a considerable amount of time in practice.

We now demonstrate the application of this procedure by applying it to the stress pattern of Amele, described in StressTyp2 (Goedemans *et al.* 2015) as follows: “In words of all sizes, primary stress falls on the leftmost heavy syllable, else on the initial syllable.” This is shown in Figure 2.

Figure 2:
The application
of the
subsequence
closure
algorithm to
Amele. At left,
the normalised
DFA representing
Amele itself; in
centre, the
addition of
 ϵ -edges; and at
right, the
normalised result



Note that Amele is fully described by a set of forbidden subsequences and one additional constraint: that some syllable with primary stress must occur, which, following Hyman (2009), we will refer to as obligatoriness. The (non-trimmed) residue of the subsequence

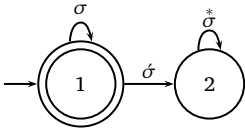


Figure 3:
The residue of Amele's Strictly Piecewise approximation

closure of Amele with Amele itself is shown in Figure 3. The complement of this residue (the *coresidue*) is exactly obligatoriness. While it does not always work out this cleanly, this certainly motivates further analysis of the residue in order to obtain additional constraints.

8 LOCAL AND PIECEWISE TESTABLE CONSTRAINTS

As Rogers *et al.* (2012) showed, three quarters of the lects in the StressTyp2 database are Strictly Local. They also identified a set of Strictly Piecewise constraints that enforce aspects of the remaining lects, although SL and SP constraints are not sufficient by themselves. Working analytically from the English glosses of the stress patterns, they derived a set of nine Locally Testable constraints that, together with the SL and SP constraints suffice to define 104 of the 106 patterns, the exceptions being Cyrenaican and Negev Bedouin Arabics, which are properly Regular.¹⁰ Thus they established that, with those two exceptions, all of the stress patterns are, at most, conjunctions of Locally Testable and Strictly Piecewise (LT + SP) constraints.

Most prominent among these LT constraints is the requirement that primary stress falls on some syllable of every word, i.e. obligatoriness. Since SP stringsets are closed under substring and SL stringsets are closed under substitution of suffixes, this requirement cannot be enforced, in general, by any conjunction of SL and SP constraints. In the lects that are SL, primary stress is required to fall within a fixed distance of either the initial or final syllable which allows obligatoriness to be enforced by initial or final forbidden factors. But SP constraints are oblivious to the ends of words, which accounts for the fact that none of the lects are purely SP.

Note that the complement of obligatoriness – that no primary stress occurs – is SL_1 , which puts it in the class of co-SL constraints.

¹⁰ Regular but not Star-Free. See McNaughton and Papert (1971).

These are constraints that are disjunctions of positive literals, a significantly restricted subset of the full set of LT constraints. From a cognitive perspective, co-SL constraints are very nearly as simple as SL constraints. Paraphrasing, SL constraints correspond to “you can’t say X ”, where the X is a fixed string of syllables, while co-SL constraints correspond to “you can’t not say X ”. Note though that, in general, since co-SL constraints are disjunctive, X may be a set of alternatives.

In the case of obligatoriness, though, there is a single disjunct. All that is required is that primary stress is identifiable somewhere. Adding obligatoriness to the sets of forbidden local and piecewise factors we extract allows us to characterise another 18 lects: 98 of the 106 lects (92.5%) are SL + SP + obligatoriness and thus extremely simple in the cognitive resources they require.

This raises the question of whether we can capture the remaining lects by extracting their co-SL constraints. Unfortunately, this is not as simple as just complementing the stringset and extracting forbidden factors. While the original stringset may (in our case, will) enforce a positive literal constraint, it will be lost in the noise of all of the strings that fail to be in the original stringset for other reasons.

Proposition 6. *If L is a stringset that forbids piecewise or free local factors, then \bar{L} cannot forbid piecewise or free, initial or final local factors. If L forbids only initial local factors, \bar{L} cannot forbid piecewise, free or final local factors, although it may forbid initial local factors. The same is true, mutatis mutandis, of final local factors.*

To see why this is so, suppose L has an initial forbidden local factor v . It follows that for all $w \in \Sigma^*$, vw is not in L . Since this applies for all w , \bar{L} has neither free nor final forbidden local factors, nor does it have any forbidden piecewise factors.

If v were instead a free forbidden local factor or a forbidden piecewise factor, these same conclusions hold. However, it is also true in that case that for all $w \in \Sigma^*$, wv is not in L . Since this applies for all w , \bar{L} has no initial forbidden factors either.

If v were instead a final forbidden local factor, then L^R has an initial forbidden local factor (v^R). Thus \bar{L}^R has neither free nor final forbidden factors. As they are equivalent, this holds for \bar{L}^R . Then \bar{L} has neither free nor initial forbidden factors.

The situation is not hopeless, though. Having identified forbidden local and piecewise factors, we can look for co-SL or co-SP constraints in the residue set, the difference between our approximation and the original stringset. Since the forbidden factor constraints have been stripped from this, we can in principle and do in fact get useful co-SL constraints in this way. This is an extension of the methodology that is demonstrated in Section 7 which resulted in the isolation of obligatoriness. This allows us to formally characterise all of the lects that are SL + SP + co-SL *ab initio*, fully computationally.

For the non-strict, non-co-strict constraints, we fall back on the LT constraints identified in the prior work. We treat these as hypotheses and systematically test each subset of the constraints to see if it suffices to complete our approximation. In this way, we determined that five of the eight non-strict, Subregular constraints identified in that work turn out to be unnecessary. The remaining three are of the form $\neg X \vee \neg \acute{H} \times$ where $X \in \{H, \grave{H}, S\}$.

Each of the two properly Regular stringsets was associated, in the prior work, with a specific Regular constraint. But since one of these constraints is just the conjunction of the other with a fourth LT constraint ($\neg \acute{L} \vee \times \acute{L} \times$, i.e., primary stress falls on a light syllable only in monosyllabic words), these can be unified with a single, simple and linguistically suggestive Regular constraint: that a light syllable with primary stress is immediately preceded by an uninterrupted sequence of an odd number of unstressed light syllables that is not itself immediately preceded by more unstressed light syllables. In the actual analysis from StressTyp2, this shows up as a requirement that unstressed syllables and those with secondary stress alternate, with the secondary stress syllables not surfacing.

9 RESULTS AND PROSPECTUS

We have designed and implemented algorithms that, given a finite-state automaton, compute a set of forbidden words, units, initial, free and final local factors that define an SL approximation of the stringset recognised by the FSA, along with a minimal DFA that recognises the residue set: the set of strings in the approximation that are not in the stringset recognised by the FSA. Similarly, we have designed and implemented an algorithm that computes a set of forbidden piecewise

factors that define an SP approximation. The intersection of these approximations is an SL + SP approximation. Then, working with the residue automaton, which recognises exactly the set of strings the approximation accepts that are not in the original stringset, we identify the forbidden factors in its complement. Those that are consistent with the original stringset are required factors of that stringset, that is, they are co-SL constraints. This gives us an SL + SP + co-SL approximation. If the FSA recognises a stringset that is in the union of these three types then the approximation is exact.

We have used these algorithms to implement a system that completely characterises any Regular stringset that is SL + SP + co-SL, fully computationally. We have applied this to the lects in the StressTyp2 database. SL + SP + co-SL constraints cover 98 (92.5%) of the 106 lects in the database that have corresponding FSAs.

For the remaining 6 Subregular lects, we have systematically tested the sufficiency of each of the subsets of the properly LT constraints that Rogers *et al.* (2012) employed in their characterisation. Of these, only obligatoriness and three other constraints were needed to fully characterise these lects. The only properly Regular constraint that is required in characterising the remaining two lects is a hidden alternation pattern that requires an odd number of syllables to occur in certain spans of the word. This is about as simple as non-trivial properly Regular constraints can get and it is reminiscent of the notion of metrical foot that plays an important role in much phonotactic analysis.

Our results sharpen the results of the prior work in the following additional ways. For the individual lects the maximum number of forbidden words is 20. Since the size of our default alphabet is 15 (five degrees of weight and three degrees of stress) and all lects have at least one weight and two levels of stress, the maximum number of forbidden units is 13. The maximum number of forbidden initial factors is 15. The maximum number of forbidden free and final factors is 386 and 117, respectively, but these are all due to Pirahã, an outlier. Without Pirahã they are 185 and 32, respectively. The maximum number of forbidden subsequences is 90, but this is due to Bhojpuri (per Shukla Tiwari), another outlier. Without this lect it is 18.

For the union factor types, there are 14 distinct forbidden units (only unstressed light syllables occur in every lect), 44 distinct for-

bidden words, 35 distinct initial forbidden factors, 904 distinct free forbidden factors and 230 distinct final forbidden factors. The maximum width of forbidden words, initial factors and free factors is 5. The maximum width of final forbidden factors is 6, due to a single lect (Içuã Tupi) which is also the only example of a properly SL_6 stringset, the other SL patterns all being SL_5 or less. There are 126 distinct forbidden subsequences with a maximum width of 4, but this again is due to Bhojpuri (per Shukla Tiwari). Without this lect, the maximum width is only 3.

That is still a lot of factors, too many to draw much insight from. But these are all in ground form, with each syllable and stress combination represented by a distinct alphabet symbol. In future work we plan to adapt the alphabet type to be tuples of features or perhaps non-reëntrant feature structures (adding full feature structures we will leave for others), which will provide opportunities to generalise across those features. We know, just from the phonology, that this will reduce the total number of exemplars significantly.

The algorithms we have presented here have asymptotic time complexity that is exponential in the size of the automaton if the stringset it recognises is Strictly Local or Strictly Piecewise. If it is not, they obtain an optimal Strictly Local approximation in time doubly exponential in the size of the automaton. (The naïve algorithm for finding initial forbidden factors, which suffices for our application, is doubly exponential.) The complexity of the algorithm for obtaining the optimum SP approximation is still just singly exponential. This relatively high degree of complexity is typical of algorithms in this domain, most of which are based on properties of the Syntactic Monoid (or Semigroup), which is exponential in the size of the automaton; the powerset graph is only marginally smaller than the Syntactic Monoid. With the exception of the naïve algorithm, they are all optimal for algorithms that return the sets of forbidden factors of the stringset.

In practice, the algorithms are quite efficient on moderate sized automata. The full methodology, running on our Haskell workbench, processes the entire set of 106 lects (≤ 33 states) in under one minute running on an AMD64 based PC with four cores at 3.7 GHz with 12GB of RAM, with a disproportionate fraction of that time spent processing the SL stringsets with large k . Thus the workbench, with only minimal

optimisation, has proven to be a useful interactive tool for exploring Regular sets of strings.

Nevertheless, the performance can certainly be improved significantly. The asymptotic bound is due to the potential size of the set of factors. This is not, however, the dominant factor in the practical performance. Rather it is the time it takes to generate a minimal DFA from the forbidden factors. This is only necessary for construction of the residue automata. If all that is required is the sets of factors it can be dispensed with. Moreover, it is an easy target for optimisation and is of the type of “embarrassingly parallel” algorithms that Haskell can parallelise extremely effectively. In essence, the performance will ultimately be bound only by the number of cores one has available.

ACKNOWLEDGEMENTS

The work reported here builds on the work of at least a dozen undergraduate students and alumni of Earlham College over the course of about ten years. We are greatly indebted to their willingness to think carefully about hard problems and to collaborate effectively both within the group and over time. We are also indebted to the collaboration of Jeff Heinz and his students in Linguistics and Cognitive Science at the University of Delaware, and to the detailed and extremely helpful suggestions of the anonymous reviewers.

REFERENCES

- Pascal CARON (2000), Families of locally testable languages, *Theoretical Computer Science*, 242:361–376.
- Jane CHANDLEE (2014), *Strictly local phonological processes*, Ph.D. thesis, University of Delaware.
- François DENIS, Aurélien LEMAY, and Alain TERLUTTE (2002), Residual finite state automata, *Fundamenta Informaticae*, 51(4):339–368.
- Matt EDLEFSEN, Dylan LEEMAN, Nathan MYERS, Nathaniel SMITH, Molly VISSCHER, and David WELLCOME (2008), Deciding strictly local (SL) languages, in Jon BREITENBUCHER, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pp. 66–73.

Extracting Subregular constraints

- Margaret FERRO, Dakotah LAMBERT, Sean WIBEL, and James ROGERS (2014), Abstract categories of phonotactic constraints, https://apps.cur.org/ncur2018/archive/Display_NCUR.aspx?id=83628, Retrieved 30 January 2018, presented at the National Conference on Undergraduate Research (NCUR'14).
- Jie FU, Jeffrey HEINZ, and Herbert G. TANNER (2011), An algebraic characterization of strictly piecewise languages, in Mitsunori OGIHARA and Jun TARUI, editors, *Theory and applications of models of computation*, volume 6648 of *Lecture Notes in Computer Science*, pp. 252–263, Springer, Berlin.
- R. W. N. GOEDEMAN, Jeffrey HEINZ, and Harry VAN DER HULST (2015), StressTyp2, <http://st2.ul1et.net/>, retrieved 30 January 2018.
- Jeffrey HEINZ (2010), Learning long-distance phonotactics, *Linguistic Inquiry*, 41(4):623–661.
- Jeffrey HEINZ (2018), The computational nature of phonological generalizations, in Larry HYMAN and Frank PLANK, editors, *Phonological typology*, volume 23 of *Phonetics and Phonology*, chapter 5, pp. 126–195, Mouton De Gruyter, Berlin.
- John E. HOPCROFT and Jeffrey D. ULLMAN (1979), *Introduction to automata theory, languages and computation*, Addison-Wesley, Boston, MA.
- Larry M. HYMAN (2009), How (not) to do phonological typology: the case of pitch-accent, *Language Sciences*, 31(2–3):213–238.
- Adam JARDINE (2016), *Locality and non-linear representations in tonal phonology*, Ph.D. thesis, University of Delaware.
- M. LOTHAIRE, editor (1983), *Combinatorics on words*, Cambridge University Press, New York.
- Robert MCNAUGHTON and Seymour PAPERT (1971), *Counter-free automata*, MIT Press, Cambridge, MA.
- Mark-Jan NEDERHOF (2000), Practical experiments with regular approximation of context-free languages, *Computational Linguistics*, 26(1). <http://aclweb.org/anthology/J00-1003>.
- Geoffrey K. PULLUM and Barbara C. SCHOLZ (2001), On the distinction between model-theoretic and generative-enumerative syntactic frameworks, in Philippe DE GROOTE, Glyn MORRILL, and Christian RETORÉ, editors, *Logical Aspects of Computational Linguistics: 4th international conference*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pp. 17–43, Springer Verlag, Berlin.
- James ROGERS (1994), Capturing CFLs with tree adjoining grammars, in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pp. 155–162, Association for Computational Linguistics, Las Cruces, NM.

James ROGERS (1996), A model-theoretic framework for theories of syntax, in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pp. 10–16, Association for Computational Linguistics, Santa Cruz, CA.

James ROGERS, Jeff HEINZ, Margaret FERO, Jeremy HURST, Dakotah LAMBERT, and Sean WIBEL (2012), Cognitive and sub-regular complexity, in Glyn MORRILL and Mark-Jan NEDERHOF, editors, *Formal Grammar 2012*, volume 8036 of *Lecture Notes in Computer Science*, pp. 90–108, Springer, Berlin.

James ROGERS, Jeffrey HEINZ, Gil BAILEY, Matt EDLEFSEN, Molly VISSCHER, David WELLCOME, and Sean WIBEL (2010), On languages piecewise testable in the strict sense, in Christian EBERT, Gerhard JÄGER, and Jens MICHAELIS, editors, *The mathematics of language: revised selected papers from the 10th and 11th biennial conference on the mathematics of language*, volume 6149 of *Lecture Notes in Artificial Intelligence*, pp. 255–265, FoLLI/Springer, Berlin.

Geoffrey SAMPSON (1975), *The form of language*, Weidenfeld and Nicolson, London.

Yves SCHABES and Richard C. WATERS (1993), Lexicalized context-free grammars, in *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pp. 121–129, Association for Computational Linguistics, Columbus, OH.

Imre SIMON (1975), Piecewise testable events, in Helmut BRAKHAGE, editor, *Automata theory and formal languages*, volume 33 of *Lecture Notes in Computer Science*, pp. 214–222, Springer Verlag, Berlin.

Wolfgang THOMAS (1982), Classifying regular events in symbolic logic, *Journal of Computer and Systems Sciences*, 25(3):360–376.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



How to embed noncrossing trees in Universal Dependencies treebanks in a low-complexity regular language

Anssi Yli-Jyrä
University of Helsinki, Finland

ABSTRACT

A recently proposed balanced-bracket encoding (Yli-Jyrä and Gómez-Rodríguez 2017) has given us a way to embed all noncrossing dependency graphs into the string space and to formulate their exact arc-factored inference problem (Kuhlmann and Johnsson 2015) as the best string problem in a dynamically constructed and weighted unambiguous context-free grammar. The current work improves the encoding and makes it shallower by omitting redundant brackets from it. The streamlined encoding gives rise to a *bounded-depth subset approximation* that is represented by a *small finite-state automaton*. When bounded to 7 levels of balanced brackets, the automaton has 762 states and represents a strict superset of more than 99.9999% of the noncrossing trees available in Universal Dependencies 2.4 (Nivre *et al.* 2019). In addition, it strictly contains all 15-vertex noncrossing digraphs. When bounded to 4 levels and 90 states, the automaton still captures 99.2% of all noncrossing trees in the reference dataset. The approach is flexible and extensible towards unrestricted graphs, and it suggests tight finite-state bounds for dependency parsing, and for the main existing parsing methods.

Keywords:
bounded stack,
coding
morphisms,
context-free
grammars,
dependencies,
finite-state,
parsing, state
complexity,
treebanks

1

INTRODUCTION

Dependency structures – rooted trees and more general digraphs – have tremendous importance in multilingual syntactic analysis and in the related semantic analysis, and its applicability to the world’s

languages have been demonstrated recently very strongly by the Universal Dependencies (UD) initiative.¹ The main approaches to produce syntactic dependency structures include *graph-based parsers* (Eisner and Satta 1999; McDonald *et al.* 2005) that usually aim at exact inference, and *transition-based parsers* (Nivre 2008) that treat parsing as beam search that runs in linear time with a small risk of missing the best analysis. *Neural network-based parsers*, such as Libovický (2016), Ma and Hovy (2017) and many more, provide additional flexibility and high accuracy. In the present work, we advance the long-term development of a new, *code-theoretic parsing* approach (Yli-Jyrä and Gómez-Rodríguez 2017) that may lend itself to unforeseen combinations with the existing approaches.

Parsing that leads to *noncrossing trees and graphs* (Kuhlmann and Johnsson 2015) is a simplification of more general approaches that produce nonprojective trees and ordered graphs with crossing edges. Although such parsing is limited in coverage, it is a very important, well-understood core for some more general parsing algorithms. Recently, Yli-Jyrä and Gómez-Rodríguez (2017) have explored an approach that embeds² *the set of noncrossing digraphs* (NXDIGRAPHS) into the string space Σ^* using an injective encoding morphism between the noncrossing digraphs and the corresponding set of code strings ($L_{\text{NXDIGRAPHS}}$) that form an unambiguous context-free language:

$$\text{NXDIGRAPHS} \rightarrow L_{\text{NXDIGRAPHS}}, L_{\text{NXDIGRAPHS}} \subseteq \Sigma^*.$$

The embedding can be used to turn the finite, sentence-specific search space of noncrossing graphs dynamically into a finite string set where each string corresponds to a distinct element in the search space. This gives us a *code-theoretic parsing approach* that has five advantages:

1. **Flexibility:** Several subfamilies of noncrossing digraphs can be treated as alternative search spaces that are treated uniformly by

¹<http://universaldependencies.org/>

²In mathematics, when some object X is said to be embedded in another object Y , the obtained embedding is given by some injective and structure-preserving map $f : X \rightarrow Y$. In this work, embedding of graphs is based on code strings over a code alphabet and should not be confused with continuous vector space representations, although such an embedding is commonly used in natural language processing and in modern neural network architectures.

a generic parser whose search space can be restricted to these subfamilies (Yli-Jyrä and Gómez-Rodríguez 2017).

2. **Context-freeness:** The search space can be represented compactly with a context-free grammar that can also have weights (ibid.).
3. **Decidability:** These grammars are unambiguous and can be related to a rich calculus of tree automata. These are then connected to monadic second-order logic whose formulas define linear-time decidable properties over ordered trees and tree-decompositions of graphs (Bojańczyk and Pilipczuk 2016).
4. **Compatibility:** It is probable that the approach can be combined with existing parsing frameworks such as graph-based parsing (McDonald *et al.* 2005; Kiperwasser and Goldberg 2016; Zheng 2017), transition-based parsing (Dyer *et al.* 2015; Kiperwasser and Goldberg 2016), encoder-decoder parsing (Vinyals *et al.* 2015), parsing as sequence labeling (Strzyz *et al.* 2019) and parsing with recurrent neural network grammars (Dyer *et al.* 2016; Kuncoro *et al.* 2017).
5. **Extensibility:** There is follow-up work that extends the encoding developed in this paper to all ordered digraphs (Yli-Jyrä 2019).

In the code-theoretic arc-factored parsing approach (Yli-Jyrä 2012; Yli-Jyrä and Gómez-Rodríguez 2017), the construction of the compact representation of the complete distribution of potential parses takes cubic time. The construction involves building, dynamically, a weighted context-free grammar for the complete parse forest. The exact decoding of the optimal parse is then carried out in time that is linear to the size of the dynamic grammar. Since the combined complexity of these tasks remains in $O(n^3)$, the complexity hits the previously known worst-case bound for parsing whose output is restricted to some families of noncrossing graphs (Kuhlmann and Johnsson 2015). But in today's terms, parsing through a cubic time procedure is often considered too expensive as real-time data applications demand low latency and high throughput. More efficient parsing algorithms are already available in the established parsing frameworks. Especially transition-based parsing is a very successful and efficient parsing framework (Nivre 2008, 2009; Bohnet *et al.* 2016) that has inspired recent work

on transition-based neural network parsing (Dyer *et al.* 2015; Kiperwasser and Goldberg 2016).

According to Covington (2001, 101–102), it is important to study the *constrained computational complexity* of parsing algorithms when they are applied to natural language data:

An important principle of linguistics seems to be that *the worst case does not occur*, i.e., people do not actually utter sentences that put any reasonable parsing algorithm into a worst-case situation. Human language does not use unconstrained phrase-structure or dependency grammar; it is constrained in ways that are still being discovered.

The code-theoretic parsing approach has a special advantage in the study of the constrained computational complexity of parsing algorithms because there the constraints are reflected immediately in the complexity of the search space embedding. The unknown complexity of the sufficiently constrained search space embedding for natural language gives rise to the following hypothesis:

Hypothesis

The practically occurring (noncrossing) dependency digraphs can be embedded into a subset approximation that has a very compact finite-state representation.

The concrete aim of this article is to investigate the existence of a *practical, very compact finite-state representation* for the search space of noncrossing trees and digraphs in dependency syntactic parsing. Given an encoding morphism and a depth bound that limits the maximal complexity of dependency digraphs, the corresponding set of digraphs will be recognized by a minimal deterministic finite automaton, where each state has a constant number of transitions. The *state complexity* of the minimal automaton depends only on the language it recognizes. Thus, the only way to reduce the state complexity is to improve the embedding of the digraphs into a regular, i.e. finite-state language. The hypothesis is valid, if a very compact finite-state representation for the practically occurring dependency digraphs exists.

Going from the cubic-time algorithms for noncrossing graphs to the linear-bounded state complexity of a depth-bounded search space means that that we are slightly closer to linear-time inference over

arc-factored weighted parses. However, the scope of the current work does not allow us to study whether also the representation of the weighted search space with the arc-factored weights actually remains linear bounded and compactly represented as a finite-state network. If the number of possible distinct arc weights in the statistical model is not bounded by a constant, the dynamic finite-state representation of the weighted search space is super-linear. But since there are techniques for pruning parse forests (Roark and Hollingshead 2008; Zhang and McDonald 2014; Zheng 2017), and the weights can be also simplified, e.g. by quantisation, we may avoid such super-linearity. It is, thus, conceivable that the dynamically weighted search space could also have a good finite-state approximation if there is a dynamic finite-state representation for the corresponding unweighted search space.

The structure of this article is as follows. Sections 2.1 and 2.2 contain the definitions and basic results required to understand how the sets of noncrossing graphs and digraphs are embedded into a context-free string language. Since a finite bound for the bracketing depth is desirable, Section 3 seeks a streamlined encoding that would improve on the proposal of Yli-Jyrä and Gómez-Rodríguez (2017) by radically reducing the bracketing depth of an average parse. A proposal for such a streamlined encoding is presented and formally analysed in Section 4. Finally, the prior and the streamlined encoding are evaluated in Section 5 from the point of view of state complexity and coverage. Section 6 concludes the article and identifies some questions that remain open after the current work.

2

DEFINITIONS

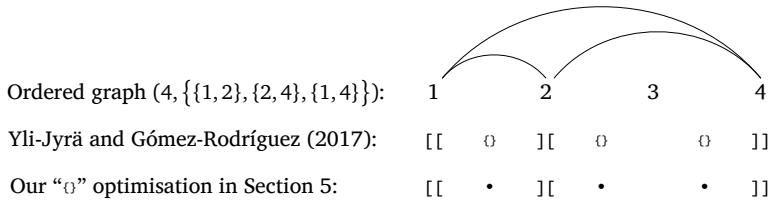
We assume that the reader is familiar with the basics of formal language theory and especially the theory of context-free grammars, finite-state automata and finite-state transducers. Algorithms will be written in a pseudo-formal language that mixes Python-like syntax with mathematical notation. In the following, we give definitions for noncrossing graphs and digraphs (Kuhlmann 2015) and the corresponding encoding that we will call *strong bracketing*, S . Strong bracketing for graphs is defined in Section 2.1, and Section 2.2 defines strong bracketing for digraphs and relates these two classes of structures.

2.1 Strong bracketing for noncrossing graphs

A (nonempty) *graph* is a pair (V, E) where V is a finite, nonempty set of vertices and $E \subseteq \{\{i, j\} \subseteq V \mid i \neq j\}$ is a set of edges. Each edge in a graph may have a label or even a multiset of labels. The *complete graph* (V, E) has all possible edges $E = \{\{i, j\} \subseteq V \mid i \neq j\}$. The vertices in graphs are usually an ordered set $V = [1, \dots, n]$ with a linear order \leq . Such an *ordered graph* (V, E) is given more simply as the pair (n, E) . By working on ordered graphs, we avoid the usual difficulty of defining equality of graphs through isomorphism: graph $(3, \{\{1, 2\}\})$ is *not* equivalent to graph $(3, \{\{2, 3\}\})$ although these graphs are isomorphic. In an ordered graph, the edge $\{i, j\} \in E$ can be viewed as an ordered pair $(\min\{i, j\}, \max\{i, j\})$. Two edges $(i, j), (k, l)$ where $i < k$ are said to be *crossing* if $k < j < l$. The *concatenation* of two ordered graphs (n, E_1) and (m, E_2) , denoted by $(n, E_1) \cdot (m, E_2)$, is $(n + m - 1, E)$ where $E = E_1 \cup \{\{i + n - 1, j + n - 1\} \mid \{i, j\} \in E_2\}$. An ordered graph is *noncrossing* if it has no crossing edges. The set of (nonempty) noncrossing graphs is denoted as NXGRAPH. Together with the trivial graph $(1, \{\})$ and concatenation, this set has the structure of a monoid.

Yli-Jyrä and Gómez-Rodríguez (2017) have proposed an encoding scheme according to which any noncrossing graph (n, E) can be represented as a string of brackets. For example, the ordered noncrossing graph $(4, \{\{1, 2\}, \{2, 4\}, \{1, 4\}\})$ is encoded as the string “[[◯] [◯ ◯]]”, see Figure 1 (the middle row).

Figure 1:
An example of
an ordered graph



The original reason for using the curly brackets “◯” in Yli-Jyrä and Gómez-Rodríguez (2017) was that, with them, the code strings respect the balanced bracketing and form a subset of a Dyck language. They also encode, intuitively, the successor edges over the vertices. Since these motivations for the curly brackets are less important in the current work, it is plausible to replace “◯” with a single character “-” to optimise the code strings; see Figure 1 (last row). This optimisa-

tion will be discussed later in this paper, in Section 5, but we stick momentarily to the original encoding (Yli-Jyrä and Gómez-Rodríguez 2017) that uses curly brackets. In both encoding schemes, the square brackets “[” and “]” connect the vertices by spanning the gaps that separate them. In particular, each pair of matching square brackets “[...]” correspond to an arc between two vertices. Because the brackets in this encoding always come in pairs, we will call this encoding a *strong bracketing*, S .

The encoding function enc_S that maps the elements of NXGRAPH to the elements of the monoid $\{[,], \{, \}\}^*$ is implemented by an algorithm that is given in Figure 2. Since the algorithm is not used during parsing, we give a simple, unoptimised version that is designed to illustrate the encoding scheme. This algorithm runs in $O(n^2)$ time, but more efficient algorithms exist.

```

def encS((n,E) ∈ NXGRAPH):
    1 str = ε
    2 for i in [1,2,...,n]:
    3     for j in [i-1,i-2,...,1]:
    4         if {j,i} in E:
    5             str += "]"
    6     for j in [n,n-1,...,i+1]:
    7         if {i,j} in E:
    8             str += "["
    9     if i<n:
    10        str += "{" + "}"
    11 return str

def decS(str ∈ LNXGRAPH,S):
    1 (n,E) = (1, {})
    2 stk = ε
    3 for c in str:
    4     if c == "[":
    5         stk.push(n)
    6     if c == "]":
    7         i = stk.pop()
    8         E += { {i,n} }
    9     if c == "{":
    10        n += 1
    11 return (n,E)

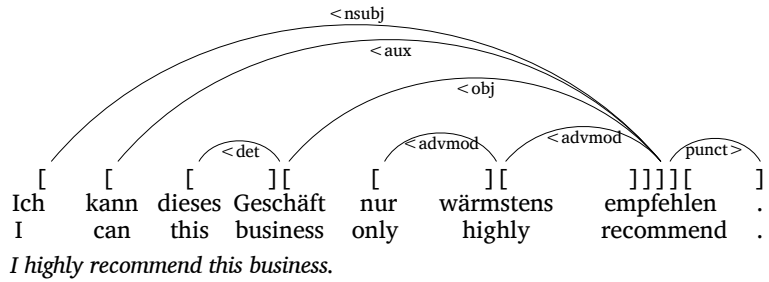
```

Figure 2:
The encoding
and decoding
algorithms
for noncrossing
ordered graphs

Lemma 2.1. *The encoding enc_S maintains an iconic correspondence between the parts of the graph and the string structure.*

Proof. The encoding function, enc_S , produces a closing square bracket for the right end of the edge, an opening square bracket for the left end of the edge and a pair of curly brackets to indicate that adjacent vertices are in a successor relation with each other. The length of the code string is exactly $2|E| + 2n - 2$ characters when $n > 0$. The empty string ε encodes the unit graph that consists of a single vertex. In other words, the encoding is based on an iconic correspondence between the graph and the bracketing. □

Figure 3:
An example
sentence with
a noncrossing
parse



To see how the encoding applies to a syntactic dependency analysis of a natural language sentence, such an analysis for an example sentence is given in Figure 3. The sentence is in German and it reads *Ich kann dieses Geschäft nur wärmstens empfehlen*. Under the line containing this sentence, there are a word-by-word English translation and a free translation in English. At the top of the figure, there is a diagram of an undirected graph that indicates the dependencies between the vertices, i.e. the tokens that constitute the sentence. The label of each edge specifies the direction of the *dependent* vertex-token and the category of this vertex-token from the perspective of the *head* vertex-token that is at the opposite end of the edge.

Between the graph diagram and the German sentence, there is a line that contains a bracket string. In this bracket string, the iconic correspondence between the brackets and the edge degree of each vertex-token is clearly recognizable, but this string does not show the curly brackets that separate the vertices of the graph. With the curly brackets, the bracket string is “[[[[[[[[[]]]]]]]] [] ”. It is also possible to add the edge labels at the corresponding brackets:

[<nsubj [<aux [<det [<obj [<advmod [<advmod []]]]]]] [<punct>] .

The encoding considered in the current article ignores the edge labels in order to keep the presentation clear. Recall that the current goal is not to develop a front-end descriptive formalism for linguists but to investigate the search space of noncrossing dependency graphs from the perspective of its state complexity.

Lemma 2.2. *The encoding function enc_s is a bijection whose inverse can be computed in linear time.*

Proof. The right side of Figure 2 presents a decoding algorithm, dec_S , that maps bracket strings to noncrossing graphs. The obtained function can be easily seen to be the inverse of enc_S . Thus, the encoding function is a bijection between its domain and the range.

Since the for-loop in the decoding algorithm dec_S needs only as many iterations as there are characters in the argument str , it computes the inverse of enc_S in linear time. \square

According to Lemma 2.2, any noncrossing graph is in a 1-to-1 relationship with the corresponding string that encodes the graph. These strings constitute a subset $L_{\text{NXGRAPH},S}$ of the free monoid $\{[,], \{, \}\}^*$. Since the input of the decoding algorithm in Figure 2 is restricted to the outputs of enc_S , the algorithm ignores the right curly bracket “}” in code strings.

Lemma 2.3. *The range of enc_S , $L_{\text{NXGRAPH},S}$, is an unambiguous context-free language.*

Proof. There is an unambiguous grammar that describes the range of the encoding function.

$$(1) \quad \begin{aligned} S &\rightarrow QS \mid \emptyset S \mid \varepsilon & S' &\rightarrow QS'' \mid \emptyset S & S'' &\rightarrow QS \mid \emptyset S \\ Q &\rightarrow [S']. \end{aligned}$$

We make three observations of the grammar:

Firstly, this grammar produces balanced bracketing over $\{[,], \{, \}\}$ where the opening and the closing curly brackets are always adjacent, like in line 10 of the enc_S algorithm.

Secondly, by the productions for the phrases S' and S'' , each level of square brackets “[]” contains one pair of curly bracket of its own or two or more nested square brackets. Thus we may have substrings “[\emptyset]”, “[\emptyset [\emptyset]]”, and “[\emptyset][\emptyset]]” but not substrings “[]” or “[[]]”. This principle avoids connecting a pair of vertices more than once and corresponds to the fact that lines 3–8 in the encoding algorithm (Figure 2) produce exactly one pair of square brackets per edge.

Thirdly, whenever two balanced substrings correspond to two subgraphs, they can be concatenated without adding any curly brackets or vertices between them. Concatenation corresponds to the S rule(s) in the grammar and the immediate succession between lines 3–5 and 5–8 in the algorithm.

These observations can be extended to an inductive formal proof over well-formed substrings and the corresponding graphs. \square

Graph concatenation generates the monoid $(\text{NXGRAPH}, \cdot, (1, \{\}))$ of noncrossing ordered graphs where the trivial graph $(1, \{\})$ is the identity element. The string concatenation of the encoded noncrossing graphs generates the monoid $(L_{\text{NXGRAPH}, S}, \cdot, \varepsilon)$ where the empty string ε is the identity element.

Lemma 2.4. *The encoding function enc_S is a homomorphism between the concatenation monoid of noncrossing graphs NXGRAPH and the concatenation monoid of code strings $L_{\text{NXGRAPH}, S}$.*

Proof. It is easy to verify that the encoding enc_S respects the monoid structure: firstly, the encoding is compositional in the sense that $\text{enc}_S(n, E_1) \cdot \text{enc}_S(m, E_2) = \text{enc}_S((n, E_1) \cdot (m, E_2))$. Secondly, the identity element of the first monoid is the trivial graph $(1, \{\})$ that is encoded as the empty string ε , the identity element of the second monoid. Thus the encoding is a homomorphism. \square

In grammars for bracketed graphs, it is often handy to use production schemas that are more expressive than the standard context-free productions. *Extended context-free grammars (ECFG)* (Salomaa 1973) extend grammar productions to production schemas whose right-hand sides are regular languages over the nonterminal and terminal symbols. ECFGs are weakly equivalent to context-free grammars but more succinct and flexible. In particular, any right linear grammar is equivalent to a ECFG that has just one rule schema and whose derivations have only one rewriting step. This expressivity of ECFG is very nice when we do not need too fine-grained derivation trees but rather want to reduce the height of derivation trees during the recognition of strings.

Lemma 2.5. *There is an extended context-free grammar that generates the language $L_{\text{NXGRAPH}, S}$ with derivation steps that correspond 1–1 to the pairs of brackets (except the topmost step).*

Proof. The original grammar of Lemma 2.3 can be written as an extended context-free grammar that removes some recursion and uses

regular expressions instead:

$$(2) \quad S \rightarrow (Q | \emptyset)^*$$

$$(3) \quad Q \rightarrow [S'] \quad S' \rightarrow (Q | \emptyset) S'' | \emptyset \quad S'' \rightarrow (Q | \emptyset)^+$$

By substitution of S' and S'' , we replace (3) to obtain:

$$(4) \quad Q \rightarrow [(\emptyset | (Q | \emptyset) (Q | \emptyset)^+)]$$

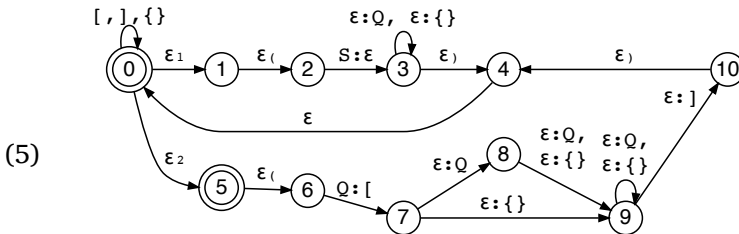
Each production schema consists of a left-hand-side (lhs) and a right-hand side (rhs) – a nonterminal and a regular language. \square

ECFGs reduce beautifully to the iterated application of finite-state transducers (FSTs). We prove just the following special case.

Lemma 2.6. *There is a finite-state transducer that represents the grammar of the proof of Lemma 2.3. Its transitive closure generates the language $L_{NXGRAPH,S}$ when restricted to the start symbol S in the input side and the terminal symbol string in the output side.*

Proof. Starting from the grammar of Lemma 2.5, we will construct one possible transducer representation. First, the two production schemas (2) and (3) compile into two finite-state transducers. Each transducer maps the lhs of the corresponding production to the corresponding rhs. A larger transducer T_G is constructed from these two subtransducers with additional epsilon transitions and self-loop transitions that accept any terminal symbols that have been produced in the earlier stages of the derivation.

The constructed grammar transducer T_G is shown in (5).



In this transducer, an edge label with a colon indicates that an input string is replaced with some other factor in the output. For example, $\varepsilon : \emptyset$ indicates that the empty string ε is replaced with the string “ \emptyset ”.

There are also some edges that do not have labels with a colon. Such labels denote a transition that copies its input to the output. Note that the production schemas (2) and (4) appear as subtransducers in the whole. The first corresponds to the transducer between states 2 and 3, and the second corresponds to the transducer between states 6 and 10. The epsilon symbols $\varepsilon_1, \varepsilon_2, \varepsilon_{(, \varepsilon)}$ denote empty strings like ε . The first two avoid cluttering the diagram and the latter two mark the beginning and end of a rule application. From the perspective of the current proof, these epsilons could have been replaced with ε and removed from the transducer all together (Mohri 1997).

The transitive closure of this transducer maps the start symbol to all the intermediate (“sentential form”) strings that can be derived from S with the production schemas. When these strings are restricted to the terminal alphabet $\{[,], (,)\}$, we obtain the string language generated by the original grammar. \square

2.2 Strong bracketing for noncrossing digraphs

A (nonempty) digraph is a pair (V, A) where V is a nonempty set of vertices and $A \subseteq \{(i, j) \in V \times V \mid i \neq j\}$ is a set of arcs. Each arc (u, v) , written as $u \rightarrow v$, is a directed edge from vertex u to vertex v . A digraph (V, A) is *inverted* if $(i, j) \in A$ implies $(j, i) \in A$. The *complete digraph* (V, A) has all possible arcs $A = \{(i, j) \in V \times V \mid i \neq j\}$. The *underlying graph* of a digraph (V, A) is the graph (V, E_A) where $E_A = \{(u, v) \mid (u, v) \in A\}$ is the set of underlying edges. Note that the cardinality $|E_A|$ of the set of underlying edges can be smaller than the cardinality $|A|$ of the set of arcs. An *ordered digraph* (n, A) is a digraph (V, A) with \leq -ordered vertices $V = [1, \dots, n]$ and a *noncrossing digraph* (n, A) is an ordered digraph whose underlying ordered graph (n, E_A) is noncrossing. The set of (nonempty) noncrossing digraphs is denoted as NXDIGRAPH. This set extends to a concatenation monoid in the same way as the carrying set of the concatenation monoid of noncrossing graphs.

Lemma 2.7. *There is a bijection between ordered digraphs and ordered graphs with 3 labels for edges.*

Proof. Let $C = \{\leftarrow, \rightarrow, \leftrightarrow\}$ be the set of three edge labels. Let f be the trivial function that maps each ordered digraph (n, A) to its underlying graph (n, E_A) and a labelling function $\lambda : E_A \rightarrow C$ such that $\lambda(\{i, j\}_{i < j}) = \leftrightarrow$ if $(i, j), (j, i) \in A$, $\lambda(\{i, j\}_{i < j}) = \rightarrow$ if $(i, j) \in A, (j, i) \notin A$

inverted digraph (V, A) where $A = \{(i, j), (j, i) \mid \{i, j\} \in E\}$. Since this is also a function, f is a bijection. \square

By Lemma 2.9, graphs can be treated as special cases of digraphs. Yli-Jyrä and Gómez-Rodríguez (2017) employ the encoding of non-crossing digraphs and Lemma 2.9 (implicitly) to construct unambiguous context-free languages that encode important families of digraphs and graphs. Such context-free languages correspond to the rooted non-crossing trees, the projective trees, the noncrossing dags, the noncrossing weakly connected dags, unoriented noncrossing trees and many other families of noncrossing digraphs and graphs.

Although digraph bracketing is more general and expressive than the graph bracketing, it has two practical disadvantages due to which we prefer to focus, in the rest of this article, on the encoding of (unlabelled) noncrossing graphs whenever possible.

- Firstly, the ordered digraph bracketing is more difficult to interpret than square brackets that contain less information. To get a possibly more readable notation, the direction of the edges can be encoded using subscripted square brackets: plain square brackets would indicate inverted or undirected edges, but a specific orientation of the corresponding edges is indicated with subscripts as in “[_<[_<[_<[_<]\[_<[_<]\[_<]\[_<]\[_>]”.
- Secondly, since the different types of left and right brackets must match each other, bracketing of digraphs require more states in the finite-state approximation. The increased complexity is needed to keep track of the open brackets. This consideration in the encoding complexity may be addressed with one-sided labelling, e.g., by dropping the subscripts of the right square brackets:

$$[<[<[<[<[<[<]]]]]>,$$

the left square brackets:

$$[[[[]\[]\[]\[]\[]],$$

or, for example, the head side brackets:

$$[<[<[<[<[<[<]]]]]>.$$

The families of noncrossing trees and noncrossing rooted trees can be treated as restrictions of the sets of noncrossing graphs and digraphs. These families are not treated separately in this article (except the observation on the state complexity of the search space of projective trees on page 205). In fact, the present work on noncrossing graphs generalizes to all 50 subfamilies of noncrossing graphs described in Yli-Jyrä and Gómez-Rodríguez (2017), although the current discussion of these is restricted to the most general case.

3 THE PROBLEM OF UNBOUNDED DEPTH

Normal-looking natural language sentences may give rise to a surprisingly high complexity when measured in terms of the depth of nested brackets and overlapping edges.

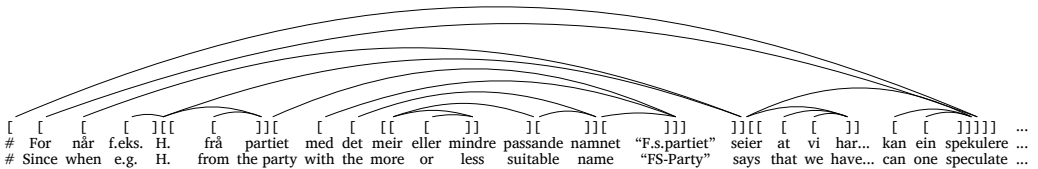


Figure 4: The underlying dependency tree of a Nynorsk (Norwegian) sentence

Figure 4 shows the parse or analysis of a sentence found in a treebank that follows the Universal Dependencies annotation scheme. The figure does not show all the details of the edge orientation and labels, but it reveals that the underlying graph of the parse is a noncrossing tree. The exceptional complexity of this ordered graph comes from its multiple levels of overlapping edges. These overlapping edges correspond to nested brackets. Due to the overlapping, the original encoding (Yli-Jyrä and Gómez-Rodríguez 2017) requires, in fact, up to 10 levels of square brackets. The sentence demonstrates that natural language sentences may involve many levels of overlapping dependency edges even though no clausal center embedding is clearly present.

Another observation from Figure 4 is that the current visualisation of overlapping edges is not very readable, and the corresponding brackets are stacked up to form almost meaningless sequences. It is, thus, obvious that this kind of bracketing requires many states in a finite-state approximation. In this section, our objective is to find a

new, simpler way to encode and draw diagrams of dependency analysis, and to reduce the state complexity of the encoding.

3.1 *Deep nesting outside dependency graphs*

Balanced bracketing has been used in many contexts that include but are not restricted to Generative Grammar, programming languages and document markup systems. In Generative Grammar, so-called *P-markers* have been used to describe phrase-structure trees. In the Lisp programming language (Teitelman 1978), a large number of parentheses are typically needed in lists that constitute the fundamental data structure of the programming language. The XML markup language and its predecessor, SGML (Goldfarb 1999), use brackets to indicate trees.

Deep nesting of brackets is a standard source of difficulties in applications of balanced bracketing. For example, it is well known that adding P-markers to context-free grammars changes their tail recursion into center-embedding (Langendoen 1975). The change converts regular, right- or left-linear context-free grammars into grammars that generate non-regular languages (except if the grammar is completely recursion-free and generates a finite language). Also, in Lisp programs and structured SGML and XML documents, brackets can be nested arbitrarily, and specialized markup editors are needed to keep track of the open brackets while editing them. Often the problem is in left- or right-linear recursion whose balanced bracketing is inconvenient due to the unbounded nesting.

To overcome the challenges of deep nesting in *strong* balanced bracketing, there are several approaches and techniques that are closely related to each other. The techniques make the bracketing unbalanced in a controlled and reversible ways. As to Generative Grammar, Chomsky (1963) already proposed omitting left or right brackets of P-markers in contexts where the original bracketing can be recovered without ambiguity. This idea of “semibrackets” was used to turn context-free grammars into grammars that produce *weak* bracketing, and to turn any non-self-embedding grammar as a whole into a finite-state transducer (Langendoen 1975; Krauwer and des Tombe 1981; Langendoen and Langsam 1984; Yli-Jyrä 2003c; Hulden and Silfverberg 2014). A complementary idea appears in InterLISP (Teitelman 1978, Section 2: Using Interlisp, page 2.4) where the pro-

grammar could close any unfinished round brackets with just one square bracket (]), a.k.a. “super-parentheses”: “a right square bracket automatically supplies enough right parentheses to match back to the last left square bracket (in the expression being read), or if none has appeared, to match the first left parentheses”. For example, this gives the following short-hand notations:

(6)	short-hand	=	expansion
	$A(B(C])$	=	$A(B(C))$
	$A[B(C(D)]E)$	=	$A(B(C(D)))E$

The role of “super-parentheses” (Teitelman 1978), or “superbrackets” in the sequel, is complementary to that of “semibrackets” that indicate the location of an initial or a final embedding (Chomsky 1963; Langendoen 1975): they close arbitrarily many one-sided brackets

It is not always easy to take advantage of weak bracketing that is based on superbrackets and semibrackets. The SGML standard (ISO 8879:1986) allowed the omission of redundant brackets, but this capacity of the standard made SGML-documents difficult to validate and parse, and contributed to the abandoning of the standard, in favour of XML. Elsewhere, a version of weak bracketing in the framework of Finite-State Intersection Grammar (Koskenniemi 1990) was used in an encoding scheme where an unbalanced clause-boundary marker “@/” indicated left or right recursion of clauses, leaving some unresolved ambiguity in the encoding on purpose: in (7), the sentence contains two levels of final clausal embedding, and in (8), there is an initial clause embedding. Thus, the markup used in the grammar framework did not indicate which clause is a subordinate clause and which is the main clause.

(7) It was a dog @/ that ate the mouse @/ that chased the cat.

(8) If the rats ate the cat @/ we were surprised.

Our present discussion does not try to advocate weak bracketing as a markup formalism for annotated data, because the benefits of weak bracketing for human-computer interaction are controversial. Instead, the focus of the research is on possible benefits for the state complexity when the subfamilies of graphs or the corresponding

search spaces are represented as string languages. The prior experiences with weak bracketing, in fact, suggest that its main advantage is related to the more natural treatment of left- and right-linear recursion and to the computational benefits of such a treatment.

3.2 Tail recursion in dependency bracketing

The idea of weak bracketing has gone almost unrecognised in the context of the dependency or edge bracketing of Yli-Jyrä and Gómez-Rodríguez (2017) since such bracketing is historically unrelated to P-markers and their recursion problems. In the dependency bracketing, right-linear embedding corresponds to local bracketing that does not introduce any recursive center-embedding. For example, tail recursion in (9) does increase the depth of dependency bracketing.

(9) It was a dog[that[[]]ate the mouse[that[[]]chased the cat.

Close to the earliest use of dependency bracketing is due to Greibach (1973) who used brackets to mark “phrase-subphrase” dependencies and to represent context-free languages via specifically bracketed Greibach Normal-Form (GNF) grammars. This bracketing maintains the regularity of the language although it contains balanced brackets: since a right-linear grammar (10) is already in a GNF, adding “phrase-subphrase” brackets converts it to another non-self-embedding context-free grammar (11) that generates a regular language. The same is not true for P-markers, which produce a grammar (12) that generates a non-regular language.

(10) $S \rightarrow aS_b \quad S_b \rightarrow bS_b \quad S_b \rightarrow \varepsilon$

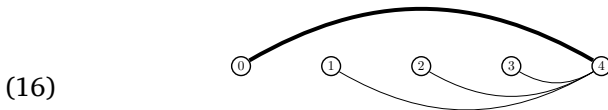
(11) $S \rightarrow a [_{S_b} S_b \quad S_b \rightarrow]_{S_b} b [_{S_b} S_b \quad S_b \rightarrow]_{S_b}$

(12) $S \rightarrow aS_b \quad S_b \rightarrow [_{S_b} bS_b]_{S_b} \quad S_b \rightarrow [_{S_b}]_{S_b}$

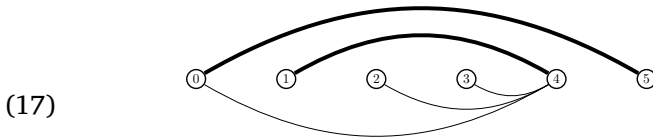
Bracketed “phrase-subphrase” dependencies have been rediscovered in projective dependency parsing by Oflazer (2003) and in nonprojective dependency parsing by Yli-Jyrä (2003b). The bracketing in projective dependency parsing has been developed further to obtain a Chomsky-Schützenberger representation for the string set and the set of structures generated by a projective dependency grammar (Yli-Jyrä 2005a) and a Link Grammar (Ginter *et al.* 2006), to obtain a cubic-time projective dependency parsing algorithm (Yli-Jyrä 2012), and

(15) $\llbracket \circ [\circ [\circ [\circ]]] \rrbracket$

The weak bracketing separates edges into three categories: The *superbracketed edges*, the *left (inner) siblings* of superbracketed edges, and the *right (inner) siblings* of the superbracketed edge. In the following, we use this classification to reduce the visual clutter of dependency diagrams: the inner siblings of the superbracketed edges are drawn below the line of vertices:

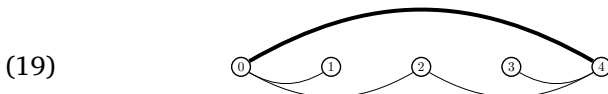


The classification of edges is based barely on the graph structure and is, therefore, not dependent on processing order. However, the category of an edge is not a local property: the category of an edge alternates between a superbracketed edge and a sibling edge. Such alternation starts from the outermost edge and proceeds transitively towards inner edges:



(18) $\llbracket \circ \llbracket \circ [\circ [\circ]] \circ \rrbracket \rrbracket$

The technique extends to situations where one vertex is connected to both ends of the outermost edge with sibling edges:



(20) $\llbracket \circ] \circ] [\circ [\circ] \rrbracket$

Figure 5 shows how the improved encoding is applied to a real dependency tree. The obtained graphical representation is immediately more readable in a very systematic way.

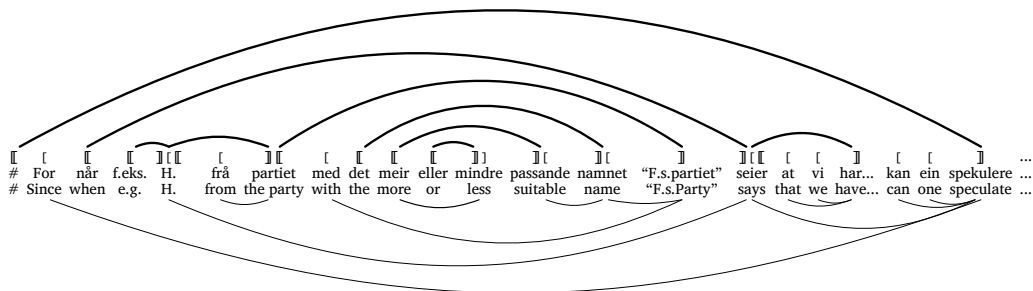


Figure 5: The application of the weak bracketing to the Nynorsk parse tree

Lemma 4.1. *There is a context-free grammar that generates the relation between the strong and the weak bracketing for noncrossing graphs.*

Proof. In the grammar of Lemma 2.6, we can distinguish three kinds of occurrences of the nonterminal Q : the initial Q_I , the central Q_C and the final Q_F . Each of these will be bracketed differently by the following grammar whose terminal alphabet is a pair alphabet $\Sigma = \{(\{, \},), [:\varepsilon,]:\varepsilon, [:[,]:], [:[,]:] \}$. The pair symbols in this alphabet are constructed from the empty string ε and the input and output symbols, and from the colon that separates them.

$$(21) \quad S \rightarrow (Q_C \mid \{:\varepsilon\})^*$$

$$(22) \quad Q_I \rightarrow [:\varepsilon S']:]$$

$$(23) \quad Q_C \rightarrow [:[S']:]$$

$$(24) \quad Q_F \rightarrow [:[S']]:\varepsilon$$

$$(25) \quad S' \rightarrow (Q_I \mid \{:\varepsilon\}) S'' \mid \{:\varepsilon\}$$

$$(26) \quad S'' \rightarrow (Q_C \mid \{:\varepsilon\})^* (Q_F \mid \{:\varepsilon\})$$

Each symbol in the terminal alphabet of this grammar is a pair $a:b$ where $a \in \Sigma_1^*$ is the input factor and $b \in \Sigma_2^*$ is the output factor. The alphabet of the input strings is $\Sigma_1 = \{(\{, \}, [,])\}$ and the alphabet of the output strings is $\Sigma_2 = \{(\{, \}, [,], [:[,]:], [:[,]:] \}$. The factor ε , in particular, is the empty string. Let $w = (a_1:b_1) \dots (a_n:b_n) \in \Sigma^*$ be a string generated by the grammar. The concatenation of the input factors a_1, \dots, a_n constitutes the input string $a_1 \dots a_n$ and the concatenation of

the output factors b_1, \dots, b_n constitutes the output string $b_1 \dots b_n$. Together, the input and the output constitute a pair $(a_1 \dots a_n, b_1 \dots b_n) \in \Sigma_1^* \times \Sigma_2^*$. In this way, the grammar defines a relation between the input strings Σ_1^* and the output strings Σ_2^* . For example, the grammar relates the input “[[\emptyset] \emptyset]” to the output “[[\emptyset] \emptyset]”. \square

In contrast to the language of strong bracketing for noncrossing graphs, $L_{\text{NXGRAPH},S}$, we denote the language of weak bracketing for noncrossing graphs with $L_{\text{NXGRAPH},W}$.

Lemma 4.2. *There is an extended context-free grammar that generates the language of weak bracketing ($L_{\text{NXGRAPH},W}$) with derivation steps that correspond 1–1 to the pairs of superbrackets (except the topmost step).*

Proof. The language $L_{\text{NXGRAPH},W}$ of the encoded noncrossing graphs is generated by an extended context-free grammar:

$$\begin{aligned}
 (27) \quad & S \rightarrow (\emptyset \mid Q)^* \\
 & Q \rightarrow \llbracket S_{\downarrow} E S_{\uparrow} \rrbracket \mid \llbracket S_{\downarrow} \rrbracket \\
 & E_{\downarrow} \rightarrow \varepsilon \mid] \quad E_{\uparrow} \rightarrow \varepsilon \mid [\quad E \rightarrow [\mid] [\\
 (28) \quad & S_{\downarrow} \rightarrow (\emptyset (E_{\downarrow} Q)^* E_{\downarrow})^* \emptyset \\
 (29) \quad & S_{\uparrow} \rightarrow (\emptyset (E_{\uparrow} Q)^* E_{\uparrow})^* \emptyset (E_{\uparrow} Q)^* \\
 (30) \quad & S_{\uparrow} \rightarrow (Q E_{\uparrow})^* \emptyset (E_{\uparrow} (Q E_{\uparrow})^* \emptyset)^*
 \end{aligned}$$

To expand the right-hand side of the production schema (27), we substitute the nonterminal symbols $S_{\downarrow}, S_{\uparrow}, S_{\uparrow}, E_{\downarrow}, E_{\uparrow}$, and E with the right-hand sides of the corresponding production schemas. One application of the expanded production schema then corresponds to exactly one level of superbrackets. \square

Lemma 4.3. *There is a finite-state transducer whose transitive closure maps the start symbol S to the language $L_{\text{NXGRAPH},W}$.*

Proof. Figure 6 shows a transducer that represents the grammar of Lemma 4.2. In this transducer, the transitions that copy the input factor to the output are indicated with simple labels that do not contain a colon. Starting from the input string S , the transitive closure of this transducer generates exactly the language of the grammar when output strings of the closure are restricted to the terminal strings. \square

How to embed trees in a regular language

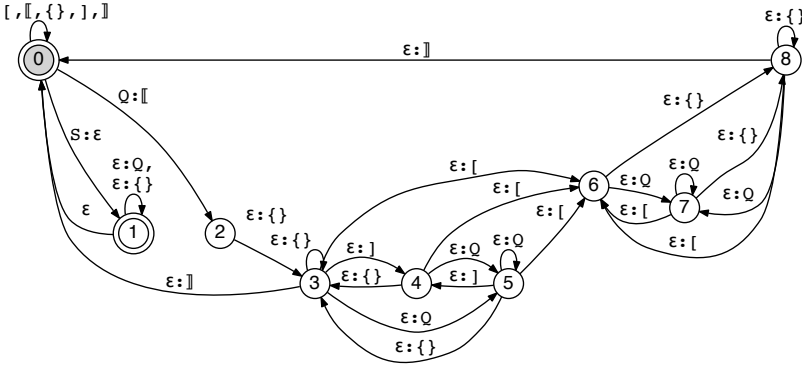


Figure 6:
A finite-state transducer that represents the grammar of Lemma 4.2

Lemma 4.4. *There is a bijective encoding morphism from the monoid of noncrossing graphs to the language $L_{NXGRAPH,W}$.*

Proof. The left column of Figure 7 contains the algorithm enc_W that maps a noncrossing graph (n, E) to an encoding that is based on weak bracketing. This algorithm works by iterating the vertex index i over the ordered vertices $[1, \dots, n]$.

- On lines 7–10, the algorithm adds a closing superbracket “ \rfloor ” if it is time to remove the corresponding edge from the stack (stk).
- On lines 11–12, the algorithm adds a semibracket “ \lrcorner ” if vertex i ends a shorter sibling of the topmost edge in the stack.

def $enc_W((n,E) \in NXGRAPH)$:

```

1 stk = [(0,n+1)]
2 str = ""
3 for i in [1,2,...,n]:
4   if i > 1:
5     str += "0";
6     (l,r) = stk.top()
7     if i == r:
8       str += "\rfloor"
9       stk.pop()
10    (l,r) = stk.top()
11    if {l,i} in E:
12      str += "\lrcorner"
13    if {i,r} in E:
14      str += "\l["
15    for j in [r-1,r-2,...,i+1]:
16      if {i,j} in E:
17        str += "\l["
18        stk.push( (i,j) )
19      break
20 return str

```

def $dec_W(str \in L_{NXGRAPH,W})$:

```

1 n = 1
2 E = {}
3 stk = []
4 for a in str:
5   if a == "\rfloor":
6     stk.push([n])
7   elif a == "i":
8     n += 1
9   elif a == "\lrcorner":
10    E += { {stk.top[0],n} }
11  elif a == "\l[":
12    stk.top() = stk.top() + [n]
13  elif a == "\l[":
14    for j in stk.pop():
15      E += { {j,n} }
16 return (n,E)

```

Figure 7:
Functions that encode/decode noncrossing graphs using weak bracketing

- On lines 13–14, the algorithm adds a semibracket “[” if vertex i starts a shorter sibling of the topmost edge in the stack.
- On lines 15–19, the algorithm adds an opening superbracket “⌈” if vertex i starts a superbracketed edge that is not a shorter sibling of the topmost edge in the stack.
- Between iterations, on lines 4–5, the algorithm adds a vertex boundary “○”.

It is easy to see that the algorithm enc_W always terminates, and it implements a mapping from all noncrossing graphs to strings in $L_{\text{NXGRAPH},W}$. It is also easy to see that the algorithm respects concatenation: $\text{enc}_W((n, E_1)) \cdot \text{enc}_W((m, E_2)) = \text{enc}_W((n, E_1) \cdot (m, E_2))$.

Conversely, the right column of Figure 7 contains the algorithm dec_W that maps strings in $L_{\text{NXGRAPH},W}$ to noncrossing graphs. The algorithm reads its input string from left to right.

- On lines 5–6, when the opening superbracket “⌈” is read, the algorithm pushes to the stack a list that just contains the current vertex n .
- On lines 7–8, when the left curly bracket “{” is read, the algorithm starts a new vertex by incrementing n . The right curly bracket “}” is just ignored in the well-formed input.
- On lines 9–10, when the right semibracket “]” is read, the algorithm looks for the first vertex number in the topmost list in the stack and adds an edge between it and the current vertex.
- On lines 11–12, when the left semibracket “[” is read, the algorithm adds the current vertex to the topmost list in the stack.
- On lines 13–15, when the closing superbracket “⌋” is read, the algorithm pops the topmost list from the stack and adds an edge between the current vertex and the vertices in this list.

It is easy to verify that the decoding algorithm dec_W runs in linear time to the length of the input string. □

Lemma 4.5. *There is an algorithm to compute the bracketing depth of graphs in weak bracketing.*

Proof. The algorithm for computing the weak bracketing depth of a graph (n, E) is given in Figure 8. □

def $\text{depth}_w((n,E))$:

```

1 d = maxd = 0
2 for c in encw(n,E):
3     if c == "[":
4         d += 1
5         maxd = max(maxd,d)
6     elif c == "]":
7         d -= 1
8 return maxd

```

Figure 8:

An algorithm for measuring the depth of the balanced brackets in weak edge bracketing

For example, the algorithm depth_w returns value 0 for graph $(2, \{\})$ and 1 for graph $(3, \{\{1,3\}, \{2,3\}\})$ because these graphs encode as “ \emptyset ” and “[$\emptyset[\emptyset]$ ”. For $(4, \{\{1,4\}, \{2,3\}\})$, the algorithm returns a depth of 2 because its code string “[$\emptyset[\emptyset]\emptyset$ ” contains two levels of superbrackets.

Lemma 4.6. *There is a conventional unambiguous context-free grammar for the streamlined encoding of noncrossing graphs, $L_{\text{NXGRAPH},W}$.*

Proof. The grammar of Lemma 4.2 is turned into a conventional context-free grammar

$$\begin{aligned}
 (31) \quad & S \rightarrow \emptyset S \mid Q S \mid \varepsilon \\
 & Q \rightarrow \llbracket S_j E S_l \rrbracket \mid \llbracket S_l \rrbracket \\
 (32) \quad & E_j \rightarrow \varepsilon \mid] \quad E_l \rightarrow \varepsilon \mid [\quad E \rightarrow [\mid] [\\
 (33) \quad & S_l \rightarrow \emptyset \mid \emptyset T ; T \rightarrow E_j \emptyset T \mid E_j \emptyset \mid E_j Q T \\
 (34) \quad & S_j \rightarrow \emptyset \mid \emptyset U ; U \rightarrow E_j \emptyset U \mid E_j \emptyset \mid E_j Q U \mid E_j Q \\
 & S_l \rightarrow W ; W \rightarrow Q E_l W \mid \emptyset E_l W \mid \emptyset
 \end{aligned}$$

To obtain this grammar, we take each right-hand-side that describes a regular language over an alphabet $\Sigma \cup V$ and replace it with a context-free subgrammar that generates this regular language. In this way, the production schemas (28)–(30) expand, respectively, to the subgrammars (32)–(34). It is now easy to verify that the resulting grammar, as a whole, is unambiguous. Especially, the production schema (31) is unambiguous, since S_l does not generate $[$ outside an embedded Q while E always generates $[$. \square

5 EVALUATION OF THE ENCODING SCHEMES

5.1 Variants of the two encoding schemes

Until now, we have given ECFG grammars for two different encodings for graphs. When these grammars are extended to brackets that indicate the direction of an edge on both sides, we obtain grammars for two different encodings for digraphs. For example, in the ECFG grammar of Lemma 2.6, the grammar is extended with two additional rules

$$(35) \quad Q \rightarrow /S'> \qquad Q \rightarrow <S'\backslash$$

We call the encoding of Yli-Jyrä and Gómez-Rodríguez (2017) the *strong bracketing (S)* and the currently proposed encoding *the weak bracketing (W)*. In addition to these, we identify three optimisations that are available to both strong and weak bracketing:

1. The first optimisation (“ \circ ”) simplifies the pair of curly brackets by replacing it with an atomic symbol: a bullet dot “ \bullet ”:

$$(36) \quad \{ \} \rightarrow \bullet$$

2. The second optimisation (“1”) is to eliminate the difference between the symbols used as a left bracket:

$$(37) \quad \llcorner \rightarrow \llbracket \qquad \llbracket / \rightarrow \llbracket \qquad \llcorner \leftrightarrow [\qquad / \rightarrow [$$

3. The third optimisation (“ $[\circ]$ ”) introduces new vertex boundaries “ \bullet_{\square} ”, “ $\bullet_{<}$ ”, and “ $\bullet_{/>}$ ” in order to compress the edges between adjacent vertices as follows:

$$(38) \quad [\bullet] \rightarrow \bullet_{\square} \qquad [\bullet] \backslash \rightarrow \bullet_{<} \qquad [/\bullet] \rightarrow \bullet_{/>}$$

$$(39) \quad \llbracket \bullet \rrbracket \rightarrow \bullet_{\square} \qquad \llbracket \bullet \rrbracket \backslash \rightarrow \bullet_{<} \qquad \llbracket / \bullet \rrbracket \rightarrow \bullet_{/>}$$

$$(40) \quad \llbracket \alpha \bullet \rrbracket \rightarrow \llbracket \alpha \bullet_{\square} \rrbracket \qquad \llbracket \alpha \bullet \rrbracket \backslash \rightarrow \llbracket \alpha \bullet_{<} \rrbracket \qquad \llbracket \alpha \bullet \rrbracket \gg \rightarrow \llbracket \alpha \bullet_{/>} \rrbracket$$

$$(41) \quad [\bullet]_{\beta} \rightarrow \bullet_{\square\beta} \qquad [\bullet]_{\beta} \backslash \rightarrow \bullet_{<\beta} \qquad [/\bullet]_{\beta} \rightarrow \bullet_{/>\beta}$$

where $\alpha \in \{\varepsilon, <, /\}$ and $\beta \in \{\varepsilon, >, \backslash\}$.

This optimisation implies the “ \circ ”-optimisation.

These optimisations are meant to optimise the state complexity of finite-state automata, but they also come with some trade-offs. The main disadvantage of the “1”-optimisation is that the information about the direction is no longer locally present at the bracket where one might need it. The “[\emptyset]”-optimisation suffers from an increased alphabet size.

The Cartesian product of two encoding schemes and three optimisations gives us twelve different bracketing schemes:

$$\{S, W\} \times \{\varepsilon, “1”\} \times \{\varepsilon, “\emptyset”, “[\emptyset]”\}.$$

We will not compare all of these schemes in detail, but we will include some of them in experiments to get an idea of their relative efficiency. The corresponding bracket alphabets for digraph encoding schemes are summarised in Table 1.

	Strong bracketing (Lemma 2.6)		Weak bracketing (Lemma 4.2)
S	[,<,/,>,\,>,<,>]	W	[[,<,[/,>],\,>,[,<,/,>,\,>,<,>]]
S\emptyset	[,<,/,>,\,>,<,>,\bullet]	W\emptyset	[[,<,[/,>],\,>,[,<,/,>,\,>,<,>,\bullet]]
S[\emptyset]	[,<,/,>,\,>,<,>,\bullet,\bullet,\bullet, □,<\ />]	W[\emptyset]	[[,<,[/,>],\,>,[,<,/,>,\,>,<,>,\bullet,\bullet,\bullet, □,<\ />]]
S1	[,>,\,>,<,>]	W1	[[,>],\,>,[,>,[,<,/,>,\,>,<,>]]
S1\emptyset	[,>,\,>,<,>,\bullet]	W1\emptyset	[[,>],\,>,[,>,[,<,/,>,\,>,<,>,\bullet]]
S1[\emptyset]	[,>,\,>,<,>,\bullet,\bullet,\bullet, □,<\ />]	W1[\emptyset]	[[,>],\,>,[,>,[,<,/,>,\,>,<,>,\bullet,\bullet,\bullet, □,<\ />]]

Table 1:
The alphabets
of different
encoding
schemes and
their variants

5.2 State complexity of finite search spaces

Table 2 reports the size and the state complexity of the search spaces as the function of the number of vertices, n . The first two columns indicate, for example, that there are 1,792 noncrossing 4-vertex digraphs. The deterministic state complexity of this “4-vertex” search space is 490, 334, 30, 106, 19, or 10 states, depending on the encoding (S or W) and the additional optimisations (“ \emptyset ”, “1 \emptyset ”, “1[\emptyset]”).

We learn from Table 2, firstly, that the state complexity of the search space grows exponentially with the number of vertices in the digraphs, regardless of the encoding scheme. The context-free representation of Yli-Jyrä and Gómez-Rodríguez (2017) is immune to the state complexity concerns, but a straightforward depth-bounded finite-state approximation of the S scheme explodes immediately.

Table 2:
The DFA state
complexity
of finite search
spaces

n	Digraphs	S	S ₀	S1 ₀	W	W1 ₀	W1[₀]
1	1	1	1	1	1	1	1
2	4	12	8	4	12	4	2
3	64	80	54	12	26	8	6
4	1,792	490	334	30	106	19	10
5	62,464	2,952	2,018	68	207	33	24
6	2,437,120	27,040	12,126	146	704	61	38
7	101,859,328	106,372	72,778	304	1,327	95	72

The rapid growth of the state complexity is explained by the fact that larger digraphs involve more open brackets and both encodings (S and W) keep a record of the type of the open brackets as well as of the intermediate constituent structure of each level, corresponding to the right-hand-side of the production schemas (4) and (27) that expand the nonterminal symbol Q in each grammar. In addition, the state space must keep track of the total number of vertices produced so far. Overall, the state complexity of the strong bracketing compares poorly against the superset approximations of context-free phrase structure grammars (Nederhof 2000).

Secondly, we learn from Table 2 that weak bracketing gives a clear advantage over the strong bracketing. The state complexity of the original encoding (S) blows up after 3 vertices and reaches 106,372 DFA states when there are 7 vertices. The state complexity of the weak bracketing scheme is substantially lower than the strong bracketing (S). With 7 vertices, W requires only 1,327 states, which is an 80-times improvement over the strong bracketing scheme. Moreover, it seems that S simply grows faster and faster in comparison to W when n increases. The lower growth rate of the complexity of W is explained by the fact that W does not open more than one superbracket “[” per every two vertices whereas S opens one pair of brackets per edge.

Similar results are obtained for subfamilies of noncrossing graphs. Figure 9 compares the state complexity of the search spaces of three different families of noncrossing graphs as a function of the number of vertices or words in the sentence. The figure indicates that the advantage of weak bracketing (W) in contrast to strong bracketing (S) is relatively robust across different families of noncrossing graphs: digraphs, projective trees and graphs are all more compactly presented with weak bracketing than with strong bracketing.

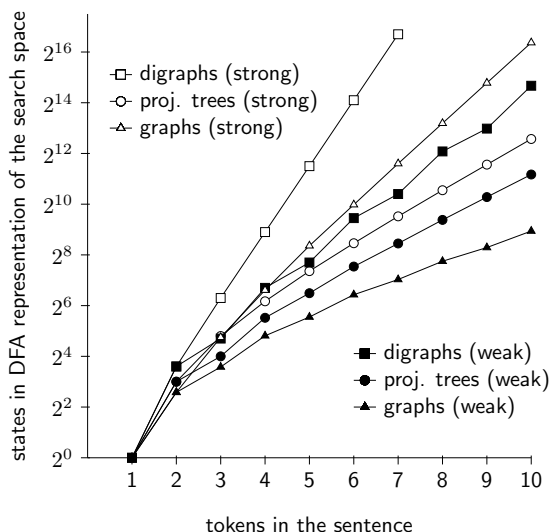


Figure 9: Weak bracketing brings exponential savings in the size of the DFA representing the search space of a sentence when the search space consists of digraphs, projective trees, or graphs

In sum, the exponential growth of the state space seems to be unavoidable for the set of noncrossing graphs, which reflects the fact that bijective encoding is more difficult than an a superset approximation of the search space. But the state space complexity improves dramatically with our new techniques: weak bracketing, search space restrictions to noncrossing subfamilies, and optimisations in the encoding scheme. The columns for S_{\square} , $S1_{\square}$, $W1_{\square}$, and $W1[\square]$ in Table 2, on page 204, show the improved state complexity of some combined optimisations. These indicate that the state complexity drops, quite dramatically, from S and S_{\square} to $S1_{\square}$ (from 106,372 and 72,778 to 304 states for $n=7$) and from W to $W1_{\square}$ (from 1,327 to 95 states). A further improvement is given by the “[\square]”-optimisation (from 95 to 72 states for 7 vertices).

Along with the weak bracketing, another big improvement in the state complexity of unweighted search space is due to the “1”-optimisation: thanks to these two improvements, we are able to build complete search spaces for relatively large ordered graphs with at least 30-vertices. Table 3 gives an idea of the implications of these improvements. In short, they can now be expressed as follows:

- The complete search space of all 10-vertex noncrossing graphs and digraphs is represented by a deterministic automaton that has 254 states.

Table 3:
State complexity
of larger search
spaces

n	Encoded graphs	$W1$	$W1_{\square}$	$W1[\square]$
10	21,292,032	492	363	254
\vdots	\vdots	\vdots	\vdots	\vdots
19	29,312,424,612,462,592	12,395	9,347	7,569
20	314,739,971,287,154 688	18,276	13,693	10,114
21	3,393,951,437,605,044,224	24,925	18,807	15,228
\vdots	\vdots	\vdots	\vdots	\vdots
30	$\approx 7.681 \cdot 10^{27}$	589,598	442,177	327,494

- The complete search space of all 30-vertex noncrossing graphs and digraphs is represented by a deterministic automaton that has 327,494 states.

5.3 State complexity of bounded-depth grammars

Bounding the bracketing depth in the encoding schemes turns the respective grammars into subset approximations. Each approximation is equivalent to a cyclic finite-state automaton that recognises a regular language. We now turn our focus to the state complexity of such bounded-depth grammars and their languages.

Table 4 illustrates the relative parsimony of $W1_{\square}$ against $S1_{\square}$ when the depth of balanced bracketing, d , grows. The table shows that $W1_{\square}$ captures the complete 7-vertex search space of 101,859,328 digraphs already with 3 levels of balanced brackets, while $S1_{\square}$ needs 6 levels of brackets to capture the same search space. When we increase

Table 4:
The state
complexity and
the largest
contained
complete search
space of
depth-bounded
grammars

d	n -Digraphs	n	$S1_{\square}$	n -Digraphs	n	$W1_{\square}$
0		1	1		1	1
1		4	2		4	6
2		64	3	62,464	5	18
3	1,792	4	18	101,859,328	7	42
4	62,464	5	38	201,889,939,456	9	90
5	2,437,120	6	78	443,939,433,742,336	11	186
6	101,859,328	7	158	1,041,383,605,688,860,672	13	378
7	4,459,528,192	8	318	$\approx 2 \cdot 10^{21}$	15	762

d	S_d	$S\circ_d$	$S1\circ_d$	$W\circ_d$	$W1_d$	$W1\circ_d$	$W1[\circ]_d$
0	2	1	1	1	2	1	1
1	11	7	3	2	9	6	7
$d + 1$	$6s_d+2$	$6s_d+4$	$2s_d+2$	$6s_d+17$	$2s_d+7$	$2s_d+6$	$2s_d+6$
2	68	46	8	23	25	18	20
3	410	280	18	155	57	42	46
4	2,462	1,684	38	947	121	90	98
5	14,774	10,108	78	5,699	249	186	202
6	88,646	60,652	158	34,211	505	378	410
7	531,878	363,916	318	205,283	1,017	762	826

Table 5:
The DFA state complexity of the bounded-depth grammar

both these bounds with one more level, the 8-vertex search space in $S1\circ$ is only 44 times larger, whereas the search space captured by $W1\circ$ has 9-vertex graphs and grows 1,998 times larger. Thus, the growth of the weakly bracketed search space is roughly quadratic to growth of the strong bracketing. This trend becomes even more striking when the bracketing gets deeper.

Table 5 shows the state complexity of the bounded-depth grammar as the function of the bracketing depth (d) and the used encoding scheme. The first impression is that strong bracketing (S) and weak bracketing (W) give rise to very similar state complexity of the bounded-depth grammars: while $W\circ$ is more compact than $S\circ$ with its 34,211 states against 60,652 states, $S1\circ$ initially looks more compact than $W1\circ$ with its 158 states against 378 states.

We already learned from Table 4 that two levels in S compare roughly to one level in W. Besides this, the depth of the latter is not sensitive to unbounded branching. Therefore, the complexity of the bounded grammar for weak bracketing is more interesting than the complexity of the bounded grammar for strong bracketing.

We now have an idea about the state complexity of a deterministic finite automaton that recognizes languages of different bounded grammars for strong and weak bracketing. As we from now on talk about the state complexity of bounded grammars, we will focus on the weak bracketing (W) and on its one-sided variant (W1). Its “ \circ ”-optimisation is even more succinct, but the “[\circ]”-optimisation appears to be harmful to the state complexity of the bounded grammars.

5.4 Formal coverage of bounded grammars

When the bound d for the depth of bracketing is fixed, the depth of bracketing does not grow arbitrarily when the length of input sentence changes. A grammar with a fixed bound will be applied to short sentences as well as to long sentences. This raises the question of what happens with the coverage and the state complexity of the restricted search space when the sentence is exceptionally long.

When we process a growing number of vertices, the first consequence of using a bounded-depth grammar is that there will be a bound k for the number of vertices beyond which the bounded grammar ceases to capture the complete search space of the noncrossing graphs. Beyond this point, the search space will be limited by the bounded depth. The state complexity of the limited search space will then grow linearly with the number of vertices when the number of vertices continues to grow enough.

For example, let us restrict the depth of bracketing to 6 levels, which gives us a bounded grammar with 505 states. The largest complete search space captured by this grammar consists of 13-vertex graphs (or digraphs, whose state complexity is the same under the **W1** encoding scheme).

If we now extract 21-vertex graphs from the same grammar, we will get only a proper subset of all 21-vertex graphs because graphs that require more than 6 levels of brackets are missing. Capturing the complete search space for 21-vertex graphs requires 10 levels of bracketing. Table 6 shows in detail what happens to the search space of 21-vertex graphs when we decrease the maximum depth of bracket-

Table 6:
98.96% coverage
of 21-vertex
graphs requires
only 6 bracket
levels and only
1/8th of the full
coverage states

d	$W1_d$	$W1_d \cap W1_{n=21}$		The number/% of 21-vertex graphs	
3	57	929	3.7%	872,294,071,717,330,944	25.70143%
4	121	1,765	7.1%	2,313,578,416,163,258,368	68.16769%
5	249	3,181	12.7%	3,131,655,209,939,369,984	92.27166%
6	505	5,501	22.1%	3,358,682,892,406,358,016	98.96084%
7	1,017	9,117	36.6%	3,391,549,974,785,294,336	99.92924%
8	2,041	14,301	57.4%	3,393,882,839,790,387,200	99.99798%
9	4,089	20,573	82.5%	3,393,950,914,747,301,888	99.99998%
10	8,185	24,925	100.0%	3,393,951,437,605,044,224	100.00000%

How to embed trees in a regular language

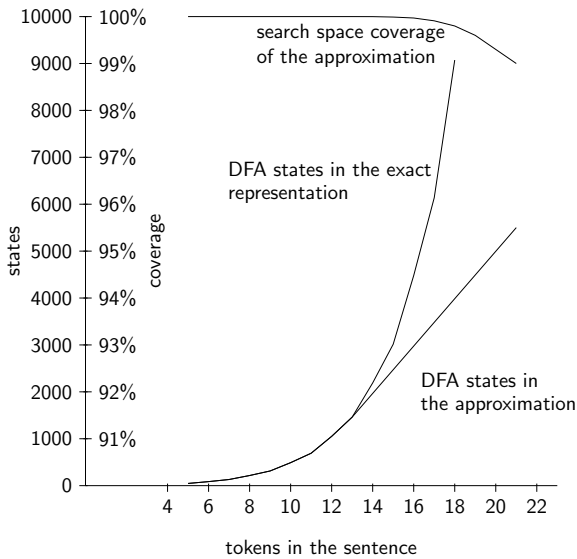


Figure 10:
The state complexity of the limited search space $W1_{d=6}$ grows only linearly with sentence length after 13 vertices, while the exact search space representation explodes quickly. At the same time, the coverage of the limited search space remains close to 100% for many more vertices

ing. Quite surprisingly, the 6-bounded grammar still contains 98.96% of all 21-vertex graphs!

The linear growth of the state complexity of the limited search space gives a huge advantage over the standard situation where the complete search space requires an exponentially growing number of states as a function of the number of vertices. This point is illustrated by Figure 10. This is also illustrated by Table 6, which shows that the state complexity of this limited search space is only 1/5 of the state complexity of the complete search space and the state complexity of the corresponding bounded grammar is only 6.2% (505 states) of the state complexity of the 10-bounded grammar (8,185 states). Thus, limiting the search space of long sentences by depth is an effective way to reduce the number of states in the grammar and search space representations. This reduction is necessary in practice because there is no fixed limit for the length of natural language sentences: as a challenge for parser developers, the Universal Dependencies treebanks contain a few really long sentences that have more than 500 tokens.

Table 7 describes the state complexity of extremely large limited search spaces that contain 8-, 16-, 32-, ..., and 512-vertex noncrossing graphs bounded to 7 levels of brackets. The 7-bounded grammar $W1_{d=7}$ can be represented, according to Table 5, with 762 states. For

Table 7:
The linear growth of the state complexity
of the limited search space
with at most 7 levels of balanced brackets

n	Upper bound $((2n - 1) \cdot 762)$	Actual states
8	11,430	157
16	23,622	3,029
32	48,006	15,221
64	96,774	39,605
128	194,310	88,373
256	389,382	185,909
512	779,526	380,981

each integer $n \geq 1$, there is a finite-state automaton whose language $W1_n$ constrains the graph size to n vertices. The state complexity of $W1_n$ is exactly $(2n - 1)$ states. By multiplying the state complexities of the depth-bounded grammar $W1_{d=7}$ and the graph size constraint $W1_n$, we obtain an upper bound for the state complexity of the limited search space of n -vertex noncrossing graphs with a maximum depth of 7 brackets. However, the actual state complexity of the intersection of the two languages is slightly smaller: instead of 779,526 states, we will need only 380,981 states to represent the limited search space of 512-vertex graphs. Thus, with seven levels of brackets, this can be summarised as follows:

- The bounded grammar of noncrossing graphs requires at most 762 DFA states.
- The largest complete search space contained in the bounded grammar consists of noncrossing graphs that have 15 vertices.
- The limited search space for 512-token sentences requires 380,981 DFA states with the $W1$ encoding scheme.

5.5 *Empirical coverage of bounded grammars*

An experiment was carried out to apply the bracketing depth measure to the noncrossing trees in the Universal Dependencies (UD) treebanks (Nivre *et al.* 2019). In order to verify that the treebank size does not significantly affect the results, we carried out the same experiment on two different releases of UD treebanks. In the v2.4 release, there are 146 treebanks and 83 languages, while in the v2.0 release, there are 70 treebanks and 50 languages and about half the number of the trees. In

the experiment, we focused on the primary dependencies that define rooted trees. Of the total 1,234,587 rooted trees in the v2.4 data set, 90.5% (1,117,332) are noncrossing, and typically nonprojective. We encoded these trees with the W encoding scheme and computed the depth of the bracketing with the algorithm depth_W shown previously in Figure 8.

For the purpose of succinct reporting of the results, the number of languages was reduced by grouping some closely related languages into larger buckets. For example, our “Scandinavian” is a group of languages containing Bokmål, Danish, Nynorsk, and Swedish, and Ancient Greek and Old Russian were grouped with their modern variants. However, we did not group Latin with Italian. Although such grouping might remove the sharpest distinctions between languages, it became as a surprise that the depth-based cross-lingual complexity differences decayed so quickly when the depth increased beyond 3 levels. Thus, the row “noncrossing” describes surprisingly well a language independent tendency where the depth of bracketing for noncrossing trees is mostly very low.

The percentage of noncrossing trees and the coverage of the measured complexity levels are shown in Table 8. The first two numerical columns show the percentage and the absolute number of noncrossing trees among all trees for the corresponding language subset. In other columns, the coverage of depth-bounded bracketing is computed against the number of noncrossing trees for the corresponding language subset. The row with the label “noncrossing” corresponds to the set of all languages. Its first two columns tell the percentage and the absolute number of noncrossing trees in the whole UD data set. The mixed data set considered all trees equal in weight regardless of the size of the tree and the size of the containing treebank.

The results in the table are illuminating in two ways. Firstly, the results indicate that a bounded search space with 7 levels of superbrackets is capable of covering 99.999% of the noncrossing trees in the v2.0 and v2.4 versions of the UD dataset. Since 7 levels require only 762 DFA states in “ $W10_d$ ”-encoding, this result supports our hypothesis according to which *the practically occurring noncrossing dependency digraphs can be embedded in a subset approximation that has a very compact finite-state representation*. Secondly, we observe that the bounded space with 4 levels of superbrackets and 90 states

Table 8: The coverage of depth-bounded W_d grammars measured against UD v2.0 and v2.4 trees that are noncrossing and do not contain *epsilon* nodes

Language	Noncrossing		Depth 1	2	3	4	5	6	7	8
all v2.0 trees	100.0%	630,518								
noncrossing	86.7 %	546,492	16.1%	57.7%	91.6%	99.2%	99.95%	99.999%	100.000%	100.000%
all v2.4 trees	100.0%	1,234,587								
noncrossing	90.5%	1,117,332	16.6%	57.8%	91.8%	99.2%	99.95%	99.998%	100.000%	100.000%
Arabic	97.2%	27,608	5.0%	21.1%	64.7%	94.5%	99.70%	99.986%	100.000%	100.000%
Catalan	95.5%	15,931	1.7%	23.0%	75.7%	97.2%	99.78%	100.000%	100.000%	100.000%
Czech	88.3%	112,595	15.0%	52.4%	91.4%	99.2%	99.95%	99.999%	100.000%	100.000%
German	92.9%	178,229	10.9%	52.5%	91.9%	99.4%	99.97%	99.998%	100.000%	100.000%
English	94.7%	32,811	16.5%	57.1%	93.5%	99.5%	99.99%	100.000%	100.000%	100.000%
Spanish	94.5%	32,789	2.3%	29.2%	82.1%	98.2%	99.90%	100.000%	100.000%	100.000%
Finnish	93.5%	32,589	39.3%	80.4%	97.3%	99.7%	99.98%	99.997%	100.000%	100.000%
French	91.6%	57,459	20.0%	59.1%	92.2%	99.3%	99.97%	99.995%	100.000%	100.000%
Greek	54.8%	18,364	31.4%	81.7%	97.7%	99.8%	99.99%	100.000%	100.000%	100.000%
Hebrew	97.0%	6,032	2.3%	28.8%	82.6%	98.9%	99.95%	100.000%	100.000%	100.000%
Hindi	86.2%	16,843	3.4%	50.6%	88.5%	98.7%	99.93%	99.988%	100.000%	100.000%
Croatian	91.0%	8,205	3.8%	39.3%	89.3%	99.3%	99.94%	100.000%	100.000%	100.000%
Hungarian	72.9%	1,312	4.0%	32.9%	79.1%	96.5%	99.70%	99.924%	100.000%	100.000%
Italian	98.1%	33,398	5.7%	50.8%	90.9%	98.9%	99.91%	99.997%	100.000%	100.000%
Japanese	99.8%	66,950	25.5%	75.3%	96.8%	99.8%	99.99%	100.000%	100.000%	100.000%
Korean	88.6%	30,758	17.5%	70.5%	96.4%	99.8%	99.99%	100.000%	100.000%	100.000%
Latin	67.2%	28,002	31.5%	74.3%	95.7%	99.6%	99.98%	100.000%	100.000%	100.000%
Latvian	90.4%	11,772	13.5%	52.1%	90.7%	98.9%	99.92%	100.000%	100.000%	100.000%
Dutch	88.3%	18,481	23.8%	68.7%	95.3%	99.4%	99.95%	99.989%	100.000%	100.000%
Polish	96.1%	38,882	16.8%	71.4%	96.2%	99.7%	99.98%	100.000%	100.000%	100.000%
Portuguese	87.1%	19,553	6.2%	42.1%	89.1%	99.2%	99.96%	100.000%	100.000%	100.000%
Romanian	93.1%	20,275	3.8%	37.6%	88.2%	99.1%	99.95%	100.000%	100.000%	100.000%
Russian	90.0%	79,657	16.3%	57.4%	91.8%	99.2%	99.95%	99.996%	99.999%	100.000%
Slovenian	86.7%	9,699	25.7%	66.5%	96.5%	99.8%	100.00%	100.000%	100.000%	100.000%
Scandinavian	91.5%	54,890	16.9%	62.6%	95.2%	99.7%	99.98%	100.000%	100.000%	100.000%
Turkish	92.8%	8,753	43.7%	85.2%	97.3%	99.7%	99.98%	100.000%	100.000%	100.000%
Chinese	99.5%	18,544	46.3%	73.9%	92.1%	98.7%	99.90%	99.995%	100.000%	100.000%
others	91.7%	136,951	18.2%	63.1%	93.6%	99.4%	99.96%	99.996%	99.999%	100.000%

is so large that it does not necessarily restrict the performance of state-of-the-art statistical parsers if the gold tree is noncrossing: the finite-state search space contains almost 99.2% of the gold noncrossing trees. The related measure – *unlabeled attachment score (UAS)* – of the best dependency parsers is typically below 98%³ but these parsers and the used benchmarks are not limited to nonprojective gold trees.

The results prompt further work on statistical explanations of this phenomenon. It would also seem extremely important to try to develop a more general encoding. If a similar depth limit works well for an encoding that covers nonprojective trees, the corresponding search space would become relevant for parser development in the future. There are already some follow-up results suggesting that this is actually the case (Yli-Jyrä 2019).

5.6 *The contrast between theory and data*

There is a striking contrast between the theoretically limited coverage of depth-bounded grammars and their surprisingly good empirical coverage:

- From the *theoretical* point of view, the bounded grammar with 6 levels (Table 6 and Figure 10) is a finite-state approximation that represents a restricted subspace of parses. The theoretical coverage of this subspace drops rapidly below 99% when the sentence length grows beyond 20 token-vertices.
- From the *empirical* point of view (Table 8), six levels of brackets seem to cover more than 99.998% of the noncrossing trees in the actual linguistic data.

The contrast between theory and practice calls for an explanation: we want to know why the limited bracketing depth gives so much better practical coverage than what we would expect from a flat distribution. The first explanation for the high coverage of the noncrossing trees is that *most trees in the data set are short*. We do not know how representative the UD treebanks actually are, as samples, and it is, indeed, perfectly possible that some treebanks are biased towards short sentences. The solid curve in Figure 11 shows how the average

³http://nlpprogress.com/english/dependency_parsing.html

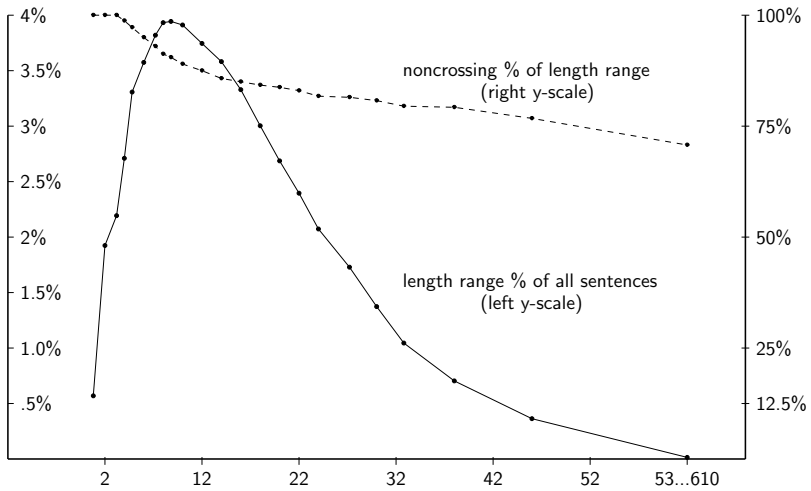


Figure 11: Solid line: the distribution of length ranges; dashed line: the percentage of noncrossing trees per length range in the v2.0 dataset

probability of the length range decreases as a function of sentence length in the data set. The length ranges in this plot were [2,2], ..., [9,9], [10,11], ..., [24,25], [26, 28], [29, 31], [32, 35], [36, 41], [42, 52], [53, 610], and each of them was represented by the median length, which is 62 for the last length range. The curve roughly indicates that relatively short sentence lengths are more probable than wide length ranges. In fact, although the tail range of lengths is quite long and nonempty, its probability mass is almost invisible in the big picture.

Another explanation for the extremely good coverage of low bracketing depths in Table 8 could be that *longer and more deeply bracketed sentences are more likely to have crossing edges*. The dashed curve in Figure 11 indicates that the percentage of noncrossing parse trees decreases when the sentence length increases. Quickly after the sentence length becomes long enough to have any crossing edges, the probability of noncrossing parses steps down to some 90% on average in the data set. Then, as the sentence length continues to increase further, this probability drops slowly until it goes below 70% of all sentences in the length range that contains the longest sentences in the data set. Thus, the parses of longer sentences are more likely to be excluded from the set of noncrossing trees than the parses of shorter

but more common sentences. As the result, a random sentence in the data set is relatively short and expected to be noncrossing with a high probability (90.5%).

The observation that the distribution of the noncrossing parses contains more shorter sentences does not mean that sentences with crossing edges are otherwise substantially “deeper”. In fact, our preliminary experiments on a more general bracketing scheme for all sentences suggest that the bracketing depth of all sentences differs very little from the bracketing depth of noncrossing sentences. Longer sentences may simply be more likely to have complex combinations of edges because they contain more places where a crossing or multiple overlapping can occur. In further work on encoding for unrestricted graphs, it would be possible to test how much crossing edges actually contribute to the local depth of the required bracketing.

The third possible explanation could be based on a psychological model that would predict the tendency to avoid long-distance dependencies (Gibson 1998) and multiple overlapping of edges when the sentence length grows arbitrarily. We could also look for an explanation from bounded memory models (Miller 1956; Kornai and Tuza 1992). With such models, it may be possible to understand why the nesting of superbrackets in the weak bracketing of data is so limited.

The language specific percentages of the noncrossing analyses depend on the choice of the annotation scheme (Havelka 2007). It is very possible that the uniform principles of the UD annotation scheme are not optimal for all languages. But we can perhaps interpret the overall low bracketing depth in the massively multilingual data set as a sign of some kind of cross-lingual uniformity in the complexity scale, which is a surprise because languages differ a lot in their strategies to minimise syntactic complexity.

The v2.0 data set contains seven sentences whose parses require seven levels of superbrackets. In the Appendix, we visualise the dependency structures of these seven noncrossing parses. The first observation from these examples is that their lengths are surprisingly high considering that these sentences are noncrossing: their lengths are between 29 and 106 tokens. This indicates that even long sentences can have noncrossing parses. Secondly, the examples indicate that the new encoding scheme is practically very effective as superbracketed edges have many sibling edges. The weak bracketing scheme divides the

set of edges into two categories, both of which contain a substantial number of overlapping edges. The divided visualisation of noncrossing trees has an advantage that although there are up to 15 overlapping edges and these sentences are pretty long, the paths in the visualised trees are relatively easy to follow from a distance, at a schematic level.⁴

5.7

On the errors in the data

It is obvious that treebanks contain a certain number of OCR errors, preprocessing errors and annotation errors. Annotation errors are typically due to the limitations or inconsistencies in the annotation manual or to other human factors that cause inconsistencies and mistakes. Although there is always a reason for annotation errors, we assume that they distribute almost randomly, having non-systematic effects on the depth of the dependency trees.

We had no realistic methods to try to estimate how often annotation errors occur. We just inspected a few most complex trees that we could find and comprehend. In such checking, we found no specific correlation between depth and errors.

6

CONCLUSION

The topic of this paper was to find a regular language that encodes noncrossing dependency graphs in treebanks. Our methodological approach used two different dependency bracketing schemes. The first encoding scheme – *strong bracketing* – has been presented previously and it has been applied to the description of several subfamilies of noncrossing graphs by Yli-Jyrä and Gómez-Rodríguez (2017). This scheme is based on balanced bracketing of edges. It uses three disjoint pairs of brackets to indicate three different orientations of edges. The second scheme – *weak bracketing* – does not properly appear in prior work and it is, therefore, a significant new contribution. In this encoding scheme, sibling edges are encoded with one-sided, weak brackets. We also considered optimisations to both bracketing schemes.

⁴The schematic nature of the Arabic sentence is a gap in the data set used, not a typographical problem.

The main result of this paper is that the new encoding scheme gives rise to a shallower and, in certain sense, less complex balanced bracketing than the previously known encoding scheme.

When we started the current work we did not know if such a shallow approach to dependency bracketing would even be possible and generalizable to noncrossing graphs. Our idea was to reduce the depth of dependency bracketing by omitting brackets when they share the same end of an edge. When this idea was conceived, we did not know if it would bring any practical benefits compared to the first scheme. But the investigation of the idea led to a few important results:

1. The discovery of a streamlined dependency bracketing

In this article, the weak dependency bracketing is presented and evaluated for the first time. Now we know that the scheme exists and corresponds to an unambiguous context-free language (Lemma 4.6), and that it has a deterministic, computable bijection from the set of (di)graphs (Lemma 4.4). This scheme constitutes a unique continuation to the history of ideas that aim at reducing complex balanced bracketing.

2. A context-free transduction between the two encodings

Now we also know that the two bracketing schemes can be related to each other with a context-free (non-deterministic push-down) transducer (Lemma 4.1). This transducer can be used to convert between the strong and the weak bracketing and to reduce the context-free encodable families of graphs (Yli-Jyrä and Gómez-Rodríguez 2017) to the weak dependency bracketing. This widens the possibilities of both bracketing schemes. Since there is a computable transduction between the strong and weak bracketing, all 50 subfamilies of noncrossing graphs characterized in Yli-Jyrä and Gómez-Rodríguez (2017) can be encoded with context-free languages that describe their weak bracketing. We observe, on page 205, that the search space of projective trees has a smaller state complexity than the noncrossing di-graphs, but the state complexity of some specialized search spaces of noncrossing subfamilies may be also slightly higher than the state complexity of the search space that contains all noncrossing graphs.

3. A low complexity bound with high empirical coverage

In dependency bracketing like Yli-Jyrä and Gómez-Rodríguez (2017), bounded-depth bracketing does not make the language finite, but the current work demonstrates that the weak bracketing is still useful because it stabilises the empirically observed depth of dependency bracketing: two levels of superbrackets cover already 58% of the noncrossing trees, three levels cover 92% and five 99.95%. Seven levels of superbrackets give the amazing 99.9998% coverage (with two excluded trees) over the massively multilingual set of dependency treebanks, UD v2.4.

The current work suggests several directions for further developments of the presented framework. We conclude this paper by introducing some of these directions.

6.1 *Fast Parsing and Neural Weighting*

The new empirical bounds open a door to new optimisations towards very efficient dependency parsing of multiple families of noncrossing graphs. An arc-factored, weighted, depth-bounded grammar for the strongly bracketed search space can be constructed in quadratic time. However, it is open whether a similar result is true for weakly bracketed search spaces.

The current work also demonstrated the existence of a high-coverage finite-state representation of a bounded grammar for noncrossing structures. When such a finite-state grammar is matched with a simplified arc weighting model, we would be very close to a linear-time graph-based parsing of bounded families of graphs.

In some state-of-the-art graph-based dependency parsers, the arc weights are computed with neural networks and the statistical inference is based on an algorithm that finds the maximum spanning tree (Libovický 2016; Ma and Hovy 2017) or best path (Rastogi *et al.* 2016). The current work is compatible with such hybrid models. It remains to be seen how the models could then be optimised together and how the search space representation interacts with the training of the weighting model.

Transition-based dependency parsing is mainly based on very expressive transition systems. If the current work could be extended

to nonprojective trees, a finite-state model of the bounded grammar could perhaps be used to handle neural transition systems and to improve nondeterministic strategies in these parsers.

Besides transition-based parsing, there are several other neural network based parsing models to which the current encoding or its unrestricted extension (Yli-Jyrä 2019) could be integrated, as mentioned in the introduction.

6.2 *Generality and Definable Properties of Graphs*

Since the noncrossing graphs have bounded treewidth, it is possible to obtain many efficient algorithms for them. Especially, there is an algorithmic metatheorem (Courcelle 1990) that states that any graph property in monadic second order logic (MSO) can be decided in linear time for bounded-treewidth graphs. Yli-Jyrä and Gómez-Rodríguez (2017) can be seen as a start for a research that reconstructs this metatheorem via dependency bracketing and context-free grammars in the case of noncrossing graphs. It is, indeed, possible to create an algorithm library that implements MSO logic for noncrossing graphs, using the currently explored encoding schemes.

The currently presented encoding is a crucial step towards more comprehensive bracket-based encoding of graphs. It is possible to develop similar encoding schemes for unrestricted ordered graphs. Indeed, we have already worked on an encoding that generalises elegantly to all ordered graphs. The description of the generalised encoding will appear separately (Yli-Jyrä 2019).

6.3 *Learnability of subregular approximations of syntax*

By showing that the positive examples in the training data have a robust bound for the depth of bracketing in the context-free encoding, the dependency structures can be seen as a regular language, with a truncated Chomsky-Schützenberger representation. It has been previously observed that such regular languages are often star-free (Yli-Jyrä 2003a, 2005a,b), but their descriptive complexity depends on the bracketing depth (Yli-Jyrä 2008, 2005c). Thus, they do not belong to any of the basic subregular classes of languages that have been shown to be learnable from positive data (Heinz and Rogers 2013). From the structure of languages in Yli-Jyrä and Gómez-Rodríguez (2017), we can infer that learning non-local properties of noncrossing graphs also

requires learning latent labeling of their bracketing. These challenges put a strain on the research on such subregular language classes that would allow us to learn finite-state approximations of syntax from treebanks. This research could be related to representation learning in neural networks.

ACKNOWLEDGEMENTS

The author has received funding from the Academy of Finland (dec. No 270354/273457/313478 – A Usable Finite-State Model for Adequate Syntactic Complexity) and the Clare Hall Fellowship from the University of Helsinki (dec. RP 137/2013 – SMT based on D-Tree Contraction Grammars and FSTs). The author is also grateful to Kimmo Koskenniemi, Lauri Carlson, Aro Voutilainen, Solomon Marcus, Filip Ginter, Tapio Salakoski, Steven Krauwer, Terence Langendoen, Fred Karlsson, Kemal Oflazer, Mans Hulden, András Kornai, Krister Lindén, James Rogers, Bill Byrne, Gonzalo Iglesias, Mark-Jan Nederhof, Jussi Piitulainen, Alexander Okhotin, Carlos Gómez-Rodríguez, Juha Kontinen, Andreas Maletti, Brian Roark, Jonathan May, Joel Mathew, Ronald Kaplan, Lauri Karttunen, Edward Gibson, Henrik Björklund, Adam Jardine, Galina Jirásková, Elin Ehsani, Eli Shamir, Gary Miles, and Tatu Ylönen for helpful discussions during earlier stages of the research, and to the anonymous reviewers for their valuable feedback on the earlier versions of this paper.

APPENDIX: HIGH-LEVEL VISUALISATION OF THE SEVEN SENTENCES
WITH SEVEN LEVELS OF SUPERBRACKETED EDGES IN THE UD V2.0 DATA SET



Figure12: Arabic (string of part-of-speech tags)

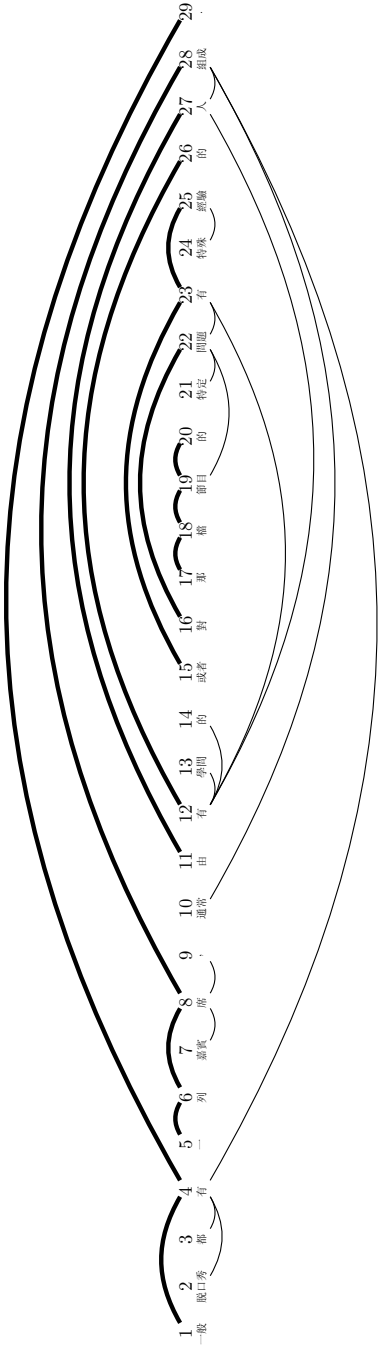


Figure 13: Chinese

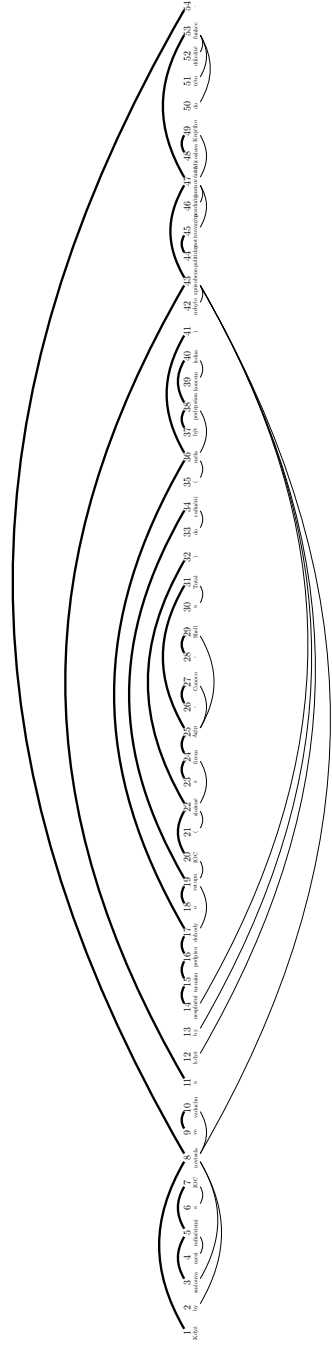


Figure 14: Czech



Figure 15: German

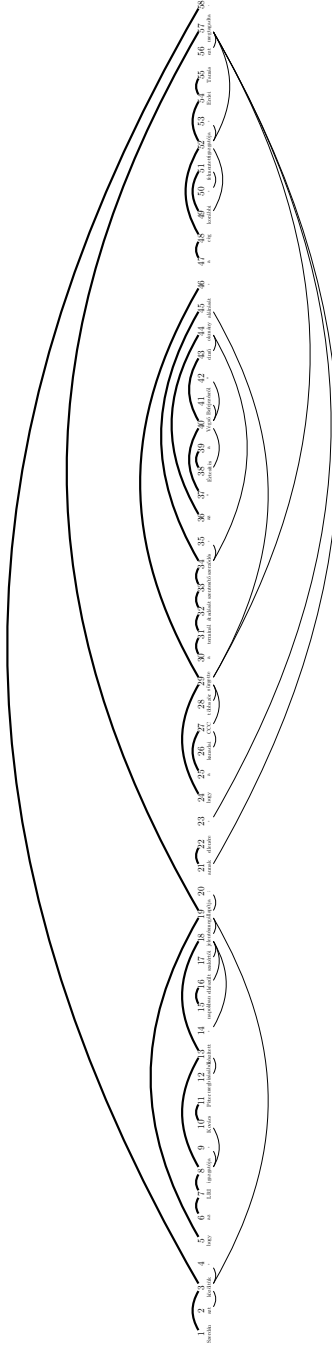


Figure 16: Hungarian

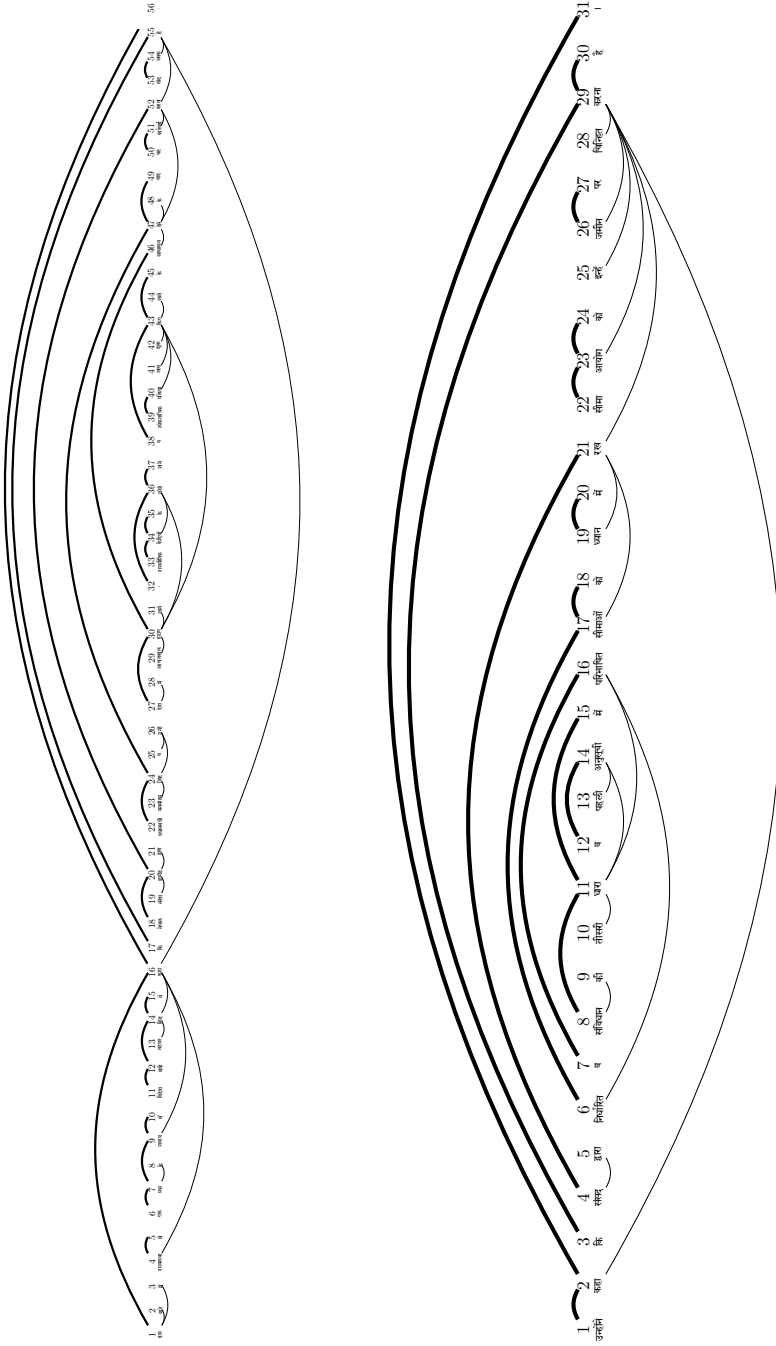


Figure 17: Hindi

REFERENCES

- Bernd BOHNET, Ryan McDONALD, Emily PITLER, and Ji MA (2016), Generalized transition-based dependency parsing via control parameters, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pp. 150–160, Association for Computational Linguistics, <http://dx.doi.org/10.18653/v1/P16-1015>.
- Mikołaj BOJAŃCZYK and Michał PILIPCZUK (2016), Definability equals recognizability for graphs of bounded treewidth, in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pp. 407–416, ACM, New York, NY, USA, ISBN 978-1-4503-4391-6, <http://dx.doi.org/10.1145/2933575.2934508>.
- Noam CHOMSKY (1963), Formal properties of grammars, in R. Duncan LUCE, Robert R. BUSH, and Eugene GALANTER, editors, *Handbook of mathematical psychology*, volume 2, pp. 323–418, John Wiley and Sons, New York.
- Bruno COURCELLE (1990), The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, *Information and Computation*, 85(1):12 – 75, ISSN 0890-5401, doi:[https://doi.org/10.1016/0890-5401\(90\)90043-H](https://doi.org/10.1016/0890-5401(90)90043-H), <http://www.sciencedirect.com/science/article/pii/089054019090043H>.
- Michael A. COVINGTON (2001), A fundamental algorithm for dependency parsing, in John A. MILLER and Jeffrey W. SMITH, editors, *Proceedings of the 39th Annual ACM Southeast Conference*, pp. 95–102, Association for Computing Machinery, <http://www.covingtoninnovations.com/mc/dgpacm.pdf>.
- Chris DYER, Miguel BALLESTEROS, Wang LING, Austin MATTHEWS, and Noah A. SMITH (2015), Transition-based dependency parsing with Stack Long Short-Term Memory, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 334–343, Association for Computational Linguistics, Beijing, China, <http://dx.doi.org/10.3115/v1/P15-1033>.
- Chris DYER, Adhiguna KUNCORO, Miguel BALLESTEROS, and Noah A. SMITH (2016), Recurrent neural network grammars, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 199–209, Association for Computational Linguistics, <http://www.aclweb.org/anthology/N16-1024>.
- Jason EISNER and Giorgio SATTA (1999), Efficient parsing for bilexical context-free grammars and Head Automaton Grammars, in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pp. 457–464, Association for Computational Linguistics, <https://doi.org/10.3115/1034678.1034748>.

Edward GIBSON (1998), Linguistic complexity: locality of syntactic dependencies, *Cognition*, 68(1):1–76, [http://doi.org/10.1016/S0010-0277\(98\)00034-1](http://doi.org/10.1016/S0010-0277(98)00034-1).

Filip GINTER, Sampo PYYSALO, Jorma BOBERG, and Tapio SALAKOSKI (2006), Regular approximation of Link grammar, in *Proceedings of the 5th International Conference on Advances in Natural Language Processing, FinTAL'06*, pp. 564–575, Springer-Verlag, Berlin, Heidelberg, ISBN 3-540-37334-9, 978-3-540-37334-6, http://dx.doi.org/10.1007/11816508_56.

Charles F. GOLDFARB (1999), Future directions in SGML/XML, in Wiebke MÖHR and Ingrid SCHMIDT, editors, *SGML und XML: Anwendungen und Perspektiven*, pp. 3–25, Springer, Berlin, Germany, https://doi.org/10.1007/978-3-642-46881-0_1.

Carlos GÓMEZ-RODRÍGUEZ and Joakim NIVRE (2010), A transition-based parser for 2-planar dependency structures, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pp. 1492–1501, Association for Computational Linguistics, <http://dl.acm.org/citation.cfm?id=1858681.1858832>.

Sheila GREIBACH (1973), The hardest context-free language, *SIAM Journal on Computing*, 2(4):304–310, <http://dx.doi.org/10.1137/0202025>.

Jiří HAVELKA (2007), Beyond projectivity: multilingual evaluation of constraints and measures on non-projective structures, in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pp. 608–615, Association for Computational Linguistics, <http://www.aclweb.org/anthology/P07-1077>.

Jeffrey HEINZ and James ROGERS (2013), Learning subregular classes of languages with factored deterministic automata, in *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pp. 64–71, Association for Computational Linguistics, Sofia, Bulgaria, <https://www.aclweb.org/anthology/W13-3007>.

Mans HULDEN and Miikka SILFVERBERG (2014), Finite-state subset approximation of phrase structure, in *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, USA, https://www.cs.uic.edu/pub/Isaim2014/WebPreferences/ISAIM2014_NLP_Hulden_Silverberg.pdf.

Eliyahu KIPERWASSER and Yoav GOLDBERG (2016), Simple and accurate dependency parsing using bidirectional LSTM feature representations, *Transactions of the Association for Computational Linguistics*, 4:313–327, http://dx.doi.org/10.1162/tac1_a_00101.

András KORNAI and Zolt TUZA (1992), Narrowness, path-width, and their application in natural language processing, *Discrete Applied Mathematics*, 36(1):87–92, [https://doi.org/10.1016/0166-218X\(92\)90208-R](https://doi.org/10.1016/0166-218X(92)90208-R).

Kimmo KOSKENNIEMI (1990), Finite-state parsing and disambiguation, in Hans KARLGREN, editor, *13th International Conference on Computational Linguistics, (COLING 1990)*, pp. 229–232, <http://aclweb.org/anthology/C90-2040>.

Steven KRAUWER and Louis DES TOMBE (1981), Transducers and grammars as theories of language, *Theoretical Linguistics*, 8(1–3):173–202, <https://doi.org/10.1515/thli.1981.8.1-3.173>.

Marco KUHLMANN (2015), Tabulation of noncrossing acyclic digraphs, arXiv:1504.04993, <https://arxiv.org/abs/1504.04993>.

Marco KUHLMANN and Peter JOHNSSON (2015), Parsing to noncrossing dependency graphs, *Transactions of the Association for Computational Linguistics*, 3:559–570, <http://aclweb.org/anthology/Q/Q15/Q15-1040.pdf>.

Adhiguna KUNCORO, Miguel BALLESTEROS, Lingpeng KONG, Chris DYER, Graham NEUBIG, and Noah A. SMITH (2017), What do recurrent neural network grammars learn about syntax?, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pp. 1249–1258, Association for Computational Linguistics, <http://aclweb.org/anthology/E17-1117>.

D. Terence LANGENDOEN (1975), Finite-state parsing of phrase-structure languages and the status of readjustment rules in grammar, *Linguistic Inquiry*, 6(4):533–554, <http://www.jstor.org/stable/4177899>.

D. Terence LANGENDOEN and Yedidyah LANGSAM (1984), The representation Of constituent structures for finite-state parsing, in Yorick WILKS, editor, *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics, Proceedings of COLING '84, July 2-6, 1984, Stanford University, California, USA.*, pp. 24–27, ACL, <http://aclweb.org/anthology/P84-1007>.

Jindřich LIBOVICKÝ (2016), Neural scoring function for MST parser, in Nicoletta CALZOLARI, Khalid CHOUKRI, Thierry DECLERCK, Sara GOGGI, Marko GROBELNIK, Bente MAEGAARD, Joseph MARIANI, Helene MAZO, Asuncion MORENO, Jan ODIJK, and Stelios PIPERIDIS, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, European Language Resources Association (ELRA), <http://www.lrec-conf.org/proceedings/lrec2016/summaries/649.html>.

Xuezhe MA and Eduard H. HOVY (2017), Neural probabilistic model for non-projective MST parsing, in *Proceedings of the The 8th International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 59–69, Asian Federation of Natural Language Processing, <http://www.aclweb.org/anthology/I17-1007>.

Ryan McDONALD, Fernando PEREIRA, Kiril RIBAROV, and Jan HAJIČ (2005), Non-projective dependency parsing using spanning tree algorithms, in *Proceedings of Human Language Technology Conference and Conference on*

Empirical Methods in Natural Language Processing, pp. 523–530, Association for Computational Linguistics,

<http://www.aclweb.org/anthology/H/H05/H05-1066.pdf>.

George A. MILLER (1956), The magical number seven, plus or minus two: some limits on our capacity for processing information, *Psychological Review*, 63(2):81–97, <http://dx.doi.org/10.1037/h0043158>.

Mehryar MOHRI (1997), Finite-state transducers in language and speech processing, *Computational Linguistics*, 23(2):1–42,

<http://www.aclweb.org/anthology/J97-2003>.

Mark-Jan NEDERHOF (2000), Practical experiments with regular approximation of context-free languages, *Computational Linguistics*, 26(1):17–44, <http://dx.doi.org/10.1162/089120100561610>.

Joakim NIVRE (2008), Algorithms for deterministic incremental dependency parsing, *Computational Linguistics*, 34(4):513–553,

<https://doi.org/10.1162/coli.07-056-R1-07-027>.

Joakim NIVRE (2009), Non-projective dependency parsing in expected linear time, in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pp. 351–359, Association for Computational Linguistics,

<http://dl.acm.org/citation.cfm?id=1687878.1687929>.

Joakim NIVRE, Mitchell ABRAMS, Željko AGIĆ, Lars AHRENBERG, Gabrielē ALEKSANDRAVIČIŪTĒ, Lene ANTONSEN, Katya APLONOVA, Maria Jesus ARANZABE, Gashaw ARUTIE, Masayuki ASAHARA, Luma ATEYAH, Mohammed ATTIA, Aitziber ATUTXA, Liesbeth AUGUSTINUS, Elena BADMAEVA, Miguel BALLESTEROS, Esha BANERJEE, Sebastian BANK, Verginica BARBU MITITELU, Victoria BASMOV, John BAUER, Sandra BELLATO, Kepa BENGOTXEA, Yevgeni BERZAK, Irshad Ahmad BHAT, Riyaz Ahmad BHAT, Erica BIAGETTI, Eckhard BICK, Agnē BIELINSKIENĒ, Rogier BLOKLAND, Victoria BOBICEV, Loïc BOIZOU, Emanuel BORGES VÖLKER, Carl BÖRSTELL, Cristina BOSCO, Gosse BOUMA, Sam BOWMAN, Adriane BOYD, Kristina BROKAITĒ, Aljoscha BURCHARDT, Marie CANDITO, Bernard CARON, Gauthier CARON, Gülşen CEBIROĞLU ERYIĞIT, Flavio Massimiliano CECCHINI, Giuseppe G. A. CELANO, Slavomír ČĚPLŮ, Savas CETIN, Fabricio CHALUB, Jinho CHOI, Yongseok CHO, Jayeol CHUN, Silvie CINKOVÁ, Aurélie COLLOMB, Çağrı ÇÖLTEKIN, Miriam CONNOR, Marine COURTIN, Elizabeth DAVIDSON, Marie-Catherine DE MARNEFFE, Valeria DE PAIVA, Arantza DIAZ DE ILARRAZA, Carly DICKERSON, Bamba DIONE, Peter DIRIX, Kaja DOBROVOLJC, Timothy DOZAT, Kira DROGANOVA, Puneet DWIVEDI, Hanne ECKHOFF, Marhaba ELI, Ali ELKAHKY, Binyam EPHREM, Tomáš ERJAVEC, Aline ETIENNE, Richárd FARKAS, Hector FERNANDEZ ALCALDE, Jennifer FOSTER, Cláudia FREITAS, Kazunori FUJITA, Katarína GAJDOŠOVÁ, Daniel GALBRAITH, Marcos GARCIA,

Moa GÄRDENFORS, Sebastian GARZA, Kim GERDES, Filip GINTER, Iakes GOENAGA, Koldo GOJENOLA, Memduh GÖKIRMAK, Yoav GOLDBERG, Xavier GÓMEZ GUINOVART, Berta GONZÁLEZ SAAVEDRA, Matias GRIONI, Normunds GRŪZĪTIS, Bruno GUILLAUME, Céline GUILLOT-BARBANCE, Nizar HABASH, Jan HAJIČ, Jan HAJIČ JR., Linh HÀ MỸ, Na-Rae HAN, Kim HARRIS, Dag HAUG, Johannes HEINECKE, Felix HENNIG, Barbora HLADKÁ, Jaroslava HLAVÁČOVÁ, Florinel HOCIUNG, Petter HOHLE, Jena HWANG, Takumi IKEDA, Radu ION, Elena IRIMIA, Olájidé ISHOLA, Tomáš JELÍNEK, Anders JOHANNSEN, Fredrik JØRGENSEN, Hüner KAŞIKARA, Andre KAASEN, Sylvain KAHANE, Hiroshi KANAYAMA, Jenna KANERVA, Boris KATZ, Tolga KAYADELEN, Jessica KENNEY, Václava KETTNEROVÁ, Jesse KIRCHNER, Arne KÖHN, Kamil KOPACEWICZ, Natalia KOTSYBA, Jolanta KOVALEVSKAITĖ, Simon KREK, Sookyoung KWAK, Veronika LAIPPALA, Lorenzo LAMBERTINO, Lucia LAM, Tatiana LANDO, Septina Dian LARASATI, Alexei LAVRENTIEV, John LEE, Phùng LÊ HỒNG, Alessandro LENCI, Saran LERTPRADIT, Herman LEUNG, Cheuk Ying LI, Josie LI, Keying LI, KyungTae LIM, Yuan LI, Nikola LJUBEŠIĆ, Olga LOGINOVA, Olga LYASHEVSKAYA, Teresa LYNN, Vivien MACKETANZ, Aibek MAKAZHANOV, Michael MANDL, Christopher MANNING, Ruli MANURUNG, Cătălina MĂRĂNDUC, David MAREČEK, Katrin MARHEINECKE, Héctor MARTÍNEZ ALONSO, André MARTINS, Jan MAŠEK, Yuji MATSUMOTO, Ryan McDONALD, Sarah MCGUINNESS, Gustavo MENDONÇA, Niko MIEKKA, Margarita MISIRPASHAYEVA, Anna MISSILÄ, Cătălin MITITELU, Yusuke MIYAO, Simonetta MONTEMAGNI, Amir MORE, Laura MORENO ROMERO, Keiko Sophie MORI, Tomohiko MORIOKA, Shinsuke MORI, Shigeki MORO, Bjartur MORTENSEN, Bohdan MOSKALEVSKYI, Kadri MUISCHNEK, Yugo MURAWAKI, Kaili MÜÜRİSEP, Pinkey NAINWANI, Juan Ignacio NAVARRO HORŇIAČEK, Anna NEDOLUZHKO, Gunta NEŠPORE-BĚRZKALNE, Lương NGUYỄN THỊ, Huyền NGUYỄN THỊ MINH, Yoshihiro NIKAIDO, Vitaly NIKOLAEV, Rattima NITISAROJ, Hanna NURMI, Stina OJALA, Adédayò OLÚÒKUN, Mai OMURA, Petya OSENOVA, Robert ÖSTLING, Lilja ØVRELID, Niko PARTANEN, Elena PASCUAL, Marco PASSAROTTI, Agnieszka PATEJUK, Guilherme PAULINO-PASSOS, Angelika PELJAK-ŁAPIŃSKA, Siyao PENG, Cenel-Augusto PEREZ, Guy PERRIER, Daria PETROVA, Slav PETROV, Jussi PIITULAINEN, Tommi A PIRINEN, Emily PITLER, Barbara PLANK, Thierry POIBEAU, Martin POPEL, Lauma PRETKALNIŅA, Sophie PRÉVOST, Prokopis PROKOPIDIS, Adam PRZEPIÓRKOWSKI, Tiina PUOLAKAINEN, Sampo PYYSALO, Andriela RĂĂBIS, Alexandre RADEMAKER, Loganathan RAMASAMY, Taraka RAMA, Carlos RAMISCH, Vinit RAVISHANKAR, Livy REAL, Siva REDDY, Georg REHM, Michael RIEßLER, Erika RIMKUTĖ, Larissa RINALDI, Laura RITUMA, Luisa ROCHA, Mykhailo ROMANENKO, Rudolf ROSA, Davide ROVATI, Valentin ROŞCA, Olga RUDINA, Jack RUETER, Shoval SADDE, Benoît SAGOT, Shadi SALEH, Alessio SALOMONI, Tanja SAMARDŽIĆ, Stephanie SAMSON, Manuela SANGUINETTI, Dage SÄRG, Baiba SAULĪTE, Yanin SAWANAKUNANON, Nathan

SCHNEIDER, Sebastian SCHUSTER, Djamé SEDDAH, Wolfgang SEEKER, Mojgan SERAJI, Mo SHEN, Atsuko SHIMADA, Hiroyuki SHIRASU, Muh SHOHIBUSSIRRI, Dmitry SICHINAVA, Natalia SILVEIRA, Maria SIMI, Radu SIMIONESCU, Katalin SIMKÓ, Mária ŠIMKOVÁ, Kiril SIMOV, Aaron SMITH, Isabela SOARES-BASTOS, Carolyn SPADINE, Antonio STELLA, Milan STRAKA, Jana STRNADOVÁ, Alane SUHR, Umut SULUBACAK, Shingo SUZUKI, Zsolt SZÁNTÓ, Dima TAJI, Yuta TAKAHASHI, Fabio TAMBURINI, Takaaki TANAKA, Isabelle TELLIER, Guillaume THOMAS, Liisi TORGA, Trond TROSTERUD, Anna TRUKHINA, Reut TSARFATY, Francis TYERS, Sumire UEMATSU, Zdeňka UREŠOVÁ, Larraitz URIA, Hans USZKOREIT, Sowmya VAJJALA, Daniel VAN NIEKERK, Gertjan VAN NOORD, Viktor VARGA, Eric VILLEMONT DE LA CLERGERIE, Veronika VINCZE, Lars WALLIN, Abigail WALSH, Jing Xian WANG, Jonathan North WASHINGTON, Maximilian WENDT, Seyi WILLIAMS, Mats WIRÉN, Christian WITTERN, Tsegay WOLDEMARIAM, Tak-sum WONG, Alina WRÓBLEWSKA, Mary YAKO, Naoki YAMAZAKI, Chunxiao YAN, Koichi YASUOKA, Marat M. YAVRUMYAN, Zhuoran YU, Zdeněk ŽABOKRTSKÝ, Amir ZELDES, Daniel ZEMAN, Manying ZHANG, and Hanzhi ZHU (2019), *Universal Dependencies 2.4*, <http://hdl.handle.net/11234/1-2988>, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Kemal OFLAZER (2003), Dependency parsing with an extended finite-state approach, *Computational Linguistics*, 29(4):515–544, <http://dx.doi.org/10.1162/089120103322753338>.

Pushpendre RASTOGI, Ryan COTTERELL, and Jason EISNER (2016), Weighting finite-state transductions with neural context, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 623–633, Association for Computational Linguistics, <http://dx.doi.org/10.18653/v1/N16-1076>.

Brian ROARK and Kristy HOLLINGSHEAD (2008), Classifying chart cells for quadratic complexity context-free inference, in *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, volume 1, pp. 745–752, Association for Computational Linguistics, <http://dl.acm.org/citation.cfm?id=1599081.1599175>.

Arto SALOMAA (1973), *Formal languages*, Academic Press, New York, NY.

Michalina STRYZ, David VILARES, and Carlos GÓMEZ-RODRÍGUEZ (2019), Viable dependency parsing as sequence labeling, in Jill BURSTEIN, Christy DORAN, and Thamar SOLORIO, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 717–723, Association for Computational Linguistics, ISBN 978-1-950737-13-0, <https://aclweb.org/anthology/papers/N/N19/N19-1077/>.

Warren TEITELMAN (1978), *INTERLISP reference manual*, Xerox Palo Alto Research Center (PARC), Palo Alto, CA.

Oriol VINYALS, Lukasz KAISER, Terry KOO, Slav PETROV, Ilya SUTSKEVER, and Geoffrey E. HINTON (2015), Grammar as a foreign language, in Corinna CORTES, Neil D. LAWRENCE, Daniel D. LEE, Masashi SUGIYAMA, and Roman GARNETT, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2773–2781, <http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language>.

Anssi YLI-JYRÄ (2003a), Describing syntax with star-free regular expressions, in *10th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Budapest, Hungary, <https://www.aclweb.org/anthology/E03-1031>.

Anssi YLI-JYRÄ (2003b), Multiplanarity – a model for dependency structures in treebanks, in Joakim NIVRE and Erhard HINRICHS, editors, *TLT 2003. Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, volume 9 of *Mathematical Modelling in Physics, Engineering and Cognitive Sciences*, pp. 189–200, Växjö University Press, Växjö, Sweden, <http://hdl.handle.net/10138/33872>.

Anssi YLI-JYRÄ (2003c), Regular approximations through labeled bracketing, in Gerald PENN, editor, *Proceedings of FGVienna: the 8th Conference on Formal Grammar*, pp. 153–166, CSLI Publications, Stanford, CA, <http://csli-publications.stanford.edu/FG/2003/ylijyra.pdf>.

Anssi YLI-JYRÄ (2004), Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings, in Geert-Jan M. KRUIJFF and Denys DUCHIER, editors, *Recent Advances in Dependency Grammar (COLING 2004 workshop)*, pp. 25–32, Association for Computational Linguistics, Geneva, Switzerland, <https://www.aclweb.org/anthology/W/W04/W04-1504.pdf>.

Anssi YLI-JYRÄ (2005a), Approximating dependency grammars through intersection of star-free regular languages, *International Journal of Foundations of Computer Science*, 16(3):565–579, <http://dx.doi.org/10.1142/S0129054105003169>.

Anssi YLI-JYRÄ (2005b), *Contributions to the theory of finite-state based grammars*, Ph.D. thesis, University of Helsinki, Faculty of Arts, Department of General Linguistics, <http://urn.fi/URN:ISBN:952-10-2510-7>.

Anssi YLI-JYRÄ (2005c), Linguistic grammars with very low complexity, in Antti Arppe *et al.*, editor, *Inquiries into Words, Constraints and Contexts: Festschrift in the Honour of Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics ONLINE, pp. 172–183, CSLI Publications, Stanford, CA, <https://web.stanford.edu/group/csli-publications/csli-publications/koskenniemi-festschrift/17-yli-jyra.pdf>.

Anssi YLI-JYRÄ (2008), Applications of diamonded double negation, in T. HANNEFORTH and K-M. WÜRZNER, editors, *Finite-State Methods and Natural Language Processing, 6th International Workshop, FSMNL-2007, Potsdam, Germany, September 14–16, Revised Papers*, pp. 6–30, Potsdam University Press, Potsdam, <https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/index/index/year/2008/docId/2519>.

Anssi YLI-JYRÄ (2012), On dependency analysis via contractions and weighted FSTs, in Diana SANTOS, Krister LINDÉN, and Wanjiku NG'ANG'A, editors, *Shall we play the Festschrift game? Essays on the occasion of Lauri Carlson's 60th birthday*, pp. 133–158, Springer, Berlin, Germany, http://dx.doi.org/10.1007/978-3-642-30773-7_10.

Anssi YLI-JYRÄ (2019), Transition-based coding and formal language theory for ordered digraphs, Paper accepted to FSMNLP 2019, The 14th International Conference on Finite-State Methods and Natural Language Processing, September 23-25, Dresden, Germany.

Anssi YLI-JYRÄ and Carlos GÓMEZ-RODRÍGUEZ (2017), Generic axiomatization of families of noncrossing graphs in dependency parsing, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 1745–1755, Association for Computational Linguistics, <http://aclweb.org/anthology/P17-1160>.

Anssi YLI-JYRÄ and Matti NYKÄNEN (2014), A hierarchy of mildly context-sensitive dependency grammars, in Paola MONACHESI, editor, *Proceedings of the Nancy Conference, Formal Grammar 2004*, pp. 155–170, CSLI Publications, Stanford, CA, <http://csli-publications.stanford.edu/FG/2004/fg-2004-paper11.pdf>.

Hao ZHANG and Ryan T. McDONALD (2014), Enforcing structural diversity in cube-pruned dependency parsing, in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pp. 656–661, Association for Computational Linguistics, <http://aclweb.org/anthology/P/P14/P14-2107.pdf>.

Xiaoqing ZHENG (2017), Incremental graph-based neural dependency parsing, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1655–1665, Association for Computational Linguistics, <http://dx.doi.org/10.18653/v1/D17-1173>.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>

