Computational locality of cyclic phonology in Armenian

A Dissertation presented

by

Hossep Dolatian

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Linguistics

Stony Brook University

May 2020

Stony Brook University

The Graduate School

Hossep Dolatian

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Jeffrey Heinz - Dissertation Advisor
Professor, Department of Linguistics

Mark Aronoff - Chairperson of Defense
Professor, Department of Linguistics

Christina Bethin
Professor, Department of Linguistics

Adam Jardine
Assistant Professor, Department of Linguistics
Rutgers University

Bert Vaux
Reader, Faculty of Modern and Medieval Languages and Linguistics
University of Cambridge

This dissertation is accepted by the Graduate School

Eric Wertheimer
Dean of the Graduate School

Abstract of the Dissertation

Computational locality of cyclic phonology in Armenian

by

Hossep Dolatian

Doctor of Philosophy

in

Linguistics

Stony Brook University

2020

The title of this dissertation indicates its goal: to determine the computational aspects of cyclic phonology as it operates in Armenian. This goal is divided into two subgoals based on empirical and computational questions.

On the empirical side, I show that Armenian requires a model of the morphology-phonology interface which is interactionist and cyclic, i.e., that morphological structure, prosodic structure, and phonological rules cyclically interact to create new words. Evidence for this nuanced organization comes from the stratal phonology of Armenian (cf. Lexical Phonology and Stratal OT: Kiparsky 1982b; Bermúdez-Otero 2018). There are phonological processes which apply differently before derivational morphology than in inflectional morphology. The main process that I examine is destressed high vowel reduction. This and other processes indicate not only different strata or levels, but also show signs of unbounded cyclicity and sensitivity to sublexical prosodic constituents, i.e., the Prosodic Stem (Downing 1999a). These processes are active in both simplex and compound words. Within compounds, the interaction of all these factors creates bracketing paradoxes. I solve these paradoxes using a mixture of cyclic prosodic phonology and Head-Operations (Hoeksema 1988). I show that counter-cyclic approaches to bracketing paradoxes, like Morphological Merger or Rebracketing (Marantz 1988), are inadequate because they contradict the rest of Armenian phonology.

There are many different incarnations of cyclic theories of phonology but there are little to no computational analyses of them. I develop an extensive computational formalization for cyclic or interactionist phonology by using Monadic-Second Order (MSO) logic, specifically graph-to-graph logical transductions. Logic is a flexible tool that lets us create iconic formalizations which faithfully replicate phonological theory. I utilize logical transductions as a way to encode the derivational nature of phonology. The formalism is a generalization of the work from one-level Declarative Phonology (Bird 1995; Coleman 1998) to a two-level framework. By formalizing the morphology, prosody, and phonological rule domains, I uncover implicit factors within cyclic phonology. The ultimate takeaway is that I show that the bulk of the morphology-phonology interface requires *local* computation, not global computation. By being local, the computational nature of the

morphology-phonology interface opens doors to understanding how we can provably learn morpho-phonological processes and how these processes relate to limitations on human cognition (cf. Heinz 2018).

Dedication Page

Ծնողքիս համար

For my parents

Acknowledgements

After 6 years, I finally made a dissertation. It's hard to verbalize the range of emotions one feels doing this. All I know is that, I am thankful to the friends and family I made along the way. I'm not sure how to achieve a balance of professional gratitude and personal gratitude. After all, the people and connections I made in the last 6 years inevitably spilled over outside the confines of a cubicle.

My first gratitude goes to my advisor and supervisor Jeff Heinz. He was the reason I got accepted into my original PhD program at Delaware, and now I followed him to Stony Brook. It has been a pleasure coauthoring with him, and buying his daughter's Girl Scout cookies. He knows his way around the classroom and can make the densest mathematical concepts be spoon-feedable. I owe half of my research profile to him, and for that I will always be grateful. Thank you for making me a computational linguist.

The other half of my research is thanks to my unofficial second advisor at Delaware, Irene Vogel. Our encounters at Delaware were brief, but continued well into my move to Stony Brook. Her precision and demands for analytical correctness are admirable, as are her vegan lasagnas and dinner parties. Thank you for making me an Armenologist.

Although I only got to be at Stony Brook for only two years (and one long-distance third year), I was no stranger to the warmness of the faculty there. Christina Bethin has been a wonderful co-advisor, who spent many hours going through my manuscripts with red ink. Besides proof-reading, she has been a joy to visit in her office, and it's been a joy to recycle her books and typewriter. Mark Aronoff leaves an indelible impression on anyone, such as when you first meet him at a workshop and he's yelling out 'where's the sushi!'[1] His whimsy is paralleled only by his grasp of morphological lore. I thank him for making me a tad less ignorant about morphology. My external committee members, Adam Jardine and Bert Vaux, deserve thanks for reading and commenting on this dissertation. Each of their feedback has made this dissertation make more sense to me, and hopefully make more sense to others. My gratitude to many other linguists whose conversations have helped me go through the Armenian data, especially Laura Downing, Ricardo Bermúdez-Otero, Donca Steriade, Thomas Graf, Ellen Broselow, and Michael Becker.

But of course, a dissertation is a culmination of years of education and perseverance. For that, I owe many people my gratitude.

My ancestors made the sacrifices to immigrate and find refuge in Lebanon. My parents suffered hardships which I'll never truly know about. All I will ever know is that their love sent me to school, to college, and abroad. It's overwhelming to try to carve out the trajectory that eventually got me into a doctoral program, the many sacrifices that my parents had to make to survive war, illiteracy, and illness. I thank them for choosing to send me to an Armenian school, lest I forget the language. I thank my father for bracing himself as he sent me off at the airport, and to my mother for giving me her blessings before her passing. I dedicate this dissertation to their sacrifices and love, neither of which I will forget. Because of a language barrier, they will never read this acknowledgement, but they know what is in my heart.

Because of my parent's sacrifices, I was the first in my family to enter college, a place where I got to learn and experience the shock of a larger world outside the confines of my neighborhood. I thank my undergraduate advisors and professors for getting me on the path from high school to graduate school, for teaching me how to think and appreciate the complexities of abstraction. Rula Diab was always kind and patient in her office hours, and Dany Badran was always willing to go the extra mile in making his lectures

---

[1]People who know him, may realize what I omitted from this quotation.

approachable. Brian Prescott-Decie was always there for a thoughtful and adventurous monologue. I owe my knowledge of Formal Language Theory to Faisal Abu-Khzam, who laid the seed of what would later be my work in subregular phonology.

Inside and outside the walls of my alma mater, I got to grow alongside many people whose memory continues with me. Dareen Shehab and her family have always been warm and welcoming, whether it's while loitering at bookshops, dining over Ramadan, and hectically finishing assignments. George Chalhoub was often behind me, planning the next escapade. Diala AlMasri was a sister-in-arms as we went through the maze of scholarships and foundations. Avedis Samuelian would haunt the library, sporadically proclaiming Neitzscheism. And Hrag Vosguerichian became my first ever conference buddy, even for just a single conference.

After I left the hustle and bustle of cosmopolitan Beirut, I had the pleasure of meeting many new and fascinating people in the most unlikely of places, Delaware. Curtis Line was my unofficial 'ambassador' as I floundered through my first summer. Stephen Benscoter gave me many fond escapes outside of the office, and a memorable poncho birthday at the opera. Cindy Wiegand and her family were the source of many fond memories and holidays in a strange new land. I'll always cherish our days at church and ice-cream brunches. I'll always remember Sue Wiegand for her kindness and generosity, which she bestowed on her unofficial exchange student.

Within the confines of the cubicle-less department at Delaware, I made friends who made my time more tolerable and, dare I say, actually fun. Nina Straitman was kind in her office, and warm in her actions. Chao Han was always there for comfortable and friendly silence as we dogged on. Lachelle Stewart knew how to be simultaneously wise and funny. Lexi Geibler could turn any of my rants into a fun pizza party. Alicia Porter made the best guide to Philadelphia's sights. Taylor Miller was an effective and bubbly mother hen, while Em Miller knew how to appreciate a good Halloween dress. It was my pleasure to smuggle tahnali ($\sim$ lady-bread) to them. Kristina Strother-Garcia was a companion in computation and in the whirlwinds of change. It was also my pleasure to smuggle for Michael Strother-Garcia some Turkish delights, which ironically don't taste that different across continents. Mai Ha Vu was one of the first deep connections I made at Delaware. Her house was a home to many cheerful parties and somber reflections, which inevitably continued for years after. Hung-Shao Cheng always wanted to unite people and friends – if only we got to have more Manhattan manicures. That I shall regret. Stefan Bartell was a pleasure to be wacky with, with his many cats and oatmeal breakfasts. I thank Irene Donovan for cleaning up the messes that Stefan framed me for.

At Delaware, I got to start off my journey into Armenian linguistics. Irene Vogel was instrumental as an early push, whether for a term paper or to churn out data for the lab. Antranik Dakessian was kind to host me in my first ever research endeavour back at Haigazian University. Nikita Bezrukov became an unexpected friend and fellow Armenologist, in the far away land of Pennsylvania. His diligence and passion for fieldwork led me to Tabita Toparlak, who knew how to make someone like me not get bored in Paris. I'm thankful to Anaïd Donabédian and Pollet Samvelian for making my Parisian experience possible in the first place, and for many pleasant discussions in French which I did not understand, but which I still enjoyed to hear.

But of course, my chapter at Delaware (partially) ended as I made my way to Stony Brook, with the ever-repeated joke that I was in Jeff's moving truck. I would not have survived the economic infrastructure of Long Island without Abraham Kohrman's car and crash-space. Lori Repetti was kind and sympathetic as I made my way through this new island. John Bailyn though was persistent in trying to get the system to work

for me. It's difficult to summarize in a pithy sentence how much warmth and frivolity was on the second floor of SBS. Sedigheh (Sophie) Moradi was always there with an infectious smile and tea. Ali Salehi would sit at his desk as I eavesdropped on his Kurdish ballads. Ji Yea Kim would share her moonpie cookies as we talked about IPA mouth diagrams. Nazila Shafiei could host the best tea-party and bureaucratic meeting. Hyunah Baek would make the hunt for beignets ever the more fun. Grace Wivell's cat dresses echoed the kindness which she exuded. Andrija Petrovic and Seoyoung Kim knew how to create parties out of nothingness (which I sadly often couldn't go to). Yaobin Liu, Khanin Chaiphet, and Enas Albasiri knew how to make late-night office work become more tolerable as the rest of floor went home. Paola Cépeda and Honaidah Ahyad were the wiser upper-years who still knew how to make you feel at home. So Young Lee was the ever-present office neighbor, whose presence made nightly walks back home more enjoyable in the cold. Sabine Laszakovits was a spiritual comrade in commuting, and in exploring odd rooms. Aniello De Santo had the honor of being my orientor, and in orienting me over the local 'food'. Alëna Aksënova was late in the game for me, but I cherish her directness as we get confused over an oddly made smoothie. Ayla Karakaş was a pleasure to befriend as we shared grunts over reduplication. Jon Rawski gave me refuge on his office couch, and I never left. I believe it's mutual. Conferences themselves brought their own collection of conference buddies: Samuel Andersson, Aleksei Nazarov, and Sophie Hao. They and many others made the effort of traveling worth it. All of this people and more were a pleasure to befriend and suffer the drudgeries of graduate school with.

It hits me that, whether I like it or not, those four chapters of my life are both over and not finished. They still come with me as I proofread the last few bits of this dissertation. The future is largely uncertain, but I'm glad to know where and who it will be with. My final gratitude is to Daniel, for his endless supply of hugs and shakshouka. It was worth waiting in No Man's Land, eating Chinese food on dorm ledges, and weekly commuting. I hope I'll always get to make you oroukh, even though I can't make much of anything else.

On a final note...

Contents

II   Computational locality of cyclic phonology   97

4   Logical notation and representations   99

Chapter 1

Introduction

The title of this dissertation indicates its goal: to compute cyclic phonology in Armenian. As a general context, this dissertation asks the following fundamental question: *What principles govern the alternation in the pronunciation of morphemes?* There is a wide history of research on this topic (Scheer 2011). To answer this question, I undertake an empirical investigation into Armenian morpho-phonology. With this empirical background, I develop a computational model that formalizes various aspects of the morphology-phonology interface. The tool I use is formal logic, specifically graph-to-graph logical transductions. The formalism is a generalization of the work from one-level Declarative Phonology to a two-level framework.

On the empirical side, Armenian is understudied but it is rich in interactions between phonology and other modules. I show that Armenian requires a model of the interface which is interactionist and cyclic, i.e., that morphological structure, prosodic structure, and phonological rules cyclically interact to create new words. Evidence for this nuanced organization comes from the stratal phonology of Armenian. There are phonological processes which apply differently before derivational morphology than in inflectional morphology. These processes indicate not only different strata or levels, but also show signs of unbounded cyclicity and sensitivity to sublexical prosodic constituents. These processes are active in both simplex and compound words. Within compounds, the interaction of all these factors creates bracketing paradoxes.

The empirical investigation demonstrates that Armenian phonology is the interaction of four principles: morphology (Selkirk 1982; Dixon and Aikhenvald 2003), prosody (Nespor and Vogel 1986; Selkirk 1986), phonological rule domains (Kiparsky 1982b, 2015), and cyclic organization (Cole 1995a; Bermúdez-Otero 2011). The question now is how are these principles computed both individually and together. Theoretically, there are roughly two extremes of thought: either the interface is computed serially with rules, or in parallel using global information. But despite many controversies in generative phonology (Bromberger and Halle 1989), there are little computational differences between parallelist vs. serialist theories once they are converted into explicit computational systems (Karttunen 1993). Their difference comes from how well they fit into subclasses of computational formalizations such as finite-state machines or regular relations (Heinz 2018). In other words, what matters is whether some theory is computationally more complicated than empirically needed. In the case of segmental phonology, global parallelist computation (as found in OT) is computationally more than what is needed (Chandlee et al. 2018; Strother-Garcia 2019). The bulk of segmental phonology can be locally computed. This dissertation provides the same result for the morphology-phonology interface.

There are many different incarnations of cyclic theories of phonology but there are little to no computational analyses of them. I develop a computational formalization using Monadic-Second Order (MSO) logic. Logic is a flexible tool that lets us create iconic formalizations which faithfully replicate phonological theory. I utilize logical transductions as a way to encode the derivational nature of phonology, i.e., to formalize phonology as a two-level system instead of a monostratal one-level system (cf. Bird 1995). By formalizing the morphology, prosody, and phonological rule domains, I uncover implicit factors within cyclic phonology.

The formalization provides a unified framework where we can examine the expressivity of these different factors. It lets us isolate parts of the theory which are computationally complex from those which are computationally simple. The ultimate takeaway is that I show that the bulk of the interface requires *local* computation, not global computation. The computational result formalizes the intuitions present in many theories of the interface (Embick 2010; Gribanova 2010), and it shows that there is little to no evidence for global computation (cf. Prince and Smolensky 2004). By being local, the computational nature of the morphology-phonology interface opens doors to understanding how we can provably learn morpho-phonological processes and how these processes relate to limitations on human cognition. I do not discuss learnability much in this dissertation, however the computational results have clear ramifications on the relative difficulty of learning morpho-phonology (cf. the learnability of local segmental phonology Ellison 1994; Heinz 2007; Chandlee 2014).

This introductory chapter gives an overview of the morpho-phonology of Armenian (§1.1), with a focus on destressed high vowel reduction. The overview sets the empirical background for the computational formalization of cyclic phonology.[1] In §1.2, I go over what must be computationally defined as part of a cyclic interactionist system for the morphology-phonology interface. As a computational tool, I do not use finite-state mechanisms because they are designed for linear inputs, not hierarchical structure (§1.3). I instead use formal logic and logical transductions within the general framework of two-level Declarative Phonology (§1.4). Finally in §1.5, I preview the results of the dissertation and give a guide to the dissertation, with a focus on computing the generative capacity of the interface. In the appendix, I provide a concise summary of different strands of work in computational or mathematical phonology.

---

[1]Data is collected from the grammars cited in the bibliography, dictionaries from www.nayiri.com, Wiktionary, and my own native (Western) judgments. Glosses are taken from Armenian-English dictionaries if available, otherwise my own translation. Data is transcribed in IPA. The tap is transcribed as /r/ while the trill is /r̄/ and the lax mid-vowels /ɛ,ɔ/ are transcribed as /e,o/. In Western Armenian, voiceless consonants are aspirated. I do not mark aspiration because it is not contrastive. Armenian citations are Romanized based on the ISO 9985 transliteration system. Glossing follows the Leipzig standards. The glosses which I use are: ABL ablative, ACC accusative, AOR aorist, DAT dative, DEF definite, DIM diminutive, GEN genitive, IMP imperative, INF infinitive, INST instrumental, LOC locative, NMLZ nominalization, NOM nominative, PL plural, POSS possessive, PRS present, PRTP participle, PST past. I refer to Classical and Modern Armenian as separate lects. Modern Western and Eastern Armenian are separate dialects or lects. The three form three lects.

1.1 Empirical landscape of the interface in Armenian

Armenian is an understudied Indo-European language. It is a primarily-suffixing, agglutinative language with two standard dialects: Standard Western and Standard Eastern Armenian, and almost 40 attested non-standard dialects of varying degrees of mutual intelligibility. In this thesis, I analyze two morphophonological processes in depth: destressed high vowel reduction and compound bracketing paradoxes. The outcome requires a cyclic interactionist model which has multiple strata (levels, cophonologies) and which references sublexical prosodic constituents. In this chapter, I give a brief overview of these processes and their factors.[2]

1.1.1 Destressed vowel reduction and cyclicity

In Armenian, stress regularly falls on the word's rightmost full vowel (1). This vowel can be part of the root (1a), a derivational suffix (1b), or an inflectional suffix (1c) as long as it not a schwa (1d).

(1) a. kórd͡z   'work'
     b. kord͡z-avór   'worker'
     c. kord͡z-avor-nér   'workers'
     d. kord͡z-avor-nér-ə   'with workers'

Although primary stress appears only once on the surface,[3] there is evidence that stress is being actively assigned and reassigned cyclically as each suffix is added. The evidence is the reduction of destressed high vowels to a schwa (2a) or nothing (2b).

---

[2]To give a larger empirical context, there is limited work on Armenian. To my knowledge, this dissertation is the first to focus on the morphophonology of Armenian. Most linguistic descriptions and analyses of the standard dialects are written in Armenian (Ačaṙyan 1971; Xačatryan 1988). Outside of Armenia, there are some structuralist grammars (Fairbanks 1948; Johnson 1954), language maintenance/attrition studies (Davidian 1987; Godson 2004; Karapetian 2014; Al-Bataineh 2015), sociolinguistic studies (Donabédian 2001a, 2018), and descriptive or teaching grammars (Gulian 1902; Kogian 1949; Bardakjian and Thomson 1977; Minassian 1980; Andonian 1999; Hagopian 2005).

There is a small but growing set of in-depth generative and non-generative work on Modern Armenian syntax-semantics (Seropian 1968; Haig 1980; Comrie 1984; Donabédian 1991; Tamrazian 1994; Sigler 1997; Ackerman 1998; Ackerman et al. 2004; Ackerman and Nikolaeva 1997, 2014; Dum-Tragut 2009; Megerdoomian 2009; Yeghiazaryan 2010; Kahnemuyipour and Megerdoomian 2011, 2017; Su 2012; Khanjian 2013; Giorgi and Haroutyunian 2016, 2019; Ouwayda 2017; Hodgson 2019; Sağ 2019), morphology-semantics (Donabédian 1993, 2001b; Kozintseva 1995; Bale and Khanjian 2008, 2014; Bale et al. 2010, 2011; Haroutyunian 2011; Giorgi 2011; Donabédian 2016; Martí 2020) phonology-phonetics (Kassabian 1971; Vaux 1998b; Hacopian 2003; Haghverdi 2016; Hovakimyan 2016; Seyfarth and Garellek 2018; Toparlak 2019; Skopeteas 2019), and morphology (Donabédian 1997; Baronian 2006; Boyacioglu 2010; Wolf 2013; Arregi et al. 2013; Daniel and Khurshudian 2015; Bezrukov 2016; Oyer 2017; Plungian 2018). There are many descriptive studies on Classical Armenian and etymology (Godel 1975; Thomson 1989; Clackson 1994; Kortlandt 2003; Ravnæs 2005; Martirosyan 2009; Olsen 2011, 2017; Macak 2017; Sayeed and Vaux 2017; Klein et al. 2017), with some in-depth generative and non-generative treatments (Connolly 1972; Hammalian 1984; Vaux 1994; Halle and Vaux 1998; Garrett 1998; Pierce 2007; Caha 2013; Meyer 2013; DeLisi 2015; Macak 2016; Balabanian 2019).

As for the non-standard dialects, most work is descriptive surveys or sketches that primarily focus on sound changes from Classical Armenian (Greppin and Khachaturian 1986; Weitenberg 2002, 2017; Martirosyan 2019). There is a growing body of descriptive and generative work on these non-standard dialects, mostly byVaux (1992, 1993, 1995a,b, 1996a,b, 1997, 1998a,b, 2000, 2001a,b, 2002, 2003, 2006, 2007, 2008, 2012, 2013) and others (Khachaturian 1983, 1992; Vaux et al. 1996; Halle et al. 2000; Fitzpatrick et al. 2004; Finley 2008a,b; Schirru 2012; Bezrukov 2016; Hodgson 2019; Bezrukov and Dolatian 2020).

[3]Armenian is reported to have secondary stress on the initial syllable (Vaux 1998b), but its acoustic cues are weak. Secondary stress likewise does not affect vowel reduction synchronically; though see DeLisi (2015) for evidence of the diachronic role of initial secondary stress.

(2) a. hín   'old'   b. teɣín   'yellow'
         hən-utjún   'oldness'   teɣn-orág   'yellowish'

In terms of its phonological factors, destressed high vowel reduction (DHR) targets only *high* vowels which surface as unstressed in the output (3a). Low and mid vowels do not reduce (3b).[4].

(3) a. makúr   'clean'   badíʒ   'punishment'
         makr-él   'to clean'   badʒ-él   'to punish'
     b. had͡ʒáχ   'frequent'   darpér   'different'   ʒoɣóv   'assembly'
         had͡ʒaχ-él   'to frequent' darper-él   'to distinguish' ʒoɣov-él   'to collect'
         *had͡ʒχ-él   *darpr-él   *ʒoɣv-él

Second, not just any unstressed high vowel in the output will reduce. Although it is cross-linguistically common for vowel reduction to target any unstressed vowel or any unstressed high vowel (Crosswhite 2001), the same cannot be said for Armenian. A high vowel will reduce only if it was stressed at some point of the derivation but subsequently lost stress, i.e. it is *destressed* (4).

(4) a. amusín   'husband'   b. irigún   'evening'
         amusn-utjúl   'marriage'   irign-ajín   'evening (adj.)'
         *amsin-utjún   *irgun-ajín

To illustrate this point, consider the example *amusín* (4a) which consists of three high vowels. In the underived base word, none of the high vowels are reduced and stress is on the final vowel. However, once a suffix *-utjun* is added onto the base, stress will shift onto the suffix and the derivative will be *amusn-utjún*. The root's final high vowel will have lost stress and be reduced in the derivative. The other high vowels in the base do not reduce: *amsin-utjún*. The same destressing process is observed in *irigún* vs. *irign-ajín* (4a).

Finally, just as stress assignment is cyclic, so is DHR. Vowel reduction can apply multiple times to a sequence of destressed high vowels (5a), can apply to suffixes (5b), and can apply in compounds (5c). This is a case of unbounded cyclicity (Orgun 1994) where every new morpheme can potentially trigger a new cycle of stress shift and reduction.

(5) a. d͡zín   'birth'   b. ázk   'nation'   c. kír   'handwriting'
         d͡zən-únt   'birth'[5]   azk-ajín   'national'   kər-ít͡ʃ   'pen'
         d͡zən-ənt-agán 'generative'   azk-ajn-utjún 'nationalist'   dúp   'box'
                                                                        kər-t͡ʃ-a-dup   'pencil-box'

Thus, DHR needs access to a word's derivational history and it can occur multiple times in a word's derivation. The analysis requires a **cyclic**, bottom-up, recursive computation. This requirement can be

---

[4]A closed set of words have the vowel /e/ undergo destressed reduction to [i]: sér→[sir-él] 'love'→'to love'. These are discussed in chapter 2.

[5]The word *dzən-unt* is a common word which is diachronically bimorphemic and derived from the root *dzín*. The root *dzín* is rarely if ever used in isolation. It is commonly found in its derivatives and deverbal compounds involving birth: *dzən-i-l* 'to be born', *asdvad͡z-a-d͡zin* 'Virgin Mary (God-√birth)'.

modeled with unbounded stem-level cycles as in traditional Lexical Phonology (Kiparsky 1982b), with recursive constraint evaluation as in Trans-derivational Correspondence Theory (Benua 1997), or with phase-based derivations as in Phonological Derivation by Phase (Newell 2008; Scheer 2011, 2012; Samuels 2011). Interestingly, Armenian likewise has destressed diphthong *uj* reduction to *u*. Like DHR, diphthong reduction reduces *destressed* diphthongs. This reinforces the role of derivational histories in Armenian.

| (6) | a. | zərújt͡s | 'conversation' | b. | hampújr | 'kiss' |
| | | zərut͡s-él | 'to converse' | | hampur-él | 'to kiss' |

1.1.2 Destressed vowel reduction and morphological strata

However, cyclicity is not enough. DHR is likewise sensitive to **morphological** structure. In the examples above, the relevant suffixes triggered stress shift *and* DHR. However, although all types of suffixes can trigger stress shift, not all of them can trigger DHR. In Western Armenian (WArm), DHR is triggered by derivational suffixes (7a-i) but not inflectional suffixes (7a-ii). Interestingly, diphthong reduction (DDR) is likewise triggered by derivation (7b-i) but blocked by inflection (7b-ii).

(7)	a.	i.	amusín	'husband'	b.	i.	zərújt͡s	'conversation'
			amusn-ut͡jún	'marriage'			zərut͡s-él	'to converse'
		ii.	amusin-óv	'husband-INST'		ii.	zərujt͡s-óv	'conversation-INST'

The morphology triggers different **phonological rule domains**, which I model with lexical strata (also called levels, cophonologies). I argue that derivational morphology creates morphological stems (8b, MStems) while inflectional morphology creates morphological words (8c, MWords). MStems trigger the stem-level phonology, i.e., a set of rules which include stress shift, high vowel reduction, and diphthong reduction. In contrast, MWords trigger the word-level phonology which only includes stress shift, but not any reduction process. Free-standing roots (8a) take covert category suffixes (Giegerich 1999; Marantz 2007). The features of suffixes are glossed; they are repeated as subscripts on MStems and MWords.

$$\text{MS}_n$$

(8) a. /amusin -∅/ b. /amusin -∅ -utjun/ c. /amusin -∅ /-ov/
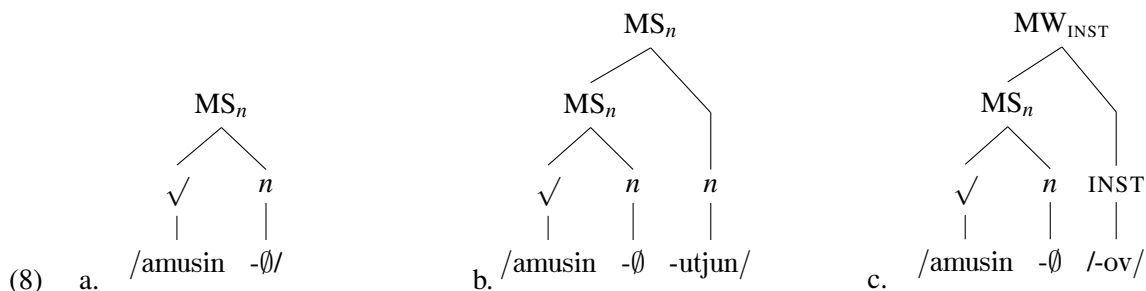
The serial derivation below illustrates the use of strata in order to derive *amusn-utjun* (7a-i) and *amusin-ov* (7a-ii). The analysis is couched in a simple interactionist model like Lexical Phonology. The derivation involves rounds of morphology and phonology. Phonological rules are applied as part of some cophonology, whether the stem-level or the word-level. Shaded cells represent inapplicable cycles or steps.

(9)   *Serial lexical-phonology spell-out for* amusn-utjún (7a-i) *and* amusin-óv (7a-ii)

				MS_n	$\mathrm{MW}_{\mathrm{INST}}$
Input				/amusin -∅ -utjun/	/amusin -∅ -ov/
Cycle 1	MORPHO		Spell-out	amusin -∅	amusin -∅
	PHONO	*SLevel*	Stress	amusín	amusín
		DHR			
Cycle 2	MORPHO		Spell-out	amusín -/utjun/	
	PHONO	*SLevel*	Stress	amusin-utjún	
		DHR		amusn-utjún	
Cycle 3	MORPHO		Spell-out		amusín -/ov/
	PHONO	*WLevel*	Stress	amusn-utjún	amusin-óv
Output				[amusn-utjún]	[amusin-óv]

1.1.3   Destressed vowel reduction and prosodic structure

This simple picture is complicated once we look at other dialects. In Eastern Armenian (EArm), DHR is likewise sensitive to the **prosodic** structure of the inflectional suffix. DHR is triggered by derivational suffixes (10b) and vowel-initial inflectional suffixes (V-Infl, 10c), but not by consonant-initial inflectional suffixes (C-Infl, 10e).

(10)   a.   amusín   'husband'
       b. Derivation   amusn-utjún   'marriage'   WArm & EArm
       c. V-initial Infl.   amusn-óv   'husband-INST'   EArm
       d. V-initial Infl.   amusin-óv   'husband-INST'   WArm
       e. C-initial Infl.   amusin-nér   'husband-PL'   WArm & EArm

Table 11 below summarizes the morphological and prosodic dichotomies for DHR across the two dialects.

(11)   *Summary of morphological factors of DHR in Western and Eastern Armenian*

Dialect	Derivation	V-initial Inflection	C-initial Inflection
Western Armenian	✓	✗	✗
Eastern Armenian	✓	✓	✗

Crucially, it is not the case that Eastern V-Infl can trigger *any* reduction process. In both dialects, destressed diphthongs reduce in derivation (12a-i) but not inflection (12a-ii).

(12) a. i. zərújts͡ʰ   'conversation'   b. i. hambújr   'kiss'
              zəruts͡ʰ-él   'to converse'   hambur-él   'to kiss'
          ii. zərújts͡ʰ-óv   'conversation-INST'   ii. hambujr-óv   'kiss-INST'
              zəruts͡ʰ-nér   'conversation-PL'   hambujr-nér   'kiss-PL'

The overapplication is DHR is triggered by *vowel*-initial inflection suffixes, not C-initial ones. This points to a prosody-based explanation in terms of syllabification and phonological representation. Within the framework of prosodic phonology, the careful analysis of agglutinative languages has pointed to a sublexical phonological constituent which straddles the boundary between derivation and inflection: the Prosodic Stem or PStem (Downing 2016). The PStem is higher than the foot but below the PWord. I argue that the prosodic misalignment of the PStem is what triggers the unexpected behavior of V-initial inflection in Eastern Armenian. In Chapter 2, I explain why alternative constituents like the foot and PWord are not sufficient.

Specifically, MStems are mapped to non-recursive PStems: *(amusín)ₛ* (13a), *(amusn-utjún)ₛ* (13b). PStems must stay aligned with syllable boundaries. Resyllabification before V-initial inflection causes the PStem to expand: *amusin-óv* (WArm) or *amusn-óv* (EArm). This expansion triggers the PStem-level cophonology which has stress shift without DHR in WArm, but stress shift with DHR in EArm. Before C-initial inflection, the PStem stays aligned with the MStem and syllables (13d).

(13) a. *Root* (10a)   b. *+ Der* (10b)   c. *+ V-Infl* (10d,e)   d. *+ C-Infl* (10e)

              PW   PW   PW   PW
              | | | /\
              PS   PS   PS   PS \
              /\   /\   /\   /\   \
          a.mu.sín   a.mus.n-ut.jún   a.mus(i).n-óv   a.mu.sin -nér

Table (14) illustrates the analysis and shows the derivation for *amusn-ov* (EArm: 10c), *amusin-ov* (WArm: 10d), and *amusin-ner* (WArm & EArm: 10e) across the two dialects. PStem boundaries are marked by (...)ₛ. The derivation involves rounds of morphology, prosody, and phonological rule application. The prosody rounds show syllabification, mapping PStems, and readjusting PStems.

(14)   *Serial lexical-phonology spell-out with prosodic constituents and misalignment for inflected items*

			EArm	WArm	EArm &WArm
			MW_{INST}	MW_{INST}	MW_{PL}
			MS_n INST	MS_n INST	MS_n PL
			$\sqrt{}$ n INST	$\sqrt{}$ n INST	$\sqrt{}$ n PL
			/amusin -∅ /-ov/	/amusin -∅ /-ov/	/amusin -∅ /-ner/
Input			/amusin -∅$_S$ -ov/	/amusin -∅$_S$ -ov/	/amusin -∅$_S$ -ner/
Cycle 1					
MORPHO		Spell-out	/amusin-/	/amusin-/	/amusin-/
PROSODY		Syllabify	amu.sin	a.mu.sin	a.mu.sín
		Map PStem	(amu.sin)$_s$	(a.mu.sin)$_s$	(a.mu.sin)$_s$
PHONO	*SLevel*	Stress	(amu.sín)$_s$	(a.mu.sín)$_s$	(a.mu.sín)$_s$
		DHR			
Cycle 2					
MORPHO		Spell-out	(a.mu.sín)$_s$ - /-ov/	(a.mu.sín)$_s$ - /-ov/	(a.mu.sín)$_s$ - /-ner/
PROSODY		Syllabify	(a.mu.sí.n)$_s$-ov	(a.mu.sí.n)$_s$-ov	(a.mu.sín)$_s$-ner
		Adjust PStem	(a.mu.sí.n-ov)$_s$	(a.mu.sí.n-ov)$_s$	
PHONO	*PStem-level*	Stress	(a.mu.sĭ.n-óv)$_s$		
		DHR (EArm)	(a.mus.n-óv)$_s$		
	WLevel	Stress		(a.mu.si.n-óv)$_s$	(a.mu.sin)$_s$-nér
Output			amusn-óv	amusin-óv	amusin-nér

In Cycle 1, the root *amusin* is spelled-out and goes through the stem-level cophonology to get stressed. Here, the MStem maps onto a PStem: *(amusin)$_s$*. In Cycle 2, the inflectional suffixes are spelled out and syllabified. The V-initial suffix *-ov* syllabifies with the stem, while the C-initial *-ner* does not.

PStems must be aligned with syllable boundaries. Before C-initial inflection, the PStem stays well-aligned with both the MStem and syllable boundaries: *(amusín)$_s$-ner*. It undergoes the word-level cophonolgy of stress shift without reduction: *amusin-nér*.

But before V-initial inflection, resyllabification makes the PStem become misaligned from syllable boundaries: **(amusí.n)-ov*. This is repaired by PStem expansion into the inflectional suffix: *(amusín-ov)$_s$*. The expansion of the PStem triggers the PStem-level cophonology. In Western Armenian, the PStem-level cophonology includes stress shift but not reduction: *(amusin-óv)$_s$*. In Eastern Armenian, the PStem-level includes stress shift, high vowel reduction, but not diphthong reduction: *(amusn-óv)$_s$*. The word-level phonology then applies for all inflected items: *amusn-óv, amusin-óv*.

The Armenian data thus provide evidence for combining cyclicity, prosodic constituents, morphological structure, and phonological rule domains. The analysis for vowel reduction necessitates using lexical phonology's concepts of cyclicity and strata, and prosodic phonology's concept of phonological constituents and non-isomorphism. In Part II, I provide a computational formalization for the different pieces of the analysis: cyclicity, morphological structure, prosodic structure, and phonological rule domains. The next section shows how these same principles arise in compounds.

1.1.4 Emergence of a bracketing paradox in compounds

The behavior of vowel reduction dissects different principles in Armenian phonology. These principles conspire in compounds to create a bracketing paradox. In general, Armenian compounds are formed by combining two word-like morphological units with a linking vowel -*a*-: an͡tsrev-a-t͡ʃúr (15a).

(15) a. an͡tsrév + t͡ʃúr   'rain + water'   b. t͡ʃár + sírd   'evil + heart'
          an͡tsrev-a-t͡ʃúr   'rain-water'   t͡ʃar-a-sírd   'evil-hearted'

The output of compounding has primary stress on the final full vowel. The presence of only one primary stress shows that a compound forms at most one phonological word in Armenian.

Although the above description looks straightforward, Armenian compounds exhibit a bracketing paradox where we see a mismatch between the morphological and phonological structures. The paradox is found in plural formation. In simplex nouns, the plural is formed by adding the suffix -*er* after monosyllabic bases (16a-i), -*ner* after polysyllabic bases (16a-ii). In some compounds, the plural counts the number of syllables in the entire compound: ʃar-a-sird-*ner* (16b-ii). But in other compounds, the plural only counts the number of syllables in the *second stem*: an͡tsrev-a-t͡ʃur-*er* (16b-i). I underline the domain of syllable counting.

(16) a. i. pág   'yard, lot'   ii. panág   'army'
              pag-ér   'yards, lots'   panag-nér   'armies'

      b. i. an͡tsrev-a-t͡ʃúr   'rain-water'   ii. t͡ʃar-a-sírd   'evil-hearted'
              an͡tsrev-a-t͡ʃur-ér 'rain-waters'   t͡ʃar-a-sird-nér   'evil-hearted people'

Within Armenian linguistics, the *existence* of this bracketing paradox is well-known (Vaux 1998b; Dum-Tragut 2009). However, there is relatively little theoretical attention on understanding the morpho-phonological factors, consequences, and correlates for compound formation in Armenian. In Chapter 3 , I fill this gap. I show that the paradox is largely due to endocentricity. The plural counts the number of syllables in the *semantic head*. If the compound is exocentric, then the plural counts the number of syllables in the entire compound (16b-ii); while if the compound is endocentric, then the plural counts the number of syllables in the second stem (16b-i). I model this paradox with cyclic head-operations (Hoeksema 1984; Aronoff 1988). I show that counter-cyclic tools such as rebracketing (Sproat 1985; Marantz 1988) would contradict the **cyclic** phonology of compounds. This is because compounding triggers the same set of stem-level rules as derivational morphology, e.g., DHR: *jergír* and *jergr-a-kúnt* (17a-i). Thus, by understanding how compounding creates the same **phonological rule domains** as derivational morphology, we weed out alternative analyses. The cyclic phonology applies regardless whether the compound has a paradoxical plural (17a-ii) or not (17b-ii).

(17) a. i. jergír + kúnt   'Earth + sphere'   b. i. aznív + sírd   'sincere + heart'
              jergr-a-kúnt   'globe'   aznəv-a-sírd   'sincere-hearted'
          ii. jergr-a-kunt-ér 'globes'   ii. aznəv-a-sird-nér 'sincere-hearted (PL)'

As an additional complication, the bracketing paradox is affected by the **prosodic** structure of bisyllabic compounds. Unsurprisingly, exocentric disyllabic compounds are always transparently pluralized: *kar-daʃ-ner*

(18a). But the bracketing paradox can variably under-apply when the bisyllabic compound is endocentric: *xat͡ʃ-kar-(n)er* (18b).

(18)   a.   kár + daʃ-él   'stone + to carve'   b.   xát͡ʃ + kár   'cross + stone'
            kar-dáʃ   'stone carver, mason'   xat͡ʃ-kár   'cross-stone'
                                                               xat͡ʃ-kar-ér   'cross-stones'
            kar-daʃ-nér   'stone carvers, masons'   xat͡ʃ-kar-nér

I argue that this variation requires the use of a prosodic constituent as the *prosodic head*, specifically the *Prosodic Stem* (Downing 1999a). I argue that in endocentric compounds, the semantic head *h* maps to a PStem. A PStem can optionally expand in bisyllabic compounds; this causes the plural suffix to count the number of syllables in the optionally expanded PStem instead of just the semantic head. As in DHR, the Prosodic Stem is thus active in different areas of the Armenian grammar. I show that alternative prosodic constituents such as feet or recursive PWords are inadequate.

1.2   Connecting theory and computation

The Armenian data showed that there are four principle factors which govern the phonology of Armenian. These are the language's morphology, prosody, phonological rule domains, and cyclic organization between them. These four principles are sketched out in (19).

(19)   *Sketch of an interactionist model*
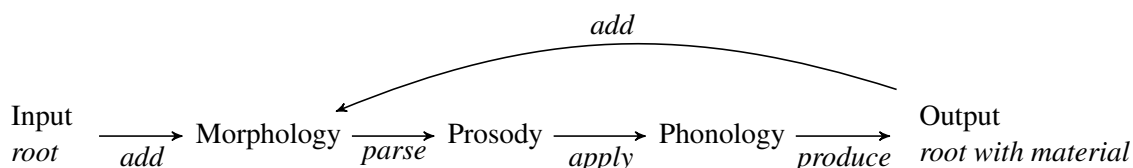
There is a wide typology of theories for the morphology-phonology interface. The above model is an **interactionist** model because the modules in the graph are interleaved together (Kaisse and Hargus 1993): morphology feeds phonology and vice versa. The model is also **cyclic** because the phonological rules reference the output of previous rounds of phonology. Cyclicity is a hallmark feature of early generative grammar (Chomsky and Halle 1968; Brame 1974), while interactionism is a later development, e.g., in Lexical Phonology (Kiparsky 1982b), Stratal OT (Bermúdez-Otero 2018), and Phase-based Phonology (Newell 2008). There is often a clear mutual dependency between morphology and prosodic structure within interactionist models (Hargus 1993; Booij and Lieber 1993). This is in contrast to non-interactionist models (Chomsky and Halle 1968; Halle and Vergnaud 1987a) where all morphological processes precede all prosodic or phonological processes. Even though non-interactionist models don't interleave the morphology and phonology, these models are still cyclic because the phonology is processed in cyclic chunks

Between these two extremes of theoretical thought, there are theories which combine some aspects of interactionism and non-interactionism, e.g., Stratal OT has a cyclic stem-level rules but post-cyclic word-level rules. Individual theories may add additional principles or stipulations such as the Strict Cyclicity Condition (Kean 1974; Mascaró 1976), Structure Preservation Myers (1991), among others. Individual

theories likewise diverge over what representations they use for the three principle components, such as if the morphology uses trees (Halle and Marantz 1993) or feature sets (Stump 2001), if the prosody allows recursive prosodic constituents (Selkirk 2011) or not (Nespor and Vogel 1986), and if the phonological rule domains reference morphological constituents (Inkelas 2014) or morphemes (Pater 2007).

But going beyond these differences, the interactionist model is an essentially common framework, and many theories of the morphology-phonology interface are couched in it. The questions now are how we *compute* the interactionist model, and what we can learn from the computation. In Parts II and III, I answer this question and develop a large-scale computational formalization for cyclic interactionist phonology. I formalize the essential factors of cyclic interactionist phonology, i.e., the four factors of morphology, prosody, phonological rule domains, and cyclic organization. Given a computational formalization, I then evaluate the model and show that the bulk of the morphology-phonology interface is computationally *local*.

1.3 Problems in computing the interface

Although cyclic phonology is a common theoretical tool, there has historically been little computational formalizations of it (Sproat 1992b:108). To my knowledge, the earliest and most extant formalization is Williams (1993) who develops a software package in Prolog for running a lexical-phonology derivation (Williams 1993, 1994; Williams et al. 1989). In terms of learnability, the most sophisticated learning algorithm for cyclic phonology is Nazarov and Pater (2017) which is couched in Stratal OT. In this section, I explain that this bleak picture is partially affected by the use of finite-state calculus which is the most common computational formalism for morpho-phonology. In the next section, I discuss how alternative logic-based formalisms do not have this problem.

1.3.1 Finite-state formalization of phonology and morphology

The most common tools in computing morphology and phonology are finite-state acceptors (FSAs) and finite-state transducer (FST). The use of finite-state machines (FSMs) has a long history in computational phonology. FSMs were applied early on in computing linguistic structure (Chomsky 1956), yet were found to be inadequate for computing syntactic structure (Chomsky 1957; Miller and Chomsky 1963; Levelt 1974). However Johnson (1972) showed that virtually any SPE-style phonological rule can be converted into an FST. His result was independently echoed by Kaplan and Kay (1994). FSTs quickly became a common formalism for computing phonological processes (Kornai 1995, 2007; Gildea and Jurafsky 1996), with ample implementational and software tools (Mohri 1997; Roche and Schabes 1997; Hulden 2009a).

FSTs were likewise found to be largely adequate for computing morphological processes (Koskenniemi 1984; Sproat 1992b; Beesley and Karttunen 2003; Roark and Sproat 2007). However, while all attested phonological processes are definable by finite-state calculus, most but not all morphological processes are finite-state. FSTs can define partial reduplication but not total reduplication (Culy 1985).[6] Furthermore, although generating a string of morphemes is within finite-state power, generating the semantic bracketing of a word is argued to not be finite-state because of center-embedding and long-distance dependencies

---

[6]As a technical clarification, this present discussion concerns 1-way FSTs which process the input in one diection. These are the types of FSTs generally used in computational linguistics. In contrast, 2-way FSTs can read the input in multiple directions (Filiot and Reynier 2016) and are adequate for computing reduplication (Dolatian and Heinz 2018b).

created by the use of prefixes and suffixes (Langendoen 1981; Selkirk 1982; Carden 1983; Hammond 1993; Oseki 2018; Oseki et al. 2019).

Putting the semantics of morphology aside, morphological functions are generally finite-state *regardless* of the differences between many morphological theories, including item-and-arrangement models (Beesley and Karttunen 2003), item-and-process models (Roark and Sproat 2007), realizational models (Karttunen 2003), Distributed Morphology (Ermolaeva and Edmiston 2018), among many others. There are little to no computational differences among different morphological theories (Roark and Sproat 2007:ch.3).

One productive line of inquiry in finite-state approaches to morpho-phonology is understanding the **generative capacity** of morpho-phonology by defining what *subclasses* of finite-state calculus are needed to compute morphology and phonology (Rogers and Pullum 2011; Jäger and Rogers 2012; Rogers et al. 2013; Chandlee 2014; Chandlee and Heinz 2017; Heinz 2018). This research program on subregular finite-state phonology and morphology has wide empirical coverage. I provide a concise summary or state-of-the-art on subregular phonology in §1.A.1.

Thus, it is clear that finite-state devices and calculus provide valuable insight on the computational nature of phonology and morphology. However, this thesis does not utilize finite-state calculus in formalizing the morphology-phonology interface. The next section explains why this is so.

1.3.2   Problems in finite-state phonology and morphology

Although finite-state tools are versatile and efficient ways to compute phonology and morphology, these tools are designed for *linear* systems. They cannot process an input which is hierarchical with multiple levels of hidden structure, i.e., morphological structure, prosodic structure, and cyclic derivations.[7] In order for an FST to compute an input which has some hierarchical structure, all this hierarchy must be flattened down and replaced with special boundary symbols. For example, the prosodic structure in the word *banána* (20a) has to be replaced with special symbols denoting the left- and right-boundaries for syllables, feet, and prosodic words (20b).

(20)   a.

b. $(_w(_\sigma \text{ba})_\sigma(_\Sigma(_\sigma \text{ná})_\sigma(_\sigma \text{na})_\sigma)_\Sigma)_w$
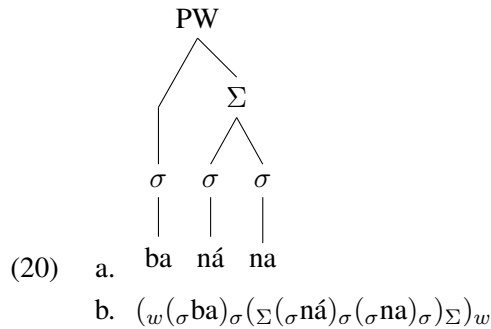
By using boundaries, FSTs can efficiently compute different morphological and prosodic processes, such as for syllabification and word-level prosody (Kiraz and Möbius 1998; Yap 2006; Hulden 2006; Idsardi

---

[7]The closest tools are finite-state tree transducers (Comon et al. 2007) which I do not consider. However, these might not be fully adequate because morphophonological structure is like a directed acyclic graph, not a single tree. This is because 1) morphological and prosodic trees are separate representations 2) with separate tree roots, but 3) they dominate the same segments, and 4) morphological nodes can have directed edges to prosodic nodes.

2009; Yu 2017), phrasal prosody or intonation (Reich 1969; Pierrehumbert 1980; Yu and Stabler 2017; Yu 2019), tone (Gibbon 1987, 2001a; Yli-Jyrä 2013, 2015; Yli-Jyrä 2019), prosodic morphology and partial reduplication (Walther 1998, 1999, 2000; Cohen-Sygal and Wintner 2006; Hulden and Bischoff 2009), speech synthesis (Laporte 1997), and debugging (Karttunen 2006a,b; Hulden 2017). Some even use context-free grammars (CFGs) to iconically capture morphological and prosodic structure (Church 1983; Cole and Coleman 1992; Coleman 1995a; Coleman and Pierrehumbert 1997; Chew 2003); though some of these formalization add various restrictions to the CFGs so that they can only generate regular languages (cf. Church 1983). Many of these case studies are found in Declarative Phonology (§1.4, §1.A.2).

Although FSTs can work with these boundaries, it is an open question if these computational models act as iconic or faithful replicas of linguistic theory. For example, they may require placing a bound on the depth of the tree, remove transformations, require trees to all be right-branching, among other stipulations. In fact, certain processes seem to use local information when computed over these hierarchical structures. However, by being flattened, what is local in the tree can be non-local in the string. Because of this difference, it is likely that the expressivity of morpho-phonological processes when given hierarchical inputs is *different* than when given flattened inputs. For example, when tonal processes are defined in terms of strings, they can be computationally more complex, less local, or more expressive than when these same processes are defined in more iconic representations like autosegmental grammars (Jardine 2016c).

1.4 Formalizing the interface with formal logic

Because of the aforementioned problems, I do not use FSTs to computational define cyclic phonology. Instead, I compute the interface by using formal logic, specifically MSO graph-to-graph transductions. I use logic because it is versatile enough to directly and faithfully encode linguistic representations. It is explicit enough to test for correctness, and it is systematic enough to measure its complexity or generative capacity.[8]

In syntax, there is substantial work on logical formalisms and model-theoretical syntax (Rogers 1997, 1998; Pullum and Scholz 2001; Morawietz 2003; Kobele 2006; Pullum 2007; ter Meulen 2012; Graf 2013).[9] This thesis is the not the first application of formal logic to morphology or phonology. The most developed research program for applying logic to phonology was Declarative Phonology or DP (Bird 1995; Coleman 1998). DP was a one-level system which did not utilize transformations between inputs to outputs. As a constraint-based formalism, its heyday was in the early 1990s but it virtually disappeared since the rise of OT (Prince and Smolensky 2004). In recent years, there has been a program of logical and model-theoretic approaches to phonology (Potts and Pullum 2002; Graf 2010b; Vu et al. 2018; Strother-Garcia 2019; Chandlee and Jardine 2019b; Danis and Jardine 2019). This program utilizes logical transductions in order to transform an input structure to an output structure. The recent program is still in its infancy and does not have a name. For ease comparison, I call this recent program two-level Declarative Phonology or 2-level DP.

This thesis utilizes the tools of 2-level DP in order to formalize cyclic phonology. In this section, I first explain how logic can be used to formalize phonological processes, both in traditional 1-level DP and in

---

[8]To clarify, I do not argue thatlogical treatments of phonology are in general better than finite-state treatments of phonology. They are two points on a continuum of computational work on phonology (Heinz prep).

[9]Similar to subregular phonology, there is likewise a recent research program on investigating what computational subclasses are needed to define syntactic-semantic processes, especially by projecting strict-locality and tier-based strict-locality to trees (Laszakovits 2018; Vu 2018, 2019; Vu et al. 2019; Graf and Shafiei 2019a; Graf 2019b; Ikawa et al. 2020; Shafiei and Graf 2020; Graf and De Santo 2019; Ji and Heinz 2020).

contemporary 2-level DP (§1.4.1). I then explain the reason why 1-level and 2-level diverge in their formal power (§1.4.2).

1.4.1   Use of logical formalizations in phonology

This section has two goals. I explain how logic can be used to formalize phonological processes, and I show how 1-level and 2-level DP diverge in their formalizations. Briefly, 1-level DP treats all phonological processes or alternations as well-formedness statements on possible surface words. There is no input-to-output transduction. In contrast, 2-level DP allows the use of logical functions or logically-defined transductions from an input structure to an output structure. The difference between 1-level and 2-level DP is similar to the dichotomy between feature-filling vs. feature-changing phonology.

Consider the non-existent language below. On the surface, this language only has the sounds [p,p^h,a]. The aspirated [p^h] and unaspirated [p] appear in complementary distribution: the stop is unaspirated intervocalically, and aspirated elsewhere (21a). Morphemes alternate in aspiration under prefixation (21b) and suffixation (21c). Aspiration is thus allophonic. Because aspiration appears in the least restricted type of environment, the underlying phoneme is aspirated /p^h/.

(21)   a.   p^{h}a   apa   ap^h
         b.   p^{h}a   a-pa
         c.   ap^h ap-a

One possible analysis would posit full specification for the phoneme /p^h/. That is, the phoneme /p^h/ is specified as aspirated or [+aspirated]. A rule of intervocalic de-aspiration (22a) then applies to change [+aspirated] to [-aspirated] intervocalically. This rule is feature-changing. Assume that a voiceless bilabial stop is specified as [+stop], and that there are no other stops in the language. In contrast, an alternative analysis would posit under-specification for the phoneme /p^h/. Underlyingly, this phonemes lacks any specification for aspiration. The phoneme /p/ is underlying neither aspirated [+aspirated] nor unaspirated [-aspirated]. The rules in (22b) inserts a specification for aspiration: [-aspirated] intervocalically, [+aspirated] elsewhere. These rules are feature-filling.

(22)   *Aspiration rule as...*
         a.   Feature-changing: [+stop,+aspirated] → [-aspirated] / V_V
         b.   Feature-filling: [+stop]   → [-aspirated] / V_V
                                                                   → [+aspirated] / elsewhere

1-level and 2-level DP differ in this regard. 1-level DP assumes that all phonological processes are feature-filling while 2-level DP makes no such commitment. Given some 'input' in 1-level DP, a process cannot be defined such that it changes anything about this input whether by deletion, epenthesis, or feature-changing rules. It can only specify information by converting underspecified properties into fully specified properties. In contrast, 2-level DP accepts the full gamut of possible phonological transformations: epenthesis, deletion, metathesis, feature changing, etc.

Because of their different premises, the two models differ in their architecture of phonological processes and in how phonology is computed. 1-level DP is non-derivational and mono-stratal: there is only one level

of representation which is simply 'further specified'. Again, 2-level DP makes no such assumption and can be multi-stratal and derivational with an input level and an output level. 1-level DP can be implemented by finite-state acceptors and by logical statements, while 2-level DP can be implemented by finite-state transducers and logical transductions. I expand on these points below.

To illustrate these differences, consider the logical formulas below. The two traditions posit different possible input forms for the aspirated [pʰa] and unaspirated [apa]. In 2-level DP, the input can be /pʰa/ and /apʰa/. A rule of deaspiration is formalized as a pair of input-to-output functions which generate the output string [pʰa] and [apa]. These functions are shown below.

(23) *Logically-defined functions for aspiration in 2-level DP*

    a. $\phi-\texttt{aspirated}(x^1) \stackrel{\text{def}}{=} +\textsf{aspirated}(x) \wedge \textbf{intervocalic}(x)$

        Some segment x is [-aspirated] in the output if it as [+aspirated] and intervocalic in the input

    b. $\phi+\texttt{aspirated}(x^1) \stackrel{\text{def}}{=} +\textsf{aspirated}(x) \wedge \neg\textbf{intervocalic}(x)$

        Some segment x is [+aspirated] in the output if it as [+aspirated] and not intervocalic in the input

The logical formulas are interpreted as follows. On the surface, a consonant x is unaspirated or [-aspirated] (23a) if it is underlyingly aspirated but intervocalic. It is aspirated or [+aspirated] (23b) if it is underlyingly aspirated and not intervocalic. The notation, font, and additional symbols are further explained in Chapter 4.

In Part II, I use logical transductions from 2-level DP to compute hierarchical outputs from hierarchical inputs. That is, I use these same types of logical formulas to compute cyclic phonology and to generate morphological structure, prosodic structure, and phonological rule domains.

In contrast in 1-level DP, the 'input' is an underspecified representation which encodes its possible allomorphs, e.g., [pʰa] and [apa] are underlyingly /{p,pʰ}a/ and /a{p,pʰ}a/. The input segment is in fact a union of possible realizations. The labial segment can either be unaspirated [p] or aspirated [pʰ] depending on its context. The job of 'rules' in 1-level DP is to filter out these possible realizations. In terms of formal logic, rules in 1-level DP can be stated in terms of relatively simple logical statements such as the following.

(24) *Logical statements for the distribution of aspiration in 1-level DP*

    a. $\textsf{stop}(x) \wedge \textbf{intervocalic}(x) \rightarrow -\textsf{aspirated}(x)$

        If some segment x is a stop and intervocalic, then it must be unaspirated

    b. $\textsf{stop}(x) \wedge \neg\textbf{intervocalic}(x) \rightarrow +\textsf{aspirated}(x)$

        If some segment x is a stop and not intervocalic, then it must be aspirated

The statement in (24a) states that if a stop is intervocalic then it must simultaneously be unaspirated. In contrast, the statement in (24b) states that if a stop is not intervocalic, then it must simultaneously be aspirated. When the input representations and the logical statements are combined, we get the right pronounced forms [pʰa], [apʰa]. I use the term 'simultaneously' to emphasize the fact there is no input-to-output transduction in 1-level DP.

1.4.2   Reasons for the logical split

Both 1-level and 2-level DP are computationally well-grounded formalizations of phonology. This thesis utilizes 2-level DP in order to faithfully replicate the derivational aspects of cyclic phonology. This section takes a step back and explains why 1-level and 2-level DP differ in the first place. Retrospectively, I argue that 1-level DP eschews transformations because of the limitations on computational and mathematical resources at the time.

There is an intimate connection between formal logic and finite state calculus (Büchi 1960; Thomas 1997; McNaughton and Papert 1971; Filiot and Reynier 2016). There are likewise connections between subclasses of finite-state calculus and logical formalisms (Rogers et al. 2013; Rogers and Lambert 2019a,b; Lambert and Rogers 2019) and work on converting between logical formalisms and finite-state calculus (Vaillette 2003, 2004; Hulden 2009b). When computed over strings, the representation and filtering 'rules' of 1-level DP can be modeled by FSAs; while the rules of 2-level DP can be modeled by FSTs.

One reason why 1-level DP assumes feature-filling rules and mono-stratal computation is because this makes 1-level DP computable by string-based FSAs (Bird and Ellison 1994). In general, FSAs are defined in terms of a single level of representation: they accept well-formed strings and reject ill-formed strings. They cannot handle transformations between multiple levels of representation. By treating phonology as mono-stratal and non-transformational, practitioners of 1-level DP aim to make their phonological models be integrated with non-transformational frameworks for syntax, such as Head-Based Phrase-Structure Grammar (Bird 1992; Klein 1993), categorial grammars (Wheeler 1981, 1988; Bach and Wheeler 1981), unification-based grammars with attribute-value matrixes (Scobbie 1991a; Bird 1991b; Bird and Klein 1994), and sign-based grammars (Orgun 1996). However, the fact that 1-level DP sought to use FSAs does not mean that phonological transformations are incomputible. Such computations are instead handled by finite-state transducers (FSTs) which transduce or map an input form to an output form. FSAs are a special type of FST.

Given the dichotomy between FSAs and FSTs, the reason why early DP was 1-level is that, at the time when 1-level DP was developing, there were limited computational resources for computing transformations. These limitations came from two sources. First, in the early 1980s and 90s, there were some feasibility problems in constructing, implementing, or composing large FSTs for natural language (Kaplan and Kay 1994; Karttunen 1993:180). This encouraged the use of FST intersection (Koskenniemi 1983b,a, 1984). However, these implementational limitations for FSTs are largely solved now. Second, the reason why 1-level DP did not use logical transductions is because 1-level DP arose before logical transductions were extensively developed. When DP first started, there was little to no work in theoretical computer science on developing logical transductions. A hallmark paper on defining logical transductions is Courcelle (1994) which was written after the bulk of 1-level DP was set out (Bird 1991a; Ellison and Scobbie 1993). The dissertations which later became Bird (1995) and Coleman (1998) were written before 1994.

In sum, despite their computational differences, both 1-level and 2-level DP are well-defined frameworks. There is substantial earlier work in 1-level DP and recent work in 2-level DP. A summary of work in each tradition is given in §1.A.2 and §1.A.3 respectively. Importantly, a major development in 1-level DP was Chew (2003) who developed a computational model of Russian, with a near complete formalization of Russian morphology, prosody, and rule domains. His formalization was non-derivational and monostratal. This dissertation has a similar goal of formalizing Armenian phonology with many of its non-phonological factors, i.e., morphology, prosody, and rule domains. But unlike Chew (2003), I formalize Armenian phonology as a cyclic, interactionist, two-level declarative model. By doing so, this dissertation is a more

iconic formalization of the morphology-phonology interface as it is commonly conceptualized in generative linguistics.

1.5 Guide to the dissertation: Generative capacity of the interface

By using logical transductions, I formalize a substantial chunk of the morphology-phonology interface. As a theoretical backdrop, I assume a simple interactionist model where the Morphology, Prosody, and Phonological Rule domains have a recursive architecture. The output of the Phonology can act as input to another round of Morphology. This interactionist architecture is what creates cyclic phonology.

(25) *Sketch of an interactionist model*

$$
\begin{array}{c}
\textit{add} \\
\text{Input} \xrightarrow[\textit{add}]{} \text{Morphology} \xrightarrow[\textit{parse}]{} \text{Prosody} \xrightarrow[\textit{apply}]{} \text{Phonology} \xrightarrow[\textit{produce}]{} \text{Output} \\
\textit{root} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textit{root with material}
\end{array}
$$
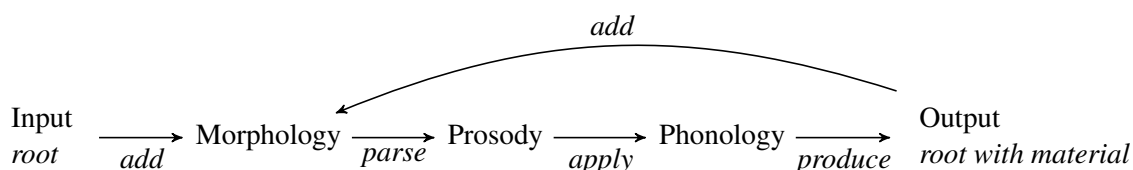
In Part I, I expand on the empirical and theoretical aspects of this model. I document and analyze a large fragment of Armenian morpho-phonology. I expand on the overview of strata, destressed reduction, and compounds given in §1.1. Readers who are interested in what Armenian brings to the theoretical table are encouraged to read these chapters. Computationally-minded readers can skip these chapters. The formalization uses data on Armenian which was already explained in the overview in §1.1.

In Chapter 2, I describe the complications present in destressed high vowel reduction in Armenian. It is a cyclic processes which is present in both Modern Western and Eastern Armenian, and inherited from Classical Armenian. I expand on the how Armenian phonology is organized into a stem-level and word-level strata based on high vowel reduction and other processes. I show evidence for the existence of the Prosodic Stem as a sublexical prosodic constituent which is larger than the foot but smaller than the Prosodic Word.

In Chapter 3, I go over compounding in Armenian and how compounds show a bracketing paradox in plural formation. I document the distribution of the paradox across the Armenian lexicon. I argue that the data requires a cyclic mechanism such as head-operations, and that it cannot be done with common counter-cyclic mechanisms such as rebraketing. By understanding the phonology and prosodic structure of compounds, I then show that the limited variation in compound pluralization is further evidence for the Prosodic Stem.

With this empirical setting in place, Parts II and III formalize the interface. In Part II, I first go over the preliminaries of the logical notation in Chapter 4. I set up the notation to handle the hierarchical structure in morphology and phonology. In Chapter 5, I use the logical notation to define the individual components of the interactionist model. I define logical transductions for computing the **Morphology**, **Prosody**, and **Phonological Rule Domains**. These are exemplified for a simplex free-standing root like *amusin* 'husband'.

In Chapter 6, I exploit the **cyclic** nature of the interface. I define additional transductions for generating derived, inflected, and compounded words. Computing these more complex words requires minimal modifications to the set of Morphological, Prosodic, and Phonological transductions. Crucially, I show that these processes

are computationally local. They are local because they do not need quantifiers, i.e., they have Quantifier-Free computation (Strother-Garcia 2018, 2019). In order to facilitate the logical definitions, I distinguish between the triggers and targets of these linguistic processes. Whereas the target of these processes are segments, the trigger is often the topmost morphological node in the tree. I encapsulate the information about this topmost node into a constant that I call the SETTINGS of the derivation. By doing this encapsulation, I make explicit the fact that the derivation is implicitly guided by regularly examining the input morphological structure. The encapsulation likewise acts as a way to factorize the potentially local target of morpho-phonological processes from the potentially non-local trigger of these processes.

(26) *Sketch of an interactionist model with an explicit stage for the* SETTINGS
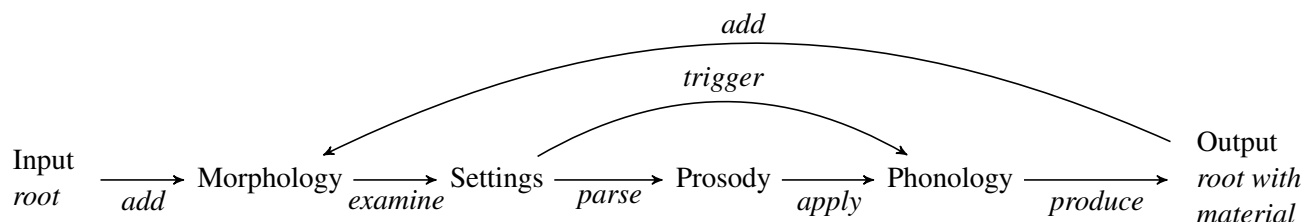
I emphasize that the formalization is meta-theoretical. As a young science, theoretical linguistics is still generating new language theories, and the field has witnessed many significant shifts in theoretical frameworks (Anderson 1985; Scheer 2011; Goldsmith 2012). I have tried to keep the computational formalism as simple as possible by formalizing only the basic aspects of interactionist models. With these basics, I have formalized a substantial chunk of the interface. Table (27) acts as a summary of some of the morpho-phonological processes or principles that I have formalized. A more detailed table is provided in the conclusion chapter §10. I do not formalize theory-specific additions such as Strict Cyclicity, phases, or output-output constraints.

(27) *Aspects of the morphology-phonology interface which are defined*

Morphology	**Prosody**	**Phonological Rule Domains**
Affix linearization	Syllabification	Domains triggered by
Zero	Generating syllables	Morphemes
Prefix	Syllable ordering	Morphological constituents
Suffix	Resyllabification	Prosodic constituents
Mobile Affix	Tiers over syllables	
Affix Allomorphy	Mapping	
Inwards-sensitive	Generating prosodic constituents	
Outwards-sensitive	Misaligning prosodic constituents	
Phono-conditioned	Restructuring prosodic constituents	
Morpho-conditioned	Recursive Prosody	
Tiers over dominance	Generating	
Compounding	Flattening	
Formation	Compound prosody	
Head-marking		

The ultimate goal of Part II is to show not only that we can use logic to define the morphology-phonology interface, but we also then know that the interface is computationally simple and local. By being local, these

results indicate that we need simple computational resources to compute the interface. This opens doors to understanding how the interface can be provably learned and how its computational properties may reflect the cognitive capacities of natural language.

To hammer down on this final point about locality and generative capacity, Part III takes a step back and examines the the various assumptions of the interactionist model.

In Chapter 7, I apply the logical formalization in order to understand the computational properties of affixation and allomorphy. Again, I show that the bulk of affixation and allomorphy are computationally local. I provide formal heuristics for determining what is local in phonologically- or morphologically-conditioned allomorphy; this chapter thus formalizes the common intuition found in morphological theories of what constitutes local allomorphy (cf. Embick 2010). Using these heuristics, I show that there does exist some non-local affixation processes (cf. Paster 2006). I furthermore show that, besides locality, affixation tends displays the computational property of order-preservation. Order-preservation is a constraint on possible input-to-output correspondence relations.

In Chapter 8, I peel away some of the assumptions of my formalization. I show that without using an encapsulation mechanism such as a SETTINGS, prosodic transductions are still local; however, phonological rule domains may potentially require non-local triggers. I emphasize *potentially* because it is difficult to find concrete cases where the morphological trigger is not within a finite bound from the target. Interestingly, if we assume a non-interactionist model, then post-cyclic prosody is largely still a local process. Pockets of non-locality are then found in more complex prosodic transductions such as compound prosody and recursive prosody.

Finally, Chapter 9 discusses computational and empirical problems within cyclicity and with the interactionist model itself. Computationally, without a bound on cycles, the computation has unrestricted expressivity (Kaplan and Kay 1994:365). Empirically, there are a class of processes which reference information that is generated from *future* morphological or phonological processes. These include cases of outward-sensitive allomorphy and postcyclic phonology. In order to compute these processes, I show that all we need is to simply enrich the input with a mechanism that keeps track of all future morphological operations. Similar mechanisms are used in Distributed Morphology (Halle and Marantz 1993) and Paradigm-Function Morphology (Stump 2001). I combine the benefits of both those theories in order to define a computationally simple way to encode morphological look-ahead. This encoding enriches our sketch of an interactionist model. This enrichment likewise provides a partial solution to the computational problem of cyclicity.

(28)   *Sketch of an interaction model with an Operation stage*
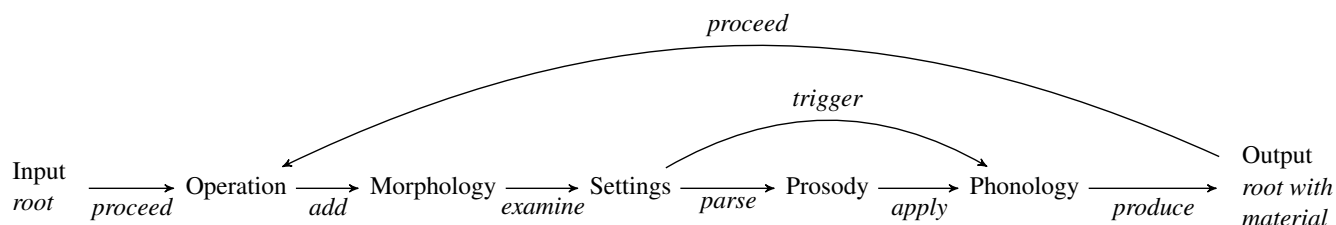
I conclude in Chapter 10.

1.A   Appendix on computational phonology

Although the computational formalization is couched within 2-level Declarative Phonology, my formalization has been heavily influenced by developments within subregular finite-state phonology and within 1-level Declarative Phonology. In this appendix, I go over the empirical coverage of these two frameworks. There is relatively little work in 2-level Declarative Phonology because of its recency. By providing the coverage of these three approaches, I hope to foster inspiration on how to combine the insights or tools of each framework for future work.

1.A.1   Developments in subregular phonology

Although finite-state calculus is sufficient to compute most morphological and phonological processes, many of these processes require specific *subclasses* of finite-state machines. There has been much recent work in defining which *subclasses* of finite-state calculus are needed to compute morphology and phonology (Heinz 2007; Rogers and Pullum 2011; Jäger and Rogers 2012; Rogers et al. 2013; Chandlee and Heinz 2017; Heinz 2018). This research program on **subregular finite-state phonology and morphology** has wide empirical coverage. I go over some these areas below.

As a general overview of its empirical coverage, there has been work on defining the right computational subclasses for:

1. local phonotactics (Chandlee 2014), including metathesis (Chandlee et al. 2012), iterative processes (Chandlee et al. 2015), opacity (Chandlee et al. 2018), and dissimilation (Payne 2014, 2017)
2. long-distance phonotactics (Heinz 2007; Chandlee and Heinz 2018), including consonant harmony (Heinz 2010; Luo 2017), vowel harmony (Gainor et al. 2012; Heinz and Lai 2013; Lai 2015), and their mappings (Burness and McMullin 2020)
3. morphological processes such as affixation (Chandlee 2017), reduplication (Chandlee and Heinz 2012), and cyclic rule domains (Bjorkman and Dunbar 2016)
4. suprasegmental processes such as tone (Jardine 2016a), tone sandhi (Chandlee 2019), stress (Heinz 2007, 2009, 2014), and sign language phonology (Rawski 2019a)
5. features or feature-based representations (Heinz and Koirala 2010; Strother-Garcia et al. 2017; Vu et al. 2018; Chandlee et al. 2019)

There have been applications to cognition (Rogers and Pullum 2011; Lai 2012, 2015; Heinz and Idsardi 2013; Hwangbo 2015; Avcu 2018, 2019; Rogers and Lambert 2019a; Lambert and Rogers 2019) and learnability (Heinz 2007; Jardine et al. 2014; Chandlee and Jardine 2014; Chandlee and Koirala 2014; Chandlee et al. 2014, 2015), including statistical learning (Heinz and Koirala 2010; Vu et al. 2018; Shibata and Heinz 2019) . One promising strand of work is discovering how these subregular classes correspond to different types of logical structures (Rogers and Pullum 2011; Strother-Garcia 2018, 2019; Chandlee and Jardine 2019b; Rawski 2019b; Rogers and Lambert 2019b; Lambert and Rogers 2020) and different neural network implementations (Shibata and Heinz 2016; Avcu et al. 2017; Rawski and Heinz 2019; Rawski 2019b; Nelson et al. 2020). A much more recent direction is finding the connection between subregular classes and information theory (Dai and Futrell 2020). In terms of implementation, the most sophisticated software support for subregular languages and processes is Aksënova (2020a,b)'s SigmaPie package.

One strand of work in subregular phonology has been defining new subclasses of finite-state calculus

based on linguistic concepts. A major development in this strand are *tier-based* languages and functions (Heinz et al. 2011). Tier-based formalisms have mostly been used to compute long-distance processes, such as consonant harmony (McMullin 2016; McMullin and Hansson 2016), vowel harmony (Aksënova and Deshmukh 2018; Mayer and Major 2018; Andersson et al. 2020), stress (Hao and Andersson 2019), morphology (Aksënova et al. 2016), semantics (Graf 2019b), and syntax when defined over trees.[10] These have results in learnabilty (Jardine 2016b; Jardine and Heinz 2016a; Jardine and McMullin 2017) and cognition (McMullin and Hansson 2019). Various extensions or refinements to tier-based formalisms have been proposed to handle diverse types of locality domains and blockers over phonological (Graf 2017; Graf and Mayer 2018; De Santo 2018; De Santo and Graf 2019; Karakaş 2020), prosodic (Baek 2018; Hao 2020), and morphological structure (Aksënova and De Santo 2018; Moradi et al. 2019). These extensions have some results on learnability (McMullin et al. 2019; Burness and McMullin 2019).

Other developments in subregular phonology have been defining more complex finite-state subclasses based on strategies such as accessing individual transitions or actions for rhythmic syncope (Hao and Bowers 2019), intermediate markup and weak determinism for tonal processes (Lamont et al. 2019), and serial decomposition for long-distance processes (Lamont 2018).

There has likewise been work on extended finite-state subclasses to more enriched types of machines, functions, and representations. These extensions include two-way transducers for reduplication (Dolatian and Heinz 2018a,b, 2019a,b, Dolatian and Heinz forthcoming; Nelson et al. 2020), autosegmental grammars for tone (Jardine 2016c, 2017a, 2019, 2020), and multi-tape transducers for Semitic templates (Kay 1987; Kiraz 2000, 2001; Hulden 2009c; Dolatian and Rawski 2020a,b), tone (Wiebe 1992; Rawski and Dolatian 2020), and features (Carson-Berndsen 1998).

1.A.2 Developments in One-level Declarative Phonology

In mainstream contemporary phonological or morphological research, 1-level DP is not widely used. Its reliance on using only feature-filling rules and no transformations had currency in the late 1980s and early 1990s due to problems in rule-based phonology and computing cascades of rules. However, even with its theoretical limitation to only feature-filling processes, there was a coherent body of work that came out of 1-level DP.

Two significant monograph-sized overviews of the tradition are Bird (1995) and Coleman (1998). They develop case studies in phonotactics and prosody to clarify the theory. Smaller conceptual overviews and theoretical motivations for DP can be found in (Scobbie 1991b; Bird et al. 1992; Scobbie 1993a; Bird 1994; Scobbie et al. 1996). Earlier developments are (Wheeler 1981; Scobbie 1991a). Contrasts with other constraint-based approaches can be found in (LaCharité and Paradis 1993; Scobbie 1993a; Coleman 2011; Scobbie 1993b). Retrospective overviews are (Coleman 2006, 2011; Hammond 2009). Two major papers are (Bird and Klein 1994; Bird and Ellison 1994). Two large edited volumes are (Bird 1991a; Ellison and Scobbie 1993) which contain multiple case studies. The most significant recent development is Chew (2003) who provides an extensive formalization of Russian morphotactics, phonotactics, and prosody using 1-level DP.

A wide-ranging yet intimate set of phonological phenomena have been modeled in 1-level DP. These include reduplication (Keane 2005), templates (Bird and Blackburn 1991; Klein 1993; Bird and Klein

---

[10]See footnote 9 for a some work on subregular syntax when defined over trees.

1994), prosodic morphology (Walther 1998, 1999), morphotactics and allomorphy (Ellison 1993; Ogden 1999), focus and intonation (Oehrle 1991; Bird 1991c; van der Linden 1991), prosodic representations for syllabification, stress, feet, and word-level prosody (Klein 1991; Walther 1993; Dirksen 1993; Coleman 1991, 1992, 1993, 2000), especially Berber (Scobbie 1993a; Coleman 1996), autosegmental structure and tone (Bird and Klein 1990; Coleman and Local 1991), defaults and exceptions (Calder and Bird 1991; Gibbon 2001b), speech synthesis (Coleman and Local 1992; Coleman 1995b), morphological domains and lexical phonology (Cole and Coleman 1992; Coleman 1995a; Orgun 1998), suprasegmental features and Firthian prosody (Ogden 1993; Broe 1991a,b). Most of these applications were theory-neutral; though Russell (1993) developed a computational formalization using Government Phonology.

1.A.3   Recent developments in Two-level Declarative Phonology

Compared to subregular phonology and 1-level DP, there has been less work in 2-level DP because of its recency. Most work has focused on phonotactics (Rogers and Pullum 2011; Rogers et al. 2013; Heinz 2018) with recent applications to syllabification (Strother-Garcia 2018, 2019), tone (Jardine and Heinz 2016b; Chandlee and Jardine 2019a; Jardine 2014, 2016c, 2017b; Koser et al. 2019; Zhu 2020; Mamadou and Jardine 2020), stress (Koser and Jardine 2019), local and long-distance phonotactics (Chandlee and Jardine 2019b), cliticization (Ashton 2012), theory-evaluation (Graf 2010a; 2010b, Payne et al. 2017, Danis and Jardine 2019, Jardine et al. to appear, Oakden to appear), learnability (Strother-Garcia et al. 2017; Vu et al. 2018; Chandlee et al. 2019), and the relationship with neural network implementations (Rawski 2019b).

A lot of work has focused on finding how various phonological theories or representations are computationally equivalent, whether in syllabification (Strother-Garcia 2019) or tone (Danis and Jardine 2019, Jardine et al. to appear, Oakden to appear). There is likewise work in using restricted types of recursive functions in computing long-distance dependencies (Chandlee and Jardine 2019b), which can be potentially extended to iterative prosody. These recursive logical functions have links with Boolean monadic recursive program schemes; this opens doors to implementation and function composition (Bhaskar et al. 2020). This thesis resembles Chew (2003) in that I provide a systematic formalization for hierarchical aspects of phonology, specifically on computing morphological structure, prosodic structure, and phonological rule domains.

Part I

Cyclic Phonology in Armenian

Chapter 2

Cyclic phonology of destressed reduction

2.1 Introduction

Phonological processes are often sensitive to morphological, prosodic, and derivational structure (Kiparsky 1982b; Nespor and Vogel 1986; Inkelas and Zec 1990; Hargus and Kaisse 1993; Wiese 1994; Scheer 2011). In this chapter, I establish the architecture of the morphology-phonology interface for Armenian.

In terms of **Morphology**, Armenian distinguishes between morphological stems (MStems) vs. morphological words (MWords). The former is created from derivational morphology, while the latter from inflectional morphology. For the **Prosody**, MWords map to prosodic words (PWords),while MStems map to a smaller prosodic constituent called the prosodic stem (PStem). For the **Phonology** of rule domains, MStems and MWords trigger stem- vs. word-level strata. Interestingly, misaligned PStems trigger a separate PStem-level cophonology or rule domain. These three aspects of the interface (Morphology, Prosody, Phonology) are **organized** in a cyclic or interactionist derivation.

I establish the above architecture based on a widespread morphophonological process: Destressed High Vowel Reduction (29). Armenian is an Indo-European language with two modern standard dialects: Western (WArm) and Eastern Armenian (EArm). In both dialects, stress is final (29a) and suffixation triggers stress shift. Derivational suffixes also trigger the reduction of destressed (29b), not unstressed high vowels (29c), in a process of Destressed High Vowel Reduction (DHR). But the dialects vary on DHR in inflection. In WArm, DHR is not triggered by inflection (29e,29f); while in EArm, DHR is triggered by V-initial but not by C-initial inflection (29d,29f).

(29)	a.	Base	amusín	'husband'	
	b.	Der. Suffix	amusn-utjún	'marriage'	
	c.		*amsin-utjún		
	d.	V-initial Infl.	amusn-óv	'husband-INST'	Eastern
	e.		amusin-óv	'husband-INST'	Western
	f.	C-initial Infl.	amusin-nér	'husband-PL'	Eastern & Western

Focusing first on Western Armenian, I show how primary stress is assigned (§2.2). Then in §2.3, I break

down DHR into its phonological, derivational, and morphological factors in WArm.

Phonologically, DHR targets high vowels which lost stress (§2.3.1). DHR is insensitive to other phonological factors such as the distance between the destressed and the newly stressed syllables . Derivationally, I show that DHR is a cyclic process that applies when certain suffixes trigger stress shift (§2.3.2). DHR thus shows a Derived Environment Effect. In fact, DHR can apply a potentially unbounded number of times, depending on how much derivational morphology we have. But morphologically, not every suffix can trigger DHR in WArm (§2.3.3). Specifically in WArm, DHR applies in derivation but not inflection. I argue that this morphological distinction is the basis for stem-level vs. word-level strata. DHR is a stem-level rule in WArm, not word-level. Section §2.4 then provides more evidence for strata in Armenian.

Moving on to the Eastern Armenian, I argue that DHR is also a stem-level rule in EArm In §2.5. The reason why V-initial inflection can trigger DHR is prosodic: syllabification and prosodic misalignment. I argue that the relevant prosodic constituent is not a metrical foot or the prosodic word (PWord). Instead, pre-inflectional DHR is due to the misalignment of the Prosodic Stem (Downing 1999a) (§2.5.4). This constituent is straddled between V-initial inflection and C-initial inflection. It triggers some but not all stem-level rules; this means that the PStem is a prosodic constituent which is indexed with its own cophonology. However, the use of the Prosodic Stem for DHR is not stipulative, and in §2.6 I give independent evidence for the PStem from appendix incorporation.

Based on DHR WArm and EArm, I posited a hierarchy of cophonologies: the stem-level, PStem-level, and word-level. In §2.7, I show that lexical variation in DHR supports the existence of the PStem-level cophonology. These cophonologies form a monotonic hierarchy where possible constraint rerankings are controlled by the Strong Domain Hypothesis (Myers 1991).

Finally, I explain the microvariation between the two dialects as due to diachrony (§2.8). Both dialects descend from Classical Armenian (CArm), where DHR was word-level. In terms of the phonological life-cycle, DHR narrowed from the word-level in CArm to the stem-level in WArm and to PStem-level in EArm (30). I argue that this divergence happened because of CArm's morpheme inventory. In Classical Armenian, nominal inflection was either V-initial suffixes like dative *-oj*, or single consonantal suffixes like plural *-k*. There were no stressable C-initial consonants like **-ner*.[1] The modern dialects developed stressable C-initial inflection like *-ner*. I argue this change created the PStem in modern Armenian. In EArm, DHR narrowed down to the PStem-level instead of all the way to the stem-level.

(30) *Domain of Destressed High Vowel Reduction across Classical, Western, and Eastern Armenian*

Lect	Morphological Domain			Cophonology
	Derivation	V-initial Inflection	C-initial Inflection	
CArm	✓	✓		word-level
EArm	✓	✓	✗	stem-level and PStem-level
WArm	✓	✗	✗	stem-level

I conclude in §2.9. In the appendix, I sketch a diachronic analysis for the origins of destressed reduction based on opacity (§2.A). I also summarize the judgment of different grammarians on the productivity of destressed reduction (§2.B).

---

[1]Throughout this chapter, the term 'C-initial inflection' is used to mean C-initial inflectional suffixes which contain a vowel like *-ner* but unlike *-k*.

2.2 Stress in Armenian

In Armenian, primary stress falls on the rightmost full vowel in the word. Primary stress assignment is edge-based, quantity-insensitive, word-bound, and avoids schwas. There is little evidence for feet. The data in this section is from Western Armenian, but the generalizations extend to Classical and Eastern Armenian.

First, stress is edge-based because it can fall on the final full vowel in a root (31a-i), derivational suffix (31a-ii), or inflectional suffix (31a-iii). Compounds are formed by concatenating two stems with the linking vowel *-a-*. They surface with final stress (31b-ii), even when inflected (31b-iii).

(31) a. i. kórd͡z   'work'   b. i. seɣán   'table'
          ii. kord͡z-avór   'worker'   ii. kord͡z-a-seɣán   'work-bench'
          iii. kord͡z-avor-nér   'workers'   iii. kord͡z-a-seɣan-nér   'work-benches'

Second, stress is quantity-insensitive. Stress ignores the coda size of the final or penultimate syllable (32a). Vowel length is not phonemic; vowel-glide sequences exist and aren't heavy (32b).

(32) a. i. kórd͡z   'work'   b. i. kájl   'wolf'
              kord͡z-í   'work-GEN   kajl-í   'wolf-GEN
              kord͡z-óv   'work-INST   kajl-óv   'wolf-INST

Third, stress is word-bound. It cannot go outside the word and fall on a clitic (33a-iii) or a clitic cluster (33a-iv). Compounds behave the same (33b). Lists of such unstressable clitics can be found in Margaryan (1997:78) and Khanjian (2013:72).

(33) a. i.   kord͡z-avór   'a worker'   b. i.   kord͡z-a-seɣán   'a work-bench'
          ii.   kord͡z-avor-óv   'with a...'   ii.   kord͡z-a-seɣan-óv   'with a...'
          iii.   kord͡z-avor-óv al   'also with...'   iii.   kord͡z-a-seɣan-óv al   'also with ...'
          iv.   kord͡z-avor-óv al e 'is also with...'   iv.   kord͡z-a-seɣan-óv al e 'is also with...'

Fourth, stress generally does not fall on the final vowel if it is a schwa.[2] Instead, stress is on the preceding full vowel (34). The final schwa can be part of either an inflectional suffix (34a) or the root (34b). The schwa can be either underlying (34c) or epenthetic, whether this epenthesis is optional (34d) or obligatory (34b).

(34) a. kórd͡z-ə   /kord͡z-DEF/   'work-DEF'   c. pítər   < Eng. [pirəɹ]   'Peter'
          b. jerpémən   /jerpemn/   'sometimes'   d. pókr ~ pókər   /pokr/   'small'

Primary stress assignment as occurring within the prosodic word (PWord). A simple stress rule is given in (35). The PWord is largely isomorphic with the morphological word (MWord), i.e., it includes derivation, compounding, and inflection, but not clitics. The only non-isomorphism is when a MWord-final consonant

___

[2]Haghverdi (2016) documents cases where final schwas can get stress in Eastern Armenian. In a larger study, Skopeteas (2019) finds similar effects but argues it is due to phrasal boundary tones. In onomatopoeic words with only schwas, both initial (Vaux 1998b:133) and final stress (Ačaṙyan 1971:339) are documented. The (un-)stressability of schwas is not crucial to this chapter.

resyllabifies with a V-initial clitic and leaves the PWord (35d). MWords and PWords are marked by the subscripts $_W$ and $_w$.

(35)   a. **Primary Stress Assignment**
       Assign stress on the rightmost syllable if it has a full vowel (σ_V). Otherwise if the final syllable has a schwa ($\sigma_ə$), then stress the penultimate full vowel
       $$\sigma_V \rightarrow \sigma_V{'} / \_ (\sigma_ə) \#$$
       b. $\{$kord͡z-ə$\}_W$ $\rightarrow$ (kórd͡z-ə)$_w$
       c. $\{$kord͡z-avor-ov$\}_W$ $\rightarrow$ (kord͡z-avor-óv)$_w$
       d. $\{$kord͡z-avor-ov$\}_W$ al e $\rightarrow$ (kord͡z-avor-ó.)$_w$v=al e

As for secondary stress, it is on the initial syllable (36) (Abeġyan 1933:20). This creates a hammock pattern because primary and secondary stress are on opposite sides of the word (Gordon 2002). Secondary stress is not iterative; long lapses of unstressed syllables are found (36d). To illustrate the lapses, I show hypothetical foot boundaries. Clitics are ignored by stress (36e).

(36)   Surface   Hypothetical Feet
       a. nàmág   (nà)(mág)   'letter'
       b. bàdasxán   (bà)(das.xán)   'answer'
       c. bàdasxan-avór   (bà.das)xa(na.vór)   'responsible'
       d. bàdasxan-avor-ner-óv   (bà.das)xa.na.vor(ne.róv)   'responsible-PL-INST'
       e. bàdasxan-avor-ner-óv al e   (bà.das)xa.na.vor(ne.ró.)v=al e 'is also with responsible people'

Although I show hypothetical foot boundaries in (36), the stress data do not show any positive evidence for feet (DeLisi 2015:42ff, 2018:115). Armenian lacks common correlates found in final-iambic languages. For example, stress is quantity insensitive and phonetically cued by pitch (Athanasopoulou et al. 2017). Turkish and French have stress patterns similar to Armenian and have been argued to lack feet (Özçelik 2017). Feet will be shown to be irrelevant for the analysis of Armenian prosody and reduction.

2.3   Destressed High Vowel Reduction in Western Armenian

In Western Armenian (WArm), there is evidence that stress is being assigned and reassigned as each suffix is added, i.e., on a cyclic basis. The evidence is Destressed High Vowel Reduction (DHR).

(37)   a. Base   amusín   'husband'
       b. Der. Suffix   amusn-utjún   'marriage'
       c.   *amsin-utjún
       d. V-initial Infl.   amusin-óv   'husband-INST'   Western
       e. C-initial Infl.   amusin-nér   'husband-PL'   Western

Adding a derivational suffix triggers a cycle of stress assignment and the reduction of the destressed high vowel *i* (37b). In contrast, the unstressed high vowel *u* does not reduce (37c). DHR distinguishes derivation from inflection: inflection triggers stress shift but *not* reduction (37d,37e). I expand on the phonological (§2.3.1), derivational (§2.3.2), and morphological (§2.3.3) factors behind DHR.

2.3.1 Phonology of reduction

Phonologically, DHR is limited to high vowels, and it is affected by stress shift and marked syllable structure (§2.3.1.1). It is insensitive to other phonological factors, such as secondary stress and the location of the destressed vs. newly stressed syllables (§2.3.1.2). I go through these factors below.

2.3.1.1 Destressing and syllabifiability

Among monophthongs, only a stress-less high *i* or *u* can reduce (38a,38b). Non-high vowels do not.[3]

(38) a. badíʒ   'punishment'   amís   'month'
          badʒ-él   'to punish'   ams-agán   'salary'

      b. makúr   'clean (adj)'   dəxúr   'sad'
          makr-él   'to clean'   dəxr-utjún 'sadness'

      c. h a d͡ʒáx   'frequent'   arák   'fast'
          had͡ʒax-él   'to frequent'   arak-anál   'to speed up'

      d. darpér   'different'   arhést   'craft'
          darper-él   'to differentiate' arhest-avór 'craftsman'

      e. ʒoɣóv   'assembly'   atór   'chair'
          ʒoɣov-él   'to collect'   ator-ág   'stool'

However, not just any stressless high vowel can reduce; it has to specifically be a destressed high vowel (Katvalyan 1989:89; Margaryan 1997:87; Vaux 1998b:148; Khanjian 2009). A destressed vowel is a vowel which is stressed in the base but not in the derivative: *badíʒ ~ badʒ-él* (38a). To illustrate, the words in (39) have multiple high vowels. But note that only the destressed high vowel can reduce in the derivative.

(39) a. irigún   'evening'   b. amusín   'husband'   c. digín   'lady'
          irign-ajín 'evening (adj.)'   amusn-anál   'to marry'   dign-utjún 'ladyship'
          *irgun-ajín   *amsin-anál   *dgin-utjún

The same applies for roots with multiple vowels but only one high vowel. If the high vowel is unstressed in the base, it stays unstressed in the derivative and does not reduce (40). It does not matter if the high vowel is in the first (40a) or second syllable (40b), or whether it has an onset or not (40c).

(40) a. d͡zid͡záɣ   'laughter'   nihár   'skinny'   kiʃér   'night'
          d͡zid͡zaɣ-él 'to laugh'   nihar-utjún   'skinniness'   kiʃer-ajín   'nocturnal'

---

[3] An exceptional set of roots with non-high vowels show apparent destressed reduction. These are discussed in §2.8.

    b. məxitár   'comforter'   vostigán   'policeman'   heɣinág   'author'
       məxitar-ánk 'comfort'   vostigan-uhí 'policewoman'   heɣinag-él 'to compose'

    c. imást   'meaning'   uráx   'happy'   iʃxán   'prince'
       imast-agán 'intellectual'   urax-anál 'to be happy'   iʃxan-utjún 'principality'

Unsurprisingly, if affixation does not trigger stress shift, then there is no destressed reduction. In (41), the nominalizer *-k* cannot trigger stress shift for phonological reasons: it lacks a full vowel.

(41) a. darí   'year'   b. kaxtní   'secret (adj)'   c. parí   'good'
      darí-k   'age'   kaxtní-k 'secret (n)'   parí-k   'charity'

DHR manifests as vowel deletion (42a), unless deletion would create an unsyllabifiable consonant cluster. In general, the maximal syllable is CVCC: complex onsets are banned while complex codas have falling sonority.[4] To avoid bad clusters, the destressed high vowel is replaced by a schwa (42b). Whether the schwa is epenthetic (Vaux 1998b; Khanjian 2009) or corresponds to the destressed vowel (Xačatryan 1988) is controversial but orthogonal to this chapter.

(42) a. amusín   'husband'   b. aznív   'honest'
      amusn-utjún   'marriage'   aznəv-utjún 'honesty'
      *amusən-utjún   *aznv-utjún

2.3.1.2 Insensitivity to other prosodic factors

Having shown what phonological factors control DHR, I now go over prosodic factors which do *not* affect reduction. First, DHR is only sensitive to primary stress and it ignores secondary stress. Initial secondary stress does not protect destressed high vowels in monosyllabic roots from DHR: *púrt* derives *pərt-a-vad͡ʒár*.[5]

(43) *DHR ignores root or suffix size in derivation*

Root size		σ		σσ	
Suffix size	-σ	púrt	'wool'	gamúrd͡ʒ	'bridge'
		pərt-éɣ	'woolly'	gamərd͡ʒ-ág	'small bridge'
	-σσ	pərt-arán	'wool shop'	gamərd͡ʒ-ají́n	'relating to bridges'
Compound size	LV-σ	pərt-a-kórd͡z	'wool-stapler'	gamərd͡ʒ-a-dúrk	'bridge-toll'
	LV-σσ	pərt-a-vad͡ʒár	'wool-seller'	gamərd͡ʒ-a-kəlúx	'bridge-head'

---

[4]There are a few known exceptions to the ban on complex onsets: consonant-glide clusters (Vaux 1998b:81), borrowed proper names (Baronian 2017), word-initial obstruent-rhotic clusters (T'oxmaxyan 1988), and word-initial obstruent clusters that are aspirated or fricated (Hovakimyan 2016). The main exception to falling-sonority complex codas are stem-final appendixes (Vaux and Wolfe 2009). This is discussed in §2.6.1.

[5]To explain this, we could argue that high vowels do not get secondary stress (Vaux 1998b:149). But this is not confirmed from acoustic data (T'oxmaxyan 1983; Athanasopoulou et al. 2017). Furthermore, it is unclear if secondary stress is relevant. Secondary stress has weaker cues in Armenian than primary stress (Abeġyan 1933:20). Although there are diachronic processes which reference secondary stress, e.g., medial syncope (§2.5.1) (DeLisi 2018), there are no attested synchronic processes which do.

Second, DHR is not pretonic reduction (43). The destressed vowel can be at any distance from the stressed vowel: *púrt ∼ pərt-a-vad͡ʒár*. When a derivational suffix is added to a root, DHR can apply regardless of the size of the root or suffix. Most derivational suffixes are monosyllabic or disyllabic (Dum-Tragut 2009:652); there are no larger suffixes. DHR likewise applies in compounds. The size of the first or second root doesn't matter (43). Most roots are monosyllabic or disyllabic, but larger roots exist (44). These can reduce just like smaller roots, whether before derivational suffixes or in compounding.

(44) *DHR for trisyllabic roots*

	$\sigma\sigma\sigma$ stem	Derivative	2nd stem	Compound
a.	ʒoɣovúrt	ʒoɣovərt-anót͡s	hamár	ʒoɣovərt-a-hamár
	'populace'	'populace'	'count'	'demography'
b.	potoríɡ	potorg-ál	ʃúnt͡ʃ	potorg-a-ʃúnt͡ʃ
	'storm'	'to bluster'	'breath'	'tempestuous'
c.	məderím	məderm-utjún	t͡sájn	məderm-a-t͡sájn
	'intimate'	'intimacy'	'voice'	'intimate voiced'

Finally, in previous examples, stress shifted to a V-initial element. But when DHR applies in derivational morphology, the syllable structure of the new element does not matter. The derivational element can be a V-initial or C-initial suffix (45a).[6] In compounding, the linking vowel *-a-* is generally used (Donabédian 2004). In some compounds, the linking vowel is exceptionally absent but DHR can still apply (45b).

(45) a. ʒoɣovúrt   'populace'   b. dún + dés   'house + see!'
          ʒoɣovərt-agán   'popular'   dən-dés   'house-keeper'
          ʒoɣovərt-ján   'popular'

2.3.2 Derivational history and reduction

Having established the relatively simple phonology behind DHR, I now show how DHR is sensitive to a word's derivatinal history. Because DHR is mainly triggered by stress shift, I show that DHR shows a Derived Environment Effect (§2.3.2.1) and it displays unbounded cyclicity (§2.3.2.2).

2.3.2.1 Destressed vowel reduction as a Derived Environment Effect

As said, the application of DHR depends on a word's derivational history. This makes DHR a Derived Environment Effect (DEE) (Kiparsky 1973) which shows Non-Derived Environment Blocking (NDEB) (Kiparsky 1993). There is a wide literature on how to formalize DEEs (Łubowicz 2002; Wolf 2008; Burzio 2011; Inkelas 2014). For Armenian, Vaux (1998b:148) uses rules with the Strict Cycle Condition while Khanjian (2009) uses constraints with Comparative Markedness (McCarthy 2003; Lubowicz 2003). Here, I represent DHR by a simple rule (46) that references 'destressed' vowels via a diacritic: ĭ,ŭ

---

[6]There is variation in DHR when the suffix *-jan* is used to form surnames, e.g. *manug* 'small' ∼ *manug-ján* 'Manougian', *sarkis* ∼ *sarkis-ján* (two names in WArm), vs. *sargis* ∼ *sargəs-ján* (two names in EArm).

(46) **Destressed High Vowel Reduction (DHR)**
      If a high vowel is destressed, it is deleted or replaced by a schwa. Deletion is used if it won't create
      an unsyllabifiable cluster; otherwise a schwa is used
      ĭ,ŭ → ∅ / VC$_1$(C$_2$)_ CV such that C$_1$C$_2$ can form a complex coda
              ə / elsewhere

I illustrate a sample derivation below.[7]

(47) a. hivánt   'sick'   hankíst   'relaxed'   amusín   'husband'
          hivant-anál   'to become sick'   hankəst-anál   'to relax'   amusn-anál   'to marry'
          *həvant-anál   *hankist-anál   *amsin-anál
      b. *Deriving base-derivative pairs for destressed high vowel reduction*

Base	Underlying form	/hivant/	/hankist/	/amusin/
	Stress	hivánt	hankíst	amusín
Derivative	Suffixation	hivánt-anal	hankíst-anal	amusín-anal
	Stress	hivănt-anál	hankĭst-anál	amusĭn-anál
	DHR		hankəst-anál	amusn-anál

From *hivánt*, we derive *hivant-anál*. The high vowel does not reduce because it is unstressed in both the
base and the derivative. For *hankíst* ∼ *hankəst-anál*, the high vowel *i* is reduced because it was destressed,
i.e., stressed in the base but not in the derivative. The vowel is reduced to a schwa because deletion cannot
form a syllabifiable cluster. And for *amusín* ∼ *amusn-anál*, the first high vowel *u* does not reduce because
it wasn't stressed in the base. The second high vowel *i* is reduced because it is destressed. It is is deleted
because the resulting cluster is syllabifiable.

2.3.2.2 Unbounded cyclicity in DHR

Because DHR depends on cyclic stress shift, DHR shows signs of unbounded cyclicity. In all previous
examples, the destressed high vowel was in a root, but destressed high vowels in *suffixes* can also reduce.

(48) a. ázk   'nation'   b. jérk   'song'   c. márz-   '√sports'
          azk-ajín   'national'   jerk-ítʃ   'singer'   marz-ítʃ   'trainer'
          azk-ajn-agán   'nationalist'   jerk-tʃ-uhí 'female singer'   marz-tʃ-agán 'of trainers'

Multiple application of DHR is found in compounds. The individual stems in a compound can consist of
zero (49a) or more derivational suffixes (49b). Compounds generally have at most two stems. Compounds
with three or more stems are rarer and restricted to technical jargon; but when formed, they can show DHR
(49c). Outside of compounds, it is rare to find a word with destressed high vowels in both roots and suffixes.

---

[7]I am currently developing a formalization using constraint conjunction (Łubowicz 2002): *i,u*[-STRESS]&IDENT[STRESS].
This is violated by unstressed high vowels that lost stress in the derivation.

(49) a. sírd   'heart'   b. kír   'handwriting'   c. jergír   'Earth'
        lár   'string'   kər-ít͡ʃ   'pen'   t͡ʃúr   'water'
        sərd-a-lár 'heart-string'   dúp   'box'   volórt   'circuit'
                                      kər-t͡ʃ-a-dup 'pencil-box'   jergr-a-t͡ʃər-olórt ...[8]

Thus, DHR is a cyclic process that can apply a potentially unbounded number of times, depending on the number of morphemes. In this way, it resembles stress preservation in English (Chomsky and Halle 1968; Collie 2007, 2008), destressed *a*-raising in Romanian (Steriade 2008a), and *a*-raising in Qashgar Uyghur (Orgun 1994; Nevins and Vaux 2008:10ff).

Unbounded cyclicity is theoretically and empirically controversial (Bermúdez-Otero 2011). It was assumed in earlier rule-based cyclic phonology (Chomsky et al. 1956; Chomsky and Halle 1968) and lexical phonology (Kiparsky 1982b). It is possible in some current frameworks via recursive computation (Kenstowicz 1996; Benua 1997) or probabilistic stem storage (Collie 2007, 2008; Bermúdez-Otero and McMahon 2006; Bermúdez-Otero 2016). The unbounded cyclicity of DHR cannot be reduced to other factors such as prosodic alignment (Liberman 1975; McCarthy and Cohn 1998) (cf. Cohn 1989; Duanmu 1999).

2.3.3 Morphology of reduction

In contrast to the previous sections, not every instance of stress shift will cause DHR. Stress shift is triggered by derivation (compounding and affixation) and by inflection (50). But DHR is more restricted. In WArm, it is triggered in derivation (50ii,iii) but not inflection (50iv-vi) (Avetisyan 2007:41, 2011:72).

(50) *Application of DHR in WArm derivation, but not inflection*

		a. monosyllabic base		b. polysyllabic base	
i	Base	túxt	'paper'	amusín	'husband'
ii	Der. Suffix	təxt-arán	'paper-mill'	amusn-utjún	'marriage'
iii	Compound	tərám	'money'	zúrg	'deprived'
		təxt-a-tərám	'paper-money'	amusn-a-zúrg	'spouse-deprived'
iv	V-initial Infl.	tuxt-óv	'paper-INST'	amusin-óv	'husband-INST'
v	C-initial Infl.	N/A		amusin-nér	'husband-PL'
vi	Stacked Infl.	tuxt-er-óv	'paper-PL-INST'	amusin-ner-óv	'husband-PL-INST'

In WArm, inflection systematically blocks reduction. It does not matter if the inflectional suffix is V-initial (50iv), C-initial (50v), or stacked (50vi), or if the root is monosyllabic (50a) or larger (50b). Monosyllabic roots do not have C-initial inflection (50a-v). In fact, regular nominal inflection is agglutinative in Western Armenian and it robustly blocks DHR. In (51), I show the regular plural and case markers for monosyllabic and polysyllabic nouns.[9] The plural shows phonologically-conditioned allomorphy: *-er* after monosyllables

---

[8]The gloss can be translated as the 'the hydrosphere of the Earth's crust'. The *v* is epenthetic (Vaux 1998b:13). The linking-vowel is absent before vowels (Donabédian 2004).

[9]Besides case and plural marking, DHR is inactive throughout Western Armenian regular inflection. Some suffixes lack full vowels and thus can't trigger stress shift or DHR: *-ə/-n* -DEF, *-əs/-s* -1SGPOSS, *-ət/-t* -2SGPOSS. The plural possessive suffix is *-ni* after polysyllabic bases but *-er-ni* after monosyllabic bases with a spurious plural (Arregi et al. 2013; Wolf 2013). It does not trigger reduction: *ʃun-er-nis* 'our dog', *axt͡ʃig-ní-s* 'our girl'. An apparent exception is verbal inflection which shows reduction. But this isn't a true exception. Verbs are formed with verbalizing theme vowels: *kír* 'writing' to *kər-é-l* 'to write'. Theme vowels act

and *-ner* elsewhere. The choice is arbitrary and generally not phonologically-optimizing (Vaux 2003).[10] But regardless, DHR generally does not apply in regular inflection, even when inflection is stacked (51).

(51)   *Regular inflection in Western Armenian without reduction*

	Base size	Citation form		Dative/Genitive	Ablative	Instrumental
Singular	σ	*tuxt*	'paper'	*tuxt-i*	*tuxt-e*	*tuxt-ov*
	$\sigma\sigma^+$	*amusin*	'husband'	*amusin-i*	*amusin-e*	*amusin-ov*
Plural	σ	*tuxt-er*	'papers'	*tuxt-er-u*	*tuxt-er-e*	*tuxt-er-ov*
	$\sigma\sigma^+$	*amusin-ner*	'husbands'	*amusin-ner-u*	*amusin-ner-e*	*amusin-ner-ov*

I formalize the above derivation vs. inflection distinction with stem vs. word strata (Kiparsky 1982b, 2000, 2015; Bermúdez-Otero 2011, 2012, 2018). Free-standing roots, derivational suffixes, and compounds form Morphological Stems (MStems or MS), while inflection creates Morphological Words (MWords or MW). MStems trigger the stem-level cophonology of stress and DHR; MWords trigger the word-level cophonology of stress without reduction. In (52), I illustrate the stratal derivation of the base *amusin*, its derivative *amusn-utjun*, and its inflected form *amusin-ov*.

(52)   *Stratal derivation of the root* amusin, *derivative* amusn-utjun, *and inflected* amusin-ov

Input			/amusin -$\emptyset_S$ -$\emptyset_W$ /	/amusin -$\emptyset_S$ -utjun$_S$ -$\emptyset_W$ /	/amusin -$\emptyset_S$ -ov$_W$ /
Cycle 1	MORPH	Spell-out	/amusin -$\emptyset_S$ /	/amusin -$\emptyset_S$/	/amusin -$\emptyset_S$/
	PROSODY	Syllabify	a.mu.sin	a.mu.sin	a.mu.sin
	PHONO *SLevel*	Stress	a.mu.sín	a.mu.sín	a.mu.sín
		DHR			
Cycle 2	MORPH	Spell-out		a.mu.sín - /-utjun$_S$/	
	PROSODY	Syllabify		a.mu.sín-u.tjun	
	PHONO *SLevel*	Stress		a.mu.šin-u.tjún	
		DHR		a.mus.n-u.tjún	
Cycle 3	MORPH	Spell-out	amusín /-$\emptyset_W$/	amusn-utjún /-$\emptyset_W$/	a.mu.sín /-ov$_W$/
	PROSODY	Syllabify			a.mu.sí.n-ov
	PHONO *WLevel*	Stress	amusín	amusn-utjún	a.mu.šin-óv
Output			amusín	amusn-utjún	amusin-óv

as derivational suffixes and are found almost everywhere in a verb's inflectional paradigm: *kər-é-m* 'I write'. Some paradigm cells replace the theme vowel with some other V-initial suffix and still show reduction: *kər-adz͡* 'written'. Thus, verbal inflection shows reduction because it requires an intermediate step of adding the theme vowel. Finally, DHR can exceptionally apply in irregular inflection; I discuss this in §2.7.2.

[10]Many cases of syllable-counting allomorphy are phonologically-optimizing and reference feet (Kager 1996; González 2005). But see Vaux (2003); Paster (2005, 2006, 2019) for more cases that do not reference feet and aren't optimizing.

Each cycle includes a round of morphological, prosodic, and phonological processes. The phonological processes are triggered by different cophonologies (Inkelas 2014). The MStems and MWords have a subscript for their morphosyntactic feature. In their underlying form, affixes have a subscript $_S$ or $_W$ to indicate that they trigger the stem-level or word-level cophonology. I assume that free-standing roots form an MStem with a covert category-head (Giegerich 1999; Marantz 2007). I assume that stems can form MWords via covert nominative case. Grey cells indicate that a given item doesn't undergo a certain step.

In Cycle 1, the root is spelled out, syllabified, and undergoes the stem-level phonology to get stressed: *amusín*. In Cycle 2, the derivative is formed. The stem-level phonology of stress shift and reduction is applied: *amusn-utjún*. In Cycle 3, covert or overt inflection is added and the word-level cophonology is applied. For the base and derivative, Cycle 3 vacuously places stress on the last syllable again. For the inflected word *amusin-ov*, the word-level phonology is applied with stress shift but no reduction: *amusin-óv*. I assume that the word-level is postcyclic, and that the stem-level is cyclic (Booij and Rubach 1987). In the next section, I provide additional evidence for strata in Western Armenian.

2.4 Lexical strata elsewhere in Western Armenian

In WArm, DHR is sensitive to derivation vs. inflection. I formalized this difference with strata. Here, I show independent evidence for strata elsewhere in WArm: Destressed Diphthong *uj*-Reduction and vowel hiatus. Readers should note that these processes are treated as diagnostics for the stem-level cophonology.

2.4.1 Destressed Diphthong *uj*-Reduction

Armenian has sequences of vowels and glides *uj, ju, aj, ja, ej, je, oj, jo*, which I descriptively call diphthongs. Of these, in general, only *uj* undergoes destressed reduction. Destressed *uj* reduces to [u] in derivation, both suffixation (53ii) and compounding (53 iii), but not in inflection (53iv).

(53)   *Destressed Diphthong uj-Reduction in roots*

	i. Base	ii. Derivative	iii. 2nd stem	Compound	iv. Inflected
a.	zərújt͡s	zərut͡s-él, *zər.t͡s-él	kír-k	zərut͡s-a-kír-k	zərujt͡s-óv
	'conversation'	'to converse'	'book'	'conversation-book'	'conversation-INST'
b.	lújs	lus-avór, *ləs-avór	nəmán	lus-a-nəmán	lujs-óv
	'light'	'luminous'	'similar'	'light-like'	'light-INST

Reduction can apply in suffixes (54a), especially when the suffix is on a bound root (54b).

(54)   *Destressed Diphthong uj-Reduction in suffixes*

	i. Root or base	Suffixed	ii.Derivative	iii. 2nd stem	Compound	iv. Inflected
a.	sovór	sovor-újt	sovor-ut-ajín	mol-utjún	sovor-ut-a-mol-utjún	sovor-ujt-óv
	'accustomed'	'custom'	'customary'	'addiction'	'routinism'	'custom-INST'
b.	jerev-íl	jerev-újt	jerev-ut-agán	díb	jerev-ut-a-díb	jerev-ujt-óv
	'to appear'	'appearance'	'visible'	'type'	'phenotype'	'appearance-INST'

Destressed *uj* reduces to [u]. Destressed Diphthong *uj*-Reduction (DDR) forms a counterfeeding chain shift with Destressed High Vowel Reduction (DHR). The [u] does not delete (53a) or further reduce to schwa (53b).[11] This creates a case of stratum-internal opacity. I represent DDR with a simple stem-level rule (55a) that is ordered after DHR.[12] I illustrate a derivation for the derivative *zəruts͡-el* (53a-ii) and inflected base *zərujts͡-ov* (53a-iv) in (55b). I omit the prosodic step for syllabification.

(55) a. **Destressed Diphthong *uj*-Reduction (DDR)**
        If a diphthong uj *vowel is destressed, it is reduced to* u
        ŭj → u

     b. *Stratal derivation of the derivative* zəruts͡-el *and* zərujts͡-ov

				MS_v derivation	MW_K derivation
Input				/zərujts͡ -∅$_S$ -el$_S$/	/zərujts͡ -∅$_S$ -ov$_W$/
Cycle 1	MORPHO		Spell-out	/zərujts͡ -∅$_S$/	/zərujts͡ -∅$_S$/
	PHONO	*Slevel*	Stress	zərújts͡	zərújts͡
			DHR		
			DDR		
Cycle 2	MORPHO		Spell-out	zərújts͡ - /-el$_S$/	
	PHONO	*SLevel*	Stress	zərŭjts͡-él	
			DHR		
			DDR	zəruts͡-él	
Cycle 3	MORPHO		Spell-out	zəruts͡-él	zərújts͡ /-ov$_W$/
	PHONO	*WLevel*	Stress		zərŭjts͡-óv
Output				zəruts͡-él	zərujts͡-óv

2.4.2 Vowel hiatus and positional faithfulness effects

Further evidence for strata comes from initial segment protection and vowel hiatus repair. In previous examples, the reduced destressed high vowel was in a closed CVC(C) syllable. But when the vowel is root-initial, i.e., in an onsetless syllable in a monosyllabic root, then it generally does *not* reduce in derivation or inflection (Margaryan 1997:94ff). Exceptions exist in derivation but not inflection (56c).

---

[11]In a handful of derivatives, destressed *uj* is deleted or replaced by schwa: *gabújd* 'blue' and *gabd-a-kújn* 'blue-colored'. This is due to a sporadic diachronic process of diphthong monophthongization which created synonymous alternating bases: *gabújd; gabúd* 'blue' (Margaryan 1997:111; Avetisyan 2011:27). If the derivative is formed from the diphthong-base, then diphthong reduction applies: *gabud-a-kújn*. Else if the derivative is formed from the monophthong-base, then high vowel reduction applies: *gabd-a-kújn*. Monophthongization also targeted some bases with *ju*: *ts͡jún; ts͡ún* 'snow' and *ts͡ən-a-márt* 'snowman'. See Greppin and Khachaturian (1986) on the dialectal distribution of monophthongization. See Khanjian (2009) for discussion on the phonetic and phonological differences between the diphthongs *uj, ju* and high vowels.

[12]Khanjian (2009) formalized this chain shift with distantal faithfulness constraints. I am currently developing a formalization using constraint conjunction

(56) a. íӡ   'viper'   b. úxt   'vow'   c. iɣt͡s   'wish'
        iӡ-agán   'viperish'   uxt-él   'to vow'   əɣt͡s-él   'to wish'
        iӡ-ér   'viper-PL'   uxt-ér   'vow-PL'   iɣt͡s-ér   'wish-PL'

The preference for stem-initial high vowels to not reduce is due to a positional faithfulness constraint (Beckman 1997) for stem-edges (57). This constraint blocks DHR at stem edges. It is is ranked higher than the output constraints which trigger destressed reduction.

(57) **Protecting high vowels at stem edges**:
     IDENT-EDGE: High vowels do not reduce if they are at the edge of a stem.

This constraint also protects stem-final high vowels in vowel hiatus (Casali 1997). This constraint must differentiate between initial and final edges, between final *i* and *u*, and between monosyllabic vs. polysyllabic roots (cf. Becker et al. 2012). To illustrate, consider the roots in (58) where the destressed high vowel *i* is in a stem-final open syllable. If the root is monosyllabic, the high vowel is not deleted in derivation or inflection. Glide epenthesis applies to repair vowel hiatus.

(58) t͡sí   'horse'   tí   'shovel'   lí   'copious'
     t͡sij-avór   'horseman'   tij-ág   'shovel'   lij-anál   'to fill'
     t͡sij-ér   'horse-PL'   tij-ér   'shovel-PL'   lij-ér   'copious-PL'

But if the stem-final *i* is in a polysyllabic root, we find different possible repairs. In derivation, vowel hiatus can be repaired by deletion (59a), coalescence (59b) (with /-agan/), glide formation (59c), or glide epenthesis (59d). The choice is unpredictable and lexeme-specific, but deletion is the most common strategy. In inflection, only glide epenthesis is allowed in Western Armenian.

(59) a. təʃnamí   'enemy'   c. gəɣzí   'island'
        təʃnam-agán   'hostile'   gəɣzj-ág   'islet'
        təʃnamij-é   'enemy-ABL'   gəɣzij-é   'island-ABL'

     b. hokí   'soul'   d. vortí   'son'
        hoke-gán   'spiritual'   vortij-agán   'filial'
        hokij-é   'soul-ABL'   vortij-é   'son-ABL'

Similar variation is found for destressed stem-final *u* in vowel hiatus. Both in monosyllabic (60a) and larger roots (60b), stem-final /u/ is turned into [v] (glide fortition) in derivation; schwa epenthesis may apply to repair the consonant cluster. Glide epenthesis can occur but it is less common (60c); deletion is rare. But in inflection, glide epenthesis is the only productive strategy in Western Armenian.[13]

(60) a. pú   'owl'   b. meɣú   'bee'   c. ot-a-t͡ʃú   'aviator'
        pəv-ál   'to screech'   meɣv-agán   'apian'   ot-a-t͡ʃuj-agán 'aeronatic'
        puj-é   'owl-ABL'   meɣuj-é   'bee-ABL'   ot-a-t͡ʃuj-é   'aviator-ABL'

---

[13]In WArm, pre-inflection glide fortition is restricted to lexicalized phrases. It is more common in EArm (Avetisyan 2007:83; Vaux 1998b:19). In §2.6.2, I show that pre-inflectional vowel hiatus repair is slightly more lax in Eastern Armenian.

To summarize, destressed reduction thus does not apply for high vowels at stem-edges because of positional faithfulness. Instead, vowel hiatus repair rules apply and these vary depending on the high vowel, syllable size, lexeme, and stratum. All of these processes are sensitive to derivation vs. inflection (Minassian 1980:46ff,95; Margaryan 1997:103ff; Vaux 1998b:19ff; Sowk'iasyan 2004:50).

In the stem-level phonology, we need different lexeme-specific rules or indexed faithfulness constraints for vowel deletion (MAX-V), glide epenthesis (DEP-j), glide formation (IDENT[SYLL]), glide fortition (IDENT[CONS]), and vowel coalescence (UNIFORMITY). In the stem-level cophonology, the markedness constraint *VV is ranked above these faithfulness constraints. The faithfulness constraints have different rankings in the stem-level, e.g., stem-level deletion needs the ranking *VV >> DEP-j >> MAX-V. In the word-level phonology, most of these faithfulness constraints are promoted over DEP-j: *VV >> MAX-V >> DEP-j. This makes glide-epenthesis the only productive hiatus repair rule in the word-level.

2.4.3   Interim summary: Strata in Armenian

The table in (61) summarizes the domains of various morphophonological processes in Western Armenian. These include Destressed High Vowel and Diphthong *uj*-Reduction, and vowel hiatus repair. Note how derivational and inflectional morphology cleanly align with the stem-level and word-level cophonologies.

(61)   *Domains of destressed reduction processes and vowel hiatus repairs in Western Armenian*

Phonological Process		Morphological domain		Cophonology	Structure-changing?
		Derivation	Inflection		
Final primary stress		✓	✓	stem-level and word-level	✓
Destressed reduction	of high vowels	✓	✗	stem-level	✓
	of diphthong *uj*	✓	✗	stem-level	✓
Vowel hiatus repair	Glide-epenthesis	✓	✓	stem-level and word-level	✗
	Glide fortition	✓	✗	stem-level	✓
	Glide formation	✓	✗	stem-level	✓
	Coalescence	✓	✗	stem-level	✓
	Deletion	✓	✗	stem-level	✓

Except for stress shift, the word-level processes are structure-building (epenthesis) and not structure-changing (no reduction, deletion). A process is structure-changing if it changes features or deletes segments. The tendency for the word-level stratum to respect the phonological structure of previous levels or cycles has been formalized by Structure-Preservation (Kiparsky 1985) or the Strong Domain Hypothesis (Myers 1991) in Lexical Phonology, higher ranked faithfulness in Stratal OT (Bermúdez-Otero 2018), and Phonological Persistence in Phase-based phonology (Newell 2008; Newell and Piggott 2014).

2.5 Destressed reduction in Eastern Armenian

We have seen so far DHR is triggered in derivation in both dialects:*amusín* ∼ *amusn-utjún* 'husband ∼ marriage'. In WArm, DHR is blocked before both V-initial and C-initial inflection: *amusin-óv* ∼ *amusin-nér* 'husband-INST ∼ husband-PL'. But in EArm, DHR applies in derivation *amusn-utjún and* in V-initial inflection *amusn-óv*, not in C-initial inflection *amusin-nér*. This section shows why reduction is possible in EArm V-initial inflection. It is not because EArm V-initial inflection is morphosyntactically different from V-initial inflection in WArm (§2.5.1) or from C-initial inflection (§2.5.2). Instead, it is the result of syllabification of MStems with inflection. I argue that this triggers the prosodic misalignment of a prosodic constituent. I argue that this constituent is not a foot or a PWord (§2.5.3), but the Prosodic Stem (§2.5.4).

2.5.1 Similarity of Eastern and Western Armenian morphology and phonology

Eastern and Western Armenian are mutually intelligible. Their main differences are in their phoneme inventory (Vaux 1998b; Vaux and Samuels 2005; Baronian 2017), verbal conjugation (Vaux 1995a; Kozintseva 1995; Donabédian 2001b; Donabédian 2016; Baronian 2006; Plungian 2018), and syntax (Sigler 1997; Ackerman et al. 2004; Megerdoomian 2009; Khanjian 2013; Donabédian 2018). But in terms of their morphophonology, there are few differences (Avetisyan 2007, 2011). Stress is final in EArm (62) just as it is in WArm. Destressed *uj*-reduction applies in the stem-level cophonology of derivation, but not in the word-level cophonology of inflection.

(62)   zərújt͡s^h 'conversation'   lújs   'light'   gújn   'color'
        zərut͡s^h-él   'to converse'   lus-avór   'luminous'   gun-avór   'colorful'
        zərújt͡s^h-óv   'conversation-INST'   lujs-óv   'light-INST'   gujn-óv   'color-INST'

Morphologically, nominal inflection is agglutinative in both dialects. The dialects differ in few slots (63). EArm has an additional locative case marker *-um*. The ablative is *-it͡s^h* in EArm but *-e* in WArm. The dative/genitive is *-i* in singular and plural nouns in EArm, but *-i* in singular and *-u* in plural nouns in WArm.

(63)   *Paradigm for regular nominal inflection in Western and Eastern Armenian*

		Citation form		DAT/GEN	ABL	INST	LOC
Singular	WArm	*pag*	'yard'	*pag-i*	*pag-e*	*pag-ov*	
	EArm	*bak*	'yard'	*bak-i*	*bak-it͡s*	*bak-ov*	*bak-um*
Plural	WArm	*pag-er*	'yards'	*pag-er-u*	*pag-er-e*	*pag-er-ov*	
	EArm	*bak-er*	'yards'	*bak-er-i*	*bak-er-it͡s*	*bak-er-ov*	*bak-er-um*

The dialects share almost the same lexicon. Besides loanwords, some differences are due to sporadic sound changes. One particular sound change is medial syncope (64). In the change from CArm to WArm and EArm, some (64a) but not all (64b) medial unstressed non-high vowels were syncopated. Medial syncope was sporadic and more prevalent in WArm than in EArm (64c,64d) (Margaryan 1997:88,109ff,124ff; Vaux 1998b:138,148; Sowk'iasyan 2004:51; Avetisyan 2007:83-5, 2011:54-60,97,123; DeLisi 2018:26).[14]

---

[14]Examples are from Hagopian (2005)'s bi-dialectal glossary. Syncope was not destressed reduction because unstressed vowels

(64) a. EArm   WArm   b. EArm   WArm
         avél   avél   'broom'   uráx   uráx   'happy'
         avl-él   avl-él   'to sweep'   urax-anal   urax-anal 'to be happy'

      c. avér   avér   'destruction'   sovór   sovór   'accustomed'
         aver-él   avr-él   'to destroy'   sovor-él   sorv-íl   'to study'

      d. mét͡s   méd͡z   'big'   artʰún   artún   'awake'
         met͡s-anál   med͡z-nál 'to get big'   artʰn-anál   artən-nál 'to wake up'

To summarize, EArm and WArm have virtually the same morphology. In both dialects, derivation creates MStems while inflection creates MWords. Destressed *uj*-reduction applies in the stem-level but not the word-level cophonology. The fact that diachronic syncope was more common in WArm than in EArm suggests that WArm should show more destressed reduction than EArm. The next section shows that this is incorrect; the domain of DHR in EArm is paradoxically larger than in WArm.

2.5.2   Reduction before V-initial inflection and syllabification

In both dialects, DHR applies in derivation. But the dialects differ when it comes to pre-inflectional DHR (Fairbanks 1948; Johnson 1954; Avetisyan 2007, 2011). In WArm, inflection cannot trigger reduction. But in EArm, V-initial inflection is morphologically word-level but it unexpectedly triggers the stem-level process of DHR. It does not trigger other stem-level process like Destressed Diphthong *uj*-reduction (§2.5.1). In this section, I show that this difference cannot be reduced to anything other than the phonological shape of inflectional suffixes.[15]

The case markers for both monosyllabic and polysyllabic roots are all V-initial (65). EArm shows DHR before these inflectional suffixes, but WArm does not. <u>Reduced</u> bases are underlined while **unreduced** bases are in bold. I show syllable boundaries to emphasize the misalignment of the MStem with syllables.

(65)   *DHR before V-initial case suffixes in Eastern but not Western Armenian*

Dialect	Base size	Citation form		DAT/GEN	ABL	INST	LOC
WArm	σ	*túxt*	'paper'	**tux.t-***í*	**tux.t-***é*	**tux.t-***óv*	
	$\sigma\sigma^+$	*amusin*	'husband'	**amusi.n-***í*	**amusi.n-***é*	**amusi.n-***óv*	
EArm	σ	*tʰúxtʰ*	'paper'	<u>*tʰəx.tʰ-í*</u>	<u>*tʰəx.tʰ-íts͡ʰ*</u>	<u>*tʰəx.tʰ-óv*</u>	<u>*tʰəx.tʰ-úm*</u>
	$\sigma\sigma^+$	*amusín*	'husband'	<u>*amus.n-ú*</u>	<u>*amus.n-íts͡ʰ*</u>	<u>*amus.n-óv*</u>	<u>*amus.n-úm*</u>

---

were targeted, e.g., the inchoative morpheme *-anal* (64d), and it was largely restricted to trisyllabic words. More cases of syncope targeting unstressed vowels come from causatives (Johnson 1954:185; Ġaragyowlyan 1979:42), inchoatives (Galstyan 2004), reduplicated verbs (Abrahamyan 1959), and compound linking-vowels (Ēloyan 1972:82). Although syncope was a sporadic diachronic process and it affected only idiosyncratic sets of words in the standard dialects, some non-standard dialects have generalized medial syncope into synchronic *destressed* a-reduction (Mkrtchyan 1952).

[15] A reviewer notes that a possible reason is language contact with Russian. Eastern Armenian is largely spoken in Iran, Russia, and Armenia. Armenia was part of the Soviet Union and its speakers are often bilingual in Russian. Russian vowel reduction may have promoted vowel reduction in EArm. But this is unlikely because the pre-inflectional DHR in EArm is documented earlier than the Soviet Union (Sargsyan 1987). Furthermore, EArm speakers in Iran are bilingual in Persian, not Russian. They still show the pre-inflectional DHR: *ʒoɣovúrt* 'populace' ∼ *ʒoɣovərt-ín* 'populace-GEN' (Megerdoomian 2009:31).

Crucially, the application of DHR varies in the plural. The plural is marked by *-er* after monosyllabic bases and *-ner* after polysyllabic bases (66). The V-initial *-er* can trigger reduction, while the C-initial *-ner* cannot. The plural allomorph *-ner* is the only productive C-initial inflectional suffix in EArm.[16]

(66)   *No DHR before WArm plurals or before EArm C-initial plural, only before EArm V-initial plural*

Dialect	Base size	Citation form	DAT/GEN	ABL	INST	LOC
WArm	σ	**tux.t**-*ér*	**tux.t**-*er-ú*	**tux.t**-*er-é*	**tux.t**-*er-óv*	
	$\sigma\sigma^+$	**amusin.**-*nér*	**amusin.**-*ner-ú*	**amusin.**-*ner-é*	**amusin.**-*ner-óv*	
EArm	σ	*tʰəx.tʰ-ér*	*tʰəx.tʰ-er-í*	*tʰəx.tʰ-er-íts*	*tʰəx.tʰ-er-óv*	*tʰəx.tʰ-er-úm*
	$\sigma\sigma^+$	**amusin.**-*nér*	**amusin.**-*ner-í*	**amusin.**-*ner-íts*	**amusin.**-*ner-óv*	**amusin.**-*ner-úm*

Inflectional suffixes are all monosyllabic and can be stacked. The plural precedes case (66) and it determines if the base reduces or not, regardless of which case-markers follow it: *tʰúxtʰ* 'paper' vs. *tʰəx.tʰ-er-óv* 'paper-PL-INST (EArm)' but *amusín* 'husband' vs. **amusin**.-*ner-óv* 'husband-PL-INST (EArm)'.

To summarize, there is as correlation between pre-inflectional DHR and between the initial segment of the inflectional suffix. Simply put, reduction applies before V-initial but not C-initial inflection.

2.5.3   High vowel reduction and prosodic misalignment

The previous section showed that pre-inflectional DHR depends on whether the inflectional suffix is V- vs. C-initial. This points to syllabification as a possible factor. Specifically, pre-inflectional DHR correlates with the misalignment of the MStem (= the base) and syllables. Before C-initial inflection, the MStem ends in a syllable: *amusin.-nér*. Before V-initial inflection, the MStem boundary and syllable boundary are not aligned: *amusi.n-óv→amusn-óv*.

Cross-linguistically, morphological constituents often get misaligned from syllable boundaries. When misaligned, phonological processes can apply differently. This suggests that the relevant phonological process references a prosodic constituent that is mapped from the morphology. For example, English level-1 suffixes are mostly V-initial and trigger the stem-level rule of stress shift, while level-2 suffixes are mostly C-initial and do not trigger stress shift (67). Dutch has similar misalignment patterns (van Oostendorp 2004).

(67)   *(médicine)$_w$*   *(sýnonym)$_w$*   *(áccurate)$_w$*   *(devélop)$_w$*
       (medícin-al)$_w$ *(synónym-ous)$_w$* *(áccurate)$_w$-ness* *(devélop)$_w$-ment*

Raffelsiefen (1999, 2005) analyzed stress shift as caused by the misalignment of the prosodic word (PWord). V-initial suffixes are outside the MWord but they incorporate into the PWord. By incorporating, these suffixes cause the PWord to expand and trigger the stem-level phonology. In other words, English stress-shift applies whenever the PWord gets larger than the MWord.

---

[16]EArm used to use *-ner* to also form plural possesives. In that case, it could trigger reduction in monosyllabic but not polysyllabic roots: *sírt*; *sərt-ner-əs* 'heart; our heart(s)' but *gəlúx*; *gəlux-nér-əs* 'head; our head(s)' (Dum-Tragut 2009:113). But the use of the suffix has unclear productivity (Nikita Bezrukov, p.c.); data and generalizations are thus limited. If productive, its ability to trigger reduction in monosyllables would likely be due to prosodic minimality restrictions (Downing 2006).

I argue for a similar analysis for Armenian. The MStem maps to some prosodic constituent *X*. Before V-initial inflection, *X* gets misaligned from syllable boundaries and must expand. In WArm, the expansion of *X* has no effect. But in EArm, the expansion of *X* triggers pre-inflectional DHR. The problem is defining what *X* is. In this section, I show that two common prosodic constituents (feet and PWords) are unlikely to be involved in DHR. The evidence points to a distinct prosodic constituent which mediates between feet and PWords: the Prosodic Stem (PStem).

2.5.3.1 Feet as the domain of pre-inflectional DHR

One hypothetical solution is to have *X* be a foot: the stem-level cophonology creates feet which later trigger pre-inflectional DHR (cf. Anttila 2006). But it is unclear how feet would trigger pre-inflectional DHR. Both V-initial and C-initial suffixes take final stress and would form iambic feet. I sketch out a failed analysis below for the two inflected items *amusn-óv, amusin-nér*.

(68) *A failed hypothetical derivation using feet*

Input			/amusin -$\emptyset_S$ -ov$_W$/	/amusin -$\emptyset_S$ -ner$_W$/
Cycle 1			a.(mu.sín)	a.(mu.sín)
Cycle 2	MORPH	Spell-out suffix	a.(mu.sín) /-ov/	a.(mu.sín) /-ner/
	PROSODY	Syllabify suffix	a.(mu.sí.n)-ov	a.(mu.sín)-ner
		Align foot with syllables	a.mu.(sí.n-ov)	
	PHONO	Stress & Shift feet		a.mu.(sĭn-nér)
		DHR in 1st syllable of foot	a.(musn.-óv)	*a.mu.(sən-nér)
Predicted Output		amusn-óv		*amusən-nér
Correct Output			amusn-óv	amusin-nér

At the end of the stem-level cycle, the base forms a final foot: *a(mu.sín)*. Upon syllabification with V-initial inflection, the foot is misaligned from syllable boundaries: *a.(mu.sí.n)-ov*. This triggers shifting the foot to the suffix: *amu(sín-ov)*. Before C-initial inflection, no misalignment occurs and thus no shift (at first): *a(mu.sín)-ner*. So far the derivation shows a prosodic difference between V-initial vs. C-initial inflection: *amu(sín-ov), a(mu.sín)-ner*. At this stage, we could argue that DHR targets the first syllable of feet. But this analysis falls apart once we actually apply stress shift in the word-level: *amu.(sĭn-nér)*. Now, both V-initial and C-initial inflection have the same foot structure: *amu(sĭn-óv), amu.(sĭn-nér)*. If DHR is formulated to apply in the first syllable of feet, then we incorrectly predict reduction in both inflected words: *amusin-óv, *amusən-ner*. Without additional machinery, pre-inflectional DHR cannot be formulated in terms of feet.[17] Furthermore, as explained in §2.2, there is no independent evidence for iambic feet in Armenian.

---

[17]One workaround is to argue that DHR applies if the destressed syllable is 1) the weak part of an iambic foot, and 2) was aligned with the MStem in the input, but 3) is no longer aligned with the MStem in the output. This works; however it is then unclear what role is played by the feet. Conditions (2,3) are the descriptive generalizations and do trigger DHR; the use of feet (1) is superfluous.

2.5.3.2 Recursive PWords and masking different domains

Instead of feet, we could argue that *X* is a PWord. But this is problematic. All inflectional suffixes are incorporated into the PWord and obligatorily trigger the word-level process of stress shift (69). Thus, the domain of final stress (inflection = the PWord) is larger than the domain of DHR (V-initial inflection = X).

(69)							*Stress shift?*	*Reduce?*
	a.	Base	amusín	(amusín)$_w$	'husband'			
	b.	V-initial Infl.	amusn-óv	(amusn-óv)$_w$	'husband-INST'		✓	✓
	c.	C-initial Infl.	amusin-nér	(amusin-nér)$_w$	'husband-PL'		✓	✗
	d.	Stacked Infl.	amusin-ner-óv	(amusin-ner-óv)$_w$	'husband-INST-PL'	✓		✗

This problem can't be fixed by pushing stress-assignment to a higher domain, e.g., a recursive PWord' (Peperkamp 1997; Selkirk 1996; Ito and Mester 2009; Kabak and Revithiadou 2009) or the Clitic/Composite Group CG (Vogel 2009, 2016). Enclitics are outside the stress domain and syllabify with word-final consonants (70). Clitics thus follow a PWord boundary and belong to the PWord' or CG.

(70) *Cliticization on...*

	a.	Base	amusín e	(amusí)$_w$n=e	'husband is'
	b.	V-initial Infl.	amusn-óv e	(amusn-ó)$_w$v=e	'husband-INST is'
	c.	C-initial Infl.	amusin-nér e	(amusin-né)$_w$r=e	'husband-PL is'
	d.	Stacked Infl.	amusin-ner-óv e	(amusin-ner-ó)$_w$v=e	'husband-INST-PL is'

Disregarding clitics, one could argue that the relevant constituent is a minimal vs. maximal PWord (Bermúdez-Otero, p.c.). I illustrate below. In the word-level stratum, the base first forms a single PWord; this is before the syllabification of V-initial or C-initial inflection: *(amusin)$_w$ /-ov,-ner/*. When V-initial inflection is syllabified, the MStem's PWord is misaligned from its syllable boundaries: *(a.mu.si.n)$_w$-ov*. This triggers PWord expansion: *(amusin-ov)$_w$*. Before C-initial inflection, there is no misalignment and the MWord is mapped to a recursive PWord: *((amusin)$_w$-ner)$_w$*.

(71) *Hypothetical derivation using recursive PWords*

Input			/amusin -∅$_S$ -ov$_W$/	/amusin -∅$_S$ -ner$_W$/
Cycle 1			(a.mu.sín)$_w$	(a.mu.sín)$_w$
Cycle 2	MORPH	Spell-out suffix	(a.mu.sín)$_w$ /-ov/	(a.mu.sín)$_w$ /-ner/
	PROSODY	Syllabify suffix	(a.mu.sí.n)$_w$-ov	(a.mu.sín)$_w$-ner
		Align PWord with syllables	(a.mu.sín.-ov)$_w$	((a.mu.sín)$_w$-ner)
		Map MWord to PWord		((a.mu.sín)$_w$-ner)$_w$
	PHONO	Stress	(a.mu.sĭn.-óv)$_w$	((a.mu.sĭn)$_w$-nér)$_w$
		DHR	(a.musn.-óv)$_w$	*blocked*
		**blocked in PWord-final σ*		
Output		amusn-óv	amusin-nér	

Moving on to the cophonologies, stress shift applies: *(amusĭn-óv)_w*, *((amusĭn)_w-nér)_w*. This analysis would argue that DHR is a word-level process and it apples inside PWords: *(amusn-óv)_w*. A faithfulness constraint protects destressed high vowels in PWord-final syllables: *((amusin)_w-ner)_w*.[18]

The problem with this analysis is that it argues that DHR is word-level in EArm. But there is evidence against this. DHR does not apply in regularized inflection (72a), inflected adjectives (72b), or loanwords (72c). PWords are a common target for post-lexical rules which can be exceptionless and blind to the diacritics of individual lexemes. But, DHR does not apply to every PWord. Furthermore, post-cyclic lexical word-level rules tend to be exceptionless (Pesetsky 1979). The data show that DHR has not generalized into a post-cyclic word-level process. This is further discussed in §2.7.3 in the context of DHR variation.

(72)   a. manúk   'child'   b. lúrd͡ʒ   'serious'   c. fílm   'film'
           mank-akán   'childish'   lərd͡ʒ-anál   'to get serious'   film-ajín   'cinematic'
           mank-án   'child-GEN (irreg.)'   lurd͡ʒ-í   'serious-GEN'   film-ér   'film-PL'
           manuk-í   'child-GEN (reg.)'

Setting aside the empirical problems above, using recursion is also conceptually problematic with unclear motivation. There is no explicit consensus on the behavior of recursive prosodic constituents. There is debate over whether recursive constituents can trigger categorically different processes vs. gradiently different processes (Ladd 1986; Ito and Mester 2009, 2012, 2013; Wagner 2010; Frota and Vigário 2013; Elfner 2015), whether they can block or trigger lexical processes (Szpyra 1989; Booij 1996; Peperkamp 1997; Raffelsiefen 2005; Kabak and Revithiadou 2009; Bennett 2018), whether they are restricted to the post-lexical phonology of clitics (Inkelas 1989; Booij 1996; Selkirk 1996; Zec 2005; Tyler 2019), and whether they act as diacritics for behaviorally different constituents (Vogel 2009, 2012, 2016; Vigário 2010; Guzzo 2018; Miller 2018, 2020).

To summarize, there is no positive evidence that the relevant prosodic constituent *X* should be the PWord. Treating *X* as a recursive PWord masks the fact that DHR is a lexical process, not a gradient word-level process. Pre-inflectional DHR instead references a prosodic constituent which is bigger than a foot yet smaller than a PWord, and bigger than an MStem yet smaller than the MWord.

2.5.4   Pre-inflectional DHR in the Prosodic Stem

In modern Armenian DHR, the relevant morphological constituent is the MStem and the relevant cophonology is the stem-level cophonology. In this section, I argue that the relevant prosodic constituent *X* is the Prosodic Stem or PStem (Downing 1999a). I first summarize the cross-linguistic evidence for the PStem, and show how the EArm data match that of other languages which have PStems.

2.5.4.1   The role of Prosodic Stems

The traditional prosodic hierarchy assumes only three levels of morphosyntactically derived constituents: the prosodic word, the prosodic phrase, and the intonational phrase (73a). However, there is cross-linguistic

---

[18]This is essentially the same analysis in Macak (2016). The difference is that he doesn't use any strata and he uses weakly bracketed feet (Hyde 2002), e.g., *a₍mu'sin₎-nér⁾*, **a₍mu'si₎n-óv⁾*.

work on morphologically complex languages which argues for a more enriched hierarchy that includes at least one constituent below the PWord: the Prosodic Stem (73b).[19]

(73) a.   b.

$$\text{Intonational Phrase } (\iota)$$
$$|$$
$$\text{Phonological Phrase } (\varphi)$$
$$|$$
$$\text{———-Prosodic Word } (\omega \text{ or PWord)—-}$$

$$\text{Intonational Phrase } (\iota)$$
$$|$$
$$\text{Phonological Phrase } (\varphi)$$
$$|$$
$$\text{———-Prosodic Word } (\omega \text{ or PWord)—-}$$
$$|$$
$$\text{Prosodic Stem } (s \text{ or PStem)}$$
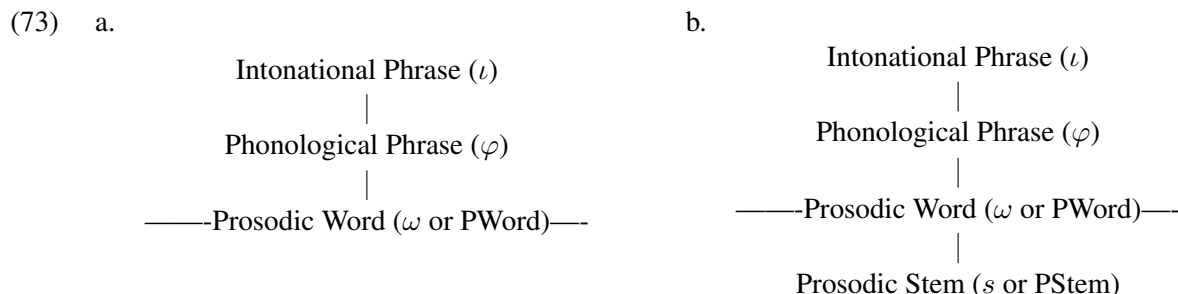
Just as syntactic phrases and morphological words are the sources of prosodic phrases and prosodic words, MStems are the source of PStems. The bulk of the evidence for the PStem comes from agglutinative languages and language families (Czaykowska-Higgins 1997; Downing 2016). The diagnostics used to detect PStems in other languages also show the PStem in Armenian. In Armenian, the overapplication of DHR straddles the boundary between the MStem and V-initial inflection. In other agglutinative languages, stem-level processes have been shown to be sensitive to the syllabification of MStems with V-initial suffixes or C-final prefixes. To illustrate, Aronoff (1988) analyzes reduplication in KiHehe as targetting the morphological stem of the word (in italics). The reduplicant (underlined) is linearly found between inflectional prefixes and the stem (74a). The reduplicant generally copies only stem segments. But when the stem is V-initial, prefix-final consonants are copied because of syllabification (74b).

(74) a. ku-*haata*   'to ferment'   b. kw-*iita*   'to pour'
          ku-haata∼*haata*   'to start fermenting'   kw-iita∼kw-*iita*   'to start pouring'

To explain prefix-copying in V-initial stems, Aronoff (1988) analyzes reduplication as also targeting a prosodic head but does not formalize this concept. Downing (1998b) reanalyzes KiHehe and formalizes the prosodic head as the PStem. It is mapped from the MStem, but it misaligns because of syllabification.

As a constituent, the PStem can be the target of reduplication, tonal processes, vowel harmony, minimality, and other sublexical processes (Downing 1998a, 1999a,b). Further evidence for the PStem as a domain for phonological rules comes from cross-linguistic work on reduplication (Fitzpatrick-Cole 1994; Inkelas and Zoll 2005; Shaw 2005), prefix-suffix asymmetries (Hyman 2008), minimality (Downing 2005, 2006), strata (Inkelas 1989, 1993), problems in bracket erasure (Inkelas 2014), and bracketing paradoxes in compounds (Han 1995). For a summary of the cross-linguistic evidence for the PStem, see Downing (2006, 2016), Downing and Kadenge (to appear).

2.5.4.2 The Prosodic Stem in Armenian

Applying the PStem to Armenian, I argue that MStems are mapped to non-recursive PStems while MWords are mapped to PWords. Before V-initial inflection, the PStem is misaligned from the MStem

---

[19]The Prosodic Root (PRoot) has also been posited as a constituent mapped from morphological roots. The evidence for the PRoots, however, is less than for PStems.

and expands. Before C-initial inflection, the PStem stays isomorphic with the MStem.

(75) *Different PStem structures in the two Armenian dialects.*

Type of structure	Base	Derivative	V-initial inflection	C-initial inflection
Morphology	MWord MStem NOM /amusin/ -∅ ∅	MWord MStem a NOM MStem /amusin/ -∅ /-utjun/ ∅	MWord MStem INST /amusin/ -∅ /-ov/	MWord MStem PL /amusin/ -∅ /-ner/
Prosody	PWord PStem amusín	PWord PStem amusn-utjún	PWord PStem amus(i)n-óv	PWord PStem σ amusin -nér

I argue that the PStem is indexed with its own cophonology. In EArm, the PStem triggers Stress Shift and Destressed High Vowel Reduction (DHR) but not Destressed Diphthong *uj*-Reduction (DDR). This is more than the stem-level cophonology, and less than the word-level cophonology. In WArm, the PStem cophonology is equivalent to the word-level stratum. It triggers stress shift but not reduction.

(76) *Distribution of processes and domains across cophonologies*

Morphological domain	Derivation	V-initial Inflection		Inflection
Relevant constituent	MStems	Misaligned PStems		MWords
Cophonology	Stem-level	PStem-level		Word-level
Dialect Process	Both	EArm	WArm	Both
Destressed Diphthong *uj* Reduction (DDR)	✓	✗	✗	✗
Destressed High Vowel Reduction (DHR)	✓	✓	✗	✗
Stress Shift	✓	✓	✓	✓

The stem-level and word-level strata are morphologically triggered by derivation and inflection. In contrast, the PStem-level cophonology is an intermediate cophonology: it applies after the stem-level and before the word-level when the right prosodic conditions are met. Specifically, it applies when V-initial inflection is added to an MStem and causes the misalignment of the PStem: $(a.mu.sin)_s \rightarrow //(a.mu.si.n)_s\text{-}ov// \rightarrow (a.mu.si.n\text{-}ov)_s$. I stipulate the the PStem-cophonology *only* occurs when we have PStem misalignment.

Without inflection, an MStem is mapped to an isomorphic PStem as a prosodic process.[20] In (77), I show the condensed cyclic derivation of the free-standing root *amusín* and its derivative *amusn-utjún*. Both are uninflected.

---

[20]In the table, the prosodic mapping of a PStem is a separate step or rule in the serial derivation (cf. Selkirk 1980; Nespor and Vogel 1986; Selkirk 1986; Güneş 2015). An alternative parallelist formalization is to adapt constraints from MATCH theory and WRAP theory (Selkirk 2011; Truckenbrodt 1999; Guekguezian 2017a,b) for PStems.

(77)　*Stratal and prosodic derivation of the root* amusin *and its derivative* amusn-utjun

Input				/amusin -∅$_S$ /	Cycle 2	(a.mu.sín)$_s$ - /-utjun$_S$/
Cycle 1	MORPH		Spell-out	/amusin -∅$_S$ /		(a.mu.si.n)$_s$-u.tjun
	PROSODY		Syllabify	a.mu.sin		(a.mu.si.n-u.tjun)$_s$
			Map PStem	(a.mu.sin)$_s$		(a.mu.sĭ.n-u.tjún)$_s$
	PHONO	*SLevel*	Stress	(a.mu.sín)$_s$		(a.mus.n-utjún)$_s$
			DHR			
Output				amusín		amusn-utjún

In Cycle 1, the base is formed. The input MStem is syllabified and mapped to a PStem: *(amusin)$_s$*. The stem-level phonological process of stress assignment applies: *(amusín)*. In Cycle 2, the derivative is formed. The larger MStem is syllabified and prosodified as a larger non-recursive PStem: *(amusín-utjun)$_s$*. The stem-level phonology applies again with stress shift and destressed high vowel reduction: *(amusn-utjún)$_s$*. In Cycle 3, the MWord maps to a PWord for both the base and derivative. The word-level cophonology vacuously applies. The PStem-level cophonology does not apply because there is no misaligned PStem.

Before inflection, the PStem-level cophonology can apply depending on the shape of the suffix and of the PStem. I illustrate below for the different inflected items with V-initial inflection: *amusn-óv* (EArm), *amusin-óv* (WArm), and C-initial inflection: *amusin-nér*.

(78)　*Stratal and prosodic derivation of* amusn-ov (EArm), amusin-ov (WArm), *and* amusin-ner (both)

				EArm /amusin -∅$_S$ -ov$_W$/	WArm /amusin -∅$_S$ -ov$_W$/	EArm &WArm /amusin -∅$_S$ -ner$_W$/
Input						
Cycle 1				(a.mu.sín)$_s$	(a.mu.sín)$_s$	(a.mu.sín)$_s$
Cycle 2	MORPH		Spell-out	(a.mu.sín)$_s$ - /-ov$_W$/	(a.mu.sín)$_s$ - /-ov$_W$/	(a.mu.sín)$_s$ - /-ner$_W$/
	PROSODY		Syllabify	(a.mu.sí.n)$_s$-ov	(a.mu.sí.n)$_s$-ov	(a.mu.sín)$_s$-ner
			Readjust PStem	(a.mu.si.n-ov)$_s$	(a.mu.sin-ov)$_s$	
	PHONO	*PStem-level*	Stress	(a.mu.sĭ.n-óv)$_s$	(a.mu.sĭ.n-óv)$_s$	
			DHR (EArm)	(a.mus.n-óv)$_s$		
	PHONO	*WLevel*	Stress	(a.mu.sn-óv)$_s$	((a.mu.sin-óv)$_s$	(a.mu.sin)$_s$-nér)
Output				amusn-óv	amusin-óv	amusin-nér

Given the base *(a.mu.sín)$_s$*, inflection is spelled out and syllabified in Cycle 2. Before C-initial inflection, the PStem stays isomorphic with the MStem and with syllable boundaries. C-initial inflection is PStem-external: *(a.mu.sin)$_s$-ner*. Before V-initial inflection, the PStem becomes misaligned from syllable boundaries: **(a.mu.si.n)$_s$-ov*. To repair this, the PStem is readjusted by incorporating the V-initial inflectional suffix: *(a.mu.si.n-ov)$_s$*.[21] The PStem cophonology is then triggered for the misaligned PStems. EArm's PStem triggers stress and DHR *amusn-óv*, WArm's PStem only triggers stress *amusin-óv*. The MWord is mapped to a PWord, and the word-level phonology applies.

---

[21]In an earlier analysis (Dolatian 2019a,b), I argued that the PStem contracted in WArm: *(a.mu.si.)$_s$n-ov*. The PStem-level cophonology was the same in both dialects. I argued that PStem expansion would trigger DHR while PStem contraction would not. I have changed the analysis to make the illustration easier. Both analyses work, but the current analysis fits better with the variation data.

2.6 Prosodic stems elsewhere: Appendixes and vowel hiatus

The use of PStems for DHR at first seems stipulative, but there are two other processes in Armenian which give independent support for the PStem: appendix incorporation in both dialects, and vowel hiatus in EArm.

2.6.1 Stem-level appendixes in modern Armenian

In both dialects, complex codas are at most two consonants and have falling sonority (79a). The exception is the morpheme *k* which can follow any cluster of one or two consonants (79b).[22] It can form two (79c) or three (79d) consonant clusters with flat sonority. Transcriptions are in WArm.

(79)   a.   ɡóɣ   'side'   lájn   'wide'   bahántʃ͡   'demand'

         b.   ɡóx-k   'book cover'   lájn-k   'width'   bahántʃ͡-k   'credit'

         c.   bét-k   'need'   daráts͡-k   'expanse'   átʃ͡-k   'eye

         d.   bárt-k   'debt'   várts͡-k   'wages'   ɡúrts͡-k   'breast'

Most instances of *-k* arose from a diachronic reanalysis and bleaching of the Classical Armenian plural suffix *-k* (§2.8.1). Morphologically, the *-k* can act as a nominalizer (79b) or empty morph: *gurts͡-k* 'breast'. Phonologically, *-k* has become an extrasyllabic consonant (Vaux 1998b:83-4; Vaux and Wolfe 2009). It is not extraprosodic; the phonology is sensitive to this *-k* because the *-k* can trigger devoicing: /ɡoɣ-k/→gox-k (cf. voicing assimilation in Polish appendixes: Rubach and Booij 1990; Rubach 1996, 1997).

As an appendix, the *-k* must be parsed into some prosodic constituent. I argue that this constituent is the PStem. In both dialects, *-k* can be found word-finally (80a), before V-initial (80b), and before C-initial inflection (80c). Because *-k* is word-internal, then it is not a PWord-appendix. Vaux (1998b:83-4) argues that *-k* is an appendix at the end of cyclic domains. I analyze this cyclic domain as the PStem-juncture between derivation and inflection. The trees in (80) illustrate this. The *-k* is a PStem-appendix word-finally and before C-initial inflection; it is an onset before V-initial inflection.

(80)   a.   aɣót-k   'prayer'   b.   aɣot-k-í   'prayer-GEN'   c.   aɣot-k-nér   'prayer-PL'
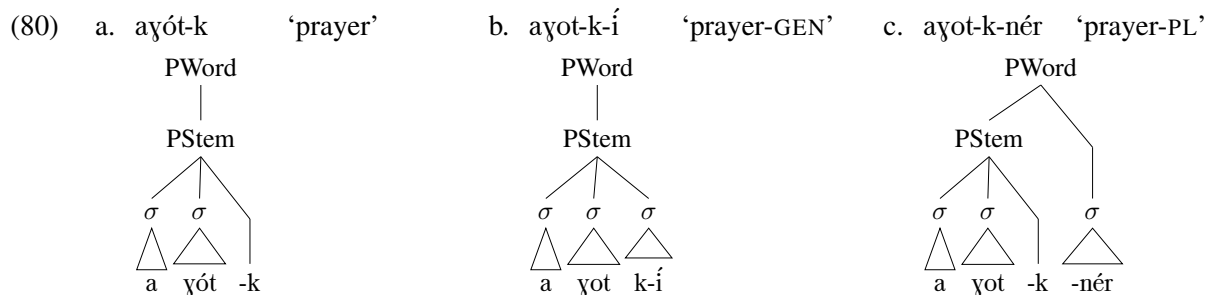
Appendix incorporation thus shows that some prosodic constituent is needed between the MStem and C-initial inflection. I argue that this is the PStem. An alternative is treating *-k* as an appendix to a recursive inner PWord: *((aɣotk)ᵥᵥ-ner)ᵥᵥ*. This would work but it faces the same problems of recursion discussed in §2.5.3. The next section provides more evidence for the PStem that comes from vowel hiatus.

---

[22]The suffix is is aspirated in both dialects, but aspiration is not contrastive in WArm. Three other rare extrasyllabic consonants exist: *s, ʃ, x* in *medaks* 'silk', *jeraʃx* 'guarantee', *naxʃ* 'type of decoration', etc. They are also appendixes.

2.6.2 Stem-final high vowels and vowel hiatus in Eastern Armenian

As explained in §2.4.2, stem-final high vowels do not undergo DHR in WArm. Instead, they undergo different vowel hiatus repair rules in derivation vs. inflection. The same applies in EArm, but EArm shows the overapplication of stem-level hiatus rules in inflection. I focus on polysyllabic roots.[23]

For final /i/, both dialects generally use deletion in derivation. In inflection, hiatus is repaired with glide epenthesis in WArm but with either deletion (81a) or glide epenthesis (81b) in EArm (Margaryan 1997:100ff), though glide-epenthesis is increasingly preferred (Sargsyan 1987:140,197). Just like in WArm (§2.4.2), other hiatus repair rules like coalescence or glide formation can apply in derivation but not inflection.

(81)	a.	WArm	EArm		b.	WArm	EArm	
		jegeɣet͡sí	jekeɣet͡sʰí	'church'		madaní	mataní	'ring'
		jegeɣet͡s-agán	jekeɣet͡sʰ-akán	'ecclesiastic'		madan-él	matan-él	'to seal'
		jegeɣet͡sij-óv	jekeɣet͡sʰ-óv	'church-INST'		madanij-óv	matanij-óv	'ring-INST'

Overapplication is also found for stem-final /u/. Both dialects generally use glide fortition to [v] in derivation. In inflection, WArm uses glide-epenthesis while EArm also uses glide fortition for some (82a) but not all roots (82b) (Minassian 1980:95; Margaryan 1997:103-7).

(82)	a.	WArm	EArm		b.	WArm	EArm	
		tətú	tʰətʰú	'sour'		jergú	jerkú	'two'
		hám	hám	'taste'				
		tətv-a-hám	tʰətʰv-a-hám	'sour-tasting'		jergv-agán	jerkv-akán	'dual'
		tətuj-í	tʰətʰv-í	'sour-GEN'		jerguj-í	jerkuj-i	'two-GEN'

To summarize, some EArm roots show the overapplication of some stem-level hiatus repair rules (deletion, glide fortition) in pre-inflectional vowel hiatus. This is analogous to pre-inflectional DHR. I argue that this due to the PStem. I argue that the PStem expands before V-initial inflection because of prosodic well-formedness, either from resyllabification (for DHR) or because the PStem cannot end with vowel hiatus. PStem expansion triggers certain stem-level rules in EArm. In both dialects, the PStem is also the site of appendix incorporation. In sum, the Prosodic Stem has independent support in Armenian.

2.7 Lexical Variation in reduction

The previous sections focused on the general behavior of cophonologies and of DHR. I posited three cophonologies: the stem-level, PStem-level, and word-level. In this section, I go over the lexical variation in DHR for different lexemes based on their morphological and prosodic context across the two dialects. As a stem-level process, DHR is not exceptionless and its application can vary by root, suffix, and by the specific root-suffix combination, i.e., on a lexeme-by-lexeme basis (Margaryan 1997:102-3). The variation

---

[23]For final /i/ in monosyllabic roots, both dialects resolve hiatus with glide-epenthesis in derivation and inflection: d͡zi 'horse'; d͡zij-avor 'horseman'; d͡zij-ov 'horse-INST (EArm)' (§2.4.2). Some earlier grammars document optional glide formation in EArm: d͡zj-ov (Johnson 1954:20). For final /u/, monosyllabic and polysyllabic roots pattern together.

data shows that the three cophonologies form a monotonic hierarchy and they respect the Strong Domain Hypothesis (Myers 1991). The end-result is support for the stratal and prosodic model.

2.7.1   Variation in pre-derivational DHR in both Armenian dialects

I first focus on the variation of DHR in derivation. The data in this section is transcribed in WArm. The generalizations apply to EArm. Some roots reduce in almost all of their derivatives (83a), in few of their derivatives (83b), or even optionally in some derivatives (83c).

(83)   a. pəlúr   'hill'   b. morúk   'beard'   c. gətúͭts   'beak'
           pəlr-ág   'hillock'   moruk-avór   'bearded'   gətͭts-él   'to peck'
           tͭsév   'shape'   moruk-ód   tͭsév   'shape'
           pəlr-a-tͭsév   'hill-shaped'   gətͭts-a-tͭsév   'beak-shaped'
                                                                                        gətuͭts-a-tͭsév

Suffixes show variation in undergoing DHR (Minassian 1980:45; Margaryan 1997:98). Some suffixes cannot reduce at all (84a).[24] Other suffixes can trigger DHR but generally resist DHR (84b-i) with exceptions after bound roots (84b-ii). The suffix *-k* can't trigger DHR but it generally blocks reduction of any preceding destressed high-vowels (84c-i), with some exceptions (84c-ii).[25] Certain suffixes like *-ik* are diachronically derived from a suffix cluster *-i-k* (J̌ahowkyan 2010); it does not undergo reduction (84c-iii).

(84)   a. *Suffixes which resist DHR*
           i. ʃárʒ   'motion'   ii. mərtͭs-íl   'to compete'
               ʃarʒ-úm   'motion, action'   ii. mərtͭs-úm   'competition'
               ʃarʒ-umn-ajín   'mobile, of movement'   ii. mərtͭs-umn-ajín   'of competitions'

       b. *Suffixes which trigger but generally resist DHR*
           i. hád   'piece'   ii. tapanͭts-él   'to penetrate'
               had-íg   'grain'   tapanͭts-íg   'transparent'
               had-ig-avór   'granular'   tapanͭts-ɡ-utjún   'transparency'

       c. *Suffixes which block DHR*
           i. tír   'posture'   ii. kír   'character'   iii. gárd͡z   'opinion'
               tír-k   'position'   kír-k   'book'   gard͡z-í-k   'thought'
               tir-k-ajín 'positional'   dún   'house'   gard͡z-i-k-agán 'suspicious'
                                                 kər-k-dún 'publishing house'

Because DHR is stem-level, we expect many exceptions and lexical variation (Kiparsky 1982b). This argues against treating DHR as word-level in EArm (§2.5.3.2). Because the word-level is postcyclic, we would except much fewer variation than we actually see.

---

[24]The nasal *n* in *ʃarʒ-um**n**-ajin* is a relic of nasal ∼ ∅ alternations from Classical Armenian, similar to nasal deletion in English *dam**n**∼damnation*.

[25]Interestingly, some grammars state that *-k* likewise resists reduction in EArm derivation and inflection: *dír-k* 'position'; *dir-k-ajín* 'positional'; *dir-k-óv* 'position-INST (41) (Minassian 1980:45; Margaryan 1997:99), though exceptions exist: *bəɾúntͭsʰkʰ* 'fist'; *bəɾəntͭsʰkʰ-a-márt* 'fist fight'; *bəɾəntͭsʰkʰ-óv* 'fist-INST'.

2.7.2 Variation in pre-inflectional DHR in Western Armenian

I next turn to the overapplication of DHR in Western Armenian inflection. In WArm, inflection generally blocks DHR. The exception is a small closed set of high-frequency roots (85) which show *optional* reduction in some of their inflected forms (Avetisyan 2007:83). The reduced form is found in idiomatic phrases.

(85) sírd   'heart'   mítk   'mind'
      sird-í   'heart-GEN'   mitk-é   'mind-ABL'
      sərd-í-t t͡ʃáp 'heart-GEN-2SGPOSS size'   mətk-é-t hané 'mind-ABL-2SGPOSS remove.IMP'
             literally... 'your heart's size'   *literally...* 'remove from your mind'
             as in... 'to your heart's content' *as in...* 'get it out of your head'

Interestingly, most of these roots are monosyllabic and thus do not take C-initial inflection (§2.3.3). One exception is the common polysyllabic root *jergir*. In WArm, it can optionally reduce before V-initial inflection but not C-initial inflection: *jerg(i)r-í, jergir-nér*. Another high-frequency word *jergin-k* behaves the same. This is analogous to DHR in EArm.

(86) *Inflection and reduction in the words* jergir 'land, world, country' *and* jergin-k 'sky, heaven'

Base		V-initial Inflection						C-initial Infl.	
		DAT/GEN		ABL		INST		PL	
jergír	jergín-k	jergir-i	jergin-k-i	jergir-é	jergin-k-é	jergir-óv	jergin-k-óv	jergir-nér	jergin-k-nér
		jergr-í	jergən-k-í	jergr-é	jergən-k-é	?jergr-óv	jergən-k-óv	*jergər-nér	*jergən-k-nér

As for irregular inflection, the DAT/GEN has a handful of allomorphs for irregular nouns (87), including ablaut (87c) (Hagopian 2005:350ff; Dum-Tragut 2009:68ff; Oyer 2017).

(87) a. ór   'day'   b. əngér   'friend'   c. hájr   'father'
       or-ván   'day-GEN'   ənger-ót͡ʃ   'friend-GEN'   hór   'father-GEN'

Irregular inflection sometimes shows the overapplication of DHR. There are a few irregular roots (88a,88b) and derivational suffixes (88c) with high vowels. Prescriptively they should take only irregular case suffixes, e.g., irregular genitive *-an*. Irregular inflection triggers DHR. But these irregular classes can optionally take regular inflection, e.g., *-i*. When they do, there is no reduction.

(88)   a. 'girl'   b. 'spring'   c. 'promise'
      Base   axt͡ʃíg   karún   xost-úm
      Irregular genitive   axt͡ʃəg-án   karn-án   xost-m-án
      Regular genitive   axt͡ʃig-í   karun-í   xost-um-í

The above facts reinforce the stratal account given for Western Armenian. Cross-linguistically, irregular or high-frequency inflected words can show traces of stem-level phonology (Kiparsky 1982b), i.e., DHR. What matters is that pre-inflectional DHR does not extend to the rest of the WArm lexicon. Furthermore, the fact that DHR in some of these exceptional words distinguishes between V-initial and C-initial inflection (86) likewise gives support to the role of the PStem.

2.7.3 Variation in pre-inflectional DHR in Eastern Armenian

Moving on to Eastern Armenian, DHR generally applies in derivation, V-initial inflection, but not C-initial inflection. But like pre-derivational DHR in Western Armenian (§2.7.1), not all derivatives and inflected forms show DHR in EArm (Minassian 1980; Sargsyan 1987; Margaryan 1997). Depending on the root, we find the following patterns: A root can reduce in derivation and V-initial inflection (89a), in derivation but not inflection (89b), or in neither derivation nor inflection (89c).[26]

(89) a. mírg   'fruit'   b. níst   'seat'   c. kírtʃ   'canyon'
         mərg-avet   'fruit-bearing'   nəst-él   'to sit'   berán   'mouth'
         mərg-í   'fruit-GEN'   nist-i   'seat-GEN'   kirtʃ-a-berán 'canyon-end'
                                                                      kirtʃ-í   'canyon-GEN'

There can also be optional reduction. There are roots which reduce in derivation and optionally in V-initial inflection (90a), optionally in both (90b), and optionally in derivation but not in inflection (90c).

(90) a. kamúrd͡ʒ   'bridge'   b. mat͡sún   'yoghurt'   c. xəlúrtʰ   'mole'
         kamərd͡ʒ-ák   'small bridge'   mat͡sun-ót   'yoghurt-filled'   xəlurtʰ-ení   'moleskin'
         kamurd͡ʒ-í   'bridge-GEN'   mat͡sən-ót   xəlórtʰ-ení
         kamərd͡ʒ-í   mat͡sun-í   'yoghurt-GEN'   xəlurtʰ-í   'mole-GEN
                                           mat͡sən-í

Furthermore, only two roots can reduce before C-initial inflection in standard EArm (91a) (Margaryan 1997:96). These roots also reduce in derivation and in V-initial inflection.[27] A small handful of other high-frequency roots can reduce in colloquial non-standard EArm (91b) (Margaryan 1997:104). For both types of roots, reduction applies everywhere: DHR is stem-, PStem-, and word-level for these roots.

(91) a. i. jerkír   'world'   ii. skízb   'beginning'
             jerkr-ajín   'earthly'   skəzbn-akán 'initial'
             jerkr-í   'world-GEN'   skəzb-í   'beginning-GEN'
             jerkər-nér   'world-PL'   skəzb-nér   'beginning-PL'

      b. i. ʒoɣovúrd   'populace'   ii. serúnd   'generation'   iii. kerakúr   'food'
             ʒoɣovərd-nér 'populace-PL'   serənd-nér 'generation-PL'   kerakər-nér 'food-PL'

Native grammars (Aṙak'elyan 1955; Sargsyan 1987; Margaryan 1997) note that variation in pre-inflectional DHR in EArm is correlated with multiple factors: frequency, dialectal vs. formal style, syllable count, syllable shape, vowel quality, part of speech, irregular inflection, and individual derivational suffixes. But despite all this variation, pre-inflectional DHR is historically robust in EArm as shown in corpus studies of early modern Eastern Armenian (Sargsyan 1987). Lexicographic and corpus data and tools are, however, too limited to conduct an extensive statistical study.

---

[26]A reviewer notes a useful minimal pair: *nist-í* 'seat-GEN' and *nəst-í* 'to sit (3SG.SUBJ)'. The word *nəst-í* is an inflected form of the verb *nəst-él*. The theme vowel *e* is replaced by verbal suffix *-i* which is homophonous with the case marker *-i*. See §2.3.3.

[27]The pre-suffixal nasal [*skəzbn-akán*] 'initial' is a relic of Classical Armenian, see footnote 24.

2.7.4 Variation in Eastern Armenian as a cue to stem-level status

For EArm, I argued that DHR is a stem-level process and that it applies in V-initial inflection because DHR is also a PStem-level process. But because of the above variation, one could instead argue that DHR is word-level and that it underapplies in C-initial inflection because of some prosodic requirement. But this is unlikely for three reasons which I previewed in §2.5.3.2. First, like WArm, regularization blocks DHR in EArm inflection (92). Irregular nouns can reduce in derivation and in irregular V-initial inflection, but they generally do not reduce when they optionally take regular V-initial inflection (Sargsyan 1987:156,172).

(92) a. manúk   'child'   b. dikín   'lady'
         mank-akán   'childish'   dikn-ík   'doll'
         mank-án   'child-GEN (irreg.)'   dikn-ódʒ̂   'lady-GEN (irreg.)'
         manuk-í   'child-GEN (reg.)'   dikin-í   'lady-GEN (reg.)'

Second, adjectives generally resist pre-inflectional DHR (93). Adjectives reduce in derivation. When inflected, they are undergo zero-conversion to become nominals. They generally do not reduce in inflection (Minassian 1980:83; Margaryan 1997:106). Non-application of DHR is more common for adjectives with destressed *u* than with destressed *i*.

(93) a. xít   'dense'   b. lúrdʒ̂   'serious'   c. amúr   'hard'
         xət-akán   'condensable'   lərdʒ̂-anál   'to get serious'   amr-ótsʰ̂   'fortress'
         xit-í   'dense-GEN'   lurdʒ̂-í   'serious-GEN'   amur-í   'hard-GEN'
                       =of (a) dense (one)   =of (a) serious (one)   =of (a) hard (one)

Third, DHR is limited to native words (94). Borrowings do not reduce in derivation or inflection (Xačatryan 1988:35,59; Margaryan 1997:95-9). The fact that loanwords behave differently than native words is not surprising (Itô and Mester 1999).

(94) a. tʰím   'team'   b. fílm   'film'   c. tʰenís   'tennis'
         tʰim-akítsʰ̂   'team-mate'   film-ajín   'cinematic'   tʰenis-a-xáɣ   'tennis game'
         tʰim-ér   'team-PL'   film-ér   'film-PL'   tʰenis-í   'tennis-GEN'

All the above is evidence that DHR is stratally deeper than the word-level. Because the word-level is postcyclic, a word-level treatment would predict that DHR is not sensitive to morpheme boundaries or to lexical classes, but that it applies generally across all or most PWords. In other words, word-level rules then to be exceptionless or have relatively few exceptions (Pesetsky 1979). This is false or DHR.[28]

2.7.5 Monotonicity in the variation of DHR across Armenian

Combining all the above data, it is clear that root-based, suffix-based, and lexeme-based variation exist in pre-derivational and pre-inflectional DHR across the two dialects. The existence of variation in morphophonological

---

[28]Further evidence comes from heritage speakers of Eastern Armenian. For heritage speakers, DHR tends to not apply in V-initial inflection (Karapetian 2014:79ff).

processes is not surprising (Inkelas et al. 1996; Anttila 2002, 2006; Wolf 2011). In my framework, variation means that roots, suffixes, and lexemes can (de-)activate rules/constraints in different cophonologies. I formalize the variation in this section.

The variation data across the two dialects showed certain subregularities. These subregularities indicate a system of implicatory relations (95). These implications support the stratal model because we do not find roots which reduce in inflection (word-level) more than in derivation (stem-level), i.e., there are no roots which reduce 1) in inflection but not derivation, 2) obligatorily in inflection but optionally in derivation, or 3) optionally in inflection but never in derivation. The data also support the prosodic model because we do not find roots which reduce in C-initial (PStem-external) but not V-initial inflection (PStem-internal).

(95)   *Implicatory relations in variation of DHR in Armenian*

	DHR in	DHR in	DHR in
Morphological domain	C-initial Inflection implies...	V-initial Inflection implies...	Derivation
Lexical cophonology	Word-level		Stem-level
Prosodic cophonology		PStem-level	

The variation data are monotonic (Graf 2019a). If DHR applies in some domain or cophonology, then it also applies in smaller ones. If DHR does not apply in some domain or cophonology, then it does not apply in larger ones. This corroborates the Strong Domain Hypothesis (Myers 1991; McPherson and Hayes 2016). The variation data indicate that the PStem-level cophonology is an intermediate cophonology between the stem-level and word-level. This is analogous to how the word-level is an intermediate cophonology between the stem-level cophonology and the post-lexical phrase-level cophonology.

In a constraint-based system, this monotonicity can be formalized by restricting possible re-rankings across the hierarchy of cophonologies. Let F, M be a faithfulness and markedness constraint which respectively block and trigger DHR. M outranks a faithfulness constraint F in a given domain. In larger domains, F can get promoted but it cannot get demoted; similarly, M can get demoted but not promoted. Applying this to Armenian, the implicatory relations can be summarized by the constraint rankings below. These constraints are indexed to individual lexemes or suffixes in order to model variation.

(96)   *Monotonicity in DHR variation as constraint re-ranking*

	Lexemes which reduce...			
	a. everywhere	b. in Derivation and V-Inflection	c. in Derivation	d. Nowhere
Cophonology				
Stem-level	$M \gg F$	$M \gg F$	$M \gg F$	$F \gg M$
PStem-level	$M \gg F$	$M \gg F$	$F \gg M$	$F \gg M$
Word-level	$M \gg F$	$F \gg M$	$F \gg M$	$F \gg M$
Example:	some EArm roots (91)	most EArm lexemes some WArm roots (86) irregular inflection (88)	most WArm lexemes some EArm roots (89b) regularized inflection (92)	loanwords (94) some roots (83b,89c) some suffixes (84)

For most words in both dialects, DHR is active in the stem-level ($M \gg F$) while inactive in the word-level ($F \gg M$). But for the PStem-level, DHR is active in EArm ($M \gg F$, 96b) while inactive in WArm ($F \gg M$, 96c). In between, variation is modeled by different lexeme-specific rankings of M or F. Some WArm

lexemes pattern like EArm with $M \gg F$ in the PStem (96b). This means that DHR applies in derivation, V-initial inflection, but not C-initial inflection. In contrast, some EArm lexemes pattern like WArm by ranking $F \gg M$ in the PStem (96c). This means that DHR applies in derivation, but not inflection. At the extremes, a given lexeme may specify the ranking $M \gg F$ in all the cophonologies, triggering DHR everywhere (96a). Another lexeme may specify the ranking $F \gg M$, blocking reduction everywhere (96d).

In sum, the variation data support the existence of three separate but interrelated cophonologies: the stem-level, PStem-level, and word-level. These cophonologies respect the Strong Domain Hypothesis because they form a monotonic hierarchy with restrictions on possible re-rankings. Between any two levels, DHR can only get turned off, not turned on. There also no lexemes which do not reduce in derivation ($F \gg M$), but do reduce in inflection ($M \gg F$); nor are there lexemes which don't reduce in V-initial inflection ($F \gg M$), but do reduce in C-initial inflection ($M \gg F$). These rerankings are banned because they violate the Strong Domain Hypothesis by demoting faithfulness constraints in later levels.

2.8 Domain narrowing from Classical to modern Armenian

The previous sections established the stratal and prosodic systems for the two modern Armenian dialects, mostly based on vowel reduction. One question is why the two dialects differ in the domain of DHR. In this section, I explain this difference by analyzing the history of reduction in Classical Armenian (CArm), the earliest attested variety of Armenian ($\sim$ 5th century AD). I show how various destressed reduction processes underwent domain narrowing from the word-level in CArm to the stem-level in WArm or fossilization (§2.8.1). The analysis highlights the role of the phonological life-cycle as the origin of modern Armenian's strata. I speculate on what morphological factors encouraged domain narrowing from Classical to Modern Armenian (§2.8.2). I argue that these factors led to the emergence of the PStem cophonology (§2.8.3).

2.8.1 Destressed Reduction in Classical Armenian is word-level

Like Modern Armenian, CArm had final stress and alternations between stressed and destressed high vowels (97a,97b) and diphthongs (97c). The Modern Western Armenian diphthong [uj] is a reflex of Classical [oj]. Like in Modern Armenian, these alternations in CArm showed a Derived Environment Effect (Macak 2017:1071). But unlike in Modern Western Armenian, these alternations were triggered by stress shift before both derivational and *inflectional* suffixes (97) (Godel 1975:12; Hammalian 1984:93ff; Thomson 1989:15ff; Beekes 2003:155ff; Matasović 2009:93ff; DeLisi 2015:33, 2018:110; Macak 2016:11ff, 2017:1045ff).[29]

(97)	a.	CArm	WArm		b.	CArm	WArm		c.	CArm	WArm	
		gír	kír	'letter'		d͡ʒúr	t͡ʃúr	'water'		lójs	lújs	'light'
		gər-él	kər-él	'to write'		d͡ʒər-akán	t͡ʃər-agán	'aquatic'		lus-awór	lus-avór	'luminous'
		gər-oj	kir-é	'letter-ABL'		d͡ʒər-oj	t͡ʃur-é	'water-ABL'		lus-oj	lujs-é	'light-ABL'

[29]To understand the different pronunciations between Classical and Western Armenian, it should be noted that Western Armenian underwent a series of consonant voicing and aspiration shifts from Classical Armenian (Baronian 2017). CArm had a three-way laryngeal contrast T^h-T-D while WArm only has a two-way one: T^h-D. This change did not affect stress and reduction. All data is taken from the sources cited in this section. When needed, data was supplemented with CArm dictionaries from `www.nayiri.com` and paradigms from Sterling (2004) and `https://lrc.la.utexas.edu/eieol/armol`. I transcribe CArm data with aspiration because aspiration is contrastive in this CArm. I do not mark aspiration in the WArm entries because aspiration is not contrastive in WArm.

Thus, Destressed High Vowel and Diphthong *uj*-Reduction were stem-level and word-level in Classical Armenian. The change from CArm to WArm involved narrowing the domain of destressed reduction to the stem-level. This follows from the life-cycle of phonological processes (Bermúdez-Otero and Trousdale 2012; Bermúdez-Otero 2014; Ramsammy 2015).

Further evidence of domain narrowing comes from fossilized alternations. In Classical Armenian, other vowels productively participated in destressed reduction: *ja* to *e*, and long *ē* to *i*.[30] But in modern Western Armenian, these alternations have become fossilized in a handful of derivatives. In CArm, the destressed non-high diphthong [ja] would reduce to [e] in derivation and inflection (98). But in modern WArm, the reduction of [ja] is no longer productive. A small closed set of (often religious) derivatives show *ja*-reduction (98a), but other derivatives do not (98b). There is no reduction in inflection (Minassian 1980:43; Xačatryan 1988:67; Margaryan 1997:112; Avetisyan 2011:82).

(98) a. CArm   WArm   b. CArm   WArm
          aṙakʰjál   arakjál   'apostle'   senják   senjág   'room'
          aṙakʰel-agán   arakel-akán   'apostolic'   senek-ík   senjag-íg   'little room'
          aṙakʰel-í   arakjal-í   'apostle-GEN'   senek-í   senjag-í   'room-GEN'

Another remnant of *ja*-reduction is the derivational suffix *-utjun* (99a). In Classical Armenian, the genitive of *-utjun* is formed by ablaut *-ytjan* (99b). The ablative is formed by the adding suffix *-ē* to the genitive form *-utjan* (99c). This triggers *ja*-reduction to *-ten-e*. In Modern Armenian, all these different forms of *-utjun* have been reanalyzed as either suppletive allomorphs or morpheme-specific rules which are conditioned by case. The suffix *-utjun* can optionally take regular inflection without any reduction.

(99) a. *Base for* 'happiness'   b. *Genitive*   c. *Ablative*
          CArm   WArm   CArm   WArm   CArm   WArm
          urax-utʰjún   urax-utjún   urax-utʰján   urax-utján   urax-utʰen-é   urax-uten-é
                                                        urax-utjun-í   urax-utjun-é

Besides *ja*-reduction, Classical Armenian distinguished between short and long front mid-vowels: /e/ vs. /ē/.[31] These two segments were phonemic and minimal pairs can be found (100). They are also spelled differently with different graphemes: <ե> for /e/ and <է> for /ē/. The short mid-vowel /e/ never reduced (100a), while the long vowel /ē/ would reduce to [i] under stress shift, both in derivation and inflection (100b).

(100) a. CArm   WArm   b. CArm   WArm
           sér   sér   'race; progeny'   sḗr   sér   'love'
           ser-él   ser-él   'to beget'   sir-él   sir-él   'to love'
           ser-ój   ser-í   'progeny-GEN'   sir-ój   ser-í   'love-GEN'

---

[30]Beekes (2003:147) argues that this diphthong was likely a vowel-vowel sequence [ea] instead of a glide-vowel sequence [ja]. Some treat the *ē*~*i* alternation as a form of destressed raising (Hammalian 1984) instead of destressed reduction (Macak 2017). Putting aside the issue of terminology, what matters is that the alternation is due to vowel becoming destressed.

[31]Because there is no data on spoken Classical Armenian, it is unknown how the two mid-vowels were phonetically different. Traditionally, the reducible vowel is transliterated as a long vowel *ē*. Diachronically, the long *ē* is a reflex of a diphthong [ei] from Proto-Indo-European (Godel 1975:6; Macak 2017:1066). Synchronically, some treat the *ē* as tense (Hammalian 1984:18; DeLisi 2015:6), closed (Godel 1975:6; Thomson 1989:14; Matasović 2009:6), a surface diphthong [ei/ej] (Beekes 2003:146), or an underlying diphthong /ei,ej/ (Vaux 1998b:19).

In modern Armenian, the two mid-vowels have collapsed into a single lax mid-vowel phoneme /e/. This creates many cases of homophonous roots: *ser* 'love; race' (100). A handful of core frequent roots inherited the alternating vowel from Classical Armenian and show destressed *ē*-reduction in their derivatives. Reduction does not apply in inflection (101a), except for frozen idioms or social phrases which often include irregular inflection (101b) (Margaryan 1997:89-93; Sowk'iasyan 2004:49-50; Avetisyan 2011:9,61). The data below is from WArm.

(101) a. i. gés   'half'   ii. véb   'novel'   iii. nəvér   'gift'
             gis-él   'to divide'   vib-él   'to narrate'   nəvir-él   'to gift'
             ges-óv   'half-INST'   vib-óv   'novel-INST'   nəver-óv   'gift-INST'

       b. i. sér   'love'   ii. dér   'lord'
             sir-él   'to love'   dir-él   'to master'
             ser-óv   'love-INST'   der-í̇   'lord-GEN'
             sir-óv   *as in* 'with my warm regards'   dir-ót͡ʃ   *as in* 'of the Lord, our God'

Furthermore, both *ja-* and *ē*-reduction are also fossilized in EArm.

(102) a. aṙakʰjál   'apostle'   b. sér   'love'
             aṙakʰel-akán   'apostolic'   sir-él   'to love'
             aṙakʰjal-í̇   'apostle-GEN'   ser-óv   'love-INST'

To my knowledge, grammarians agree that *ja-* and *ē*-reduction are not productive in Modern Armenian (Dum-Tragut 2009). These two reduction patterns are fossilized in a finite handful of words and their derivatives. Because they are highly morpheme-specific, they are likely a case of stem-allomorphy instead of the product of synchronic rules/constraints (Haugen 2016). Thus, these two processes have reached one of the last stages of domain-narrowing in the phonological life-cycle (Bermúdez-Otero and Trousdale 2012).

2.8.2 Morphological change and confounds in syllabification

The previous section that many reduction processed narrowed in scope from the word-level in CArm to the stem-level in WArm. In this section, I argue that contributing factors to the domain narrowing are the significant changes in Armenian inflection. I later use these changes to explain how DHR diverged between EArm and WArm, and how they created the PStem.

Nominal inflection in Classical Armenian was largely syncretic and fusional with the same suffix encoding number and case (Adjarian 1909:5, Halle and Vaux 1998:15; Donabédian 2000; Caha 2013; Sayeed and Vaux 2017:1154). But in modern Western Armenian, nominal inflection became agglutinative and less syncretic, with separate morphemes for number and case. The partial paradigms in (103) illustrate this for regular inflection.[32]

---

[32]For simplicity, I omit the locative because it hasn't survived into Western Armenian, but it has survived into Eastern Armenian (§2.5.1). The segment *-o-* can be segmented as a nominal theme vowel (Halle and Vaux 1998); these theme vowels have not survived into modern Armenian nominal inflection as separate morphs.

(103) *Regular nominal inflection in Classical and Modern Western Armenian for the word 'mouth'*

		NOM	ACC	DAT	GEN	ABL	INST
CArm	Singular	beran	beran	beran-oj	beran-oj	beran-oj	beran-ow
	Plural	beran-k	beran-s	beran-ot͡sʰ	beran-ot͡sʰ	beran-ot͡sʰ	beran-ow-k
WArm	Singular	peran	peran	peran-i	peran-i	peran-e	peran-ov
	Plural	peran-ner	peran-ner	peran-ner-u	peran-ner-u	peran-ner-e	peran-ner-ov

By becoming agglutinative, nominal inflection became more *morphologically* distinguishable from derivation. This would make it easier to treat inflection as also *phonologically* distinguishable from derivation. An additional change is the introduction of the stressable C-initial inflectional suffix *-ner*. Unlike the modern dialects, CArm nominal inflection *lacked* a stressable C-initial suffix which contains vowels.[33] It only had unstressable lone-consonant suffixes: *-k, -s*. In the next section, I argue that the creation of this new contrast between C-initial and V-initial inflection in modern Armenian plays a role in the development of DHR in Eastern Armenian.

2.8.3   Incomplete narrowing and the Prosodic Stem

This section synthesizes the data from this chapter in order to explain the microvariation of the two dialects as the consequence of morphological, prosodic, and historical factors. Three prosodic processes were studied: Stress Shift, Destressed High Vowel Reduction (DHR), and Destressed Diphthong *uj*-Reduction (DDR). As shown in (104), these processes have different morphological domains in the three Armenian lects: Classical (CArm), modern Western (WArm), and modern Eastern (EArm).[34] These different morphological domains correspond to different cophonologies. The narrowing of reduction processes from CArm to the modern dialects is predicted from the life-cycle of phonological processes (Ramsammy 2015).

(104) *Domain of prosodic processes across Classical, Western, and Eastern Armenian*

Process	Lect	Morphological Domain				Cophonology
		Derivation	V-initial Infl.	C-initial Infl.	Clitics	
Stress	CArm	✓	✓		✗	word-level
	EArm	✓	✓	✓	✗	stem-, PStem-, word-level
	WArm	✓	✓	✓	✗	stem-, PStem-, word-level
i,u Reduction	CArm	✓	✓		✗	word-level
	EArm	✓	✓	✗	✗	stem-, PStem-level
	WArm	✓	✗	✗	✗	stem-level
uj Reduction	CArm	✓	✓		✗	word-level
	EArm	✓	✗	✗	✗	stem-level
	WArm	✓	✗	✗	✗	stem-level

---

[33]On the surface, aorist formation in CArm verbal conjugation creates C-initial suffixes: *sir-e-t͡sʰ-i* 'I loved'. But see Hammalian (1984:217) and Macak (2016:205) on how this is actually derived from an underlying V-initial suffix /-it͡sʰ/. Similar V-initial analyses are also extended to other apparent C-initial suffixes in the subjunctive (Hammalian 1984). Besides, verbal inflection which is morphologically stem-based in Armenian

[34]The sources listed in §2.8 do not explicitly state that clitics do not trigger stress shift or reduction in CArm. I assume they do not because I have found no mention of it.

Classical Armenian did not show any word-internal stratification. I assume it only has one cophonology: a cyclic word-level cophonology. This cophonology has to be cyclic because of destressed reduction. Final stress has been constant in the three lects: it applies in every cophonology at every cycle. Reduction was word-level in CArm but it has narrowed in scope in the modern dialects. For most lexemes in WArm, both DHR and DDR have completely narrowed from the word-level to the stem-level. But in EArm, narrowing is partial. DDR has completely narrowed to the stem-level, but DHR occupies an intermediate zone: it applies in derivation, V-initial but not C-initial inflection. I modeled this zone as a separate cophonology: the PStem-level. This cophonology is triggered by prosodic misalignment of the Prosodic Stem.

The question now is why DHR did not completely narrow down to the stem-level. The answer is the interaction of diachronic change and prosody. Domain narrowing from CArm to the modern dialects was confounded with syllabification. Classical Armenian nominal inflection *lacked* a stressable C-initial suffix which contains a vowel, while modern Armenian developed one: the plural -*ner*.[35] I argue that this confound caused the emergence of the Prosodic Stem, i.e., the prosodic constituent which is mapped from misaligned (resyllabified) MStems before V-initial inflection. CArm only had V-initial inflection. When EArm learners developed a C-initial suffix, they reanalyzed DHR as applying in V-initial inflection, not C-initial inflection. They narrowed DHR down to the PStem-level. In contrast in WArm, the PStem was still created; but, the above confound was ignored and DHR completely narrowed to the stem-level.

2.9 Conclusion

This chapter examined destressed high vowel reduction (DHR) in modern Armenian. I showed that DHR is cyclic; and it is sensitive to difference diachronic, morphosyntactic, and prosodic factors across different Armenian lects. Briefly, these factors are the existence of two lexical strata: the stem-level and word-level. These two morphologically-derived strata exist alongside the Prosodic Stem and its prosodically-derived cophonology.

DHR was word-level in Classical Armenian. Over the millennia, reduction underwent domain narrowing and is now a stem-level rule in modern Western Armenian. But in Eastern Armenian, confounds in strata and prosody cause reduction to apply in derivation, V-initial inflection, but not C-inflection. I argued that this is because the dialects developed a new prosodic constituent: the Prosodic Stem (PStem). The PStem is straddled between derivational morphology and vowel-initial inflectional morphology. It triggers its own cophonology. In Eastern Armenian, reduction narrowed down to the stem-level and PStem-level cophonologies. The PStem-level cophonology is triggered by the prosodic misalignment of morphological stems and syllables before V-initial inflection.

All of the above factors require a model of the morphology-phonology interface such that i) it cyclically creates prosodic structure and ii) rule application is sensitive to this prosodic structure. Previous work has shown that lexical phonology and prosodic phonology are partially sensitive to each other (Booij and Rubach 1984; Nespor and Vogel 1986; Szpyra 1989; Cohn 1989; Inkelas 1989, 1993; Booij and Lieber 1993; Hall 1999). What Armenian shows is that prosodic constituents can trigger their own cophonology and trigger cyclic processes (cf. Mansfield 2017).[36]

---

[35]This suffix appeared sometime during medieval or Middle Armenian (Karst 1901). Future work will examine the stratal phonology of Middle Armenian.

[36]An open question is if this PStem analysis can extend to superficially similar cases of conflicts between stem-level strata and

2.A   Diachronic origins of destressed reduction

As shown, destressed reduction is pervasive in Classical and Modern Armenian. The difference between them is that destressed reduction narrowed in scope, ubiquity, and domain during the change from CArm to modern Armenian. But the question is, how did destressed reduction appear in the first place? The most likely answer is phonological reanalysis due to opacity (cf. similar developments in Korean: Cho 2009).[37]

There is a substantial body of diachronic work on what possible *segmental* sound changes occurred from Proto-Indo-European to Classical Armenian via Proto-Armenian (Ačaṙyan 1971; Kortlandt 2003; Beekes 2003; Macak 2017). But, there is fewer work on *prosodic* changes besides the emergence of final stress (DeLisi 2015, 2018; Macak 2016). Because of the lack of data on Proto-Armenian, it is difficult to determine how destressed vowel reduction became a phonological process in CArm in the first place.

Ačaṙyan (1971:329) speculates that there was a connection between Classical Armenian *destressed* reduction and a diachronic process of *unstressed* reduction from Proto-Indo-European to Classical Armenian (Beekes 2003:156;Macak 2016:27). Unstressed high vowels were deleted or replaced by schwas while unstressed midvowels and diphthongs reduced to high or midvowels. I illustrate this below, using PIE reconstructions for roots from J̌ahowkyan (2010) and of the verbal suffix *-em* from Olsen (2017). Relevant vowels are in bold and underlined.

(105)   *Distribution of high vowels in the change from PIE to Classical Armenian*

	Surface high vowels which...		
	... don't alternate	...alternate under stress shift	
	a. Root	b. Root	c. Derivative
PIE	*ue**i**keros	*u**e**ro	*u**e**ro + *-im
CArm	gi**ʃ**er	k**i**r	k**ə**r-em
Gloss	'night'	'letter'	'I write'

Building off of Ačaṙyan, it is possible that because of rampant unstressed reduction, CArm speakers were exposed to surface unstressed high vowels which *diachronically* derived from underlying unstressed diphthongs: *giʃér* (105a). But *synchronically*, there was no evidence for them being anything besides unstressed high vowels. In contrast, this diachronic reduction process created alternating pairs of stressed high vowels and their unstressed schwa/deleted counterparts: *kir ∼ kər-él* (105b-c). In order to capture this alternation, speakers had to posit a reduction rule. In order to prevent the rule from over-applying in surface unstressed high vowels, speakers analyzed this rule as targeting only *destressed* vowels.

The above scenario is simplified for illustration. The origins of reduction get complicated by postulated orderings of diachronic sound changes. Ravnæs (1988, 2005) argues that unstressed high vowel reduction preceded unstressed diphthong reduction which preceded an influx of Parthian loans with unstressed high vowels. Ultimately though, the synchronic answer is still reanalysis due to opacity

A similar reanalysis happened in Romanian. Romanian has a process of destressed *a*-raising to [ʌ] (Steriade 2008a:4). It targets destressed (106a) but not unstressed *a* (106b). Suffixes which don't trigger

---

syllabification-induced processed, e.g., in Kashaya (Buckley 2017).

[37]My gratitude to Christina Bethin, Ricardo Bermúdez-Otero, and Donca Steriade for discussing the diachronic data.

stress shift, don't trigger *a*-raising (106c-i; stress-taking suffixes do (106c-ii). Raising applies before derivation and inflection (Steriade, p.c.).

(106)   a.   sáni-e   'sled'   pomád-ʌ   'pomade'   bárb-ʌ   'beard'
             sʌni-útsʌ   'sled-DIM'   pomʌd-uí   'apply'   bʌrb-ós   'bearded-MASC'
        b.   farfurí-e   'plate'   kartóf   'potato'   papúk   'slipper'
             farfuri-útsʌ   'plate-DIM'   kartof-jór   'potato-DIM'   paputʃ-él   'slipper-DIM'
        c.   i.   ispráv-ʌ   'brave deed'
                  ispráv-nik   nobleman's title
             ii. isprʌv-nitʃ-él   nobleman's title (DIM)

Diachronically, Steriade (2017:20) argues that destressed a-raising happened via the following steps: 1) In native words, unstressed [a] was raised to ʌ early in the language. 2) Loans were then introduced which had unstressed [a]. 3) In order to handle both the alternating and non-alternating vowels, *a*-raising was reanalyzed as targeting destressed instead of unstressed vowels, e.g., in novel derivatives.

(107)   *Diachronic trajectory of destressed a-raising in Romanian (Steriade 2017)*

	Steps	Example		
1.	Unstressed /a/→[ʌ]	kʌmárʌ	'room'	< Latin *cámera*
2.	Borrowed unstressed [a]	vagabońd	'vagabond'	< French
3.	Destressed /a/→[ʌ]	komplikát	'complicated'	
		komplikʌt-él	'complicated-DIM'	

I suggest virtually the same diachronic analysis for Armenian.

2.B   Productivity of high vowel reduction

As explained in §2.7.4, DHR is not a word-level process in either Western or Eastern Armenian. For example, DHR is systematically blocked in loanwords. This section gives an overview on how grammarians have judged the productivity of DHR in modern Armenian.

Because DHR does not apply to loanwords, many grammars of Armenian argue that DHR is unproductive, whether for Western Armenian (Bardakjian and Thomson 1977:241; Hagopian 2005:26) or Eastern Armenian (Sevak 2009:86; T'oxmaxyan 1983:25). For Eastern Armenian, some grammarians explicitly state that it is unproductive and fossilized (Xačatryan 1988:35,59), unproductive but active (Khachaturian 1985), unproductive but conventional (Katvalyan 1989), or unproductive but lexical (Macak 2016:233).

Some argue that reduction is productive with some limitations and tendencies, i.e., that DHR is productive mostly for monosyllabic bases (Haghverdi 2016:4), derivational morphology, (Minassian 1980:82), frequent or core vocabulary (Dum-Tragut 2009), diphthong reduction (Hagopian 2005:56), etc. There is some extension of DHR to neologisms made from native roots (Hagopian 2005:36). It is generally under-learned by heritage speakers (Karapetian 2014:79ff).

The most extensive discussion on the subregularity of vowel reduction comes from (Margaryan 1997:89ff) who argues that vowel reduction is synchronically productive *but* that it displays high degree of variation across words and word classes. He extensively catalogues variation data in Armenian which I presented throughout this chapter. Sargsyan (1987) shows similar corpus results based on diachronic fluctuations in vowel reduction in early Modern Eastern Armenian.

Chapter 3

Compounds: Heads and paradoxes

3.1 Introduction

The previous chapter established the cyclic architecture of Armenian based on its morphology, phonology, and prosody. In this chapter, I discuss a bracketing paradox whereby these different modules create non-isomorphic, mismatching, or paradoxical structures. The data is transcribed in Western Armenian, but the generalizations extend to Eastern Armenian.

In simplex nouns, the plural suffix is *-er* after monosyllabic bases (108a-i), *-ner* after polysyllabic bases (108a-ii). But plural formation creates a bracketing paradox in compounds. Compounds are formed by concatenating two stems, STEM1 and STEM2, normally with a linking vowel *-a-*. In some cases, the plural counts the number of syllables in the entire polysyllabic compound and surfaces as *-ner* (108b-ii). But in other compounds, the plural only counts the number of syllables in the monosyllabic *second stem* and surfaces as *-er* (108b-i). I underline the domain of syllable counting.

(108) a. i. pág   'yard, lot'   ii. panág   'army'
                pag-ér   'yards, lots'   panag-nér   'armies'

       b. i. an͡tsrév + t͡ʃúr   'rain + water'   ii. t͡ʃár + sírd   'evil + heart'
                an͡tsrev-a-t͡ʃúr   'rain-water'   t͡ʃar-a-sírd   'evil-hearted'
                an͡tsrev-a-t͡ʃur-ér   'rain-waters'   t͡ʃar-a-sird-nér   'evil-hearted people'

For certain compounds (108b-i), the plural counts the number of syllables in a morphological subconstituent of the base. This constitutes a bracketing paradox. In this chapter, I show that it is largely due to endocentricity. The plural counts the number of syllables in the *semantic head*. If the compound is exocentric, it counts the number of syllables in the entire compound (108b-ii); while if the compound is endocentric, it counts the number of syllables in the second stem (108b-i).

I analyze the bracketing paradox using cyclic Head-Operations (Hoeksema 1984) and Prosodic Phonology (Nespor and Vogel 1986). I argue that the interaction between the bracketing paradox and the rest of compound phonology requires the use of stratal levels and cyclicity. I argue that counter-cyclic approaches like Morphological Merger (Marantz 1988) are inadequate because they contradict Armenian strata.

In §3.2, I go over theories and types of bracketing paradoxes, with a focus on morphology-phonology paradoxes. In §3.3, I discuss the basic Armenian data. I show that compounds largely match simplex stems in their phonology. They form a single prosodic word and undergo stem-level rules. The only bracketing paradox is how endocentric compounds are pluralized. In §3.4, I show that the bracketing paradox is productive and based on endocentricity. This signals that Armenian inflection is head-marking. I formalize the bracketing paradox in §3.5, and I argue for using cyclic Head-Operations and against the use of counter-cyclic approaches like Morphological Merger. In §3.6, I discuss prosodically-conditioned variation in pluralizing endocentric compounds. This variation requires combining cyclic Head-Operations with Prosodic Phonology. I argue that the head of compounds maps to a prosodic constituent p. This constituent is not a foot or recursive PWord, but is a Prosodic Stem (Downing 1999a). I conclude in §3.7.

3.2   Bracketing paradoxes in morphophonology

Given some word, a **bracketing paradox** is when the word has two or more contradictory constituency structures. I focus on cases where the mismatch is between the phonology and morphology, e.g., the English comparative: *happier* ∼ *unhappier*. I first provide a classification of theories for bracketing paradoxes, and I set up different subtypes of phonology-morphology paradoxes.[1]

3.2.1   Theories and tools for bracketing paradoxes

Since Pesetsky (1979), there have been different theories for modeling bracketing paradoxes (Newell 2019) and debates over the validity of these paradoxes (Kitagawa 1986; Light 1991; Sproat 1992a; Kang 1993). Here, I go over some classifications for these theories.

One common classification is based on what constituency is posited as primary vs. derived (Sproat 1985). Most approaches treat the morphological structure as primary, while the phonological representation is derived. Theories which do this include Morphological Rebracketing (Sproat 1985, 1988), Morphological Merger (Marantz 1988), Prosodic Phonology (Aronoff and Sridhar 1983; Nespor and Vogel 1986; Cohn 1989; Booij and Lieber 1993), and Local Dislocation (Embick and Noyer 2001; Haugen and Harley 2013; Deal 2016). A less frequent approach is to let the phonological representation be primary while the morphological representation is derived; this includes theories like Affix Raising (Pesetsky 1985; Hoeksema 1987) and Morphological Reanalysis (Kiparsky 1983). A third set of approaches assumes that the morphological and phonological representations are always identical but they utilize additional tools to let the paradox emerge, such as counter-cyclicity or the ability of an affix to look inside its base. This set includes Late Adjunction (Newell 2005, 2008), Paradigm Function Morphology (Stump 1995b,a, 2001), Head Operations (Hoeksema 1984), and autosegmental planes (Halle and Vergnaud 1987a,b; Falk 1991).

To facilitate the application of the above types of theories to Armenian, I reclassify them in terms of the cyclicity vs. counter-cyclicity of the phonological representation. Some theories assume that the

---

[1]For space, I don't discuss some solutions that developed in non-Chomskyan frameworks, e.g., Autolexical Theory (Chelliah 1995), CG (Chae 1990, 1993), CCG (Bozsahin 1999), HPSG (Crysmann 1999; Müller 2003), LFG (Kim 1991, 1992), Dependency Grammars (Gross 2011a,b), a.o. I also don't discuss work that focuses on paradoxes between the morphological, syntactic, and semantic representations, such as in the phrases *transformational grammarian* and *beautiful dancer* (Williams 1981; Strauss 1982; Sadock 1985; Spencer 1988; Beard 1991; Becker 1993; Fukushima 1999, 2015, 2014; Ackema and Neeleman 2004; Belk 2019).

temporal order in which both the morphological and phonological structures are generated *matches* with the order in which the word's meaning (semantics) is interpreted, i.e., cyclically. This includes theories such as Head-Operations and Prosodic Phonology. In contrast, in counter-cyclic theories, the phonological representation (morpheme spell-out) is generated in a temporal order which does *not match* the semantic order. Depending on the theory, the morphological representation can be generated in a temporal order that either does or doesn't match the semantics. Most theories for bracketing paradoxes are counter-cyclic, including Morphological Merger, Morphological Rebracketing, Late Adjunction, among others.

(109) *Cyclicity-based classification for theories in bracketing paradoxes*

Counter-cyclic	Cyclic
Morphological Rebracketing (Sproat 1985), Morphological Merger (Marantz 1988), Paradigm Functions (Stump 2001), Linear Dislocaiton (Embick and Noyer 2001), Late Adjunction (Newell 2005) , ...	Head-Operations (Hoeksema 1984), Prosodic Phonology (Nespor and Vogel 1986), Autosegmental Planes (Halle and Vergnaud 1987a)

To illustrate, Prosodic Phonology is a cyclic model. Even though the prosodic and morphological structures don't match, the morphemes are spelled-out in the temporal order that matches the semantics (cf. Booij and Rubach 1984; Cohn 1989; Inkelas 1989). For example in *un-(happi-er)$_w$*, the prefix *un-* is spelled-out as outside the root's PWord temporally before the suffix *-er* is added. In contrast, Morphological Merger is countercyclic. The morphological structure is first generated in the right semantic order, but it is later modified. The morphemes are phonologically-spelled out based on this modified representation. For *unhappier*, the suffix forms a constituent with the root in the modified representation [*un*[*happy*-COMP]], and it spelled out as *-er* temporarily before the prefix *un-* is phonologically spelled-out.

3.2.2 Definition and types of bracketing paradoxes

Having set a cyclicity-based classification for the above theories, I now refine the types of representations that are involved in morphology-phonology paradoxes. I classify paradoxes into allomorphy-based vs. process-based bracketing paradoxes based on the source of the phonological representation.

A bracketing paradox is **allomorphy-based** if the paradox between the morphological constituency and the phonological constituency is based on the allomorphy of a morpheme. There is such a contradiction in *unhappier* because the comparative counts the syllables of the root 'happy' instead of the base 'unhappy'.

(110) *unhappier*   Morphology   Allomorphy-based Phonology   Prosody
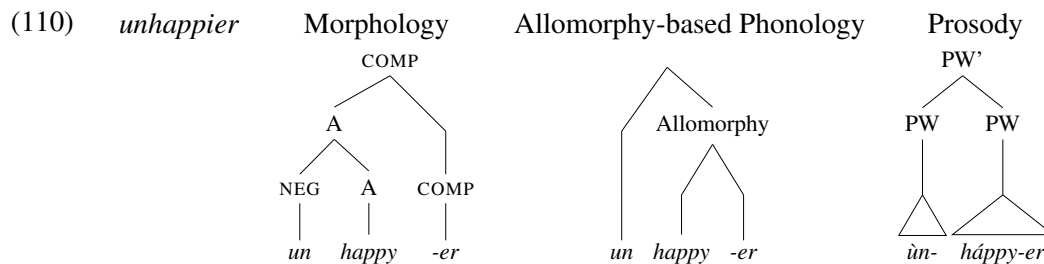
In contrast, a **process-based** bracketing paradox occurs for the word *ungrammaticality* because morphological constituency contradicts the phonological constituency that's based on phonological rule domains. Morphologically,

the adjective *grammatical* forms a constituent with the prefix *un-*, not the suffix *-ity*. But, the domain of stem-level (SLevel) or Level 1 stress assignment includes the suffix *-ity*, not the prefix *un-*. The entire construction is the domain of word-level (WLevel) processes.

(111)   *ungrammaticality*   Morphology   Process-based Phonology   Prosody
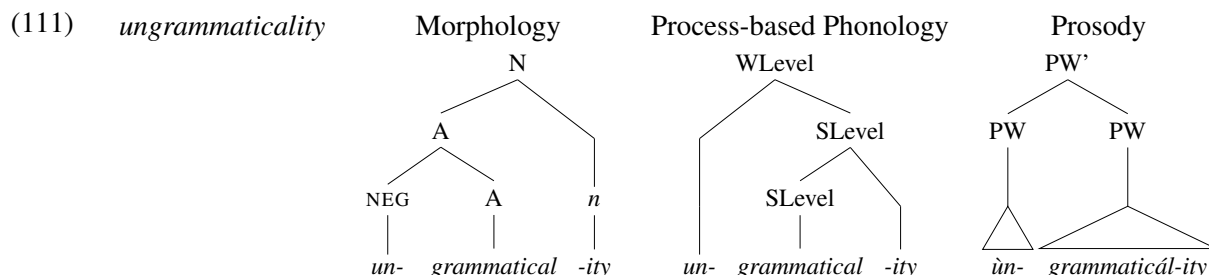
Cross-linguistically for most bracketing paradoxes, the allomorphy-based and process-based representations tend to be the same and they tend to match prosodic structure. In English, both types of phonological representations loosely match the prosody, e.g., *un-* forms its own PWord. Because of this tendency, most work in morphology-phonology bracketing paradoxes conflate these two types of phonological constituencies.

In Armenian compounds, I argue that these two phonological representations are not identical. In contrast, the morphological representation matches the domain-based representation, the domain-based representation does *not* match the allomorphy-based representation, and the allomorphy-based representation matches the prosodic representation. Because of this split in which representations match, Armenian is a useful case study to evaluate different tools for bracketing paradoxes. In the next section, I go over the data and I later argue that the Armenian data works best with cyclic theories, not counter-cyclic theories.

3.3   Constituencies in Armenian compounds

In this section, I set up the morphological, phonological, and prosodic structure for Armenian simplex words and compound words. Compound words tend to show isomorphic representations (§3.3.1), and the only bracketing paradox is present in compound pluralization (§3.3.2).

3.3.1   Matching constituencies in compounds

In terms of morphology, Armenian is primarily suffixing and agglutinative. A root can take on derivational (112a-ii) and inflectional suffixes (112a-iii). Compounds are formed by combining two stems (STEM1 & STEM2), normally with a linking vowel *-a-* (112b-ii).[2] A compound can itself take a suffix (112b-iii).

(112)   a.   i.   kórd͡z   'work'   b.   i.   seɣán   'table'

           ii.   kord͡z-avór   'worker   ii.   kord͡z-a-seɣán   'work-bench

           iii.   kord͡z-avor-nér   'workers'   iii.   kord͡z-a-seɣan-nér   'work-benches'

---

[2]Most compounds consist of only two stems but there are compounds which have three or more stems: *manər + lujs + nəgar* 'small + light + picture' → *manr-a-lus-a-nəgar* 'micro-photograph'. A linking vowel is used between each stem. These large compounds are rarely used and mostly restricted to higher registers. Data on their phonology is limited and I do not discuss them.

In terms of prosody, compounds and simplex words are similar. Primary stress is on the rightmost full vowel of the word, whether on a root (112a-i) or a suffix (112a-iii). Final schwas are unstressed (113iii). Clitics are word-external and are not stressed (113iv). Based on stress, both simplex and compounds form a single prosodic word. Armenian is thus like Greek in that compounds form a single PWord, contra the cross-linguistic tendency for compounds to form two PWords (Nespor and Vogel 1986; Nespor 1999; Vogel 2010).

(113) a. i.   kord͡z-avór   'a worker'   b. i.   kord͡z-a-seɣán   'a work-bench'
           ii. kord͡z-avor-nér   'workers'   ii. kord͡z-a-seɣan-nér   'work-benches'
           iii. kord͡z-avor-nér-ə   'the workers'   iii. kord͡z-a-seɣan-nér-ə   'the work-benches'
           iv. kord͡z-avor-nér al   'also workers'   iv. kord͡z-a-seɣan-nér al 'also work-benches'

Besides stress, recall from Chapter 2 that there are phonological processes which apply differently between derivational and inflectional morphology. For example, there is a process of Destressed High Vowel Reduction (DHR) whereby destressed high vowels are deleted (114a), or reduced to a schwa (114b) if deletion would create an unsyllabifiable consonant cluster (Khanjian 2009). Similarly, Armenian has a rule of Destressed Diphthong *uj*-Reduction (DDR) whereby a destressed *uj* is reduced to *u* (114c). Both types of reduction apply before derivational suffixes (114ii), compounding (114iii) but not inflectional suffixes (114iv).[3]

(114)   a. *DHR via deletion*   b. *DHR to schwa*   c. *DDR*
       i.   d͡zaɣíg   'flower'   t͡ʃúr   'water'   hújn   'a Greek'
       ii.   d͡zaɣg-avéd   'flowery'   t͡ʃər-ajín   'watery'   hun-agán   'Greek (adj.)'
       iii. tért   'paper'   pós   'hole'   háj   'Armenian'
             d͡zaɣg-a-tért 'flower-petal'   t͡ʃər-a-pós 'water-hole'   hun-a-háj 'Greek-Armenian'
       iv.   d͡zaɣig-óv   'flower-INST'   t͡ʃur-óv   'water-INST'   hujn-óv   'Greek-INST'

I analyze these processes in terms of lexical strata or cophonologies. Derivational morphology forms morphological stems (MStems) and triggers the stem-level phonology, while informational morphology forms morphological words and triggers the word-level phonology. The reduction processes are stem-level processes, not word-level. Morphologically, compounds form a single MStem and undergo the same set of stem-level rules as simplex stems.

Vowel-hiatus repairs likewise shows stratal distinctions. Before derivational suffixes, vowel hiatus with a base final *i* is commonly repaired by of deletion (115a-ii); other possible repairs are coalescence (115b-ii, /-agan/). Similarly for base-final *u*, hiatus is commonly repaired by glide fortition to *v* (115c-ii). Compounding behaves the same and triggers the same set of stem-level rules (115iii). But before inflectional affixes, hiatus is generally repaired by glide epenthesis (115iv).[4]

---

[3]In Eastern Armenian, V-initial inflection can cause the overapplication of destressed high vowel reduction because of prosodic misalignment: t͡ʃər-ér 'waters'. The same extends to compounds, especially endocentric compounds: and͡zrev-a-d͡ʒər-er 'rain-waters'. The derivation above is just for Western Armenian. There are also fossilized rules of destressed *e*-to-*i* reduction and destressed *ja*-to-*e* reduction. These rules apply in some derivatives: *sér* 'love' vs. *sir-agan* 'affectionate', *arakjál* 'apostle vs. arakel-agán* 'apostolic'. They don't apply in not inflection: *ser-óv* 'love-INST', *arakjal-óv* 'apostle-INST'. Both rules apply in compounding: *túxt* for *sir-a-túxt* 'love-letter', *gérb* 'manner' for *arakel-a-gérb*.

[4]Before derivational suffixes, some rare repair rules are glide formation for *i* (*i→j*), glide epenthesis for *i* and *u*, and vowel deletion for *u*. Before inflectional suffixes, Eastern Armenian allows vowel deletion for *i* and glide fortition for *u*. But these rules are much less common than glide epenthesis.

(115)   a. *Deletion*   b. *Coalescence*   c. *Fortition*
        i.   aɣavní   'pigeon'   kiní   'wine'   lezú   'language'
        ii.   aɣavn-óts͡   'pigeon-coop'   kin-egán   'vinic'   lezv-agán   'linguistic'
        iii. dún   'house'   dún   'house'   xúmp   'group'
              aɣavn-a-dún   'pigeon-coop'   kin-e-pós   'wine-shop'   lezv-a-xúmp   'language-family'
        iv.   aɣavni[j]-óv 'pigeon-INST'   kini[j]-óv   'wine-INST'   lezu[j]-óv   'language-INST'

3.3.2   Paradoxical constituencies in compounds

So far, I have not shown a bracketing paradox for compounds. However, a bracketing paradox occurs in pluralization. For simplex words, the plural has two phonologically-conditioned allomorphs: *-er* after monosyllabic bases (116a-i), *-ner* after polysyllabic bases (116a-ii,116a-iii). I use the realization rules in (116b). They will be later revised.

(116)   a. i. pág   'yard, lot'   ii. panág   'army'   iii. akarág   'farm'
                pag-ér   'yards, lots'   panag-nér   'armies'   akarag-nér   'farms'

        b. PL: *counting syllables* (Version 1)
            PL → *-er* / #σ _
            PL → *-ner* /elsewhere

In simplex stems, the allomorphy is a simple case of syllable-counting.[5] I show its simple distribution below. The allomorphy is insensitive to stress because the suffix takes final stress. It not based on the number of feet; both bi- and trisyllabic words are pluralized the same. And, the allomorphy is insensitive to the syllable structure of the final syllable, e.g. the suffix *-ner* is added to a polysyllabic base even if the base ends in a consonant cluster. Likewise, *-er* is added to V-final monosyllabic bases; vowel hiatus is repaired by epenthesis.

(117)   *Distribution of the plural in simplex nouns*

	Syllable count								
Final σ	σ	σ-er		σσ	σσ-ner		σσσ	σσσ-ner	
CV	ts͡í	ts͡i[j]-ér	'horses'	ʃugá	ʃuga-nér	'stores'	mekená	mekena-nér	'machines'
CVC	pát	pat-ér	'ducks'	d͡ʒagád	d͡ʒagad-nér	'foreheads'	kaɣapár	kaɣapar-nér	'concepts'
CVCC	pánd	pand-ér	'prisons'	daɣánt	daɣant-nér	'talents'	aʃxadánk	aʃxadank-nér	'works'
CVCCC	bártk	bartk-ér	'debts'	lusánt͡sk	lusant͡sk-nér	'margins'	alabást(ə)r	alabastər-nér	'alabasters'

Compounds are always polysyllabic, thus we expect them to always be pluralized with *-ner*. This is the case for compounds where STEM2 is polysyllabic; these compounds are always pluralized with *-ner*.

---

[5]The allomorphy does not optimize phonological well-formedness. The only trace of optimization are CVCC bases which end in a rising-sonority cluster (Vaux 2003; Macak 2016). These clusters optionally take an excrescent or epenthetic schwa: *man(ə)r* 'small'. Their plural can be bisyllabic with *-er*: *manr-er*, or trisyllabic with *-ner*: *manər-ner*. The choice varies by dialect, speaker, and item. I put these cases aside. A few other morphemes also show suppletion based on syllable count, e.g. the indicative prefix (Vaux 1998b) and possessive plurals (Arregi et al. 2013; Wolf 2013). None of these processes reference stress-assignment. Macak (2016) argues that they reference unstressed feet.

(118)   a.   gárd͡ʒ + haság   'short + height'   b.   kórd͡z + seɣán   'work + table'
           gard͡ʒ-a-haság   'short'   kord͡z-a-seɣán   'work-bench'
           gard͡ʒ-a-hasag-nér   'short (people)'   kord͡z-a-seɣan-nér   'work-benches'

But, pluralization exhibits a bracketing paradox in compounds when STEM2 is monosyllabic. In this situation, some compounds are pluralized with *-ner* (119a), but some are pluralized with *-er* (119b). As a mnemonic, I say that compounds of the first type are transparently pluralized with *-ner*, while compounds of the second type are paradoxically pluralized with *-er*. I underline the syllables which are counted.

(119)   a.   t͡ʃár + sírd   'evil + heart'   b.   ant͡srév + t͡ʃúr   'rain + water'
           t͡ʃar-a-sírd   'evil-hearted'   ant͡srev-a-t͡ʃúr   'rain-water'
           t͡ʃar-a-<u>sird</u>-ner   'evil-hearted (people)'   ant͡srev-a-t͡ʃ<u>ur</u>-er   'rain-waters'

In (120), I show the different types of constituencies t for the transparent vs. paradoxical compound plurals. The compounds are identical in their morphological constituency, process-based phonological constituency, and in their stress-based prosodic constituency. Here, the compound stems form a stem-level constituent that excludes the word-level plural suffix.[6] Crucially, the compounds differ in their allomorphy-based representation. The allomorphy-based representation of the transparent plural matches the morphology, while that of the paradoxical plural does not. In the transparent case, the compound takes a suffix *-ner* because the plural counts the number of syllables in the compound. But in the paradoxical case, the plural counts the syllables in STEM2, meaning that STEM2+PL form an allomorphy-based constituent to the exclusion of STEM1.

(120)   *Constituencies in compound plurals with monosyllabic* STEM2

---

[6]I assume a simple morphological model for compounds (Selkirk 1982). In the morphological representation, I omit the linking vowel. I assume it is a semantically empty morph (Aronoff 1994; Ralli 2008) which is added during phonological spell-out in PF as a dissociated morpheme (Oltra-Massuet 1999b; Tat 2013; Embick 2015). In the prosodic representation, I omit feet. In §3.6, I show that two types of compounds have different prosodic constituencies below the PWord-level.

The bracketing paradox is found for compound pluralization, but not for the rest of the compound phonology. Visually, the paradoxical plural's two different phonological representations do not match. The compound undergoes the same stem-level processes as the transparent plural. I discuss this more in §3.5. In the next section, I show that this bracketing paradox is robust and that it is triggered by endocentricity

3.4   Endocentricity and Head-marking in Armenian compounds

Having explained how compound plurals form a bracketing paradox, I provide a simple explanation for the paradox based on endocentricity: only endocentric compounds like 'rain-water' can trigger a paradoxical plural. It is cross-linguistically common for headedness to affect compound structure and phonology (Williams 1981; Zwicky 1985; Hoeksema 1984, 1988, 1992; Di Sciullo and Williams 1987; Stump 1995b; Revithiadou 1999; Scalise et al. 2009; Scalise and Fábregas 2010; Moskal and Smith 2019). For Armenian, the role of endocentricity is not a novel claim, but it is an established fact in Armenian linguistics (Vaux 1998b; Dum-Tragut 2009).

Because the paradox is largely unfamiliar in generative linguistics, I go through a representative sample of compound plurals (§3.4.1). I show that across different types of compounds, the single most common predictor for pluralization depends on endocentricity. The bracketing paradox is insensitive to other factors such as the part of speech of the compound members or the semantic relationship between these members. I show that bracketing paradox is productive (§3.4.2). I show that inflection displays head-marking in endocentric compounds (§3.4.3), including the percolation of irregular inflection.

The examples is in this section are my native Western judgments for how to pluralize a compound. The judgments match the prescriptive and descriptive generalizations that are found across grammars on Armenian (Sowk'iasyan 2004:232; Ezekyan 2007:248; Sevak 2009:152; Dum-Tragut 2009:670-5, a.o.) and philological research on Armenian compounds (Mkrtčyan 1972, 1973, 1977, 1980; Sargsyan 1979, 1987; Donabédian 2004; Xačatryan 2009a,b; Karapetyan 2016).

3.4.1   Distribution of the bracketing paradox

I use the following working definition for endocentricity. A compound is endocentric if it is hyponymic; it is hyponymic if the compound is interpreted as a subtype of STEM2 (cf. Allen 1979's 'IS A' condition). In this case, STEM2 acts as the semantic head *h* of the compound. In this section, I go through a taxonomy of compounds. The three most common constructions are nominal, possessive, and deverbal compounds. Nominal compounds are endocentric, while possessive and deverbal compounds are exocentric. I show that only endocentric compounds trigger the bracketing paradox.

For example, the word *dun* can form the STEM2 of either an endocentric nominal or exocentric possessive compound. The exocentric one is transparently pluralized with *-ner* (121a), while the endocentric one is paradoxically pluralized with *-er* (121b). In the rest of this section, I don't gloss the plural.

(121) a. médz͡ + dún   'big + house'   b. aɣavní + dún   'pigeon + house'
           medz͡-a-dún   'opulent'   aɣavn-a-dún   'pigeon-coop'
           <u>medz͡-a-dun</u>-nér   'opulent people'   aɣavn-a-<u>dun</u>-ér   'pigeon-coops'

3.4.1.1 Bracketing paradox in endocentric nominal compounds: X-N=N

In a nominal compound, both STEM2 and the compound are nouns (X-N=N). These are estimated to constitute around 30% of Armenian compounds (Donabédian 2004). These compounds are endocentric and are paradoxically pluralized with *-er*.

(122) dón + d͡zár   'holiday + tree'   ad͡zúx + hór   'coal + pit'   pərún͡tsk + márd   'fist + fight'
       don-a-d͡zár   'Christmas tree'   ad͡zx-a-hór   'coal-pit'   pərən͡tsk-a-márd   'fist-fight'
       don-a-d͡zar-ér   ad͡zx-a-hor-ér   pərən͡tsk-a-mard-ér

Semantically, STEM1 acts as an adjunct modifier for STEM2. The specific specific semantic relationship between the two stems is wide-ranging and unpredictable, but it does not affect the bracketing paradox. As long as the compound stays hyponymic, it is paradoxically pluralized.

(123) a. *2 of 1*   arév + ʃóɣ   'sun + ray'   jergír + kúnt   'Earth + sphere'
                          arev-a-ʃóɣ   'sunray'   jergr-a-kúnt   'world globe'
                          arev-a-ʃoɣ-ér   jergr-a-kunt-ér
       b. *2 made from 1* méɣr+móm   'honey + candle'   medáks + kórk   'silk + carpet'
                          meɣr-a-móm   'beeswax, wax candle'   medaks-a-kórk   'silk carpet'
                          meɣr-a-mom-ér   medaks-a-kork-ér
       c. *2 in 1*   kedín + xórʃ   'ground + pit, cavern'   mit͡ʃín + bád   'middle + wall'
                          kedn-a-xórʃ   'ditch'   mit͡ʃn-a-bád   'dividing wall'
                          kedn-a-xorʃ-ér   mit͡ʃn-a-bad-ér

STEM1 is usually a noun as in the above examples, but STEM1 can range over different parts-of-speech, such as an adjective, infinitival verb, or a verb root.[7] Again, the category of STEM1 does not matter for the bracketing paradox; the compound is still endocentric and paradoxically pluralized.

(124) a. mán(ə)r + véb   'small + novel'   náx + hájr   'first + father'
          manr-a-véb   'novella'   nax-a-hájr   'forefather'
          manr-a-veb-ér   nax-a-hajr-ér
       b. kord͡z-é-l + gérb   'to work + manner'   kər-é-l + t͡sév   'to write + manner'
          kord͡z-e-l-a-gérb   'strategy'   kər-e-l-a-t͡sév   'writing style'
          kord͡z-e-l-a-gerb-ér   kər-e-l-a-t͡sev-ér
       c. aʃxad-í-l + várt͡s   'to work + payment'   marz-é-l + táʃt 'to exercise + field'
          aʃxad-a-várt͡s   'wage'   marz-a-táʃt   'sports field'
          aʃxad-a-vart͡s-ér   marz-a-taʃt-ér

---

[7]Free-standing verbs consist of a root followed a theme vowel *-e,-i,-a* and tense/agreement marking such as the infinitival *-l*: *aʃxad-i-l* 'to work'. STEM1 can be an infinitival verb or bound verbal root. In these compounds, STEM1 is verbal because the compound is interpreted as one involving the activity of STEM1. These compounds can be translated with English gerunds.

3.4.1.2 No bracketing paradox in possessive compounds: X-N=A

Exocentric possessive compounds constitute 5% of common Armenian compounds (Donabédian 2004). Here STEM2 is a noun but the compound is an adjective (X-N=A). These compounds are interpreted as metonymic or possessive compounds, whereby the compound *possesses* STEM2. They are analogous to English *bahuvrihi* compounds (*blue-eyed*) except that there is no overt suffix on STEM2. As adjectives, they can be substantivized and take nominal inflection. These compounds are non-hyponymic and exocentric: an evil-hearted person is not a type of heart. They are transparently pluralized with *-ner*.

(125)   t͡ʃár + sírd   'evil + heart'   ard͡zát + várs   'silver + locks'   táng + kín   'costly + price'
        t͡ʃar-sírd   'evil-hearted'   ard͡zat-a-várs   'silver-locked'   tang-a-kín   'valuable
        t͡ʃar-a-sird-nér   ard͡zat-a-vars-nér   tang-a-kin-nér

STEM1 is usually an adjective (126a). STEM1 can range over other parts-of-speech such as nouns (126b). In a few rare cases, STEM1 can also be a verbal root (126c). But regardless of what STEM1 is, possessive compounds are always transparently pluralized.

(126)   a.   tetév + kájl   'light + footstep'   zəvárt + t͡sájn   'cheerful + voice'
             tetev-a-kájl   'light-footed'   zəvart-a-t͡sájn   'cheerful-voiced'
             tetev-a-kajl-nér   zəvart-a-t͡sajn-nér
        b.   arjun + kújn   'blood + color'   t͡sjún + pájl   'snow + brightness'
             arjun-a-kújn   'blood-colored'   t͡sjún + pájl   'snow-white'
             arjun-a-kujn-nér   t͡sjun-a-pajl-nér
        c.   xéɣt-e-l + t͡sájn   'to strangle + voice'   xéɣt-e-l + máh   'to strangle + death'
             xeɣt-a-t͡sájn   'strangled-voiced'   xeɣt-a-máh   'asphyxiated'
             xeɣt-a-t͡sajn-nér   xeɣt-a-mah-nér

3.4.1.3 No bracketing paradox in deverbal compounds: X-V$_{\text{Root}}$=N/A

Finally, the most common class of compounds are deverbal compounds which are estimated to form at least 48.6% of Armenian compounds (Donabédian 2004).[8] In these compounds, STEM2 is derived from a verb while the entire compound is a noun or adjective. Because of this mismatch in their parts of speech, deverbal compounds are non-hyponymic and exocentric. They are transparently pluralized.

(127)   goʃíg + gar-é-l   'show + to sew'   gərág + baʃt-é-l   'fire + to worship'
        goʃg-a-gár   'shoe-maker'   gərag-a-báʃt   'fire-worshipper'
        goʃg-a-gar-nér   gərag-a-baʃt-nér

Armenian deverbal compounds have clear argument structure with STEM2 acting as a verb. They are analogous to English synthetic compounds like *truck driver*. But unlike in English, these compounds lack

---

[8]This figure is for deverbal compounds where STEM1 is a noun. This figure is larger if we include other categories for STEM1.

an overt nominalizer suffix on STEM2. Traditionally in Armenian linguistics, STEM2 is called a verbal root (V_{root}),[9] because STEM2 lacks the corresponding verb's theme vowel and tense/agreement suffixes

Semantically, the verbal root STEM2 is salient in these compounds because it assigns a thematic role to STEM1. In most cases like above, STEM2 is the root of a transitive verb. STEM1 acts as the internal argument of the verbal root while the entire compound is interpreted as the external argument. As an internal argument, STEM1 can be a noun (128a), adjective (128b) or even a verbal root (128c).[10] But in pluralization, STEM1's category does not matter and the compound is transparently pluralized with *-ner*.

(128) a. manúg + varʒ-é-l   'child + to instruct'   ʒoɣovúrt + var-é-l   'populace + to lead'
           mang-a-várʒ   'school-teacher'   ʒoɣovərt-a-vár   'demagogue
           <u>mang-a-varʒ-nér</u>   <u>ʒoɣovərt-a-var-nér</u>

       b. zazír + xos-í-l   'filthy + to speak'   láv + des-n-é-l   'good + to see'
           zazr-a-xós   'lewd, coarse'   lav-a-dés   'optimist'
           <u>zazr-a-xos-nér</u>   <u>lav-a-des-nér</u>

       c. əntert͡s-an-é-l + sir-é-l 'to read + love'[11]   hajhoj-é-l + sir-é-l   'to swear + to love'
           əntert͡s-a-sér   'lover of reading'   hajhoj-a-sér   'lover of swearing'
           <u>əntert͡s-a-ser-nér</u>   <u>hajhoj-a-ser-nér</u>

Other argument structures also don't trigger the bracketing paradox. In some compounds, STEM2 is the root of an intransitive verb. Here, STEM1 acts as a modifier adjunct. It can be an adverb or rarely a noun.

(129) a. jergár + dev-é-l   'long + to last'   b. kedín + soɣ-á-l   'ground + to creep'
           jergar-a-dév   'long-lasting'   kedn-a-sóɣ   'ground-crawler'
           <u>jergar-a-dev-nér</u>   <u>kedn-a-soɣ-nér</u>

In some compounds, STEM2 is interpreted as a passive verb, while the compound is interpreted as the promoted internal argument. Passive verbs have an overt passive affix *-v-* after the root; but passivization is covert in compounds. STEM1 can be a noun that acts as an external argument, a noun that acts as an instrumental adjunct, or an adjective/adverb that modifies the verbal action. As before, passive deverbal compounds are transparently pluralized with *-ner*.

(130) a. tév + har-v-í-l   'demon + to be beaten'   vodn + gox-v-í-l   'foot (archaic) + to be trodden'
           tiv-a-hár   'demon-possessed'   vodn-a-góx   'foot-trodden'
           <u>tiv-a-har-nér</u>   <u>vodn-a-gox-nér</u>

       b. aváz + bad-v-í-l   'sand + to be enclosed'   d͡zár + zart-v-í-l   'tree + to be decorated'
           avaz-a-bád   'enclosed with sand'   d͡zar-a-zárt   'decorated with trees'
           <u>avaz-a-bad-nér</u>   <u>d͡zar-a-zart-nér</u>

       c. təʒvár + mars-v-í-l   'hard + to be digested'   nór + hjus-v-í-l   'new, newly + to be woven'
           təʒvar-a-márs   'indigestible'   nor-a-hjús   'newly woven'
           <u>təʒvar-a-mars-nér</u>   <u>nor-a-hjus-nér</u>

---

[9]The Armenian term is "բայական արմատ" /pajagan armad/ 'verbal root', or the compound "բայարմատ" /paj-armad/ 'verb-root' (Sowk'iasyan 2004:251; Karapetyan 2016:77, a.o.).

[10]In some cases, STEM1 is morphologically ambiguous between a verbal root *hajhoj-e-l* 'to swear' or a truncated noun *hajhoj-ank* 'swearing (n)' for *hajhoj-a-ser* 'lover of swearing'.

Crisscrossing both intransitive and passive STEM2 are cases where STEM1 is interpreted as an internal argument that is *possessed* by the compound. These are also transparently pluralized.

(131) a. derev + tap(-v)-í-l 'leaf + to be shed'   b. hújs + xap-í-l   'hope + to be tricked'
           derev-a-táp   'leafless'   hus-a-xáp   'hopeless, desperate'
           derev-a-tap-nér   hus-a-xap-nér

In sum, when STEM2 is a verbal root, the bracketing paradox is not triggered.[12] It doesn't matter whether STEM1 is an argument, an adjunct, a noun, or another category. Deverbal compounds are non-hyponymic, exocentric, and transparently pluralized as polysyllabic bases with *-ner*. As for possessive compounds, they are also non-hyponymic, exocentric, and transparently pluralized. The ONLY compounds which trigger the bracketing paradox are *endocentric* compounds, such as nominal compounds.

3.4.2 Productivity of the bracketing paradox

The previous section showed that endocentricity is the most common denominator in predicting the bracketing paradox. In this section, I extend the argument by showing that the bracketing paradox is productive in new types of compounds and for morphologically ambiguous compounds.

The most common compound constructions are nominal, possessive, and deverbal compounds; they are estimated to constitute almost 90% of existing compounds (Donabédian 2004). But the bracketing paradox is not restricted to nominal compounds. It is slowly extending to *new* endocentric compound constructions, such as *adjectival* compounds where both STEM2 and the compound are adjectives.

(132) a. derév + xíd   'leaf + dense'   b. márt + ʃád   'man + many'
           derev-a-xíd   'dense with leaves'   mart-a-ʃád   'populous'
           derev-a-<u>xid</u>-ér   'ones dense with leaves'   mart-a-<u>ʃad</u>-ér   'ones that are populous'

These compounds are very rare in Armenian (Donabédian 2004). Their pluralized forms are rarely if ever found online, in corpora, grammars, or philological works (cf. Karapetyan 2016:75). But in my own judgments, these plurals are paradoxically pluralized because they're hyponymic and endocentric.[13]

The second piece of evidence comes from ambiguity. In principle, some identical pairs of stems are ambiguous between hyponymic and non-hyponymic readings. For example, some compound can alternate between an endocentric nominal and exocentric possessive reading (133a), or between an endocentric nominal and exocentric deverbal reading (133b). In the latter case, STEM2 can be morphologically parsed

---

[12]The above compounds are unambiguously parsed as deverbal because of their clear argument structure. But in some compounds, STEM2 is morphologically ambiguous between a free-standing noun or an intransitive/passive verbal root: *bád* 'cover' or *bad-v-í-l* 'to be walled' for *baɣeɣ-a-bad* 'ivy-walled' where STEM1 is *baɣéɣ* 'ivy'. Because of this ambiguity, these compounds can be parsed either as a possessive compound or a deverbal compound. Both readings are non-hyponymic, exocentric, and take the transparent plural: *baɣeɣ-a-<u>bad</u>-ner*.

[13]Speaker judgments vary on how to pluralize these compounds. I speculate that morphological ambiguity may play a role. In many of these adjectival compounds, STEM2 could be parsed an adjective or a verbal root: *xid* 'dense' vs. */xid-/* '$\sqrt{dense}$' in *xəd-an-a-l* 'to be dense'. An adjectival reading would make the compound hyponymic and take a paradoxical plural: *derev-a-<u>xid</u>-er*, while a verbal parse would make the compound be non-hyponymic and take a transparent plural: *derev-a-<u>xid</u>-ner*. Here, the verb would be interpreted as an intransitive inchoative. I suspect that the deverbal reading is generally dominant.

as either a noun or a verbal root. But regardless, the endocentric reading is paradoxically pluralized with *-er* while the exocentric one is transparently pluralized with *-ner*.

(133) a.

garmír + tév	red + wing'	méɣm + t͡sájn	'mild + voice'
garmr-a-tév	'red-winged (est.)';	meɣm-a-t͡sájn	'mild-voiced (est.)';
	'a red wing'		'a mild voice'
<u>garmr-a-tev</u>-nér	'red-winged things'	<u>meɣm-a-t͡sajn</u>-nér	'mild-voiced people'
garmr-a-<u>tev</u>-ér	red wings'	meɣm-a-t͡sajn-ér	'mild voices'

     b.

pájd + hád	'wood + piece'	márkarid + hád	'pearl + piece'
pájd + had-é-l	'wood + to cut'	márkarid + had-é-l	'pearl + to cut'
pajd-a-hád	'wood-cutter (est.)';	markard-a-hád	'piece of pearl (est.)';
	'piece of wood'		'pearl-cutter'
<u>pajd-a-had</u>-nér	'wood-cutters'	<u>markard-a-had</u>-nér	'pearl-cutters'
pajd-a-<u>had</u>-ér	'pieces of wood'	markard-a-<u>had</u>-ér	'pieces of pearl'

For these potentially ambiguous compounds, one reading is the *established* reading (est.) which is listed in dictionaries. For *garmr-a-tev*, the default reading is the exocentric 'red-winged'. This meaning is transparently pluralized: *garmr-a-tev-ner* 'red-winged things'. But if a speaker encounters this compound with a paradoxical plural, then the speaker is forced to reinterpret it as a endeocentric nominal compound: *garmr-a-<u>tev</u>-er* 'red wings'.[14]

To summarize, the bracketing paradox is productive and ultimately driven by endocentricity. The bracketing paradox is slowly extending from nominal compounds to adjectival compounds. Furthermore, speakers can alternate their interpretation of a plural compound based on the shape of the plural suffix.

3.4.3 Endocentricity and percolation of irregular inflection

The analysis so far is that pluralization a form of head-marking, and that the plural counts the number of syllables in the semantic head. Further evidence for the role of headedness comes irregular plurals. Certain nouns form irregular plurals (134a); their irregular plural is inherited in endocentric compounds (134b). However, Armenian irregular inflection has been leveling out (Sargsyan 1984). Some endocentric compounds with an irregular head can optionally get a regular plural (134c) (Sargsyan 1987:212) .[15]

(134) a.

márt	'man, person'	b.	dəɣá + márt	'boy + man'	c.	ájd͡z + márt	'goat + man'
			dəɣa-márt	'young man'		ajd͡z-a-márt	'satyr'
mart-íg	'men, people'		dəɣa-<u>mart</u>-íg	'young men'		ajt͡d͡z-a-<u>mart</u>-íg	'satyrs'
						ajt͡d͡z-a-<u>mart</u>-ér	'satyrs'

Cross-linguistically, if one type of inflectional process is head-marking in a language, then other inflectional processes tend to be head-marking as well (Stump 1995b, 2001). Specifically, we predict that the semantic head is also the *morphological head* and it percolates *all* of its inflectional features (Lieber 1989; Don 2005, 2004). This is borne out in Armenian. Besides irregular plurals, endocentric compounds also inherit the

---

[14]These judgments are my own and those of other Armenian speakers consulted.

[15]I could not find an exocentric compound where STEM2 is *mart* 'man'.

irregular case of their head (Sargsyan 1984, 1987). For example, the genitive of *majr* can use either the regular genitive suffix *majr-i* or irregular ablaut *mor* (135a). When this irregular noun forms the STEM2 of an endocentric compound, the compound is paradoxically pluralized and *inherits* the stem's optional irregular inflection (135b). In contrast, exocentric compounds don't inherit irregular morphology at all (135c).

(135)		a.	'mother'	b.	'mother + seal = godmother'	c.	*country* + *mother* = *capital*
			májr		gəník + májr		kaɣák + májr
					gənk-a-májr		kaɣak-a-májr
	Plural		majr-ér		gənk-a-<u>majr</u>-ér		<u>kaɣak-a-majr</u>-nér
	Reg. Gen.		majr-í		gənk-a-<u>majr</u>-í		<u>kaɣak-a-majr</u>-i
	Irreg. Gen.		mór		gənk-a-<u>mór</u>		

Endocentric compounds thus inherit both *morphological* irregular inflection and *phonological* syllable-counting allomorphy. This reinforces the role of heads in the bracketing paradox.[16]

3.5   Formalizing the bracketing paradox

The previous section established the robustness of the bracketing paradox and the role of headedness. In this section, I go over how different theories of bracketing paradoxes can or cannot describe the Armenian data. I argue that only a cyclic model can describe the bracketing paradox. I specifically use cyclic Head-Operations, and I show that counter-cyclic approaches like Morphological Merger/Rebracketing are inadequate. I use the running examples below.

(136)	a.	t͡ʃár + sírd	'evil + heart'	b.	ant͡srév + t͡ʃúr	'rain + water'
		t͡ʃar-a-sírd	'evil-hearted'		ant͡srev-a-t͡ʃúr	'rain-water'
		<u>t͡ʃar-a-sird</u>-ner	'evil-hearted (people)'		ant͡srev-a-<u>t͡ʃur</u>-er	'rain-waters'

Head-Operations are a cyclic approach (Hoeksema 1984; Hoeksema and Janda 1988; Aronoff 1988; Rainer 1993). Below, I derive the transparent and paradoxical plurals from (136) below. The theory assumes that the temporal order in which the phonological representation is spelled-out matches the hierarchical layers of the morphology. The two stems are first phonologically realized in Cycle 1. In Cycle 2, they are concatenated and the linking vowel is added. We determine that the head *h* of the exocentric compound is the entire compound (in brackets), while that of the endocentric compound is STEM2. The plural suffix is then spelled-out in Cycle 3. The plural allomorphy is determined by a head-operation. The realization rule for plural (137b) counts the number of syllables in the semantic head *h*.[17] The bracketing paradox thus emerges.

---

[16]There is evidence that compounds in Classical Armenian likewise displayed head-marking inflection. Classical Armenian did not phonologically-conditioned suppletion for the plural, but it had many declension classes based on case assignment. Compounds tended to inherit the class of their head STEM2 when endocentric (Olsen 2011).

[17]I have reformulated the realization rule (137b) so that it can encode the intuition behind a head-operation. Formally, in Categorial Morphology (Hoeksema 1984), an inflectional process F is a head-operation if when given a morphologically complex input $W = XY$ where Y is the head of W, then the output of $F(Z)$ is $XF(Y)$.

(137) a. *Deriving the bracketing paradox with Head-Operations*

		Exocentric 'evil-hearted (people)'	Endocentric 'rain-waters'
	UR	/t͡ʃar + sird + PL/	/ant͡srev + t͡ʃur + PL/
	Morphology	[[[t͡ʃar][sird]] PL]	[[[ant͡srev][t͡ʃur]] PL]
Cycle 1	Spell-out stems	t͡ʃar sird	ant͡srev t͡ʃur
Cycle 2	Concatenate stems	t͡ʃar-a-sird	ant͡srev-a-t͡ʃur
	Determine *h*	[t͡ʃar-a-sird]ₕ	ant͡srev-a-[t͡ʃur]ₕ
Cycle 3	Spell-out PL	t͡ʃar-a-sird-ner	ant͡srev-a-t͡ʃur-er

 b. PL: *counting syllables in the semantic head (h) (Version 2)*

 PL → *-er* / [σ]ₕ _

 PL → *-ner* / elsewhere

In contrast to Head-Operations, a countercylic approach like Morphological Merger (Marantz 1988) posits that the order in which morphemes are phonologically spelled-out does not match the underlying morphology. In the morphology, the stems first form a constituent under the scope of the plural. A rebracketing operation then applies whereby the semantic head of the compound is merged with the plural suffix. This operation is vacuous for exocentric compounds, but is crucial for endocentric compounds because now STEM2 forms a constituent with the plural. With this modified constituency, we apply bottom-up spell-out. The two compounds are spelled-out in two different orders and thus the paradox emerges.

(138) *Deriving the bracketing paradox with Morphological Merger*

		Exocentric 'evil-hearted (people)'			Endocentric 'rain-waters'
	UR	/t͡ʃar + sird + PL/			/ant͡srev + t͡ʃur + PL/
	Morphology	[[[t͡ʃar][sird]] PL]			[[[ant͡srev][t͡ʃur]] PL]
	Rebracketing				[[ant͡srev] [[t͡ʃur]PL]]
Cycle 1	Spell-out stems	t͡ʃar sird		Spell-out stems	ant͡srev t͡ʃur
Cycle 2	Concatenate stems	t͡ʃar-a-sird		Spell-out PL	ant͡srev t͡ʃur-er
Cycle 3	Spell-out PL	t͡ʃar-a-sird-ner		Concatenate stems	ant͡srev-a-t͡ʃur-er

So far, both cyclic and counter-cyclic approaches can describe the bracketing paradox in compound pluralization. In fact, Vaux (1998b:57) formalizes the bracketing paradox in Armenian using Morphological Merger, specifically in the form of head-to-head movement. Conceptually though, rebracketing is more redundant than head-operations because both reference the semantic head but the former also uses rebracketing.

However, I argue that counter-cyclic approaches are inadequate once we contrast the bracketing paradox with the rest of compound phonology. Recall from §3.2.2 that I distinguish between process-based vs. allomorphy-based bracketing paradoxes. The former occurs in examples like English stress assignment in *ungrammaticality*, while the latter occurs in the English comparative *unhappier*. The two types of paradoxes involve the contradiction between the morphological structure vs. the process-based and allomorphy-based representations respectively. The above counter-cyclic approaches assume that in any bracketing paradox, the allomorphy-based and process-based representations are the same and that they both contradict the morphological representation. This is where counter-cyclic analyses fall short in Armenian.

In a counter-cyclic analysis like Morphological Merger, we incorrectly predict that the second member and the plural form a single domain-based constituent, and that the first member is phonologically non-cohering. This is not borne out. Both exocentric and endocentric compounds undergo the same set of stem-level rules between their stems. As explained in §3.3.1, the stem-level rule of destressed high vowel reduction applies in derivation, but not inflection. It applies in both exocentric (139a) and endocentric compounds (139b).

(139)   a. aznív + sírd   'sincere + heart'   b. jergír + kúnt   'earth + sphere'
            aznəv-a-sírd   'sincere-hearted'   jerg-a-kúnt   'globe'
            <u>aznəv-a-sird</u>-ner   'sincere-hearted (people)'   jergr-a-<u>kunt</u>-er   'globes'

The two stems form a domain for the application of stem-level rules such as high vowel reduction, among other rules. The compound forms a larger word-level domain with the plural suffix. Visually, the process-based representations for endocentric and exocentric compounds are the same, and they match the morphological representation. The allomorphy-based representation for an endocentric compound matches neither of them. In the morphology, I show the stems in their underlying form without the linking vowel.

(140)   *Constituencies in compound plurals with monosyllabic* STEM2 *and vowel reduction*

When combined with stratal phonology, cyclic head-operations can account for the matching process-based representation and the mismatching allomorphy-based representation. In Cycle 1, the two stems are first spelled-out and undergo the stem-level (SLevel) phonology to get stressed. They are then concatenated into an MStem and their heads are determined. Both concatenations undergo the same stem-level rules of reduction. The plural suffix is then added and counts the number of syllables in the head. Destressed high vowels are marked by the diacritic ĭ,ŭ.

(141)   *Deriving the bracketing paradox with Head-Operations and strata*

		Exocentric 'sincere-hearted (people)'	Endocentric 'globes'
	UR	/azniv + sird + PL/	/jergir + kunt + PL/
	Morphology	[[[azniv][sird]] PL]	[[[jergir][kunt]] PL]
Cycle 1	Spell-out stems	azniv sird	jergir kunt
	SLevel: stress	aznív sírd	jergír kúnt
	SLevel: reduction		
Cycle 2	Concatenate stems	aznív-a-sírd	jergír-akúnt
	Determine *h*	[aznív-a-sírd]$_h$	jergír-a-[kúnt]$_h$
	SLevel: stress	[aznǐv-a-sírd]$_h$	jergǐr-a-[kúnt]$_h$
	SLevel: reduction	[aznəv-a-sírd]$_h$	jergr-a-[kúnt]$_h$
Cycle 3	Spell-out PL	aznəv-a-sírd-ner	jergr-a-kúnt-er
	WLevel: stress	aznəv-a-sird-nér	jergr-a-kunt-ér

In contrast, a derivation using Morphological Merger and strata does not work. In the original morphological structure of an endocentric compound, the two stems form a larger MStem which then merges with the PL to form an MWord. But in the rebracketed version, STEM2 and the plural form a constituent, an inflected MWord. When concatenated with STEM1, this stem-word compound forms a larger MWord. This is because MStems generally can't dominate MWords; stem-word compounds tend to show prosodic differences from simple stem-stem compounds (cf. stem-word compounds in Greek: Nespor and Ralli 1996; Nespor 1999; Ralli 2012; Nikolou 2009). Below, I show the original and modified morphological structure for an endocentric compound.
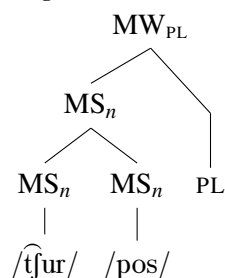
(142)   a. *Original*   b. *Rebracketed*
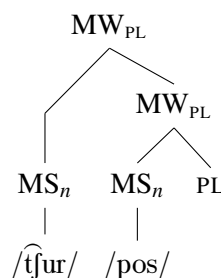
The two stems are spelled out and undergo the stem-level rules to get stressed. In the exocentric case, the two stems are concatenated, undergo the stem-level rules, and get pluralized. But in the endocentric case, the second stem had formed an MWord with the plural. In Cycle 2, we spell-out the plural suffix and trigger the word-level phonology. Then at Cycle 3, we concatenate STEM1 with the MWord to form a larger MWord. But, as an MWord, stem-level rules are blocked and we get the incorrect output without reduction.

(143)   *Failed derivation of the bracketing paradox with Morphological Merger and strata*

		Exocentric 'evil-hearted (people)'			Endocentric 'rain-waters'	
	UR	/azniv + sird + PL/			/t͡ʃur + pos + PL/	
	Morphology	[[[azniv][sird]] PL]			[[[jergir][kunt]] PL]	
	Rebracketing				[[jergir] [[kunt]PL]]	
Cycle 1	Spell-out stems	azniv	sird	Spell-out stems	jergir	kunt
	SLevel: stress	aznív	sírd	SLevel: stress	jergír	kúnt
	SLevel: reduction			SLevel: reduction		
Cycle 2	Concatenate stems	aznív-a-sird		Spell-out PL	jergír	kúnt-er
	SLevel: stress	aznĭv-a-sírd		WLevel: stress PL	jergír	kŭnt-ér
	SLevel: reduction	aznəv-a-sírd				
Cycle 3	Spell-out PL	aznəv-a-sírd-ner		Concatenate stems	jergír-a-kunt-ér	
	WLevel: stress	aznəv-a-sird-nér		WLevel: stress	jergĭr-a-kunt-ér	
Output:		✓ aznəv-a-sird-ner			✗ *jergir-a-kunt-er	
expect					jergr-a-kunt-er	

To summarize, counter-cyclic analyses can work for the compound plurals, but they cause inconsistencies with the rest of compound phonology. In endocentric compounds, these analyses require that STEM2 and the plural form a single domain-based constituent, while STEM1 is excluded from this constituent and treated as a phonologically non-cohering element. This requirement is inconsistent with the fact that STEM1 *is* phonologically cohering and that it forms a stem-level domain with STEM2. Faced with this inconsistency, we either need to abandon counter-cyclic analyses or to abandon lexical strata. In this chapter, I chose the former approach and I settle on using cyclic Head-Operations. I show in the appendix that there are problems when replacing strata with phases (Newell 2008) or with the free interleaving of cyclic and non-cyclic phonology (Halle and Vergnaud 1987b), also known as selective spell-out (Scheer 2011:9).

3.6   Prosodic variation

In this chapter, I focused on the general rules in pluralizing compounds. In the previous section, I developed a cyclic analysis for the bracketing paradox in compounds by using Head-Operations. The Head-Operations analysis can adequately describe cases where endocentric compounds are paradoxically pluralized as monosyllabic, as well as cases where they inherit irregular inflection from their head. For these cases, another cyclic analysis like Prosodic Phonology (Nespor and Vogel 1986) is by itself inadequate.[18] A prosodic analysis would be unsuited for the percolation of irregular inflection.

In this section, I discuss a type of prosodically-determined variation which cannot be analyzed with Head-Operations, but requires prosodic constituents like the Prosodic Stem (Downing 1999a).[19] I argue that we need both Head-Operations and Prosodic Phonology in order to describe the full extent of the bracketing

---

[18]Using cyclically created autosegmental planes (Halle and Vergnaud 1987a) is also inadequate because it would not distinguish exocentric and endocentric compounds because both are cyclically formed and have identical stress planes (cf. a similar problem noted by Cohn 1989). Though in the appendix, I show that counter-cyclically created planes don't have this problem.

[19]In this chapter, I only discuss prosodically-conditioned variation. There is limited variation based on generation, dialect, semantic shift, semantic opacity, metaphoricity, animacy, loanwords, grammaticalization, lexicalization, and frequency (Sargsyan 1979, 1987; Marowt'yan 2003; Avetisyan 2007:43). All relevant data and researh are unfortunately only available in Armenian.

paradox. Head-Operations handle both the norm and irregular inflection, while Prosodic Phonology handles the norm and the variation.

3.6.1 Prosodic heads and bisyllabic minimality

In general, compounds are at least trisyllabic because they surface with the linking vowel *-a-*. But, some compounds *arbitrarily* lack a linking vowel. I call such compounds 'unlinked'. These unlinked compounds are less common than linked compounds. They arose from collocations, dialectal borrowings, or sporadic diachronic syncope of the linking vowel. They are estimated at fewer than a 100 compounds (Ēloyan 1972).

(144) a. xát͡ʃ + kár   'cross + stone'   b. kár + daʃ-el   'stone, to carve'
           xat͡ʃ-kár   'cross-stone (ornament)'   kar-dáʃ   'stone carver, mason'
           xat͡ʃ-<u>kar</u>-ér   'cross-stones'   *kar-daʃ-ér   'stone carvers, masons'
           xat͡ʃ-<u>kar</u>-nér   kar-da<u>ʃ</u>-nér

Because the linking-vowel is absent, the above compounds are bisyllabic. Unsurprisingly, exocentric bisyllabic compounds are transparently pluralized with *-ner* (144b). But bisyllabic endocentric compounds are variably pluralized as transparent or paradoxical (144a). Judgements for bisyllabic compounds vary by speaker, dialect, and era. For the endocentric unlinked compound *xat͡ʃ-kar* (144a), the paradoxical plural with *-er* is prescriptive and considered standard in Eastern Armenian. Some consider the transparent plural as obsolete (Sargsyan 1979:39, Sargsyan 1987:206, Marowt'yan 2003:57). But in my judgments of Western Armenian, the transparent plural is more common and sounds more natural.

The above variation indicates an emergent minimality effect. In §3.6.2, I provide more examples in order to understand the extent of the variation. I first formalize this variation using prosodic constituents (145). The semantic head *h* is mapped to a prosodic constituent *p* (the prosodic head). *p* is optionally restructured when it is monosyllabic and follows an unparsed word-initial syllable. I discuss the identity of *p* in §3.6.3. I argue that the most likely label is the *Prosodic Stem* (Downing 1999a).

(145) *Prosodic mapping* (Version 1)
    a. *Mapping the semantic head* h *to the prosodic head* p
        $[...]_h \rightarrow (...)_p$
    b. *Optional restructuring the prosodic head* p *in a bisyllabic compound*
        $\#\sigma\ (\ \sigma\ )_p \rightarrow (\ \sigma\ \sigma\ )_p$

I show the prosodic structure of these bisyllabic compounds, before parsing the suffix.

(146) a. Exocentric   b. Endocentric & Paradoxical   c. Endocentric & Transparent

a. Exocentric
PW — p — kar-daʃ + *-ner*
kar-daʃ-ner 'stone carvers'

b. Endocentric & Paradoxical
PW — p — xat͡ʃ -kar + *-er*
xat͡ʃ-<u>kar</u>-er 'cross-stones'

c. Endocentric & Transparent
PW — p — xat͡ʃ-kar + *-ner*
xat͡ʃ-kar-ner 'cross-stones'

Inflectional processes are sensitive to both the semantic head *h* and its prosodic head *p*. For the plural, if the *h* has any irregular inflection features, then the plural is spelled-out as an irregular plural. Otherwise, when *h* is regular, the plural counts the number of syllables in both *h* and *p*. If both *h* and *p* are monosyllabic, then the allomorph *-er* is chosen. Otherwise, we use the allomorph *-ner*. In the latter case, *h* can be monosyllabic while its *p* is polysyllabic; this still triggers the allomorph *-ner*

(147)   PL: *counting syllables in semantics and prosodic heads* (Version 3)

> PL → *-ig* / [+IRREGULAR]$_h$
>
> PL → *-er* / [(σ)$_p$]$_h$ __
>
> PL → *-ner* / elsewhere

Below, I show a partial derivation for bisyllabic and trisyllabic compounds. After the stems are concatenated, the semantic head *h* is determined in Cycle 2. *h* is parsed into *p*. *p* is optionally restructured in the bisyllabic endocentric compound (b), while *p* and *h* stay isomorphic for exocentric and trisyllabic compounds (a,c,d). In Cycle 3, The plural is then spelled-out and counts the syllables in the heads. The optional restructuring creates two possible plurals for *xatʃ-kar*. I omit the application of stem-level and word-level rules.

(148)   *Deriving optionality in the bracketing paradox*

		Bisyllabic Compound		*Trisyllabic Compound*	
		a. Exocentric ‘stone carvers’	b. Endocentric ‘cross-stones’	c. Exocentric ‘evil-hearted (PL)’	d. Endocentric ‘rain-waters’
	UR	/kar + daʃ + PL/	/xatʃ + kar + PL/	/tʃar + sird + PL/	/an͡tsrev + t͡ʃur + PL/
	Morphology	[[[kar][daʃ]] PL]	[[[xatʃ][kar]] PL]	[[[tʃar][sird]] PL]	[[[an͡tsrev][t͡ʃur]] PL]
Cycle 1	Spell-out stems	kar daʃ	xatʃ kar	tʃar sird	an͡tsrev t͡ʃur
Cycle 2	Concatenate stems	kar-daʃ	xatʃ-kar	tʃar-a-sird	an͡tsrev-a-t͡ʃur
	Determine *h*	[kar-daʃ]$_h$	xatʃ-[kar]$_h$	[tʃar-a-sird]$_h$	an͡tsrev-a-[t͡ʃur]$_h$
	Map *p*	(kar-daʃ)$_p$	xatʃ-(kar)$_p$	(tʃar-a-sird)$_p$	an͡tsrev-a-(t͡ʃur)$_p$
	Restructure *p*		xatʃ-(kar)$_p$, (xatʃ-kar)$_p$		
Cycle 3	Spell-out PL	kar-daʃ-ner	xatʃ-<u>kar</u>-er, xatʃ-kar-ner	tʃar-a-sird-ner	an͡tsrev-a-t͡ʃur-er

3.6.2   Prosodic minimality across bisyllabic compounds

As said, unlinked compounds are rare (Ēloyan 1972). Thus, endocentric bisyllabic ones are rarer. Fortunately, another source for bisyllabic compounds comes from cases where STEM2 is monosyllabic and V-initial. When STEM2 is V-initial, no linking vowel is used and the two stems are syllabified together. If the compound is endocentric and at least trisyllabic, then it is paradoxically pluralized (149a). But if the compound is bisyllabic, then it can be paradoxically or *transparently* pluralized (149b).

(149)   a.   god͡zág + án͡tsk   ‘button + passage’   b.   kít + án͡tsk   ‘nose + passage’
            god͡za.g-án͡tsk   ‘water canal’   kə.t-án͡tsk   ‘nasal cavity’
            god͡za.g-an͡tsk-ér   ‘water canals’   kə.t-<u>an͡tsk</u>-ér   ‘nasal cavities’
            *god͡za.g-an͡tsk-nér   <u>kə.t-an͡tsk</u>-nér

Optional prosodic restructuring is widespread among endocentric bisyllabic compounds which lack a linking vowel, either arbitrarily (150a) or because of vowel hiatus (150b). They take either a transparent or paradoxical plural. Judgments are taken from Sargsyan (1979:39, 1987:205-6) based on Eastern Armenian.

(150)   a.   t͡sár + xód   'horsehair + grass'   t͡ʃúr + hór   'water + well'

            t͡sar-xód   'fern'   t͡ʃər-hór   'well'

            t͡sar-<u>xod</u>-ér   'ferns'   t͡ʃər-<u>hor</u>-ér   'wells'

            t͡sar-<u>xod</u>-nér   t͡ʃər-hor-nér

   b.   mah + azt   'death + notice'   kéd + áp   'river + coast'

            ma.h-ázt   'death notice'   ke.d-áp   'river-bank'

            ma.h-<u>azt</u>-ér   'death-notices'   ke.d-ap-ér   'river-banks'

            <u>ma.h-azt</u>-nér   ke.d-ap-nér

However, for some endocentric bisyllabic compounds, prosodic restructuring is obligatory. For bisyllabic compounds where STEM2 is the V-initial morpheme *ant͡sk* 'passage', some of them only accept the transparent plural (151a), others accept both (151b). Bisyllabic unlinked compounds where STEM2 is *gáb* 'tie, knot' behave the same (151c,d). Additionally, compounds with *gab* can optionally appear with the linking vowel *-a-* and become trisyllabic. When they do, only the paradoxical plural is grammatical.

(151)   a.   lújs + ánt͡sk   'light + ...'   t͡ʃúr + ánt͡sk   'water + ...'

            lu.s-ánt͡sk   'margin'   t͡ʃə.r-ánt͡sk   'canal'

            lu.s-<u>ant͡sk</u>-nér   'margins'   t͡ʃə.r-<u>ant͡sk</u>-nér   'canals'

   b.   hért͡s + ánt͡sk   'leaf + ...'   soɣ-á-l + ánt͡sk   'to creep + ...'

            her.t͡s-ánt͡sk   'leaf-vein'   so.ɣ-ánt͡sk   'loophole'

            her.t͡s-<u>ant͡sk</u>-ér   'leaf-veins'   so.ɣ-<u>ant͡sk</u>-ér   'loopholes'

            <u>her.t͡s-ant͡sk</u>-nér   so.ɣ-<u>ant͡sk</u>-nér

   c.   víz + gáb   'neck + ...'   póɣ + gáb   'neck + ...'

            vəz-gáb   vəz-a-gáb   'necktie   poɣ-gáb   poɣ-a-gáb   'necktie'

            vəz-gab-nér   vəz-a-<u>gab</u>-ér   'neckties'   poɣ-gab-ér   poɣ-a-<u>gab</u>-ér   'neckties'

   d.   kár + gáb   'stone + ...'   tév + gáb   'arm + ...'

            kar-gáb   kar-a-gáb   'gorge'   tev-gáb   tev-a-gáb   'cuff'

            kar-gab-ér   kar-a-<u>gab</u>-ér   'gorges'   tev-gab-ér   tev-a-<u>gab</u>-ér   'cuffs'

            <u>kar-gab</u>-nér   tev-gab-nér

Given an endocentric bisyllabic compound, it is unpredictable whether it can be pluralized transparently, paradoxically, or both. Sargsyan (1979, 1987) speculates that the choice is diachronic, while Marowt'yan (2003) speculates that the choice depends on the degree of semantic bleaching and lexicalization. Two open questions are thus 1) the probabilistic distribution of these different plurals and 2) the additional factors which determine this distribution. Corpus resources on Armenian are too limited to answer these questions.

The last piece of evidence for this constituent *p* inside endocentric compounds comes from the syllabification of sibilant-stop clusters. Word-initially, a schwa is epenthesized before sibilant-stop clusters (sT) in Western Armenian; epenthesis is less common in Eastern Armenian. In exocentric compounds with an sT-initial STEM2, there is no schwa epenthesis (152a) (Ġaragyowlyan 1979, Sowk'iasyan 2004:27-28,61). But in endocentric compounds, schwa epenthesis can apply (152b).

(152) a.   /steɣd͡z-e-l/ → əsteɣd͡z-é-l   'to create'   /skest/ → əskést   'dress'
            pán + /steɣd͡z-e-l/   'thing, word + ...'   sév + /skest/   'black + ...'
            pan-a-s.téɣd͡z   'poet'   sev-a-s.kest   'dressed in black
            (pan-a-s.téɣd͡z)$_p$ (sev-a-s.kest)$_p$
       b.   /skest/ → əs.két   'dress'   /sgisp/ → əs.gísp   'start'
            súk + /skest/   'grief + dress'   darí + sgisp   'year + start'
            sək-a-əskést   'mourning dress'   dar-e-əsgísp   'start of the year'
            sək-a-(əskést)$_p$ dar-e-(əsgísp)$_p$

Compound-internal epenthesis can be explained by positing that epenthesis applies at the left edge of
p: STEM2 in endocentric compounds.[20] Epenthesis inside endocentric compounds is not discussed in the
previous literature and it may be a recently emerging bracketing paradox. In my judgment, the above
epenthetic schwas in endocentric compounds are optional but preferred, while epenthesis is banned in
exocentric compounds. Unfortunately, epenthetic schwas are not marked in the orthography; the data is
thus limited to Wiktionary entries which show syllabified forms.

3.6.3   Identity of the prosodic head

To summarize, I repeat below the prosodic structure for trisyllabic and bisyllabic compounds. They all
form a single PWord based on stress. For now, I omit feet and syllables. The constituent *p* is isomorphic
with the semantic head *h* in most cases, but it can optionally mismatch in endocentric bisyllabic compounds.
In the previous section, I did not give a label to the prosodic constituent involved in prosodic restructuring.
In this section, I discuss the identity of the prosodic head *p*.[21] I argue that it is unlikely to be a foot or PWord,
but is likely a *Prosodic Stem*.[22]

(153)   *Prosodic structure of different compounds*

Trisyllabic		Bisyllabic	
Exocentric	Endocentric	Exocentric	Endocentric
'evil-hearted (PL)'	'rain-waters'	stone carvers'	'cross-stones

---

[20]Note that these epenthetic schwas are included in syllable-counting allomorphy: *əs.kest-ner* 'dresses', *əsgisp-ner* 'starts'.

[21]The prosodic head *p* does not affect stress placement (cf. headedness and accent in Revithiadou 1999; Roon 2005; Gouskova and Roon 2008; Gouskova 2010). Stress stays final in all compounds.

[22]An alternative is to let *p* stay unlabelled, i.e., the morphological structure is cyclically mapped unlabelled prosodic constituents. Unlabelled prosodic constituents have been proposed before for the *recursive* mapping of phrasal cycles to *gradient* and relative prosodic structure (Wagner 2005, 2010). However, this alternative only recapitulates the problem of not having a label for *p*.

3.6.3.1 Feet as prosodic heads

One strategy is to argue that the semantic head *h* must be right-aligned with a foot. In this analysis, degenerate feet are optionally restructured in bisyllabic compounds, and the plural counts the number of syllables in the rightmost foot.

(154) *Deriving optionality in the bracketing paradox with feet*

		Exocentric 'stone carvers'	Endocentric 'cross-stones'
	UR	/kar + daʃ + PL/	/xat͡ʃ + kar + PL/
Cycle 1	...		
Cycle 2	Concatenating stems	kar-dáʃ	xat͡ʃ-kar
	Determine *h*	[kar-dáʃ]*h*	xat͡ʃ-[kar]*h*
	Map *f*	(kar-dáʃ)*f*	xat͡ʃ-(kár)*f*
	Restructure *f*		xat͡ʃ-(kár)*f*, (xat͡ʃ-kár)*f*
Cycle 3	Spell-out PL	kar-daʃ-nér	xat͡ʃ-<u>kar</u>-ér, xat͡ʃ-kar-nér

Treating *p* as a foot would work for modeling *just* the bracketing paradox. However, this analysis is inconsistent with secondary stress assignment. Secondary stress is assigned on the first syllable, both in simplex (155a) and compound words (155b) (Abeġyan 1933:20; Sowk'iasyan 2004:29).[23] Armenian stress is a hammock system: we need feet at both word edges (Vaux 1998b; Gordon 2002; DeLisi 2015, 2018).

(155) a. bàdasxán 'answer'   b. t͡sév 'shape, manner'
bàdasxan-avór 'responsible'   bàdasxan-a-t͡sév 'style of answering'
bàdasxan-avor-óv 'responsible-INST'   bàdasxan-a-t͡sev-óv 'style of answering (INST)'
(bàd.as)xa.na(vo.róv)   (bàd.as)xa.na(t͡se.vóv)

The feet required for secondary stress contradict the feet that we need from equating *p* with feet. For example, the bisyllabic simplex words in (156) have initial secondary stress and final primary stress, and thus have two feet. Secondary stress is weakly perceivable in the free-standing roots in (156a), but it is highly perceivable for words with the negative prefix *an-* (Ġaragyowlyan 1974:133, T'oxmaxyan 1975:179). This prefix can be category-changing (156b) or category-preserving (156c). All of these words are transparently pluralized.

(156) a. àntám 'member'   àndár 'forest'   ànkám 'time'
<u>àntam</u>-nér 'members'   <u>àndar</u>-nér 'forests'   <u>ànkam</u>-nér 'time'
b. hám 'taste'   váx 'fear'   xélk 'mind'
àn-hám 'tasteless'   àn-váx 'fearless'   àn-xélk 'mindless'
<u>àn-ham</u>-nér 'tasteless (PL)'   <u>àn-vax</u>-nér 'fearless (PL)'   <u>àn-xelk</u>-nér 'mindless (PL)'

---

[23]In compounds, there are conflicting *impressionistic* reports of secondary stress on the linking vowel (Margaryan 1997:76): *ot-à-nav-a-gaján* 'air-ship-station (=airport)', or the first syllable of STEM2 (T'oxmaxyan 1971:63, 1983:74). These reports are restricted to mostly tri-stem compounds, and are not acoustically supported (Toparlak 2019).

c. géɣd͡z   'false'   kój   'existent'   hájd   'evident'
    àn-géɣd͡z   'sincere'   àn-kój   'non-existent'   àn-hájd   'non-evident'
    àn-geɣd͡z-nér   'sincere (PL)' àn-koj-nér 'non-existent (PL)' àn-hajd-nér 'non-evident (PL)'

Prosodically, these bisyllabic words form two degenerate feet, one foot for each level of stress. If we define the constituent *p* as the foot, then we *incorrectly* predict that these words should be paradoxically pluralized with *-er* because they end in a monosyllabic degenerate foot, as shown below. This shows that *p* cannot be a foot, and that *p* must be larger than a foot.[24]

(157)   *Foot structure and plural formation for bisyllabic non-compounds*

| | 'member' | 'sincere' | 'tasteless' |
| | antam | an-geɣd͡z | an-ham |
Primary and Secondary Stress	àntám	àn-géɣd͡z	àn-hám
Foot-structure for stress	(àn)(tám)	(àn)-(géɣd͡z)	(àn)-(hám)
Predicted plural if *p* is foot	*(àn)(tam-ér)	*(àn)-(geɣd͡z-ér)	*(àn)-(ham-ér)
Correct plural	àntam-nér	àn-geɣd͡z-nér	àn-ham-nér

3.6.3.2   Recursive prosodic words as prosodic heads

Alternatively, we could treat *p* as a recursive minor PWord (Selkirk 1996; Peperkamp 1997; Ito and Mester 2009; Macak 2016). Exocentric compounds form a single PWord, while endocentric ones consist of two PWord layers. Bisyllabic endocentric compounds optionally fuse to a single PWord. The plural counts the number of syllables in the rightmost PWord. This analysis would work for the paradox, but it contradicts the stratal phonology of Armenian. Stem-level rules like destressed high vowel reduction apply in derivation, not inflection (158a). They also apply across the compound-boundary to STEM1. These rules apply in endocentric compounds, both trisyllabic (158b) or bisyllabic (158c).

(158)   a. t͡ʃúr   'water'   b. jergír + kúnt   'earth + sphere'   c. t͡ʃúr + hór   'water + well'
            t͡ʃər-ajín   'aquatic'   jergr-a-kúnt   'globe'   t͡ʃər-hór   'well
            t͡ʃur-óv   'water-INST'   jergr-a-kunt-er 'globes'   t͡ʃər-hor-ér   'wells'
                                                                                      t͡ʃər-hor-nér

By equating the compound-boundary with a PWord-boundary, it is surprising that stem-level rules can apply in this context. Because endocentric compounds have an internal PWord-boundary, we incorrectly predict that stem-level rules like reduction won't apply to STEM1.

---

[24]Besides the empirical problem of secondary stress, there is relatively little positive evidence for feet in Armenian. Primary stress in Armenian is final, cued by pitch (Athanasopoulou et al. 2017), non-iterative, and shows a hammock pattern with initial secondary stress. Because of these properties, Armenian has been argued to be footless (DeLisi 2015); its prosody has been modeled with just the metrical grid (Gordon 2002). French and Turkish have similar stress patterns, and they have also been argued to be footless (Özçelik 2017).

(159) *Inconsistencies between strata and recursive PWords in endocentric compounds*

		Bisyllabic 'wells'		Trisyllabic 'globes'
	UR	/t͡ʃur + hor + PL/		/jergir + kunt + PL/
Cycle 1	...			
Cycle 2	Concatenating stems	t͡ʃúr-hór		jergír-a-kúnt
	Determine *h*	t͡ʃúr-[hor]$_h$		jergír-a-[kúnt]$_h$
	Map PWords	(t͡ʃúr-(hor)$_w$)$_w$		(jergír-a-(kúnt)$_w$)$_w$
	Restructure PWords	(t͡ʃúr-(hor)$_w$)$_w$	(t͡ʃúr-hor)$_w$	
	SLevel: stress*		(t͡ʃŭr-hór)$_w$	
	SLevel: reduction*		(t͡ʃər-hór)$_w$	
	blocked across PWords			
Cycle 3	Spell-out PL	t͡ʃúr-<u>hór</u>-er,	t͡ʃər-<u>hór</u>-ner	jergír-a-<u>kúnt</u>-er
	WLevel: stress	t͡ʃŭr-<u>hor</u>-ér,	t͡ʃər-<u>hor</u>-nér	jergír-a-<u>kŭnt</u>-ér
Output:		✗ t͡ʃur-<u>hor</u>-ér	✓ t͡ʃər-<u>hor</u>-nér	✗ jergir-a-<u>kunt</u>-ér
expect		t͡ʃər-<u>hor</u>-ér		jergr-a-<u>kunt</u>-ér

This stratal problem is aggravated by the lack of explicit consensus on the behavior of recursive prosodic constituents. There is debate over whether recursive constituents can trigger categorically different processes vs. gradiently different processes (Ladd 1986; Ito and Mester 2009, 2012, 2013; Wagner 2010; Frota and Vigário 2013; Elfner 2015), whether they can block or trigger lexical processes (Szpyra 1989; Booij 1996; Peperkamp 1997; Raffelsiefen 2005; Kabak and Revithiadou 2009; Bennett 2018), whether they are restricted to the post-lexical phonology of clitics (Inkelas 1989; Booij 1996; Selkirk 1996; Zec 2005; Tyler 2019), and whether they act as diacritics for behaviorally different constituents (Vogel 2009, 2012, 2016; Vigário 2010; Guzzo 2018; Downing and Kadenge pear; Miller 2018, 2020).

3.6.3.3 Prosodic stems in as prosodic heads

Faced with these problems, I argue that *p* is actually a Prosodic Stem. The traditional prosodic hierarchy assumes only three levels of morphosyntactically derived constituents: the prosodic word, the prosodic phrase, and the intonational phrase (160a). However, there is cross-linguistic work on agglutinative and polysynthetic languages which argues for a more enriched hierarchy that includes at least one constituent below the PWord: the Prosodic Stem (160b).

(160) a.

Intonational Phrase (ι)

|

Phonological Phrase (φ)

|

——-Prosodic Word (ω or PWord)——-

b.

Intonational Phrase (ι)

|

Phonological Phrase (φ)

|

——-Prosodic Word (ω or PWord)——-

|

Prosodic Stem (PStem)
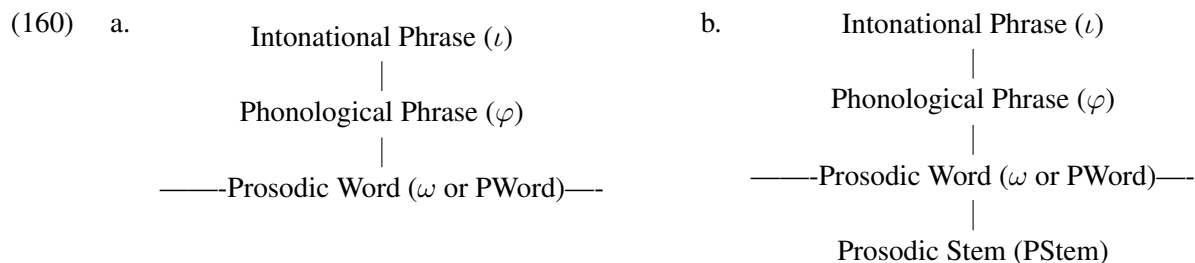
For Armenian, there is conceptual evidence for labelling *p* as the PStem. Cross-linguistically, there are

correlations between 1) morphosemantic heads and the prosodic stem, 2) minimality and PStems, and 3) morphological stems and prosodic stems. All three correlations are intermingled. To illustrate, Aronoff (1988) analyzes reduplication in KiHehe as a head-operation which targets the head of the word (in italics). Reduplication in KiHehe linearly occurs between prefixes and stems (161a) (Odden and Odden 1985). The reduplicant (underlined) generally copies only stem segments but it semantically scopes over the entire word. Furthermore, when the stem is V-initial, prefix-final consonants are overcopied with the stem because of syllabification (161b).

(161)   a. ku-*haata*   'to ferment'   b. kw-*iita*   'to pour'
            ku-<u>haata</u>~*haata*   'to start fermenting'   kw-<u>iita</u>~kw-*iita*   'to start pouring'

Aronoff analyzes the reduplication as a head-operation which is attached to the morphological head of the word: the stem. To explain prefix-copying in V-initial stems, he discusses reduplication as also targeting a *prosodic head* but does not formalize this concept. Downing (1998b) reanalyzes KiHehe and formalizes the prosodic head as the *Prosodic Stem* (PStem). The PStem is a prosodic constituent which is mapped from the morphological stem, analogous to the mapping of prosodic words from morphological words. The bulk of the evidence for the PStems comes from morphologically complex languages (Downing 1999a; Czaykowska-Higgins 1997). In Chapter 2, I showed the role of the PStem as the domain of appendix incorporation and for dialectal variation in vowel reduction (Dolatian 2019a,b).

As a constituent, the PStem can be the target of reduplication, tone, vowel harmony, minimality, and other sublexical processes (Downing 1998a, 1999a,b). It shows misalignment from the MStem based on syllabification. Further evidence for the PStem as a domain for phonological rules comes from cross-linguistic work on reduplication (Fitzpatrick-Cole 1994; Inkelas and Zoll 2005; Shaw 2005), prefix-suffix asymmetries (Hyman 2008), minimality (Downing 2005, 2006), strata (Inkelas 1989, 1993), and even bracketing paradoxes in **compounds** (Han 1995). For more cross-linguistic evidence for the PStem, see Downing (2006, 2016), Downing and Kadenge (to appear).

I argue that the constituent *p* is the PStem *s*. I illustrate a possible geometry below for singular compounds. For completeness, I show the foot and PStem structure for a bisyllabic prefixed word *àn-hám* from (§3.6.3.1).

(162)   *Prosodic structure of different types of compounds and prefixed-words where* p *is the PStem*

	Exocentric	Endocentric Trisyllabic	Endocentric Bisyllabic	Endocentric Bisyllabic Restructured	Prefixed Bisyllabic
Singular Plural	*s* àznəv -a- sírd (aznəv-a-sird)$_s$-nér	*s* *s* tʃər -a- pós (jergr-a)$_s$-(kunt)$_s$-ér	*s* *s* xàtʃ kár (x-atʃ)$_s$-(kar)$_s$-ér	*s* xàtʃ - kár (xatʃ-kar)$_s$-nér	*s* *f* *f* àn hám (àn-ham)$_s$-nér

The semantic head of the compound is mapped to a PStem *s* (163a). In simplex words, the entire input stem is the head and is mapped to a PStem. A sequence of two monosyllabic PStems is optionally

restructured to form a single PStem (163b). The plural counts the number of syllables in the rightmost PStem (163c).

(163)   *Prosodic mapping* (Final Version)

    a. *Mapping the semantic head* h *to a Prosodic Stem* s

        $[...]_h \rightarrow (...)_s$

    b. *Optional restructuring the Prosodic Stem* s *in a bisyllabic compound*

        $\#(\sigma)_s \, (\, \sigma \,)_s \rightarrow (\, \sigma \, \sigma \,)_s$

    c. PL: *counting syllables in the semantic head* h *and rightmost PStem* s (Final Version)

        PL $\rightarrow$ *-ig* / [+IRREGULAR]$_h$

        PL $\rightarrow$ *-er* / [(σ)$_s$]$_h$ _

        PL $\rightarrow$ *-ner* / elsewhere

By using a PStem in our cyclic derivation, there is no inconsistency with secondary stress assignment for feet, nor with the stem-level phonology. I don't show secondary stress in the below examples because initial schwas tend to be unstressed (Fairbanks 1948; Johnson 1954).

(164)   *Deriving optionality in the bracketing paradox with PStems in endocentric compounds*

		Bisyllabic 'wells'		Trisyllabic 'globes'
	UR	/t͡ʃur + hor + PL/		/jergir + kunt + PL/
Cycle 1	...			
Cycle 2	Concatenating stems	t͡ʃúr-hór		jergír-akúnt
	Determine *h*	t͡ʃúr-[hor]$_h$		jergír-a-[kúnt]$_h$
	Map PStem	t͡ʃúr-(hor)$_s$		jergír-a-(kúnt)$_s$
	Restructure PStem		(t͡ʃúr-hor)$_s$	
	SLevel: stress	t͡ʃŭr-(hór)$_s$	(t͡ʃŭr-hór)$_s$	jergǐr-a-(kúnt)$_s$
	SLevel: reduction	t͡ʃər-(hór)$_s$	(t͡ʃər-hór)$_s$	jergr-a-(kúnt)$_s$
Cycle 3	Spell-out PL	t͡ʃər-<u>hór</u>-er,	t͡ʃər-hór-ner	jergr-a-<u>kúnt</u>-er
	WLevel: stress	t͡ʃər-<u>hor</u>-ér,	t͡ʃər-hor-nér	jergr-a-<u>kunt</u>-ér

The above derivation is serial, however the prosodic mapping can be equivalently formalized with parallelist constraints, such as by adapting constraints from MATCH theory (Selkirk 2011) and WRAP theory (Truckenbrodt 1999) for stems. We would need a specialized constraint MATCHENDO that requires that semantic heads are parsed to PStems. The output constraint $*(\sigma)_s(\sigma)_s$ bans a string of a monosyllabic PStems. Recursive structures would be blocked by a constraint NONREC. I do not flesh out these constraints.

3.7   Conclusion

Linguistic theory posits that words have complex internal structure in terms of their morphology and phonology. Bracketing paradoxes arise when the morphological structure of the word contradicts its phonological

structure. This phonological structure is required by the domain of phonological processes or required by the domain of affix allomorphy.

In Armenian, the morphological structure of a compound is isomorphic to its process-based phonological structure, i.e., its stratal phonology. However, the plural form shows a bracketing paradox between its morphological structure and its allomorphy-based phonological structure. The plural suffix has two allomorphs based on syllable count: *-er* after monosyllabic bases, *-ner* after polysyllabic bases. Endocentric compounds paradoxically surface with *-er* if the second stem is monosyllabic.

This bracketing paradox signals a type of head-marking inflection. I argue that the bracketing paradox is resolved by using a combination of two cyclic tools: as Head-Operations and Prosodic Phonology. A cyclic approach like Head-Operations can derive the paradoxical allomorphy-based phonological structure, and it can also derive the non-paradoxical process-based phonological structure. Counter-cyclic approaches like Morphological Merger cannot do both. I also argued that the existence of prosodically-conditioned variation in some compounds means that Head-Operations are not enough. We also need sublexical prosodic constituents such as the Prosodic Stem in order to fully capture the bracketing paradox. The end result is a demonstration that bracketing paradoxes are cross-modular, and that different tools for bracketing paradoxes can contradict different aspects of the larger phonological system.

3.A Counter-cyclicity with alternatives to stratal phonology

Setting aside prosodic variation, I showed that counter-cyclic approaches like Morphological Merger are partially adequate for modeling the bracketing paradox but they contradict the stratal phonology of Armenian. I instead argued that the bracketing paradox required a cyclic approach like head-operation which did not contradict the stratal phonology. In this appendix, I sketch out two alternative analysis that combines counter-cyclic approaches with two alternative to stratal phonology: phase-based phonology and Halle and Vergnaud (1987a)'s free interleaving of cyclic and non-cyclic phonology. I argue that these alternatives have problems in modelling the entirety of compound phonology and morphology.

3.A.1 Empirical problems with phase-based phonology

Phonological Derivation by Phase (PDb) is a conceptual offshoot of lexical phonology (Marvin 2002; Newell 2008; Samuels 2011, 2012). While several theoretical assumptions of PDbP are still being debated (Matushansky and Marantz 2013; Siddiqi and Harley 2016; Newell et al. 2017), the following discussion refers to the more commonly held assumptions. I argue that when combined with Morphological Merger, PDbP can model the bracketing paradox and the application of stem-level rules in endocentric compounds. But it has problems in the application of stem-level rules in exocentric compounds.

PDbP assumes that words are cyclically derived where cycles are defined in terms of phase head *n, v, a* (= derivational suffixes). Non-phase heads (= inflectional suffixes) cannot trigger spell-out and thus do not trigger any cycles. They are not just post-cyclic, but they are part of the same cycle as derivation (Embick 2010). PDbP generally assumes that there is only one cophonology (= no stem vs. word strata) and that there are no prosodic constituents (Scheer 2011, 2012). Cyclic derivations generally respect the phonological structure created on previous cycles, whether by the Phase Impenetrability Condition (Marvin 2002), Phonological Persistence (Newell and Piggott 2014), or phase-based faithfulness (McPherson and Hayes 2016).

Recall that destressed high vowel reduction (DHR) applies in derivation, but not inflection. It applies in both endocentric and exocentric compounds.

(165) a. i. $\widehat{\text{t\textipa{S}}}$úr   'water'   ii. aznív   'sincere'
               $\widehat{\text{t\textipa{S}}}$ər-ajín   'aquatic'   aznəv-utjún   'sincerity'
               $\widehat{\text{t\textipa{S}}}$ur-óv   'water-INST'   azniv-óv   'sincere-INST'

       b. i. jergír + kúnt   'earth + sphere'   ii. aznív + sírd   'sincere + heart'
               jergr-a-kúnt   'globe'   aznəv-a-sírd   'sincere-hearted'
               jergr-a-<u>kunt</u>-ér   'globes'   <u>aznəv-a-sird</u>-nér   'sincere-hearted (PL)'

First, without strata, PDbP cannot directly capture the fact that inflection can trigger some but not all stem-level processes. The null hypothesis is that inflection cannot trigger its own cycle because inflection is a non-phase head. Instead, inflection is incorporated into the same phase-cycle as the previous (covert or overt) derivational suffix (Embick 2010). For simplex stems, this would allow inflection to trigger stress shift but not DHR.

(166)   *Phase-based derivation for vowel reduction*

		Base 'water'	Derivation 'aquatic'	Inflection 'water-INST'

Base 'water':
```
        n
       / \
      √   n
      |   |
     t͡ʃur  ∅
```

Derivation 'aquatic':
```
          a
         / \
        n   a
       / \   \
      √   n  -ajin
      |   |
     t͡ʃur  -∅
```

Inflection 'water-INST':
```
          INST
         /  \
        n    INST
       / \     \
      √   n    -ov
      |   |
     t͡ʃur  -∅
```

		Base 'water'	Derivation 'aquatic'	Inflection 'water-INST'
Cycle 1	Spell-out	t͡ʃur - ∅	t͡ʃur - ∅	t͡ʃur - ∅ - ov
	Stress	t͡ʃúr	t͡ʃúr	t͡ʃur-óv
	Reduction			
Cycle 2	Spell-out		t͡ʃúr - ajin	
	Stress		t͡ʃŭr-ajín	
	Reduction		t͡ʃər-ajin	
Output		t͡ʃúr	t͡ʃər-ajín	t͡ʃur-óv

When combined with Morphological Merger, PDbP can model the bracketing paradox. Destressed high vowel reduction correctly applies in endocentric compounds, but not in exocentric compounds. I assume the following morphological structure for endocentric and exocentric compounds. These structures are adapted from conventions in Distributed Morphology (Harley 2009; Harðarson 2016, 2017, 2018). Each stem is made up of a root and covert category affix. In the endocentric compound, STEM1 is adjoined to STEM2, as indicated by the different indexes for *n*. In exocentric compounds, STEM1 is also adjoined to STEM2. The entire compound takes a covert adjectivizer *a* (Steddy 2019). As before, I assume that linking vowels are generated in phonological spell-out in the PF component as dissociated morphemes (Embick 2015).

(167)   *DM-based morphology for compounds*

Exocentric possessive 'sincere-hearted (people)'	Endocentric nominal 'globes'	Endocentric nominal (rebracketed)

Exocentric possessive 'sincere-hearted (people)':
```
                PL
               /  \
              a    PL
             / \
            n   \
           / \   \
          a   n   \
         /|  /|\   \
        √ a √ n a  PL
        |  | | | |
     azniv -∅ sird -∅ -∅  PL
```

Endocentric nominal 'globes':
```
            PL
           /  \
          n₂   PL
         /  \
        n₂   n₂
       / \   / \
      √  n₁ √  n₂
      |  |  |  |
   jergir -∅ kunt -∅  PL
```

Endocentric nominal (rebracketed):
```
            PL
           /  \
          /    PL
         /    /  \
        n₂   n₂   PL
       / \   / \
      √  n₁ √  n₂
      |  |  |  |
   jergir -∅ kunt -∅  PL
```

I go through a derivation below. Following morphological rebracketing, the PL forms a constituent with STEM2 in endocentric compounds. The PL is spelled-out as *-er* in the first cycle with STEM2. The two stems are concatenated in Cycle 2 and reduction correction applies.

(168)   *Deriving compounds with phases & Morphological Merger*

Morphology		Endocentric 'globe'	Exocentric 'water-colored (PL)'
Morphology	Input Rebracketed	$[\ [\ [\text{jergir -}\emptyset]_{n1} \ [\text{kunt -}\emptyset]_{n2} \ ]_{n2} \text{ -PL}]_{\text{PL}}$ $[\ [\text{jergir -}\emptyset]_{n1} \ [\ [\text{kunt -}\emptyset]_{n2} \text{ -PL}]_{\text{PL}} \ ]_{\text{PL}}$	$[\ [\ [\ [\text{azniv -}\emptyset]_{a} \ [\text{sird -}\emptyset]_{n}]_{n} \text{ -}\emptyset \ ]_{a} \text{ -PL}]_{\text{PL}}$
Cycle 1	Spell-out Stress Reduction	jergir - ∅ kunt - ∅ - er jergír kunt-ér	azniv - ∅ sird - ∅ aznív sírd
Cycle 2	Spell-out Stress Reduction	jergír-a-kunt-ér jergĭr-a-kunt-ér jergr-a-kunt-ér	aznív-a-sírd aznĭv-a-sírd aznəv-a-sírd
Cycle 3	Spell-out Stress Reduction		aznəv-a-sírd - ∅ - ner aznəv-a-sírd-nér aznəv-a-sərd-nér
Output *expect*		✓ jergr-a-kunt-ér	✗ aznəv-a-sərd-nér aznəv-a-sird-nér

But, reduction is incorrectly predicted to apply to STEM2 in exocentric compounds. In the first two cycles, each stem is spelled-out and then concatenated. By the second cycle, the compound underwent stress shift to STEM2 and reduction on STEM1: *aznəv-a-sírd*. At this stage, the singular compound is an endocentric noun, but it must be reinterpreted as an exocentric possessive by taking a covert adjectivizer *a* (Steddy 2019). But this causes a problem for the phonology. In Cycle 3, the *a* category-node for the compound is spelled-out alongside the PL. Although the correct plural is generated, the monostratal phase-based phonology must apply stress shift to PL and reduction on STEM2: **aznəv-a-sərd-nér*.

An additional conceptual problem is that the above PDbP system contradicts Phonological Persistence. Phonological Persistence (Newell and Piggott 2014) predicts that structure-changing processes (reduction) should be possible within the same phase. By being in the same phase-cycle as the root, inflection would not be blocked from changing any structure: reduction would be incorrectly predicted to be preferred. In contrast, derivation would be incorrectly predicted to only trigger structure-building processes (no reduction) because it is in a separate phase.

In order to get the right reduction patterns, the simplest solution is to allow inflection to trigger its own cycles (Bobaljik and Wurmbrand 2013).[25] Phonological Persistence would then block inflection from triggering DHR. However, this implies an informal stratification of phonological processes. Furthermore, we still need a separate mechanism that allows derivation to trigger structure-changing processes like reduction.

To handle Armenian, PDbP can be tweaked to include strata via final vs. non-final phases (Lochbihler 2017), separation of phonological and morphosyntactic cycles (Embick 2014; d'Alessandro and Scheer 2015), and allowing inflection to trigger its own cycles (Bobaljik and Wurmbrand 2013; Shwayder 2015; Kilborne-Ceron et al. 2016). Doing so reduces the differences between PDbP vs. Stratal/Lexical Phonology. This brings us back into the original problem of how Morphological Merger contradicts stratal phonology.

---

[25]Morphological evidence for the separation of inflection from a root's phase-cycle is that regular inflection does not trigger root suppletion. The closest case of suppletion is irregular inflection which also triggers reduction (§2.7.2).

3.A.2   Conceptual problems in the free interleaving of cyclic and non-cyclic phonology

Phase-based phonology assumes that it is morphosyntactically predictable if a given morpheme will trigger a cycle or not. In contrast, the model of phonological cyclicity in Halle and Vergnaud (1987a,b) removes this predictability. Whether an affix can trigger cyclic rules or not is also diacritically determined. There are no constraints on the ordering of cyclic and non-cyclic affixes. Scheer (2011:9) calls this system "selective spell-out". Furthermore, PDbP assumes a single cophonology or set of rules. In contrast, Halle and Vergnaud (1987a,b) assume two cophonologies: cyclic vs. noncyclic rules. Whether a phonological process is cyclic or not is diacritically determined. I show that this lack of constraints allow Halle and Vergnaud (1987a) to model the Armenian data. But on the other hand, this model does not show independent evidence elsewhere in Armenian.[26]

Consider again the application of destressed high vowel reduction in non-compounds. To make the comparison easier, I decompose stress shift into two processes: rightmost stress assignment and destressing. Rightmost Stress Assignment place stress on the rightmost full vowel; Destressing will remove stress from a stressed vowel which is not the rightmost stressed vowel. I treat these as informal rules.

In my analysis, reduction is a stem-level rule which applies in derivation (the MStem) but not inflection (the MWord). Stress-assignment and destressing are both stem-level and word-level rules. In Halle and Vergnaud (1987a,b)'s system, stem-level rules are called cyclic rules while word-level rules are noncyclic. All derivational and inflectional suffixes are diacritically marked as triggering cyclic ($_c$) and noncyclic ($_{nc}$) rules respectively. The concept of MStems vs. MWords is not relevant.

(169)   *Deriving vowel reduction in simplex words with Halle and Vergnaud (1987a)'s system*

		Base 'water'	Derivation 'aquatic'	Inflection 'water-INST'
		N_c | t͡ʃur	A_c / \ N_c A_c | | t͡ʃur -ajin	$INST_{nc}$ / \ N_c $INST_n$ | | t͡ʃur -ov
Cycle 1	Spell-out	t͡ʃur$_c$	t͡ʃur$_c$	t͡ʃur$_c$
	Rightmost Stress Assignment	t͡ʃúr	t͡ʃúr	t͡ʃúr
	Destressing			
	Reduction			
Cycle 2	Spell-out		t͡ʃúr - ajin$_c$	t͡ʃúr - ov$_{nc}$
	Rightmost Stress Assignment		t͡ʃúr-ajín	
	Destressing		t͡ʃŭr-ajín	
	Reduction		t͡ʃər-ajin	
Noncyclic Cycle	Rightmost Stress Assignment	t͡ʃúr	t͡ʃər-ajín	t͡ʃúr-óv
	Destressing			t͡ʃŭr-óv
Output		t͡ʃúr	t͡ʃər-ajín	t͡ʃur-óv

---

[26]Halle and Vergnaud (1987a) assume a non-interactionist system whereby all morphological processes precede phonological processes. This means that they have problems in defining phonologically-conditioned allomorphy (Hargus 1993), e.g., the Armenian plural suffix. I set this problem aside for illustration.

To match the representations used in Halle and Vergnaud (1987a), I do not assume covert category suffixes on roots. In Cycle 1, the noun is spelled-out. Because the noun has a cyclic diacritic, it triggers the cyclic rules of stress and (vacuous) reduction to form $\widehat{tʃ}úr$. In Cycle 2, the derivational suffix -*ajin* is spelled out and triggers the cyclic rules of stress and reduction: $\widehat{tʃ}ər$-*ajín*. But, in Cycle 2, we also spell-out the noncyclic inflectional suffix -*ov*. Halle and Vergnaud (1987a,b) argue that non-cyclic affixes can't trigger cyclic rules. They also can't trigger non-cyclic rules. Thus, no rules apply at all. Finally, all the words undergo a final nonlcyclic cycle where only noncyclic rules apply. This gets us final stress in $\widehat{tʃ}ur$-*óv*.

For compounding, I assume that the compound's category is not due to adjunction (Selkirk 1982). When combined with Morphological Merger or any other rebracketing process, Halle and Vergnaud (1987a)'s model can handle the bracketing paradox. It can also *arguably* trigger the right cyclic rules. The problem is that they rely on the Strict Cyclicity Condition (Mascaró 1976), a debunked principle (Kiparsky 1993).

(170)   *Deriving compounds with Morphological Merger and Halle and Vergnaud (1987a)'s system*

Morphology	Input	Endocentric 'globe'		Exocentric 'water-colored (PL)'	
	Rebracketed				
Cycle 1	Spell-out	$jergir_c$	$kunt_c$	$azniv_c$	$sird_c$
	Rightmost Stress Assignment	jergír	kúnt	aznív	sírd
	Destressing				
	Reduction				
Cycle 2	Spell-out	jergír	kúnt-er_{nc}	aznív-a-sírd	
	Rightmost Stress Assignment			azniv-a-sírd	
	Destressing			aznĭv-a-sírd	
	Reduction			aznəv-a-sírd	
Cycle 3	Spell-out	$\{jergír$-a-kúnt-er$\}_c$		aznəv-a-sírd - ner	
	Rightmost Stress Assignment				
	Destressing	jergĭr-a-kúnt-er			
	Reduction	jergr-a-kúnt-er			
Noncyclic Cycle	Rightmost Stress Assignment	jergr-a-kúnt-ér		aznəv-a-sírd-nér	
	Destressing	jergr-a-kŭnt-ér		aznəv-a-sĭrd-nér	
Output		jergr-a-kunt-ér		aznəv-a-sird-nér	

The exocentric plural has a straightforward derivation. In Cycle 1, the two stems undergo the cyclic rules and get stressed. In Cycle 2, they are concatenated and undergo the cyclic rules of stress and reduction. Note that Rightmost Stress Assignment does not remove primary stress from STEM1. Removal is done by Destressing. In Cycle 3, the plural suffix is added. Because the suffix is non-cyclic, it does not trigger cyclic stress shift and reduction. Without the SCC, the suffix would get rightmost stress, while STEM2 would destressed and reduce. The word eventually undergoes the non-cyclic stratum and gets final stress.

For the endocentric plural, the first two cycles are straightforward. In Cycle 1, the stems are stressed. In Cycle 2, the second stem gets the plural suffix -er. Because the suffix is non-cyclic, then no rules apply. In Cycle 3, the stems are are concatenated. The suffix does not get any stress because of the Strict Cyclicity Condition (SCC). SCC blocks rightmost stress assignment on the suffix -er because the substring er was created in the previous cycle. We still get destressing and reduction on STEM1's u because concatenation made u no longer a final vowel. In the final noncyclic cycle, the compound undergoes the noncyclic rules to get final stress.

As shown, Halle and Vergnaud (1987a)'s system would work for Armenian. However, it has three conceptual problems. First, the derivation mainly worked because of the SCC. In Cycle 3, the input was *jergír-a-kúnt-er* and the output was *jergr-a-kúnt-er* with reduction on STEM1. In this cycle, the SCC blocked stress shift to the suffix -er, which itself blocked destressing and reduction in **jergr-a-kənt-ér*. Current incarnations of Halle and Vergnaud (1987a)'s system still assume that the SCC plays some role (Halle and Nevins 2009). However, the SCC is a dubious principle and has been debunked (Kiparsky 1993). An alternative blocking systems like the Phrase Impenetrability Condition is likewise dubious (Embick 2014; Newell 2017). [27]

Second, Halle and Vergnaud (1987a)'s assumes it is arbitrary and unpredictable whether an affix is cyclic or not. This misses the fact that in Armenian all cyclic and non-cyclic affixes are respectively derivational and inflectional morphology. Third, Halle and Vergnaud (1987a)'s system allows cyclic and noncyclic affixes to be linearly precede and follow each other, e.g., English *patent-abl-ity* where *-able* is Level 2, while *-ity* is Level 1 (Halle and Kenstowicz 1991; also in Salishan: Czaykowska-Higgins 1993). It is then surprising that these ordering paradoxes are not found in Armenian. All noncyclic affixes are inflectional, and they follow all other derivational morphology.

---

[27]It is possible that the SCC could be replaced with Shwayder (2015)'s 'phonocyclic buffer'. The phonocyclic buffer is a diacritically determined linear span of morphemes for cyclic phonological processes. When combined with Morphological Merger, the plural suffix would get counter-cyclically spelled-out but it would not enter the buffer. The plural would latter undergo the post-cyclic rule of stress shift. In this way, the buffer lets us separate between the domain of cyclic processes and the domain of allomorphy (cf. §3.2.2). I leave exploring the use of this phonocyclic buffer to future work.

Part II

Computational locality of cyclic phonology

Chapter 4

Logical notation and representations

Morphological and phonological transformations are functions from one representation or another. To describe these functions, I use Model Theory with monadic second order (MSO) logic and logically-defined graph-to-graph transductions. Doing so requires that we are explicit on the following questions:

- What does the input consist of?
- What does the output consist of?
- How are individual aspects of the output related to individual aspects of the input?

Model Theory is not a grammar formalization. It is a *descriptive* tool that I use to understand the computational properties of different functions and structures. In this chapter, I explain the logical notation via illustration. There are many formal and complete treatments of formal logic, model theory, and logical transductions (Büchi 1960; Courcelle 1994, 1997; Engelfriet and Hoogeboom 2001; Enderton 2001; Courcelle and Engelfriet 2012; Libkin 2013). This thesis acts as an accessible illustration and application to morphology and phonology. For another accessible illustration, see Strother-Garcia (2019) on syllabification and Jardine (2016c) on autosegmental structure.

I first provide a simple informal illustration of word models in §4.1. I then formalize this in §4.2 by introducing the formal notions of domains, labels, and relations. In §4.3, I introduce other formal yet minor notions through linguistic examples, specifically the tools of negation (§4.3.1), modifying binary relations (§4.3.2), copy sets (§4.3.3), helper predicates (§4.3.4), and logical consistency (§4.3.5).

In §4.4, I discuss the generative capacity of logical transductions. I illustrate a simple hierarchy of possible logical transductions, with a focus on Quantifier Free transductions (§4.4.1). In §4.4.2, I show how various intuitively local processes can be formalized as Quantifier Free transductions.

In §4.5, I use the same notation to formalize the hierarchical structure present in morphology (§4.5.1) and prosody (§4.5.2). As with simple linear strings, we can determine if the computation is local or not over these hierarchical representations (§4.5.3).

Conclusions are in §4.6. Because later chapters add on to our word signature, I provide the entire word signature or model signature in an appendix. I list all constants, unary labels, and binary relations with references to where they were introduced or significantly used. I do not list any user-defined predicates.

4.1   Informal illustration

Consider a process of intervocalic stop voicing: *pata~pada*. In a traditional rule-based analysis, the function is modeled by the rule in (171a) which changes the underlying [-voice] feature of a stop to [+voice] between two vowels. The symbols T and D are underspecified segments representing voiceless and voiced stops. In a parallelist constraint-based analysis (171b), the output constraint against intervocalic voiceless stops *VTV outranks the faithfulness or identity constraint IDENT(voice) against changing underlying voicing.

(171)   a. T→D/ V_V   b. *VTV » IDENT(voice)

The two formalizaitons in (171a) and (171b) are descriptively equivalent though they differ in their architectural premises: *Are transformations serialist or parallelist?* Stepping back from this distinction, however, the two treatments *imply* the same representation for the input and output:

(172)   *Informal representation for strings*
         1.   a string of symbols
         2.   each with its own attributes (vowel, stop, voiced, etc.),
         3.   which are ordered from left to right,
         4.   the two strings are in correspondence with each other
         5.   there is a change from input to output

Given an input string x and output string y, the above five aspects are formalized with the following tools:

(173)   *Formal representation for strings*
         1.   The **domain** of the input as a set of nodes
         2.   The set of **unary labels** that can be defined for nodes in the domain
         3.   The set of **binary relations** among nodes in the domain
         4.   The set of **copies** of the domain (= **copy set**)
         5.   There is a **logical transduction** which utilizes a set of **output functions**

For the input-output pair *pata~pada*, the representation of the individual strings and the relationship between them are graphically visualized either implicitly in (174a) or explicitly in (174b). The latter explicitly encodes the **binary relation** of **successor** between any two segments via directed edges with the label '◁'.

(174)   a.   Input: p   a   t   a   b.   Input: p ⟶◁ a ⟶◁ t ⟶◁ a

         Output: p   a   d   a   Output: p ⟶◁ a ⟶◁ d ⟶◁ a

The representation can be made more explicit (175) by dissecting the *input segments* into individual **domain** nodes marked by a pair of two indices: '0' to denote that they're part of the input and a natural

number in 1-4 to denote which individual segment they index. A segment's phonological attributes (its IPA symbol) or **unary labels** are visualized as the node's name; the index is a subscript. As for the output segments, the '0' index is replaced by '1', signifying that they are part of a separate **copy** of the input segments.

(175)   Input: $p_{0.1} \longrightarrow a_{0.2} \longrightarrow t_{0.3} \longrightarrow a_{0.4}$

   Output: $p_{1.1} \longrightarrow a_{1.2} \longrightarrow d_{1.3} \longrightarrow a_{1.4}$

The representation assumes that IPA symbols are **atomic** and part of the **unary labels**; but we can exchange the IPA symbols with a set of phonetic features as in (176) where a node's name is now its index. The transformation involves changing the features or **unary labels** [-voice] to [+voice] in bold.

(176)   Input: $0.1_{+cons,...} \longrightarrow 0.2_{+vowel,...} \longrightarrow 0.3_{+cons,\textit{-voice},...} \longrightarrow 0.4_{+vowel}$

   Output: $1.1_{+cons,...} \longrightarrow 1.2_{+vowel,...} \longrightarrow 1.3_{+cons,\textbf{\textit{+voice}},...} \longrightarrow 1.4_{+vowel}$

4.2 Formalization

We showed that the superficially simple idea of a word as 'a string of segments' encodes multiple distinct facts about the relevant representation. These facts are summarized as a **word signature** or a template on possible words. A word signature is a tuple $\langle D, L, R \rangle$ where D is the domain of a word, L is the set of possible unary labels, and R is the set of possible binary relations. The **word signature** of the single word *pata* is visualized as the graphs in (177). We don't include an index '0' because no transformation is involved.

(177)   a. $p \longrightarrow a \longrightarrow t \longrightarrow a$ b. $p_1 \longrightarrow a_2 \longrightarrow t_3 \longrightarrow a_4$

The two graphs in (177) are descriptively equivalent. In this chapter, both the left and right types of graphs will be used for maximal clarity and illustration. In later chapters, only the right type of graph will be used. All the information encoded in these graphs can be written out as the set of logical statements in (178).

(178)   *For the word* pata *with **word signature** $\langle D, L, R \rangle$*
   a.   **Domain** D: {1,2,3,4}
   b.   **Unary labels** L:
      - $a(x) =$ TRUE for {2,4}

- $\mathsf{p}(x) = \textsc{true}$ for $\{1\}$
- $\mathsf{b}(x) = \textsc{true}$ for $\emptyset$
- $\mathsf{t}(x) = \textsc{true}$ for $\{1\}$
- $\mathsf{d}(x) = \textsc{true}$ for $\emptyset$
- $\mathsf{k}(x) = \textsc{true}$ for $\emptyset$
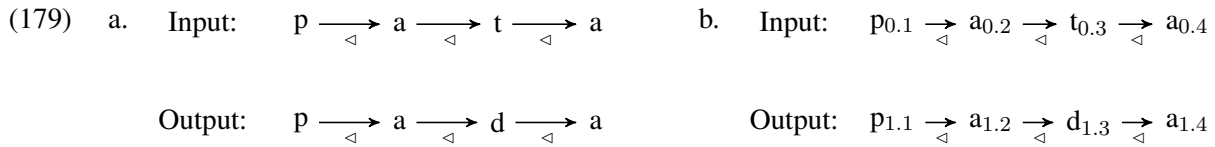- $\mathsf{g}(x) = \textsc{true}$ for $\emptyset$

c. **Binary relations** R: succ:seg(x,y)=$\textsc{true}$ for (x,y) in the set:

- (1,2)
- (2,3)
- (3,4)

The **domain** (178a) gives a unique index to the existing segments in the word. The set of **unary labels** (178b) includes the labels for existing phonemes in this language: $\mathsf{a}(x), \mathsf{p}(x), \mathsf{b}(x), \mathsf{t}(x), \mathsf{d}(x), \mathsf{k}(x), \mathsf{g}(x)$. A label like $\mathsf{p}(x)$ is evaluated to $\textsc{true}$ for domain elements (= segments) which are interpreted as the voiceless stop /p/. The set of unary labels can be replaced with labels for phonological features like $\mathsf{syllabic}(x)$, as in Strother-Garcia (2019). Feature-based labels encode the fact that phonological features are the fundamental units of phoneme representation. I use both feature-based and segment-based labels for easier illustration.

As for **binary relations** (178c), I only show the relation of successor which is true for any pair of domain elements x, y which are interpreted as y immediately succeeding x. I often use the term *precedence* in prose when referring to the successor relation succ:seg(x, y), e.g., b precedes c in the string $abcd$. In section §4.4.1, I show how general long-distance precedence can be computed from successor. In the graphs, I mark successor with $\lhd$. I refine immediate successor to a specific type of immediate successor among segments succ:seg(x). Later sections (§4.5.2) use other refined types of immediate successor for prosodic structures.

To formalize input-to-output transformations or input-to-output **logical transductions**, we include two additional tools: the **copy set** and **output functions**. Consider the input-output pair *pata*∼*pada* in (179a).

(179) a.   Input: $\mathsf{p} \xrightarrow{\lhd} \mathsf{a} \xrightarrow{\lhd} \mathsf{t} \xrightarrow{\lhd} \mathsf{a}$ b.   Input: $\mathsf{p}_{0.1} \xrightarrow{\lhd} \mathsf{a}_{0.2} \xrightarrow{\lhd} \mathsf{t}_{0.3} \xrightarrow{\lhd} \mathsf{a}_{0.4}$

Output: $\mathsf{p} \xrightarrow{\lhd} \mathsf{a} \xrightarrow{\lhd} \mathsf{d} \xrightarrow{\lhd} \mathsf{a}$ Output: $\mathsf{p}_{1.1} \xrightarrow{\lhd} \mathsf{a}_{1.2} \xrightarrow{\lhd} \mathsf{d}_{1.3} \xrightarrow{\lhd} \mathsf{a}_{1.4}$

The visualization of the input-output pair in (179) uses the same template or **word signature** $\langle D, L, R \rangle$ as that for a single word *pata* (177). The input string *pata* has the same domain, labels, and relations as the string *pata* (178). When defined over the input, the domain, labels, and relations are represented in this font.

To denote the output string, we use a **copy set** of size 1. A copy set consists of some fixed number of copies of the input. Because the output string has as many segments as the input, then we only need a copy set of size 1. To denote the relationship between the input string in the domain and the output string in the copy set, we use **output functions** and **user-defined predicates**. The change from input to output is called a **logical transduction**, and it constitutes the entire set of output functions.

User-defined predicates are custom-made shorthand that summarize useful information about some input item. They are represented in **this font**. To model intervocalic voicing, we use the following predicates

(180). The predicate **vowel**(x) is TRUE of some input symbol x if and only if (iff) x has the label of /a/. The predicate **intervocalic**(x) is TRUE of any input symbol x iff there are two segments w, y such that w precedes x, x precedes y, and w, y are vowels.

(180)   *User-defined predicates for intervocalic voicing*
- **vowel**$(x) \stackrel{\text{def}}{=} \mathsf{a}(x)$
- **intervocalic**$(x) \stackrel{\text{def}}{=} \exists w, y[\mathsf{succ{:}seg}(w, x) \wedge \mathsf{succ{:}seg}(x, y) \wedge \textbf{vowel}(w) \wedge \textbf{vowel}(y)]$

These predicates are constructed using common logical operators (181). The symbol $\stackrel{\text{def}}{=}$ is used to mean 'this formula is defined as...'; the equality between any two items uses the normal equal symbol: $x = y$.

(181)   *Common logical operators*

Operator	$\wedge$	$\vee$	$\neg$	$\stackrel{\text{def}}{=}$	$\exists$	$\forall$
Meaning	*and*	*or*	*not*	*defined as*	*there exists*	*for all*

User-defined predicates can likewise be defined over the *output* by referencing output correspondents in some copy: $\phi\textbf{vowel}(x^1)$. I explain this notation below. Defining predicates over the output makes certain computations easier to read, especially for syllabification. However, since the output is defined in terms of the input, predicates defined over the output are also ultimately defined over the input.

As for **output functions**, they have the template $\phi\mathtt{lab}(x^c)$ for labels and $\phi\mathtt{rel}(x^c, y^d)$ for relations. For the unary labels (182), the symbol ϕ denotes that the following formula is defined for an output symbol in some copy. The index c denotes that we are changing the label *lab* on an item x in the cth copy. For intervocalic voicing, there's only one copy so all c's are 1. We unpack each output function below.

(182)   *Output functions for unary labels for intervocalic voicing*
   a. For vowels
   - $\phi\mathsf{a}(x^1) \stackrel{\text{def}}{=} \mathsf{a}(x)$
   b. For voiceless stops
   - $\phi\mathsf{p}(x^1) \stackrel{\text{def}}{=} \mathsf{p}(x) \wedge \neg\textbf{intervocalic}(x)$
   - $\phi\mathsf{t}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \neg\textbf{intervocalic}(x)$
   - $\phi\mathsf{k}(x^1) \stackrel{\text{def}}{=} \mathsf{k}(x) \wedge \neg\textbf{intervocalic}(x)$
   c. For voiced stops
   - $\phi\mathsf{b}(x^1) \stackrel{\text{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{intervocalic}(x)]$
   - $\phi\mathsf{d}(x^1) \stackrel{\text{def}}{=} \mathsf{d}(x) \vee [\mathsf{t}(x) \wedge \textbf{intervocalic}(x)]$
   - $\phi\mathsf{g}(x^1) \stackrel{\text{def}}{=} \mathsf{g}(x) \vee [\mathsf{k}(x) \wedge \textbf{intervocalic}(x)]$

For the label $\mathsf{a}(x)$ in (182a), an output symbol is marked as an [a] if it is an /a/ in the input. Notice that the formula is represented as $\phi\mathsf{a}(x^1) \stackrel{\text{def}}{=} \mathsf{a}(x)$ where the difference in font shows the difference in status: a in the output, a in the input. The output function is defined over the 1$^{\text{st}}$ copy of the input. The label $\mathsf{a}(x^1)$ is TRUE only for the output node '1.1' in (179b), where x^c is interpreted as the node $c.x$.[1]

---

[1]The notation implies that x^1 is a different variable from x, but this is false. Here, x^1 is the first copy of the x. We interpret the

For the labels *p-t-k* (182b), matters are more complicated. Via $\phi\mathrm{p}(x^1)$, an output symbol x is marked as a [p] in the output iff it is a /p/ in the input and it is *not* intervocalic in the input. x is not intervocalic if it doesn't satisfy the user predicate **intervocalic**(x).

User-defined predicates are optional. For an output functions like $\phi\mathrm{p}(x^1)$ in (182), we *could* replace any user-defined predicate like **intervocalic**(x) with its definition (183). However, the result is awkward to use. These predicates have no consequence on the computation, but they make it easier to read and write.

(183)   *Illustrating utility of user-defined predicates*
    a. *Formula with user-defined predicate*
        - $\phi\mathrm{p}(x^1) \stackrel{\text{def}}{=} \mathrm{p}(x) \wedge \neg\mathbf{intervocalic}(x)$
    b. *Formula without the user-defined predicate* **intervocalic**(x)
        - $\phi\mathrm{p}(x^1) \stackrel{\text{def}}{=} \mathrm{p}(x) \wedge \neg\exists w, y[\mathsf{succ:seg}(w,x) \wedge \mathsf{succ:seg}(x,y) \wedge \mathbf{vowel}(w) \wedge \mathbf{vowel}(y)]$
    c. *Formula without the user-defined predicate* **vowel**(x)
        - $\phi\mathrm{p}(x^1) \stackrel{\text{def}}{=} \mathrm{p}(x) \wedge \neg\exists w, y[\mathsf{succ:seg}(w,x) \wedge \mathsf{succ:seg}(x,y) \wedge \mathsf{a}(w) \wedge \mathsf{a}(y)]$

As for the labels of voiced stops (182c), their interpretation is straightforward. Via $\phi\mathrm{d}(x^1)$, an output symbol is labeled as a [d] iff it is either a /d/ in the input *or* it is an underlying intervocalic /t/. For the input-output pair *pata*~*pada*, the 3rd domain item /t/ maps to [d] because it satisfies $\phi\mathrm{d}(x^1)$ by being an intervocalic /t/.

The transformation is almost complete. We also define output functions for the binary relation $\mathsf{succ:seg}(x,y)$. Since intervocalic voicing does not change the ordering among segments, the output function is straightforward. The only caveat is that, as a binary relation, the output function $\phi\mathsf{succ:seg}(x^1, y^1)$ needs to specify that the two variables x, y belong to some copy of the input. Here, both belong to the same 1st copy of the input.

(184)   *Output function for binary relations for intervocalic voicing*
    - $\phi\mathsf{succ:seg}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{succ:seg}(x,y)$

Like user-defined predicates, output functions over some output copy c can reference information from another output copy d, or even from the same copy c as long as there is no circularity; that is, the two output functions f, g do not reference each other.

4.3   Other minor aspects in logical formalization

Having illustrated the core aspects of the logical notation, I now go over some minor aspects. These are explained through simple phonological illustrations.

---

superscript in x^c not as a new variable, but as a composite identity that indicates the index *c.x*. This is different form other notations where the index is a superscript and the label is a subscript on ϕ: $\phi^c_{lab}(x)$ (Engelfriet and Hoogeboom 2001). That notation is standard and has been used in phonology (Jardine 2017b). Following Strother-Garcia (2019), I have chosen the alternative notation for easier readability.

(185)   *Minor aspects in the logical notation*

a.	negation	word edges	§4.3.1
b.	modifying binary relations	deletion	§4.3.2
c.	copy sets	epenthesis	§4.3.3
d.	helper predicates	metathesis	§4.3.4
e.	disjunction and consistency	multiple processes	§4.3.5

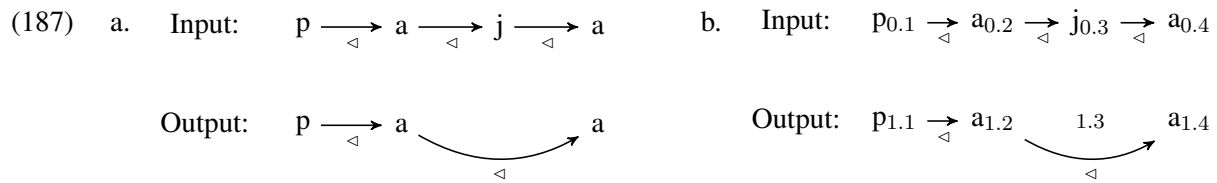### 4.3.1   Negation and edge-position

Given some input segment x in a string vwx (xyz), there are many ways to encode the fact that x is final (or initial). A brief list includes: x precedes a boundary symbol #, x precedes itself, or x precedes nothing. I use the last encoding by using user-defined predicates (186).

(186)   *Encoding initial and final segments*
- $\textbf{initial:seg}(x) \overset{\text{def}}{=} \neg \exists w[\textsf{succ:seg}(w, x)]$
- $\textbf{final:seg}(x) \overset{\text{def}}{=} \neg \exists y[\textsf{succ:seg}(x, y)]$

4.3.2   Deletion and modifying binary relations

In the previous examples, the output did not differ from its input in terms of its binary relations. Here, I illustrate a case where the output does differ. Consider a hypothetical process of intervocalic glide *j*-deletion: *papa~papa* but *paja~pai*. The input-output pair *paja~paa* is represented in (187a).

(187)   a.   Input: $p \longrightarrow a \longrightarrow j \longrightarrow a$ b.   Input: $p_{0.1} \rightarrow a_{0.2} \rightarrow j_{0.3} \rightarrow a_{0.4}$

  Output: $p \longrightarrow a \qquad a$ Output: $p_{1.1} \rightarrow a_{1.2} \qquad 1.3 \qquad a_{1.4}$

Note that in the more explicit representation (187b), a copy of the *j* is generated as an index '1.3'. This copied node is vacuous and doesn't surface because it does not have any labels or relations with other nodes. In the more explicit representation, this is visualized by the lack of any segment label.

Assume that we have a simple phoneme inventory of {p,a,j}. The set of unary labels is then just the set: $\{\textsf{p}(x), \textsf{a}(x), \textsf{j}(x)\}$. To model the transformation, we need to delete the intervocalic /j/ and change the successor relations around the /j/. The first part is modeled with the following output functions (188) which are defined for the unary labels. Every label except for an intervocalic /j/ is faithfully outputted.

(188)   *Output functions for unary labels for glide deletion*
- $\phi \textsf{a}(x^1) \overset{\text{def}}{=} \textsf{a}(x)$
- $\phi \textsf{p}(x^1) \overset{\text{def}}{=} \textsf{p}(x)$

- $\phi\mathsf{j}(x^1) \overset{\text{def}}{=} \mathsf{j}(x) \wedge \neg\mathbf{intervocalic}(x)$

To change the binary relations, we use the user-defined predicate **to_be_deleted**(x) in (189a) to pick out the segments which will delete because they are an intervocalic /j/. With this predicate at hand, we output the successor relations (189b) *carefully* so that we skip the intervocalic glide and make its flanking segments be adjacent. The formula consists of two disjuncts. Given two input nodes x, y and their output correspondents x^1, y^1, x^1 will precede y^1 if x, y satisfy either of the two disjuncts.

(189) a. *User-defined predicates for glide deletion*
- $\mathbf{to\_be\_deleted}(x) \overset{\text{def}}{=} \mathsf{j}(x) \wedge \mathbf{intervocalic}(x)$

b. *Output functions for binary relations for glide deletion*
- $\phi\mathsf{succ:seg}(x^1, y^1) \overset{\text{def}}{=} [\mathsf{succ:seg}(x, y) \wedge \neg\mathbf{to\_be\_deleted}(x) \wedge \neg\mathbf{to\_be\_deleted}(y)] \vee$
$\exists w[\mathsf{succ:seg}(x, w) \wedge \mathsf{succ:seg}(w, y) \wedge \mathbf{to\_be\_deleted}(w)]$

For the first disjunct in $\phi\mathsf{succ:seg}(x^1, y^1)$, the output node x^1 precedes the output node y^1 iff x underlyingly precedes y and neither of them are deletable. For a different input-output pair like *pajp∼pajp*, this disjunct is true for the symbols *a,j* because neither is an intervocalic glide.

For the second disjunct, the output node x^1 precedes the output node y^1 iff there exists another segment w such that x underlyingly precedes w and w underlyingly precedes y as in the string *xwv*, and the input segment w is an intervocalic glide that is supposed to delete. For the vowels *a,a* in *paja∼paa*, they satisfy this disjunct and are thus adjacent in the output string.

4.3.3   Copy sets and epenthesis

In the previous illustrations, the transformations used only *one* copy of the input. This section illustrates when more copies will be needed. A simple case is schwa epenthesis between any two consonants: *apta∼apəta*.

(190) a.   Input: b.   Input:

The input consists of 4 segments. The output consists of 5 because of the epenthetic schwa. In order to produce this many segments, we must include an additional copy of the input into our copy set: Copy 2 on the third row in (190b). In the second copy, the 'correspondent' of the 'p' is a schwa and it surfaces at index '2.2'. The correspondents of the other input segments do not surface at all in the second copy ('2.1', '2.3', and '2.4' are vacuous). The transformation involves the following steps which can be done in any order.

1. output the labels on the first copy *apta*

2. order the segments on the first copy *except* for the bad **pt* cluster
3. output the labels on the second copy's schwa
4. order the schwa on the second copy *with* the bad **pt* cluster on the first copy

We assume that the language has a simple phoneme inventory $\{p,t,a,ə\}$ with unary labels $\{p(x), t(x), a(x), ə(x)\}$. In the first copy, we faithfully output all segment labels (191). We use the shorthand $\mathsf{lab} \in L$.

(191) *Output functions for unary labels over Copy 1 for epenthesis*
- for every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x^1) \overset{\text{def}}{=} \mathsf{lab}(x)$

As for binary relations, we use a predicate **bad_cluster**(x, y) to pick out bad consonant clusters (192a), i.e., any pair of two consonants x, y which are in a sequence. For the output function $\phi\mathtt{succ:seg}(x^1, y^1)$ (192b), consider any two input symbols x, y such that x precedes y. In their output in the first copy, x^1 precedes y^1 as long as x, y don't form a bad cluster. Thus for the input-output pair *apta*∼*apəta*, only the underlined substrings in the output *ap*ə*ta* are in a successor relationship over Copy 1.

(192) a. *User-defined predicates for epenthesis context*
- **consonant**$(x) \overset{\text{def}}{=} p(x) \vee t(x)$
- **bad_cluster**$(x, y) \overset{\text{def}}{=} \textbf{consonant}(x) \wedge \textbf{consonant}(y) \wedge \mathsf{succ:seg}(x, y)$

b. *Output functions for binary relations over Copy 1 for for epenthesis*
- $\phi\mathtt{succ:seg}(x^1, y^1) \overset{\text{def}}{=} \mathsf{succ:seg}(x, y) \wedge \neg\textbf{bad\_cluster}(x, y)$

As for Copy 2, a schwa is 'added' inside a consonant cluster, i.e., a schwa is 'added' after a consonant which precedes another consonant. The user-defined predicate **cluster_initial**(x) (193a) captures this context. It is satisfied if x is a consonant which precedes another consonant y.

(193) a. *User-defined predicate for finding cluster-initial consonant in epenthesis*
- **cluster_initial**$(x) \overset{\text{def}}{=} \textbf{consonant}(x) \wedge \exists y[\mathsf{succ:seg}(x, y) \wedge \textbf{consonant}(y)]$

b. *Output functions for unary labels over Copy 2 for epenthesis*
- $\phi ə(x^2) \overset{\text{def}}{=} \textbf{cluster\_initial}(x)$
- For every other label $\mathsf{lab} \in L - \{ə(x)\}$:
  $\phi\mathtt{lab}(x^2) \overset{\text{def}}{=} \text{FALSE}$

c. *Output functions for binary relations across Copy 1 and 2 for epenthesis*
- $\phi\mathtt{succ:seg}(x^1, y^2) \overset{\text{def}}{=} (x = y) \wedge \textbf{cluster\_initial}(x)$
- $\phi\mathtt{succ:seg}(x^2, y^1) \overset{\text{def}}{=} \textbf{cluster\_initial}(x) \wedge \mathsf{succ:seg}(x, y)$

With $\phi ə(x^2)$ (193b), a schwa surfaces in the second copy iff it 'corresponds' to a cluster-initial consonant in the input. For the input-output pair *apta*∼*apəta*, the schwa is in correspondence with the underlined input segment *t* in *ap*t*a* which is cluster-initial.

To order this schwa with the underlying cluster, it has to be ordered with both the cluster-initial and cluster-final consonants via $\phi\mathtt{succ:seg}(x^1, y^2)$ and $\phi\mathtt{succ:seg}(x^2, y^1)$ respectively (193c). For the

cluster-initial consonant via ϕsucc:seg(x^1, y^2), a schwa in the second copy y^2 follows the consonant x^1 in the first copy. Both segments correspond to the *same* input segment or domain element $(= x = y)$. This domain element x is underlyingly cluster-initial in the input. The above graphs from (190) are repeated below to better illustrate this for the input-output pair *apta~apəta*.

(194)   a.   b.

Input: a $\longrightarrow$ p $\longrightarrow$ t $\longrightarrow$ a   Input: $a_{0.1} \longrightarrow p_{0.2} \longrightarrow t_{0.3} \longrightarrow a_{0.4}$

Output: a $\longrightarrow$ p   t $\longrightarrow$ a   Output: $a_{1.1} \longrightarrow p_{1.1}$ $t_{1.2} \longrightarrow a_{1.3}$

                          ə   2.1 $ə_{2.2}$ 2.3   2.4

Analogously for the cluster-final segment, the schwa must be ordered before it via ϕsucc:seg(x^2, y^1) (193c). A schwa x^2 from the second copy will precede a consonant y^1 from the first copy iff x^2 corresponds to an underlying cluster-initial consonant x in the input, and the underlying correspondent x precedes the underlying correspondent y in the input.

4.3.4   Helper predicates and metathesis

Deletion illustrates that certain transformations require modifying binary relations. This section illustrates a case where the modification is affected by multiple separate conditions in the input. To facilitate writing out these different factors, I introduce **helper predicates** as a shorthand. Consider metathesis whereby post-CV intervocalic /r/ in a string CVr is metathesized to CrV as in *uparit~uprait* (195).

(195)   a.   Input: u $\longrightarrow$ p $\longrightarrow$ a $\longrightarrow$ r $\longrightarrow$ i $\longrightarrow$ t

        Output: u $\longrightarrow$ p   a   r   i $\longrightarrow$ t

        b.   Input: $u_{0.1} \longrightarrow p_{0.2} \longrightarrow a_{0.3} \longrightarrow r_{0.4} \longrightarrow i_{0.5} \longrightarrow t_{0.6}$

        Output: $u_{1.1} \longrightarrow p_{1.2}$ $a_{1.3}$ $r_{1.4}$ $i_{1.4} \longrightarrow t_{1.4}$

As shown, the following changes occur because of metathesis:

(196)   *Changes created by metathesis*
        a.   **Create Cr**: the onset *p* now precedes *r*
        b.   **Create ra**:the *r* now precedes the (underlyingly) pre-rhotic vowel *a*

    c. **Create ai**: the (underlyingly) pre-rhotic vowel *a* now precedes the (underlyingly) post-rhotic vowel *i*

On the one hand, modeling metathesis involves changing the output's successor relationship in much the same way as for deletion. However, its context and change can look convoluted. To illustrate, assume the language has a small phoneme inventory of {u,p,a,r,i,t} with labels $\{u(x), p(x), a(x), r(x), i(x), t(x)\}$. To pick out the relevant segments involved in metathesis, we use the user-defined predicates in (197).

(197)   *User-defined predicates for metathesis*

    a. **vowel**$(x) \stackrel{\text{def}}{=} a(x) \vee i(x)$

    b. **consonant**$(x) \stackrel{\text{def}}{=} \neg\textbf{vowel}(x)$

    c. **bad_r**$(x) \stackrel{\text{def}}{=} r(x) \wedge \exists v, w, y[\text{succ:seg}(v, w) \wedge \text{succ:seg}(w, x) \wedge \text{succ:seg}(x, y) \wedge$
                        $\textbf{consonant}(v) \wedge \textbf{vowel}(w) \wedge \textbf{vowel}(y)$

    d. **post-rhotic_V**$(x) \stackrel{\text{def}}{=} \textbf{vowel}(x) \wedge \exists z[\text{succ:seg}(z, x) \wedge \textbf{bad\_r}(z)]$

    e. **pre-rhotic_V**$(x) \stackrel{\text{def}}{=} \textbf{vowel}(x) \wedge \exists z[\text{succ:seg}(x, z) \wedge \textbf{bad\_r}(z)]$

    f. **pre-rhotic_C**$(x) \stackrel{\text{def}}{=} \textbf{consonant}(x) \wedge \exists y, z[\text{succ:seg}(x, y) \wedge \text{succ:seg}(y, z) \wedge$
                       $\textbf{pre-rhotic\_V}(y) \wedge \textbf{bad\_r}(z)]$

    g. **metathesis_context**$(x) \stackrel{\text{def}}{=} \textbf{bad\_r}(x) \vee \textbf{pre-rhotic\_C}(x) \vee$
                             $\textbf{pre-rhotic\_V}(x) \vee \textbf{post-rhotic\_V}(x)$

The predicates **vowel**(x) and **consonant**(x) pick out vowels and consonants. The predicate **bad_r**(x) picks out the /r/ which will metathesize based on its context. By using the variables v, w, y, the predicate picks out an /r/ which is in some substring $C_v V_w r V_y$: the variables v, w, x, y are linearly ordered via successor and v, w, y are a consonant and two vowels respectively.

The predicates **post-rhotic_V**(x), **pre-rhotic_V**(x), **pre-rhotic_C**(x) pick out the neighboring segments based on straightforward definitions. The predicate **metathesis_context**(x) picks out any segment that's part of the metathesis context, i.e. a symbol in the input substring CVrV.

The unary labels in the output stay the same. The only change is in the succ:seg(x, y) relation. There are two ways to write out the relevant changes, either with a single large output function or with a relatively small output function that uses **helper predicates**. A helper predicate is a type of user-defined predicate which is used to categorize different conditions for binary relations. To illustrate, consider again the input-output pair *uparit*∼*uprait* (198) where 3 of the 5 underlying successor relations are modified.

(198)   a.   Input:

           Output:
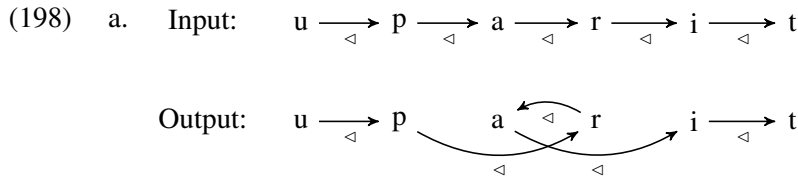
Informally, any segment outside of the local context of metathesis maintains its underlying order: it *stays* in the same position. This is formalized using the helper predicate **should__succ__stay**(x, y).

Helper predicates have the template **should__lab__cond**(x) for labels and **should__rel__cond**(x, y) for binary relations. They start with the word *should__* in their name with two underscores. Their names ends with mnemonic name *__cond* on the relevant condition.

(199)   *A helper predicate for metathesis*

- **should__succ__stay**$(x, y) \stackrel{\text{def}}{=} \mathsf{succ{:}seg}(x, y) \wedge$
$$\neg[\mathbf{metathesis\_context}(x) \wedge \mathbf{metathesis\_context}(y)]$$

The helper predicate **should__succ__stay**(x, y) picks out any underlying successor relation between a pair of input segments x, y such that both segments are *outside* the metathesis context. For the input-output pair *uparit~uprait*, this predicate picks out the underlined successor relations in the input *up̲ar̲it*.

The helper predicates in (200) pick out what *should* become the successor relations for the pre-rhotic consonant and the rhotic (200a), the rhotic and the pre-rhotic vowel (200b), and the pre-rhotic vowel and the post-rhotic vowel (200c). Their names match with the changes listed in (196).

(200)   *Other helper predicates for metathesis*

a. **should__succ__Cr**$(x, y) \stackrel{\text{def}}{=} \mathbf{pre\text{-}rhotic\_C}(x) \wedge \mathbf{bad\_r}(y) \wedge$
$$\exists w[\mathsf{succ{:}seg}(x, w) \wedge \mathsf{succ{:}seg}(w, y) \wedge \mathbf{pre\text{-}rhotic\_V}(w)]$$

b. **should__succ__ra**$(x, y) \stackrel{\text{def}}{=} \mathbf{bad\_r}(x) \wedge \mathbf{pre\text{-}rhotic\_V}(y) \wedge \mathsf{succ{:}seg}(y, x)$

c. **should__succ__ai**$(x, y) \stackrel{\text{def}}{=} \mathbf{pre\text{-}rhotic\_V}(x) \wedge \mathbf{post\text{-}rhotic\_V}(t) \wedge$
$$\exists z[\mathsf{succ{:}seg}(x, z) \wedge \mathsf{succ{:}seg}(z, y) \wedge \mathbf{bad\_r}(z)]$$

With the above helper predicates, the *actual* output function for successor is the *disjunction* of all of them.

(201)   *Output function for binary relations for metathesis*

- $\phi\mathsf{succ{:}seg}(x^1, y^1) \stackrel{\text{def}}{=}$ **should__succ__stay**$(x, y) \vee$ **should__succ__Cr**$(x, y) \vee$
should__succ__ra$(x, y) \vee$ **should__succ__ai**(x, y)

The table below summarizes the roles played by these different helper predicates.

(202)   *Division of labor across the helper predicates for metathesis for* uparit→uprait

should__succ__stay(x,y)	**should__succ__Cr**(x,y)	**should__succ__ra**(x,y)	**should__succ__ai**(x,y)
Keep underlying order	Consonant precedes /r/	/r/ precedes the vowel	Vowels are adjacent
up̲ra̲it	*up̲rait*	*upr̲a̲it*	*upra̲it̲*

Without using these helper predicates, the output function $\phi\mathsf{succ{:}seg}(x^1, y^1)$ would have been substantially larger and harder to understand. The function would have each helper predicate be replaced by its definition. Helper predicates will be used in this thesis to encode different contexts for different rules. The next section shows their utility for encoding multiple processes that affect the same labels or relations.

4.3.5 Consistency and disjunction

Using a computational formalism forces us to make all information be explicit. Lack of full explicitness can cause our computational model to be inconsistent. However, full explicitness can make it harder for us to understand the formalism. This happens when we have multiple processes affecting the same labels or relations in the output. I explain this problem, and present one solution with disjunction.

Consider intervocalic voicing again in (203). The output function $\phi\mathsf{b}(x^1)$ will change an underlying /p/ into an output [b] if the segment is intervocalic.

(203)   *Output functions for unary labels for intervocalic voicing*

    a. $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{intervocalic}(x)]$

A problem arises if the language has a separate process which also involves the same output function $\phi\mathsf{b}(x^1)$, e.g., consonants are voiced after nasals: *anpa→anba*. To encode just this process, we need an output function which changes an underlying /p/ to [b] if the consonant is post-nasal. This context is encoded in the user-defined predicate **post-nasal**(x) and utilized in the output function $\phi\mathsf{b}(x^1)$.

(204)   *User-defined predicate and output function for post-nasal voicing*

    a. $\textbf{post-nasal}(x) \stackrel{\mathsf{def}}{=} \exists y[\mathsf{nasal}(y) \wedge \mathsf{succ{:}seg}(y, x)]$

    b. $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{post-nasal}(x)]$

If the grammar contains both of these rules, then we have an inconsistent definition for the output function $\phi\mathsf{b}(x^1)$ (205a). There cannot be two output functions which have the same name. To fix this, we need to redefine $\phi\mathsf{b}(x^1)$ as the disjunction of the two original processes or output functions (205b).[2]

(205)   a. *Inconsistent grammar with two definitions for* $\phi\mathsf{b}(x^1)$

      • $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{intervocalic}(x)]$

      • $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{post-nasal}(x)]$

    b. *Consistent grammar which uses disjunction for* $\phi\mathsf{b}(x^1)$

      • $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} [\mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{intervocalic}(x)]]\vee$
             $[\mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{post-nasal}(x)]]$

The problem with this solution is that it can affect readability. It masks the fact that there are two conditions for voicing based on two separate processes. Another way to solve the inconsistency is to redefine the two original output functions as two separate helper predicates (206a). The output function $\phi\mathsf{b}(x^1)$ is then defined as the disjunction of these two helper predicates (206b).

(206)   a. *Helper predicates for the two processes which generate* [b]

      • $\textbf{should\_b\_intervocalic}(x) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{intervocalic}(x)]$

      • $\textbf{should\_b\_postnasal}(x) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge \textbf{post-nasal}(x)]$

---

[2]The function can be simplified to $\phi\mathsf{b}(x^1) \stackrel{\mathsf{def}}{=} \mathsf{b}(x) \vee [\mathsf{p}(x) \wedge [\textbf{intervocalic}(x) \vee \textbf{post-nasal}(x)]]$.

    b. *Consistent grammar which uses disjunction of helper predicates for $\phi b(x^1)$*

- $\phi b(x^1) \stackrel{\text{def}}{=} \textbf{should\_\_b\_\_intervocalic}(x) \vee \textbf{should\_\_b\_\_postnasal}(x)$

Because the goal of this thesis is to design a formalization of the morphology-phonology interface, I have run into the problem of consistency.[3] For example, I treat prosodic mapping as a set of logical transductions, each for different possible parses. Chapter 5 sets up one logical transduction which uses one definition of the output function $\phi\texttt{PStem}(x^2)$. This function generates a PStem when given an unparsed MStem which dominates an MRoot (207a-i); the formula of this function is explained in the corresponding chapter. But in Chapter 6, I set up a separate transduction that uses an output function of the same name. This generates a PStem when given an unparsed MStem which dominates an MStem (207a-ii). To unify these two logical transductions, we need to unify their output functions. To improve readability, I do not do the unification in this dissertation. However, their unification can be easily done with disjunction over helper predicates, as demonstrated in (207b).

(207)   a. *Inconsistent grammar that use two output functions from Chapters 5 and 6*

      i. *Output function for generating a PStem given a non-recursive MStem*

- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \textsf{Parse:MStem:nonrecursive}(\textsc{settings}) \wedge \textsf{MStem}(x)$

      ii. *Output function for generating a PStem given a recursive MStem*

- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \textsf{Parse:MStem:recursive}(\textsc{settings})\wedge$
  $\textsf{MStem}(x) \wedge \textbf{MTopmost}(x)\wedge$
  $\neg \exists y[\textsf{PStem}(y) \wedge \textsf{Match:stem}(x,y)]$

   b. *Consistent grammar that uses helper predicates*

      i. *Helper predicates based on the original output functions*

- $\textbf{should\_\_PStem\_\_nonrecursive}(x) \stackrel{\text{def}}{=} \textsf{Parse:MStem:nonrecursive}(\textsc{settings})\wedge$
  $\textsf{MStem}(x)$
- $\textbf{should\_\_PStem\_\_recursive}(x) \stackrel{\text{def}}{=} \textsf{Parse:MStem:recursive}(\textsc{settings})\wedge$
  $\textsf{MStem}(x) \wedge \textbf{MTopmost}(x)\wedge$
  $\neg\exists y[\textsf{PStem}(y) \wedge \textsf{Match:stem}(x,y)]$

      ii. *Output function based on these helper predicates*

- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \textbf{should\_\_PStem\_\_nonrecursive}(x) \vee$
  $\textbf{should\_\_PStem\_\_recursive}(x)$

4.4 Generative capacity of logical formalization

Just as with rewrite grammars and finite-state machines, there are hierarchies of possible logical transductions. I go over this hierarchy and show how it corresponds to how much local vs. non-local information is used. I use this hierarchy to show that the bulk of the morphology-phonology interface uses local transductions.

---

[3]The consistency problem is not a major problem for computational *implementations* of logical programming. In Prolog at least, if any two rules have the same name, they are interpreted with implicit disjunction.

4.4.1   Locality and non-locality in logical transductions

Compared to finite-state transductions, there is relatively little work on setting up a hierarchy of subclasses for logical transductions. One hierarchy is defined in terms of how quantifiers are used. Quantifier-Free (QF) logic does not allow the use of quantifiers at all. First-Order (FO) logic allows the use of quantifiers over variables. Monadic Second-Order (MSO) logic allows quantifiers over variables or sets.

(208)   *Hierarchy of logical transductions*

Monadic Second Order (MSO)	*Quantifiers over variables $\exists x$ and sets $\exists X$*
│	
First Order (FO)	*Quantifiers over variables $\exists x$*
│	
Quantifier-Free (QF)	*No quantifiers*
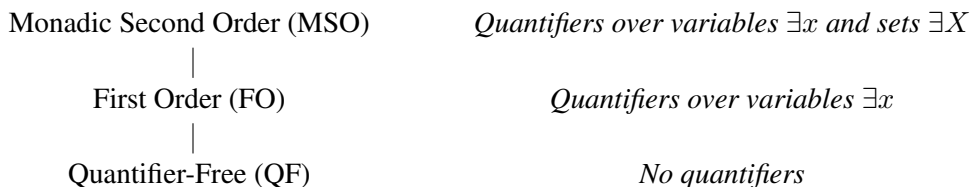
There are some correspondences between this hierarchy and our intuitions of what counts as a local process. QF logic can only capture local dependencies. FO logic can compute long-distance dependencies which do not reference direction. MSO logic can compute long-distance dependencies which depend on direction. To illustrate, consider the process of frication of /t/ to [s]: $t \rightarrow s$. This change can be triggered by different phonological factors in the input. I list some logically possible triggers below.

(209)   *Rules of varying locality*
1. **Contextless**:   /t/ becomes a [s] anywhere in the word: $t \rightarrow s/$ _
2. **Distant**:   /t/ becomes a [s] if there is an /i/ in the word: $t \rightarrow s/$ _...i or i..._
3. **Directional**:   /t/ becomes a [s] if there is an /i/ anywhere after the /t/: $t \rightarrow s/$ _...i

Assume that the langauges don't have a phonemic /s/; i.e., no underlying form has an /s/. The application of these rules is illustrated below. The rules correspond to QF, FO, and MSO functions.

(210)   *Application of rules with different locality*

Rule		Input			Expressivity
		tata	*itat*	*atupi*	
Contextless	$t \rightarrow s/$ _	sasa	isas	asupi	Quantifier Free (QF)
Distant	$t \rightarrow s/$ _...i or i..._		isas	asupi	First Order (FO)
Directional	$t \rightarrow s/$ _...i			asupi	Monadic Second Order (MSO)

With the **Contextless** rule, /t/ always becomes [s]: /tata/→[sasa]. This process is local and only looks at the target input symbol. It is computed by the QF output function in (211) which doesn't use any quantifiers.

(211)   *QF output function for the 'contextless' rule*
- $\phi_{\mathsf{s}}(x^1) \stackrel{\mathrm{def}}{=} \mathsf{t}(x)$

With the **Distant** rule, /t/ becomes [s] if there is an /i/ anywhere in the word: /tati/→[sasi] but /tata/→[tata]. The /i/ and /t/ can be at any distance from each other. The /i/ is thus a non-local trigger. The rule is computed by the FO function (212) which uses a quantifier to find this segment /i/.

(212)   *FO output function for the 'distant' rule*
- $\phi_\mathsf{S}(x^1) \overset{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\mathsf{i}(y)]$

With the **Directional** rule, /t/ becomes [s] if there is an /i/ which is at any distance *after* the /t/. We need MSO logic to determine if the long-distance trigger /i/ is after /t/ instead of just being anywhere in the input. Given the input string *atupi*, we can compute the fact that the /t/ non-immediately precedes the /i/ by using the *transitive closure* of immediate successor. Let X be some set of segments. X is said to be closed under immediate successor iff for any two segments x, y, if x is in X and x immediately precedes y, then y must also be in X. Visually, the set X is a set of segments which form an unbroken line of immediate successor from some point in the string all the way to the end, e.g. the underlined segments in *a<u>tupi</u>* could form a set X, but the underlined segments *a<u>tu</u>pi* cannot.

(213)   *MSO definition for the transitive closure of immediate successor*
- **closed:succ:seg**$(X) \overset{\text{def}}{=} \forall x, y[(x \in X \wedge \mathsf{succ:seg}(x, y)) \rightarrow y \in X]$

With X defined as the transitive closure of immediate successor, we can compute any long-distance or general precedence in a string. Intuitively, the trick is to make x be the point where the line X starts. Visually, for two segments x, y, x generally precedes y if we can draw an unbroken line from x to the end of the string *and* this line passes through y. Formally, x generally precedes y if x, y are different points, and if for any transitively closed set X, if x is in X (meaning x is part of a line), then y is also is part of *X*.

(214)   *MSO definition of general precedence from the transitive closure of immediate successor*
- **gen_prec:seg**$(x, y) \overset{\text{def}}{=} \forall X[(x \in X \wedge \textbf{closed:succ:seg}(X)) \rightarrow y \in X] \wedge x \neq y$

To illustrate with *atupi*, if x is the input segment /a/, then X is the substring *a<u>tupi</u>*. y can be any segment in this substring. For the segment /t/, X is the substring *at<u>upi</u>*, and for /p/, X is the substring *atu<u>pi</u>*. For /t/, X includes at least the segment /i/.

The predicate **gen_prec:seg**(x, y) uses MSO logic because it uses a universal quantifier $\forall$ over a set X. Any function which uses this predicate is also MSO. With this predicate, the MSO formula in (215) will change /t/ to [s] if it non-immediately precedes an /i/.

(215)   *MSO output function for the 'directional' rule*
- $\phi_\mathsf{S}(x^1) \overset{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\mathsf{i}(y) \wedge \textbf{gen\_prec:seg}(x, y)]$

4.4.2   Reduction to Quantifier-Free logic

The previous section showed that a process is QF if it does not reference any non-local information by using quantifiers. However, certain processes seem local at first but are not QF. This is because the use of binary-relations themselves requires quantifiers. In this section, I show that certain types of binary relations can be converted to **unary functions**. When local transformations are defined over these unary functions, the computation is QF. However, not every binary relation can be converted into a unary function; this means that processes which use these nonconvertible binary relations are necessarily non-local.

Consider again the process of frication of /t/ to [s]: $t \rightarrow s$. The table below now lists two additional processes which intuitively look local at first but are not QF.

(216) *Rules of varying locality*
1. **Contextless**: /t/ becomes a [s] anywhere in the word: $\quad$ $t \rightarrow s/$ _ $\quad$ *k-ISL* for *k*=1
2. **Adjacent**: /t/ becomes a [s] before an /i/: $\quad$ $t \rightarrow s/$ _*i* $\quad$ *k-ISL* for *k*=2
3. **Close**: /t/ becomes a [s] before a segment+/i/: $\quad$ $t \rightarrow s/$ _Σi $\quad$ *k-ISL* for *k*=3
4. **Distant**: /t/ becomes a [s] if there is an /i/ in the word: $\quad$ $t \rightarrow s/$ _...*i or i*..._ $\quad$ **Not** *k-ISL*
5. **Directional**: /t/ becomes a [s] if there is an /i/ after the /t/: $\quad$ $t \rightarrow s/$ _...*i* $\quad$ **Not** *k-ISL*

The Contextless, Adjacent, and Close rules are all computationally local. One formalization of locality are *k*-Input-Strictly Local functions for some natural number *k* (Vaysse 1986; Chandlee 2014, 2017; Chandlee et al. 2014, 2015, 2018; Chandlee and Heinz 2018). Informally, a rule is *Input Strictly Local* (ISL) for a number *k* if we only need to examine a finite number of *k*-1 segments before and after the target of the rule. The Contextless rule is ISL for a context size of *k*=1. The '1' is reserved for checking that the target itself is /t/. The Distant and Directional rules are not ISL for any *k* because they use long-distance information. In contrast, the Adjacent and Close rules are local but with larger *k* values.

The **Adjacent** rule will change a /t/ to an [s] if it precedes an /i/: /tati/$\rightarrow$[ta<u>s</u>i]. It uses a local context of size 2. The rule uses the output function below. The function checks that the target is /t/ and that there exists some segment y which follows /t/ and is an /i/.[4]

(217) *FO output function for the 'adjacent' rule*
- $\phi \mathsf{s}(x^1) \overset{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\mathsf{succ{:}seg}(x, y) \wedge \mathsf{i}(y)]$

In the **Close** rule, a /t/ becomes an [s] if the second segment after the /t/ is /i/: /atpi/$\rightarrow$[a<u>s</u>pi]. This /i/ and /t/ are not adjacent but they are within a finite window of each other. This window is of size *k*=3; thus the rule is local. The rule is formalized by the output function below. The function checks that the /i/ precedes some segment y, that y precedes a segment z, and that z is an /i/.

(218) *FO output function for the 'close' rule*
- $\phi \mathsf{s}(x^1) \overset{\text{def}}{=} \mathsf{t}(x) \wedge \exists y, z[\mathsf{succ{:}seg}(x, y) \wedge \mathsf{succ{:}seg}(y, z) \wedge \mathsf{i}(z)]$

Intuitively, the **Adjacent** and **Close** rules are local because they examine the local context of the target /t/. However, their output functions use quantifiers $\exists y, z$ and this makes them not be QF. These output functions can be converted to QF formulas by binding the variables y, z introduced by these quantifiers. Binding can be accomplished if the binary relations can be turned to unary functions. By doing so, we can replace the quantifier's variables y, z with terms which can be derived from x. The conversion is possible because the interpretation of the binary relation of immediate successor is a function: given some variable x, we can deterministically find these bounded variables y, z.

Consider the **Adjacent** rule (217) and the input *tati*. The variable y introduced by $\exists y$ picks out the segment which follows the target x. The selection is done by the binary relation $\mathsf{succ{:}seg}(x, y)$ which conceptually

---

[4]An additional output function $\phi \mathsf{t}(x^1)$ is needed in all these different rules in order to generate any faithful /t/s, not shown here.

encodes a function. Any given node x will precede either one unique node y or no node at all. This binary relation can be converted to two functions $\mathsf{F_L}$:succ:seg(x) and $\mathsf{F_R}$:succ:seg(y). The subscripts $_{L,R}$ tells us if we're searching from the left variable x vs. the right variable y. Thus, the function $\mathsf{F_L}$:succ:seg(x) returns the node y which follows the left ($_L$) variable x; the function $\mathsf{F_R}$:succ:seg(y) finds the node x which precedes the right ($_R$) variable y. I list below the values of these two functions for the input word *tati*.

(219)   a. *Immediate successors as a binary relation for* tati *or* $t_1a_2t_3i_4$
        - succ:seg(x, y) is TRUE for $\{(1,2), (2,3), (3,4)\}$
    b. *Immediate successors as two unary functions for* tati *or* $t_1a_2t_3i_4$
        - $\mathsf{F_L}$:succ:seg(x) is defined as
            – $\mathsf{F_L}$:succ:seg$(1) = 2$
            – $\mathsf{F_L}$:succ:seg$(2) = 3$
            – $\mathsf{F_L}$:succ:seg$(3) = 4$
        - $\mathsf{F_R}$:succ:seg(x) is defined as
            – $\mathsf{F_R}$:succ:seg$(2) = 1$
            – $\mathsf{F_R}$:succ:seg$(3) = 2$
            – $\mathsf{F_R}$:succ:seg$(4) = 3$

I use the term **unary function** to refer to derived functions to distinguish them from output functions.

Initial and final segments are picked out by the user-defined predicates in (220a), repeated from (186). These predicates negate an existential quantifier. To make these predicates QF, we replace the binary relation succ:seg(x, y) with the unary functions $\mathsf{F_L}$:succ:seg(x) and $\mathsf{F_R}$:succ:seg(y) (220b). We check that the unary function $\mathsf{F_L}$:succ:seg(x) ($\mathsf{F_R}$:succ:seg(x)) is undefined for the final (initial) segment.

(220)   a. *FO encoding initial and final segments*
        - **initial:seg**$(x) \overset{\text{def}}{=} \neg \exists w[\text{succ:seg}(w, x)]$
        - **final:seg**$(x) \overset{\text{def}}{=} \neg \exists y[\text{succ:seg}(x, y)]$
    b. *QF encoding initial and final segments*
        - **initial:seg**$(x) \overset{\text{def}}{=} \mathsf{F_R}$:succ:seg$(x) = $ NULL
        - **final:seg**$(x) \overset{\text{def}}{=} \mathsf{F_L}$:succ:seg$(x) = $ NULL

I use the term NULL such that any element which is not defined is equal to NULL. However, the logical formalism doesn't let us define undefined terms. Intuitively, I treat the term NULL as a constant in our domain D; it acts similarly to a sink state in finite-state automata.[5]

When finding the successor of the successor of some node x, we need to use two layers of $\mathsf{F_L}$:succ:seg(x). For readability, we can collapse these two layers with a superscript $\mathsf{F_L}$:succ:seg$^2(x)$.

(221)   *Finding the successor of a successor of* x
    a. *FO: with binary relations*
        - $\exists y, z[\text{succ:seg}(x, y) \wedge \text{succ:seg}(y, z)]$

---

[5]An alternative formalization is to define **initial:seg**(x) as true when x precedes itself (Thomas 1982; Strother-Garcia 2019). I did not use this approach because it can get problematic in certain morphological configurations like gemination or reduplication where a node in the graph can precede itself (Raimy 2000; Papillon 2020).

b. *QF: with unary functions*
- $\mathsf{F_L}\text{:succ:seg}(\mathsf{F_L}\text{:succ:seg}(x))$
- $\mathsf{F_L}\text{:succ:seg}^2(x)$

Consider again the different types of local and non-local rules. Below, I replace the binary relation succ:seg(x, y) with the unary functions $\mathsf{F_L}$:succ:seg(x) and $\mathsf{F_R}$:succ:seg(y) whenever possible. Using these unary functions lets us replace some of the existential quantifiers, e.g., in the **Adjacent** and **Close** rules.

(222) *Output functions with and without binary relations for the...*

a.	*'contextless' rule*	$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x)$	QF
b.	*'adjacent' rule*	$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\text{succ:seg}(x, y) \wedge \mathsf{i}(y)]$	FO
		$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \mathsf{i}(\mathsf{F_L}\text{:succ:seg}(x))$	QF
c.	*'close' rule*	$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \exists y, z[\text{succ:seg}(x, y) \wedge \text{succ:seg}(y, z) \wedge \mathsf{i}(z)]$	FO
		$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \mathsf{i}(\mathsf{F_L}\text{:succ:seg}(\mathsf{F_L}\text{:succ:seg}(x)))$	QF
		$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \mathsf{i}(\mathsf{F_L}\text{:succ:seg}^2(x))$	QF
d.	*'distant' rule*	$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\mathsf{i}(y)]$	FO
e.	*'directional' rule*	$\phi\mathsf{s}(x^1) \stackrel{\text{def}}{=} \mathsf{t}(x) \wedge \exists y[\mathsf{i}(y) \wedge \textbf{gen\_prec:seg}(x, y)]$	MSO

In the **Adjacent** rule, the variable y is replaced by directly using the successor of x via the unary function $\mathsf{F_L}$:succ:seg(x). In the **Close** rule, the variable z is replaced by the unary function $\mathsf{F_L}$:succ:seg(y). But the variable y itself is also replaced by $\mathsf{F_L}$:succ:seg(x). Ultimately, this means that z is replaced by $\mathsf{F_L}$:succ:seg$^2(x)$. No such replacements can be made in the **Distant** and **Directional** rules.

By replacing the binary relations with unary functions, the output functions for the **Adjacent** and **Close** rules no longer use quantifiers. They are now QF output functions. When the input is a simple string of segments, then its word-signature uses immediate successor as its only binary relation. This relation can always be turned into two unary functions. Strother-Garcia (2018, 2019) is an extended illustration which shows how QF treatments for successor can be used to show that syllabification is a local QF process. However, not everything can be made QF. There's no way to remove quantifiers from truly long-distant processes like the Distant and Directional Rules.

4.5 Hierarchical representation in the morphology and prosody

Having defined a word signature for linear strings, I now add new labels and relations to this word signature in order to formalize the hierarchical structures present in the morphology-phonology interface. Beyond being a string of segments, a word consists of two separate but related representational hierarchies or data structures (Booij and Lieber 1993): a morphological tree and a prosodic tree. These two trees meet at the segments. I formalize these two separate representational systems and how they can locally computed.[6] I define the unary labels and binary relations which create these tree structures for morphology (§4.5.1) and prosody (§4.5.2). I show how unary functions can be created for some but not all of these binary relations (§4.5.3). This shows that some but not all logically possible operations for morpho-prosodic trees are local.

---

[6]See Bird (1995); Coleman (1998) on previous formalizations of these trees as directed acyclic graphs.

There are many different theories and types of representations for morphology (Hockett 1942; Aronoff 1976; Anderson 1992; Stump 2001; Embick 2015). For the sake of concreteness, I assume a simple representation where words form morphological trees, e.g., in word-syntax (Lieber 1980; Selkirk 1982) or Distributed Morphology (Halle and Marantz 1993). An alternative is to treat words as a bundle of morphosyntactic features instead of a tree (Stump 2001). As I discuss in Chapter 7 (§7.2), there is little computational difference among these different morphological theories (Roark and Sproat 2007:ch3). I essentially adopt an item-and-arrangement approach to morphological representation, and an item-and-process approach to morphological generation (cf. word-formation rules Aronoff 1976).

4.5.1   Representation of morphological structure

Morphologically, a simple analysis of 'apple' decomposes it into a **morphological root** (MRoot) or $\sqrt{}$ (223a). Explicitly, the segments are **morphologically dominated** ($\sim$ *MDom* or *m*) by the root node (223b).

(223)   a.

A more sophisticated analysis decomposes this word even further. In the morphological structure, specific nodes in the graph are **morphological nodes** (MNode) of different 'chunks' (224). The smallest chunk corresponds to a **morph** (Aronoff 1994; Wolf 2008; Haspelmath 2020). It dominates zero or more segments. A morph is dominated by a **morpheme** which is marked by some number of morphosyntactic features (Halle and Marantz 1993). For 'apple', the morpheme is a simple MRoot.

(224)   a.

A morpheme can be a morphological root or an **affix**. An affix is specified for its status (derivational vs. inflectional). Derivational affixes are also called category suffixes with the features *n,v,a*; they form nouns, verbs, and adjectives respectively. Inflectional affixes mark some morphosyntactic feature like plurality or gerundive. To illustrate, consider the derivational suffix *-ness* and inflectional suffix *-ing* (225). Each is a morpheme that dominates some morph which dominates some string of segments. The morpheme *-ness* has the labels noun(x) and der(x), while *-ing* has gerund(x) and infl(x). I illustrate their feature label as their main label; the feature labels are illustrated as single letters *n* or in glossing conventions GRND but they are logically defined as the multicharacter symbols noun(x) and gerund(x).

(225)   a.
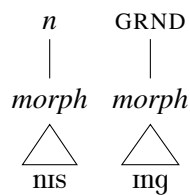
b.
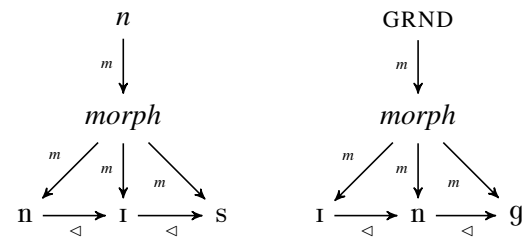
Higher types of morphological nodes are **morphological stems** (MStem) and **morphological words** (MWord). These are called **morphological constituents** (MConc). In the case of the morpheme 'apple', it is a free-standing root and forms an MStem with a covert derivational suffix of category *n* (Marantz 1997, 2007; Giegerich 1999). Explicitly, all of these morphological items are organized by **morphological dominance**. Zero morphs do not dominate any segment.

(226)   a.
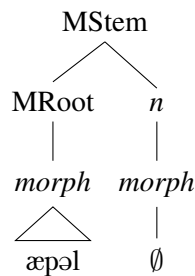
b.
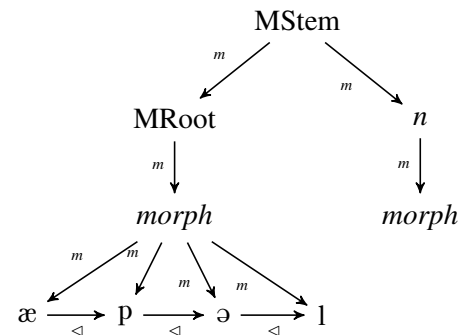
MStems are formed by adding more derivational suffixes, e.g., the overt suffix *-ness* on the adjective *dumb*. MWords are formed by adding covert or overt inflectional suffixes like *-ing* on the verb *love*.

(227)   a. *dumbness*

   i.

   ii.

b. *loving*

i.

MWord
MStem
MRoot *v* *grnd*
morph *morph* *morph*
lʌv ∅ -ɪŋg

ii.

MStem
MWord
MRoot *v* *grnd*
morph *morph* *morph*
l → ʌ → v → ɪ → ŋ → g
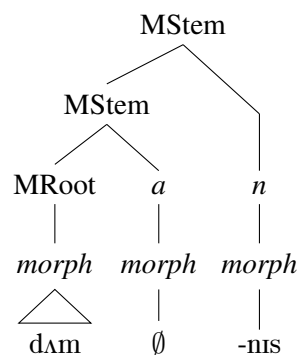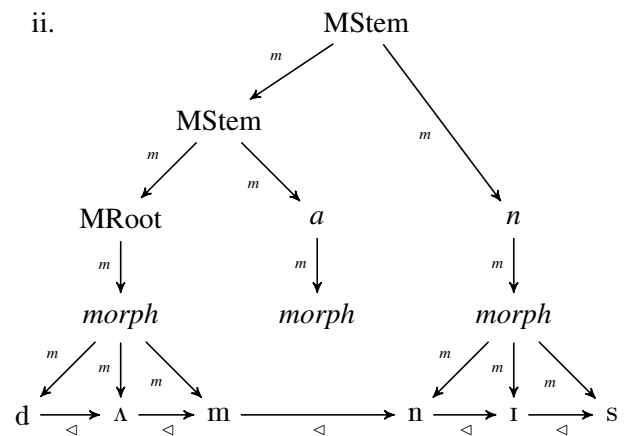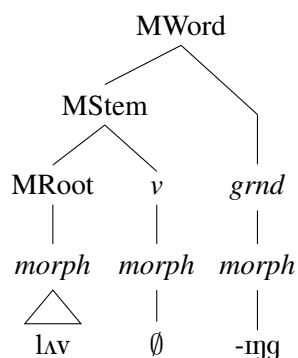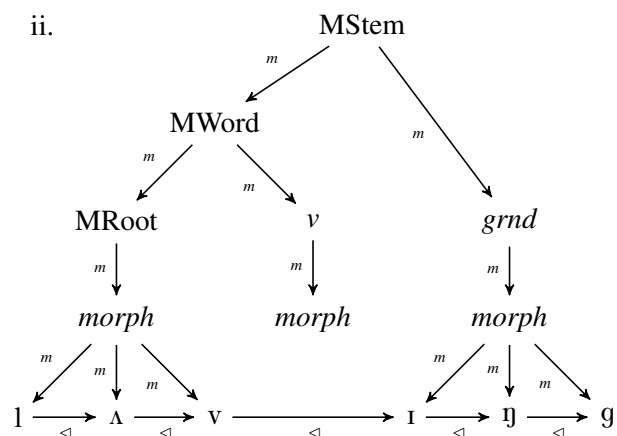
In general, *abstract* morphological information is relatively stable cross-linguistically. For Armenian, lexical items are similarly defined in terms of morphological nodes like morphs, morphemes, MRoots, MStems, and MWords. To illustrate, consider the simplex stem *amusin*, its derivative *amusn-agan*, and its inflected form *amusin-ov* below.

(228) a. i. amusin   'husband'   ii.

MStem
MRoot *n*
morph *morph*
amusin ∅

MStem
MRoot *n*
morph *morph*
a → m → u → s → i → n

b. i. amusn-agan   'marital'   ii.

MStem
MStem *a*
MRoot *n* *a*
morph *morph* *morph*
amusn ∅ -agan

MStem
MStem *a*
MRoot *n* *a*
morph *morph* *morph*
a → m → u → s → í → n → a → g → a → n
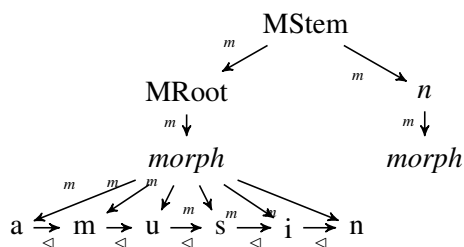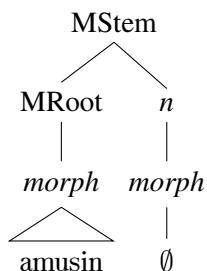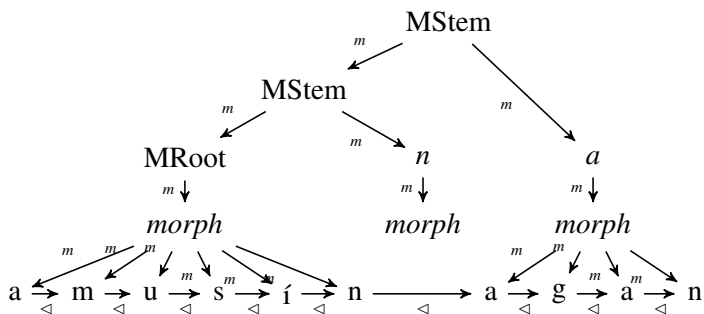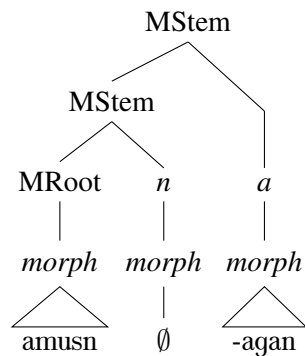
c.   i.   amusin-ov   'husband-INST' (WArm) ii.
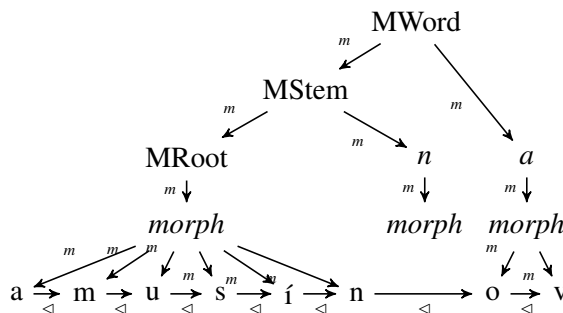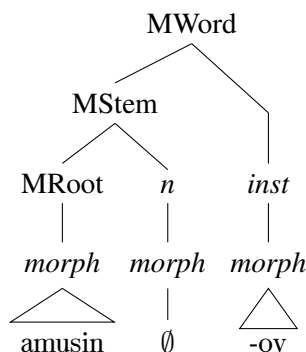
For either English or Armenian, all this information is represented by a simple set of unary labels (229a). To reiterate, a morphological node can be a MRoot, MStem, MWord, morpheme, or morph. A morpheme is an MRoot or Affix. A morphological constituent is an MStem or MWord. The set of labels can be enriched by marking the specific morphosyntactic features of the MNode (229b), such as the features of an inflectional affix, the category of a derivational suffix, or class diacritics for larger MNodes. All these MNodes are related via the binary relation of morphological dominance (229c).

(229)   a.   *Basic unary labels for morphological nodes*
   - $MNode(x)$: x is a morphological node (any morph, morpheme, MStem, or MWord)
   - $MConc(x)$: x is a morphological constituent (any MStem or MWord)
   - $morpheme(x)$: x is a morpheme (any root or affix)
   - $morph(x)$: x is a morph
   - $affix(x)$: x is an an affix
   - $der(x)$: x is an a derivational affix
   - $infl(x)$: x is an an inflectional affix
   - $MRoot(x)$: x is a morphological root
   - $MStem(x)$: x is a morphological stem
   - $MWord(x)$: x is a morphological word

   b.   *Additional unary labels for the features of morphological nodes*
   - $past(x), plural(x), gerund(x), inst(x), def(x), infinitival(x), irregular(x) \dots$
   - $noun(x), verb(x), adj(x)$
   - $Class{:}E(x), Class{:}A(x), Class{:}I(x) \dots$

   c.   *Basic binary relations for morphological nodes*
   - $MDom(x, y)$: a morphological node x morphologically dominates some other morphological node or segment y

I assume that a *morph* can dominate any number of segments. In this thesis, a morpheme can dominate only one morph. Cases of multiple exponence are not discussed. Morphological constituents (MStems, MWords) can dominate only two other MNodes (MWords, MStems, MRoots, morphemes). This makes the morphological dominance from MNodes to MNodes binary branching in *general*. To formalize compounds with linking vowels, I introduce new types of morphological nodes in Chapter 6 (§6.2.2) and I use ternary branching trees.

I assume that morphological nodes are not linearly (horizontally) organized amongst themselves, i.e., morphemes don't precede other morphemes (cf. Sproat 1985; Embick and Noyer 2001). Any apparent linear ordering among morphemes or morphs is because of the linear ordering of their segments. I also do not assume that affixes are labeled with the property of being a prefix or suffix. That property can be derived from examining the linear order between the segments of the affix and the segments of the affix's morphological sister in the tree. For this thesis, there is no gain in computational expressivity if we assume that morphemes are linearly ordered or are labeled as a prefix vs. suffix. Potential problems would come from zero morphs and deeply-embedded morphological nodes (Chapter 7:§7.7.3).

A MNode of label x is said to recursively dominate another MNode of the same label x, e.g., an MStem (MWord) recursively dominates another MStem (MWord). If the two MNodes have different labels, then they are not in recursive dominance, e.g., an MStem (MWord) non-recursively dominates an MRoot (MStem). This distinction plays a role in prosodic parsing (Chapter 5:§5.3.1).

As with segment successor and long-distance precedence (§4.4.1), we need MSO logic to determine if some MNode x non-immediately dominates some other node y. The predicate **closed:MDom**(X) defines a set X of nodes which form an unbroken chain of morphological dominance from some point x in the tree, by traversing all branches from x, all the way down to one or more leaves in the tree. Long-distance or general morphological dominance is then defined in **gen_MDom**(x, y) using this transitive closure. The utility of this type of long-distance computation is needed in post-cyclic prosodic parsing (Chapter 8:§8.3.2), but is generally avoided in cyclic prosody (Chapter 6:§6.5).

(230) *MSO user-defined predicates for long-distance morphological dominance*
- *Defining transitive closure of morphological dominance*
  closed:MDom$(X) \stackrel{\text{def}}{=} \forall x, y[(x \in X \land \mathsf{MDom}(x, y)) \rightarrow y \in X]$
- *Defining long-distance or general morphological dominance*
  gen_MDom$(x, y) \stackrel{\text{def}}{=} \forall X[(x \in X \land \textbf{closed:MDom}(X) \rightarrow y \in X] \land x \neq y$

This concludes the basic labels and relations needed for morphology. Later in Chapters 5-3, I use logical transductions to create larger morphological trees, similar to word-formation rules (cf. Aronoff 1976).

4.5.2 Representation of prosodic structure

As for the prosody of the word 'apple' [æpəl], the segments form syllables, which form feet, which form a prosodic word: $æ < σ < F < PWord$. Explicitly, this prosodic organization is a data structure where nodes of the same type are (horizontally) ordered by successor, while nodes of different types are (vertically) ordered by **prosodic dominance** ($\sim pdom$ or p).

(231)   a.
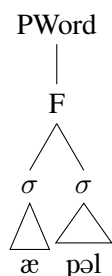
b.
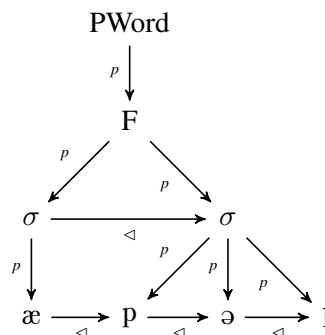
English prosody uses all unary labels in (232) except for **prosodic stems** (PStems). Armenian prosody uses PStems but not feet.[7] I use the unary label $\text{syll}(x)$ for syllables but I designate it with the label σ in trees. All these labels correspond to possible **prosodic nodes** (PNodes) in the language

(232)   *Basic unary labels for prosodic structure*
- $\text{syll}(x)$: x is a syllable
- $\text{foot}(x)$: x is a foot
- $\text{PStem}(x)$: x is a prosodic stem
- $\text{PWord}(x)$: x is a prosodic word
- $\text{PNode}(x)$: x is a prosodic node (syllable, foot, PStem, PWord)

Within a syllable, a segment can have different structural roles: onset, nucleus, coda. For syllable margins, Armenian syllables are generally at most CVCC: complex onsets are banned and complex codas must have falling sonority. Consider the words [sur] and [surp] (233). A coda is called an *inner coda* when it is either a simplex coda as in [su<u>r</u>] or the first member of a complex coda as in [su<u>r</u>p]. A coda is an *external coda* if it is the second member of a complex coda: [sur<u>p</u>].
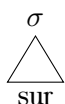
(233)   a.   i.   sur   'sharp'   ii.   surp   'holy'
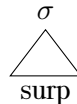
   b.   i.
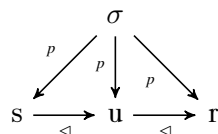
   ii.
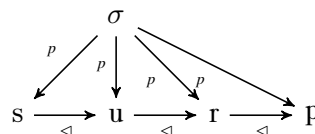
In order to formalize the structural roles within a syllable,[8] I refine the relation of *prosodic dominance* (PDom) into different subtypes for onset, nucleus, inner coda (*coda1*), and outer coda (*coda2*):

---

[7]See Özçelik (2017) on arguments for how feet are likely not present in every language. See Chapter 2:§2.2 on the lack of evidence for feet in Armenian.

[8]This is one of many equivalent formalizations (cf. Strother-Garcia 2019).

(234)   *Types of prosodic dominance from syllable to segment*
    a. PDom:syll_ons(x, y): a syllable x dominates a segment y as an onset
    b. PDom:syll_nuc(x, y): a syllable x dominates a segment y as a nucleus
    c. PDom:syll_coda1(x, y): a syllable x dominates a segment y as an inner coda
    d. PDom:syll_coda2(x, y): a syllable x dominates a segment y as an outer coda

A user-defined predicate is defined for the prosodic dominance between a syllable and a segment, or between a syllable and a coda.

(235)   *User-defined predicates for syllable-to-segment prosodic dominance*
    • **PDom:syll_seg**$(x, y) \stackrel{\text{def}}{=}$ PDom:syll_ons$(x, y) \lor$ PDom:syll_nuc$(x, y) \lor$
                         PDom:syll_coda1$(x, y) \lor$ PDom:syll_coda2(x, y)
    • **PDom:syll_coda**$(x, y) \stackrel{\text{def}}{=}$ PDom:syll_coda1$(x, y) \lor$ PDom:syll_coda2(x, y)

Throughout this thesis, I will not visualize PDom edges with the relevant structural roles (234). I use a generic label *p* as in (233); context makes the meaning of the label clear. I do this because explicitly showing these PDom types is hard to read, e.g., (236).
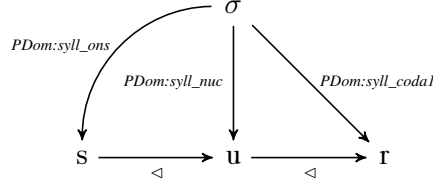
(236) a. i.
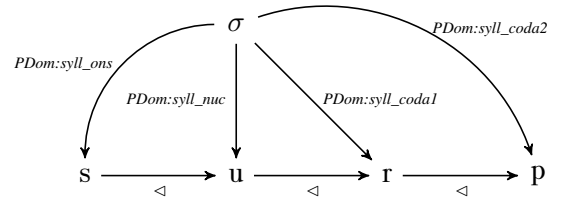
Going up higher in the word, Armenian lacks feet. Instead, syllables are organized into morphologically-derived prosodic constituents: **prosodic stems** (PStems) and **prosodic words** (PWords). A free-standing root like *amusin* (237a) forms its own PStem and PWord. When inflected with a C-initial inflectional suffix like *-ner* in *amusin-ner* (237b), the inflectional suffix is outside the PStem but within the PWord.

(237) a. amusín   'husband'
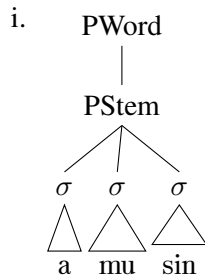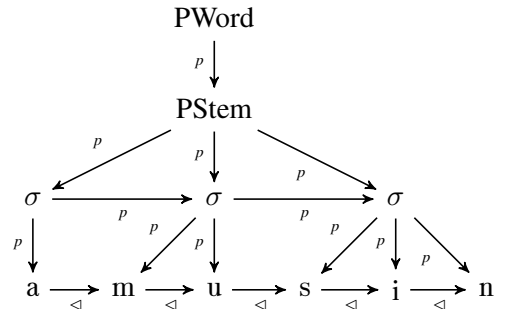
b. amusin-ner   'husband-PL'

i.

ii.
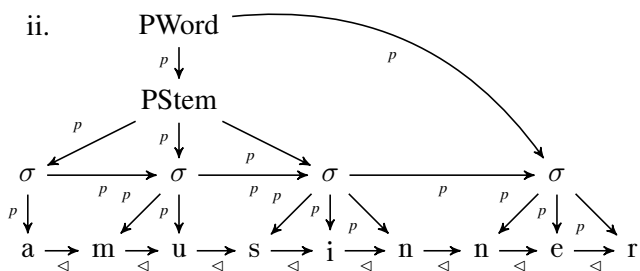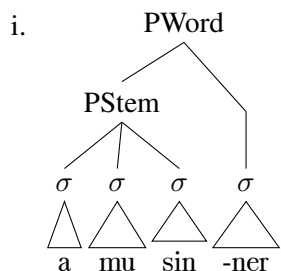
Syllables are linearly ordered with a successor relation that is reserved for syllables: $\mathsf{succ{:}syll}(x,y)$. I assume that PStems and PWords can succeed other constituents. For now, I only define a successor relation among PStems because it will be useful for compounds: $\mathsf{succ{:}PStem}(x,y)$.

(238) *Binary relations for immediate successor for prosodic structure*
   - $\mathsf{succ{:}syll}(x,y)$: x,y are syllables and x precedes y
   - $\mathsf{succ{:}PStem}(x,y)$: x,y are PStems and x precedes y

Sometimes, it seems that a constituent of level x precedes a lower constituent of level x-1, e.g., the foot in *(hap.py)-ly* precedes a syllable. Similarly in Armenian *amusin-ner*, the PStem precedes a syllable in *(amusin)$_s$-ner*. I do not formalize the adjacency of feet with syllables (or of PStems with syllables) as a special type of immediate successor. The precedence between prosodic constituents of different levels can be determined by examining the successor relations between their syllables, i.e., we know if a foot or PStem precedes some syllable y if the final syllable x of the foot/PStem precedes y.

As with syllable-to-segment prosodic dominance, we are explicit about the type of prosodic dominance among higher prosodic constituents. I use the additional subtypes of prosodic dominance in (239a), which are generalized in (239b). Further subtypes can be introduced to capture appendixes and phrasal phonology.

(239) a. *Types of prosodic dominance from higher prosodic constituents*
      i. $\mathsf{PDom{:}PStem\_syll}(x,y)$: a PStem x dominates a syllable y
      ii. $\mathsf{PDom{:}PWord\_PStem}(x,y)$: a PWord x dominates a PStem y
      iii. $\mathsf{PDom{:}PWord\_syll}(x,y)$: a PWord x dominates a syllable y
   b. *User-defined predicate for general prosodic dominance*
      - $\mathbf{PDom}(x,y) \overset{\text{def}}{=} \mathbf{PDom{:}syll\_seg}(x,y) \lor \mathsf{PDom{:}PStem\_syll}(x,y) \lor$
 $\mathsf{PDom{:}PWord\_PStem}(x,y) \lor \mathsf{PDom{:}PWord\_syll}(x,y)$

Some theories of prosodic phonology argue that prosodic dominance can be recursive, i.e., a PWord can dominate another PWord (Selkirk 1996; Peperkamp 1997). Such relations are formalized below. I generally do not use recursive prosody, yet it is still logically definable.

(240)   *Types of prosodic dominance for prosodic recursion*

  a. PDom:PStem_PStem(x, y): a PStem x dominates a PStem y

  b. PDom:PWord_PWord(x, y): a PWord x dominates a PWord y

The morphology and prosody act as a set of two trees which meet at the segments. Above syllables and feet, higher prosodic constituents like the PStem or PWord are associated with (= mapped, parsed from) MStems and MWords. I formalize this mapping as a **Match** relation (cf. Selkirk 2011), visualized as a dashed line. This is a form of correspondence between morphological and prosodic constituents. I use an English example for PWord matching and an Armenian example for PStem matching. I assume English MStems map to PWords.

(241)   a.   i.
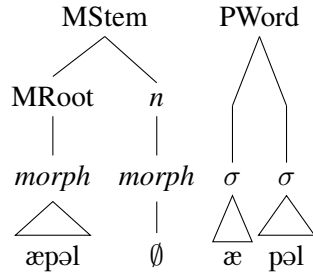
   ii.
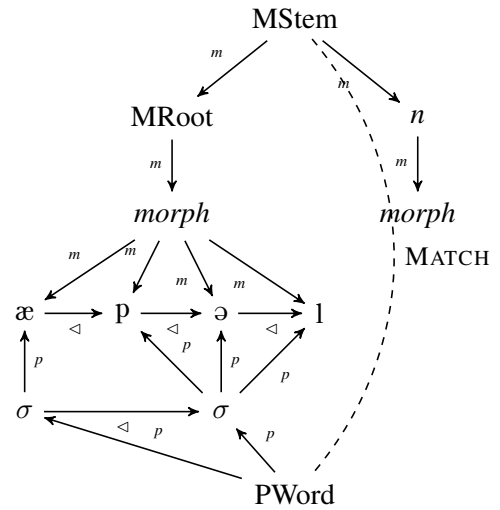
  b.   i.   amusin   'husband'
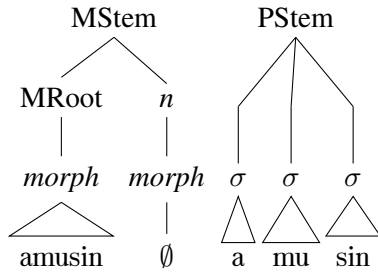
   ii.
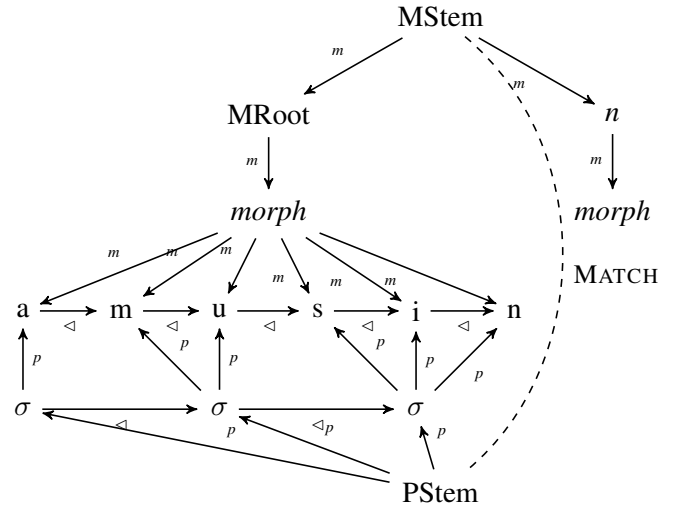
Possible MATCH relations are listed below. I focus only on Armenian where MStems map to PStems, not to PWords. Among morphological nodes, I assume that the only parsable MNodes are MStems and MWords.

(242)   *Binary relations for prosodic mapping or* Matching
- Match:stem(x, y): the MStem x is matched with (mapped to) the PStem y
- Match:word(x, y): the MWord x is matched with (mapped to) the PWord y

A more specialized type of prosodic mapping between morphological and prosodic constituents is *wrapping* (Truckenbrodt 1995, 1999). Assume you have two MNodes of the same label: MNode1 and MNode2. Morphologically, let MNode2 dominate MNode1 (243b-i). Prosodically, let MNode2 get matched with its own PNode2. MNode1 is either matched with its own unique PNode1 (243b-ii), or it is not matched with any PNode (243b-iii). In the latter case, MNode1 is said to be *wrapped* into the PNode2 of its dominating MNode2. Prosodic wrapping is illustrated in Chapter 6 where I formalize the prosody of derivatives (§6.5.1).

(243)   a. *Binary relations for prosodic wrapping*
- Wrap:stem(x, y): the MStem x is wrapped into PStem y
- Wrap:word(x, y): the MWord x is wrapped into PWord y

b.   i. *Input morphology*   ii. *MNode1 matches with PNode1*   iii. *MNode1 wrapped into PNode2*
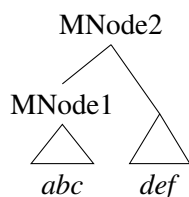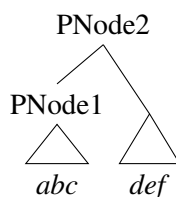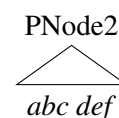
Match and **Wrap** are inspired from Match theory (Selkirk 2011) and Wrap theory (Truckenbrodt 1995, 1999). I also posit a new specialized type of prosodic association called **subsuming**. Prosodic subsumption is utilized in compounding (Chapter 6:§6.5.3). In certain compounds, the two individual stems *MStem$_L$* and *MStem$_R$* are matched with their own PStems. But the entire compound itself is its own compound MStem or *MStem$_C$*. This larger MStem$_C$ is not matched with any of the component PStems. Instead, the MStem$_C$ is subsumed into the rightmost PStem (= the PStem of *MStem$_R$*). As explained in Chapter 6, prosodic subsumption is restricted to compound MStems in endocentric compounds. I adopt prosodic subsumption alongside prosodic matching and wrapping so that every MStem/MWord will correspond to some prosodic constituent.

(244)   a. *Binary relations for prosodic subsumption in compounds*
- Subsume:stem(x, y): the MStem x is subsumed into PStem y

b. *Illustrating prosodic subsumption*
          Morphology   *Prosody* | *Associations*
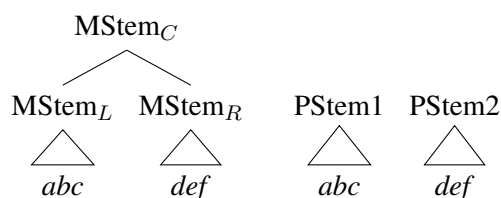
Associations

$MStem_L$ *matches with PStem1*
$MStem_R$ *matches with PStem2*
$MStem_C$ *subsumed into PStem2*
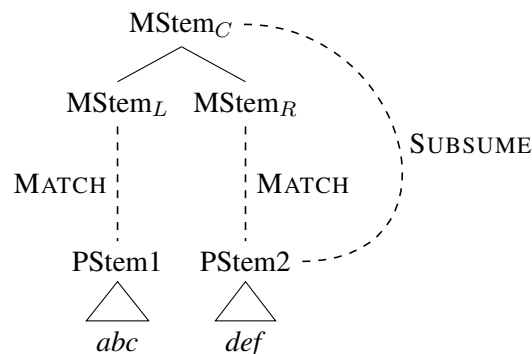
A MNode is said to be parsed if there exists some PNode such that the two are in some prosodic association relation, such as matching, wrapping, or subsuming. An MNode is unparsed if it has no associated PNode.

### 4.5.3   Local computations over hierarchical structures

Earlier in this chapter, the input and output representations were a simple string of segments. With that linear representation, it was easy to determine if a process is computationally local or not. The process is local if it can be computed without quantifiers, potentially by turning binary relations like immediate successor into unary functions. The previous two sections introduced formal definitions for hierarchical structure in the morphology and prosody. I use the same method to determine locality over hierarchical representations. Recall from §4.4.2, that not all logically-possible binary relations can be converted to unary functions. I illustrate this here. Over trees, some binary relations do not encode two unary functions. Briefly, a relation can be turned into a unary function if finding the relevant nodes is deterministic by involving a bounded number of mothers or daughters.

#### 4.5.3.1   Problems in local computations over hierarchical structure

When working with hierarchical inputs, non-local computations arise from n-ary branching trees where we cannot deterministically find the relevant daughter of some mother node. All daughter-to-mother dependencies can be turned into a unary function because every daughter has at most one mother. But, not all types of mother-to-daughter dependencies can be turned into a unary function. To illustrate, consider the morph 'apple' [æpəl] which dominates 4 segments.

(245)   *Morphological structure of the morph 'apple'* [æpəl]
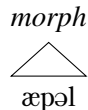
    a.   b.
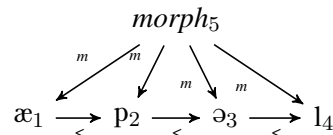
For the morph *apple*, the relation $MDom(x, y)$ is true for the following pairs of nodes, i.e., it is true when x is the morph at index '5' and y is any of the segments at indexes '1'-'4'.

(246)   *Values for the binary relation* $MDom(x, y)$ *for the morph* 'apple' [æpəl]
- $MDom(x, y)$ is TRUE for $\{(5,1), (5,2), (5,3), (5,4)\}$

Over strings, the relation $succ{:}seg(x, y)$ was broken down to two unary functions $F_L{:}succ{:}seg(x)$, $F_R{:}succ{:}seg(y)$ where $_{L,R}$ tells us if we're searching from the left variable x vs. the right variable y. Analogously for trees, the task of the unary function is to start searching from the mother node x or the daughter node y. The relation $MDom(x, y)$ can be broken down into two relations: $F_M{:}MDom(x) = y$ finds the daughter node y given the mother $(_M)$ node x, while $F_D{:}MDom(y) = x$ looks for the mother node x given the daughter $(_D)$ node y. The relation $F_D{:}MDom(y) = x$ is a unary function but $F_M{:}MDom(x) = y$ is not; the latter is a one-to-many relation. In other words, for the relation $MDom(x, y)$, going from the daughter variable y to the mother variable x is predictable, but not from the mother x to daughter y. This is because every segment y is dominated by only one morphological node ($\sim$ has at most one mother), but the morph x dominates multiple segments ($\sim$ has many daughters).

(247)   *Breaking* $MDom(x, y)$ *into two relations or functions*
- a. $F_M{:}MDom(x)$ is defined as a one-to-many relation
  - $F_M{:}MDom(5) = \{1, 2, 3, 4\}$
- b. $F_D{:}MDom(x)$ is defined as a function
  - $F_D{:}MDom(1) = 5$
  - $F_D{:}MDom(2) = 5$
  - $F_D{:}MDom(3) = 5$
  - $F_D{:}MDom(4) = 5$

With this input and its converted relations, we can formulate both QF and non-QF processes. If a process uses segment-to-morph dependencies, then it is QF and local. An example of a QF process is a process like *"give a segment [p] the label +F if it is part of a morph"*. The output function is shown in (248a-ii); it can be adapted with unary functions instead of binary relations (248a-iii). We can deterministically find the morph node from segment *p*. In contrast, if a process uses morph-to-segment dependencies, then it is not QF or local. For example, a non-QF process is a process like *"give a morph the label +G if it dominates a [p] segment"*. We cannot deterministically find the segment node *p* from the morph node. A quantifier is needed (248b-ii).

(248)   a. *Computation over segment-to-morph dependencies*

        i. Prose: *Give a segment [p] the label +F if it is part of a morph*

       ii. FO output function: $\phi_{+\mathrm{F}}(x^1) \stackrel{\text{def}}{=} \mathsf{p}(x) \wedge \exists y[\mathsf{MDom}(y,x) \wedge \mathsf{morph}(y)]$

      iii. QF output function: $\phi_{+\mathrm{F}}(x^1) \stackrel{\text{def}}{=} \mathsf{p}(x) \wedge \mathsf{morph}(\mathsf{F_M}{:}\mathsf{MDom}(x))$

   b. *Computation over segment-to-morph dependencies*

        i. Prose: *Give a morph the label +G if it dominates a [p] segment*

       ii. FO output function: $\phi_{+\mathrm{G}}(x^1) \stackrel{\text{def}}{=} \mathsf{morph}(x) \wedge \exists y[\mathsf{MDom}(x,y) \wedge \mathsf{p}(y)]$

Given this problem in determining locality over hierarchical structure, the next two sections sketch out how we can set up unary functions for some but not all types of morpho-prosodic dominance.

4.5.3.2   Local computations over morphological structure

When computed over either morphological or prosodic structure, finding the mother of some node is QF because a node can at have at most one mother. Finding the daughters however is not-QF unless we a priori have limitations on how many daughters some node can have. Certain aspects of morpho-prosodic structure does assume such limitations.

Consider first morphological structure. There is no bound on how many segments can be dominated by a morph. Any process which references morph-to-segment morphological dominance cannot be QF. But besides morphs, other types of MNodes have bounds on how many daughters they can have. A morpheme can dominate only one morph. Affixed constituents like MStems and MWord can dominate at most two MNodes (binary branching). Compounds can be ternary branching because of linking vowels.

Exploiting the above limitations on mother-to-daughter dependencies requires refining the types of MDom relations. In this thesis, I use a generic binary relation $\mathsf{MDom}(x,y)$ which is true when x is a morph, morpheme, or higher. To exploit how some morphological dominances are QF-definable, the relation $\mathsf{MDom}(x,y)$ must be refined to the following types of atomic binary relations which must be in the input:

(249)   *Refined types of morphological dominance*

     a. $\mathsf{MDom{:}nary}(x,y) \stackrel{\text{def}}{=} \mathsf{MDom}(x,y) \wedge \mathsf{morph}(x) \wedge \mathsf{segment}(y)$

     b. $\mathsf{MDom{:}first}(x,y) \stackrel{\text{def}}{=} \mathsf{MDom}(x,y) \wedge \neg\mathsf{morph}(x) \wedge$ "*y* is the first daughter"

     c. $\mathsf{MDom{:}second}(x,y) \stackrel{\text{def}}{=} \mathsf{MDom}(x,y) \wedge \neg\mathsf{morph}(x) \wedge$ "*y* is the second daughter"

     d. $\mathsf{MDom{:}third}(x,y) \stackrel{\text{def}}{=} \mathsf{MDom}(x,y) \wedge \neg\mathsf{morph}(x) \wedge$ "*y* is the third daughter"

The relation $\mathsf{MDom{:}nary}(x,y)$ is defined when there is no bound on how many daughters some node x can have, i.e., when x is a morph and y is a segment. In contrast, the other three relations are defined when there is such a bound. The bound is 1 when the mother node is a morpheme, 2 for an MStem or MWord, and 3 when the mother node is a compound.[9] The relations $\mathsf{MDom{:}first}(x,y)$, $\mathsf{MDom{:}second}(x,y)$, $\mathsf{MDom{:}third}(x,y)$ pick out the first, second, or third daughter if it exists.

---

[9]Compound nodes are formalized as $\mathsf{MStem{:}Comp}(x)$ in Chapter 6 (§6.2.2).

The binary relation $\mathsf{MDom{:}nary}(x, y)$ can be converted to a unary function to find the mother node x of some node y. The other binary relations can be converted to a pair of unary functions to find the mother or daughter nodes.

(250)   *Deriving unary functions from types of morphological dominance relations*
   a. *Morphological dominance with unbounded number of daughters*
      - From the relation $\mathsf{MDom{:}nary}(x, y)$:
        – $\mathsf{F_M{:}MDom{:}nary}(x)$ is not definable
        – $\mathsf{F_D{:}MDom{:}nary}(y)$: return the mother node x from the daughter node y
   b. *Morphological dominance with bounded number of daughters*
      - From the relation $\mathsf{MDom{:}first}(x, y)$ where y is the first daughter of x:
        – $\mathsf{F_M{:}MDom{:}first}(x)$: return the daughter node y from the mother node x
        – $\mathsf{F_D{:}MDom{:}first}(y)$: return the mother node x from the daughter node y
      - From the relation $\mathsf{MDom{:}second}(x, y)$ where y is the second daughter of x:
        – $\mathsf{F_M{:}MDom{:}second}(x)$: return the daughter node y from the mother node x
        – $\mathsf{F_D{:}MDom{:}second}(y)$: return the mother node x from the daughter node y
      - From the relation $\mathsf{MDom{:}third}(x, y)$ where y is the third daughter of x:
        – $\mathsf{F_M{:}MDom{:}third}(x)$: return the daughter node y from the mother node x
        – $\mathsf{F_D{:}MDom{:}third}(y)$: return the mother node x from the daughter node y

The above unary functions are defined for mutually exclusive domains. They are summarized into the two unary functions below.

(251)   *Unary functions for morphological dominance*
   a. $\mathsf{F_M{:}MDom}(x) \stackrel{\text{def}}{=} \mathsf{F_M{:}MDom{:}first}(x) \cup \mathsf{F_M{:}MDom{:}second}(x) \cup \mathsf{F_M{:}MDom{:}third}(x)$
   b. $\mathsf{F_D{:}MDom}(y) \stackrel{\text{def}}{=} \mathsf{F_D{:}MDom{:}nary}(y) \cup \mathsf{F_D{:}MDom{:}first}(y) \cup$
   $\mathsf{F_D{:}MDom{:}second}(y) \cup \mathsf{F_D{:}MDom{:}third}(y)$

With these definitions for these unary functions, the rest of the dissertation will utilize simple morphological dominance $\mathsf{MDom}(x, y)$. In order to show if some process is QF local or not, I replace the binary relation $\mathsf{MDom}(x, y)$ with the two unary functions $\mathsf{F_M{:}MDom}(x)$ and $\mathsf{F_D{:}MDom}(y)$ if possible.

4.5.3.3   Local computations over prosodic structure

Like morphological structure, some but not all hierarchical relations in prosody are convertible to two unary functions. As with the morphology, conversion depends on being able to deterministically find one node y from another node x. Likewise, some but not all prosodic association relations are convertible. Successor relations are convertible.

Given some daughter node, finding its mother node is local and deterministic via unary functions. But among mother-to-daughter prosodic dependencies, some are convertible to unary functions, and some are not. The conversion depends on if there is a bounded number of daughters. Briefly, syllable-to-segment and foot-to-syllable dependencies are QF, but all other prosodic mother-to-daughter dependencies are not QF. I

assume syllables dominate a *bounded* number of segments (for discusssion, see Strother-Garcia 2019). Thus syllable-to-segment dependencies are convertible to a unary function. The same applies to foot-to-syllable dependencies in a footed language like English. However, there is no bound on how many syllables or feet are dominated by a PStem or PWord. In compounds, there is no bound on how many PStems/PWords are dominated by a single PWord.

For this dissertation, I already use refined types of prosodic dominances for syllable-to-segment dependencies, e.g., PDom:syll_ons(x, y), etc. I do not use feet. I use refined types of PDom for PStem-to-syllable, PWord-to-PStem, PWord-to-syllable, PStem-to-PStem, and PWord-to-PWord dependencies.[10] Finding the mothers of these prosodic nodes is definable with a unary function. Finding the daughters of syllables is likewise definable. All other mother-to-daughter dependencies are not definable with a unary function.

(252) *Deriving unary functions from types of prosodic dominance relations*
   a. From the relation PDom:syll_ons(x, y):
       - F_M:PDom:syll_ons(x): return the onset y of syllable x
       - F_D:PDom:syll_ons(y): return the syllable x of onset y
   b. From the relation PDom:syll_nuc(x, y):
       - F_M:PDom:syll_nuc(x): return the nucleus y of syllable x
       - F_D:PDom:syll_nuc(y): return the syllable x of nucleus y
   c. From the relation PDom:syll_coda1(x, y):
       - F_M:PDom:syll_coda1(x): return the inner coda y of syllable x
       - F_D:PDom:syll_coda1(y): return the syllable x of inner coda y
   d. From the relation PDom:syll_coda2(x, y),:
       - F_M:PDom:syll_coda2(x): return the outer coda y of syllable x
       - F_D:PDom:syll_coda2(y): return the syllable x of outer coda y
   e. From the relation PDom:PStem_syll(x, y):
       - F_M:PDom:PStem_syll(x) is not definable
       - F_D:PDom:PStem_syll(y): return the PStem x of syllable y
   f. From the relation PDom:PWord_PStem(x, y):
       - F_M:PDom:PWord_PStem(x) is not definable
       - F_D:PDom:PWord_PStem(y): return the PWord x of PStem y
   g. From the relation PDom:PWord_syll(x, y):
       - F_M:PDom:PWord_syll(x) is not definable
       - F_D:PDom:PWord_syll(y): return the PWord x of syllable y
   h. From the relation PDom:PStem_PStem(x, y):
       - F_M:PDom:PStem_PStem(x) is not definable
       - F_D:PDom:PStem_PStem(y): return the PStem x that dominates a PStem y
   i. From the relation PDom:PWord_PWord(x, y):
       - F_M:PDom:PWord_PWord(x) is not definable
       - F_D:PDom:PWord_PWord(y): return the PWord x that dominates PWord y

Based on the predicate F_D:PDom:syll_seg(x, y), I define a unary function which returns the syllable x of a segment y.

---

[10]The Armenian data does not use prosodic recursion: PWords (PStems) do not dominate other PWords (PStems).

(253) *QF user-defined predicates for segment-to-syllable prosodic dominance*
- $\mathsf{F_D{:}PDom{:}syll\_seg}(y) \overset{\text{def}}{=} \mathsf{F_D{:}PDom{:}syll\_ons}(y) = x \cup \mathsf{F_D{:}PDom{:}syll\_nuc}(y) = x\cup$
$\mathsf{F_D{:}PDom{:}syll\_coda1}(y) = x \cup \mathsf{F_D{:}PDom{:}syll\_coda2}(y) = x$

As for prosodic association relations, Match relations are convertible to two unary functions. A given MNode (the left variable) can only get matched with a single PNode (the right variable); likewise a given PNode can only get matched with a single MNode. Wrap relations and Subsume relations however are not convertible to two unary functions. An MNode can be wrapped or subsumed into a single PNode, however a given PNode can have multiple MNodes be wrapped or subsumed into it.

(254) *Deriving unary functions from types of prosodic association relations*
   a. From Match relations
      i. From the relation $\mathsf{Match{:}stem}(x, y)$:
         - $\mathsf{F_L{:}Match{:}stem}(x)$: return the PStem y which is matched with an MStem x
         - $\mathsf{F_R{:}Match{:}stem}(y)$: return the MStem x which is matched with an PStem y
      ii. From the relation $\mathsf{Match{:}word}(x, y)$:
         - $\mathsf{F_L{:}Match{:}word}(x)$: return the PWord y which is matched with an MWord x
         - $\mathsf{F_R{:}Match{:}word}(y)$: return the MWord x which is matched with an PWord y
   b. From Wrap relations
      i. From the relation $\mathsf{Wrap{:}stem}(x, y)$:
         - $\mathsf{F_L{:}Wrap{:}stem}(x)$: return the PStem y which is wrapped into by the MStem x
         - $\mathsf{F_R{:}Wrap{:}stem}(y)$ is not definable
      ii. From the relation $\mathsf{Wrap{:}word}(x, y)$:
         - $\mathsf{F_L{:}Wrap{:}word}(x)$: return the PWord y which is wrapped into by the MStem x
         - $\mathsf{F_R{:}Wrap{:}word}(y)$ is not definable
   c. From the Subsume relation $\mathsf{Subsume{:}stem}(x, y)$:
      - $\mathsf{F_L{:}Subsume{:}stem}(x)$: return the PStem y which is subsumed into by the MStem x
      - $\mathsf{F_R{:}Subsume{:}stem}(y)$ is not definable

Finally, the successor relations among syllables or PStems are convertible to two unary functions.

(255) *Deriving unary functions from types of immediate successor relations in prosodic structure*
   a. From the relation $\mathsf{succ{:}syll}(x, y)$:
      - $\mathsf{F_L{:}succ{:}syll}(x)$: return the syllable y which follows a syllable x
      - $\mathsf{F_R{:}succ{:}syll}(y)$: return the syllable x which precedes a syllable y
   b. From the relation $\mathsf{succ{:}PStem}(x, y)$:
      - $\mathsf{F_L{:}succ{:}PStem}(x)$: return the PStem y which follows a PStem x
      - $\mathsf{F_R{:}succ{:}PStem}(y)$: return the PStem x which precedes a PStem y

To summarize, finding the daughters of most prosodic constituents is not doable with a QF unary function. The main exception is finding the bounded number of daughters of a syllable. As for morphology-prosody correspondences, it depends on the type of association or binary relation. Given a morphological node x and

a prosodic node y, we can use unary functions to locally find out if x, y are in a prosodic match relationship. Given a morphological node x, we can use a unary function to locally find out if x is wrapped or subsumed into some prosodic node y. However, given prosodic node y, we can't use unary functions to locally find the morphological nodes which are wrapped or subsumed into y.

4.6   Conclusion

In this chapter, I introduced and defined the logical notation and representations that I use to formalize the morphology-phonology interface. After going over the basics of the notation, I introduced a hierarchy of logical transductions. At the bottom of the hierarchy are Quantifier Free (QF) transductions which compute local processes. I discussed how some binary relations can be replaced with unary functions in order to expand the set of possible local or QF processes. I then defined the hierarchical structure of morphology and prosody using the logical notation. I showed how we can determine if the computation over these hierarchical structures is local or not. In general, finding the mother nodes of a node is locally-compatible, while finding the daughter nodes of a node might not be depending on if there is any bound on tree branching. This means that some of our binary relations are functional in both directions, while some are functional in one direction but not another. The next chapter will use the notational system to logically define the morphology-phonology interface, the generation of morphological and prosodic structure, and the formulation of phonological rule domains. A key finding is that the bulk of this interface is computationally local and can be computed with QF transductions.

4.A   Word signature for the formalization

Recall from Chapter 4: §4.2, that a word signature is of the template $\langle D, L, R \rangle$ where D is the domain (set of domain elements), L is the set of unary labels, and R is the set of binary relations. In the dissertation, I used a single word signature which I first defined in §4.2, and to which I then subsequently added new elements. Specifically, §4.5 added morphological and prosodic labels/relations. In Chapter 5, I added a constant SETTINGS to the domain. In Chapter 9, I added labels and relations for Operation Nodes. This appendix compiles all of these unary labels and binary relations. This can be skipped on the first reading.

4.A.1   Domain D

The domain D consists of a set of elements $\{1, 2, ..., n\}$ where the n elements can represent segments, morphological nodes, or prosodic nodes. The domain likewise has two constants: NULL and the SETTINGS.

The constant NULL was introduced in §4.4.2. Its role is to act as a 'sink state' when a unary function $F(x)$ is undefined for some input x.

The constant SETTINGS encodes certain global information about the morphological structure in a given cycle. It was introduced in Chapter 5 in §5.3, utilized in §5.4.3 and §5.5, and further explained in §5.6. It was further utilized in generating morphologically complex words in Chapter §6: §6.3-6.6. In Chapter §9, the SETTINGS was used to connect the derivation with its derivational history.

4.A.2   Unary labels L

The set L of unary labels encodes relevant properties of individual nodes. This information can be segmental, morphological, prosodic, or derivational.

4.A.2.1   Segmental labels

The set of segmental unary labels are a set of labels for segments $\{a, b, c, d, \ldots, z\}$ and segmental features $\{vowel, consonant, \ldots\}$. These were introduced in Chapter 4: §4.2 and used throughout the dissertation. Sometimes for clarity, I used user-defined predicates to encode features instead of unary labels, e.g. **vowel**(x). Chapter 5: §5.5 introduced labels for stress and destressing.

- stressed:vowel(x): x is a stressed vowel
- stressed:syll(x): x is a stressed syllable
- destressed:vowel(x): x is a destressed vowel
- destressed:syll(x): x is a destressed syllable

4.A.2.2   Morphological labels

The set of morphological unary labels are a set of labels for morphological nodes. The basic set of labels was first introduced in Chapter 4: §4.5.1.

- MNode(x): x is a morphological node (any morph, morpheme, MStem, or MWord)
- MConc(x): x is a morphological constituent (any MStem or MWord)
- morpheme(x): x is a morpheme (any root or affix)
- morph(x): x is a morph
- affix(x): x is an an affix
- der(x): x is an a derivational affix
- infl(x): x is an an inflectional affix
- MRoot(x): x is a morphological root
- MStem(x): x is a morphological stem
- MWord(x): x is a morphological word

Additional morphological features were likewise introduced in Chapter 4: §4.5.1, and were used especially in Chapter 7.

- past(x), plural(x), gerund(x), inst(x), def(x), infinitival(x), irregular(x)...
- noun(x), verb(x), adj(x)
- Class:E(x), Class:A(x), Class:I(x), ...

The set of labels for compounding was introduced in Chapter 6: §6.2.2.2.

- MConc(x): a morphological node x is the morphological concatenation of stems and a linking vowel
- LV_morpheme(x): a morpheme x is the linking vowel used in compounding
- MStem:Comp:Left(x): the MStem x is the left stem or MStem$_L$ of a compound
- MStem:Comp:Right(x): the MStem x is the right stem or MStem$_R$ of a compound
- MStem:Comp(x): the MStem x is a compound
- MStem:Comp:Endo(x): the MStem x is a hyponymic or endocentric compound
- MStem:Comp:Exo(x): the MStem x is a non-hyponymic or exocentric compound

4.A.2.3   Prosodic labels

The basic set of unary labels for prosodic structure was introduced in Chapter 4: §4.5.2.

- syll(x): x is a syllable
- foot(x): x is a foot
- PStem(x): x is a prosodic stem
- PWord(x): x is a prosodic word
- PNode(x): x is a prosodic node (syllable, foot, PStem, PWord)

4.A.2.4   Derivational labels

These labels were used to direct the derivation by specifying what to prosodically parse, what rules to apply, what dialect to use, and what morphological operations to undertake.

Parse lables encode what should be prosodically parsed. They are determined by examining the morphological and prosodic context of the morphologically topmost node. They were introduced in Chapter 5: §5.3.1.

- Parse:MStem:nonrecursive(SETTINGS):

  is TRUE for SETTINGS iff we need to parse a non-recursive MStem into a PStem

  Introduced in Chapter 5: §5.3.1, used in §5.4.3

- Parse:MStem:recursive(SETTINGS):

  is TRUE for SETTINGS iff we need to parse an unparsed MStem which dominates a matched (parsed) MStem

  Introduced in Chapter 6: §6.3.2.1, used in §6.5.1

- Parse:MWord:nonrecursive(SETTINGS):

  is TRUE for SETTINGS iff we need to parse a non-recursive MWord into a PStem

  Introduced in Chapter 6: §6.3.2.2, used in §6.5.2

- Parse:MStem:Comp:Endo(SETTINGS):

  is TRUE for SETTINGS iff we need to parse an endocentric compound

  Introduced in Chapter 6: §6.3.2.3, used in §6.5.3.4

- Parse:MStem:Comp:Exo(SETTINGS):

  is TRUE for SETTINGS iff we need to parse an exocentric compound

  Introduced in Chapter 6: §6.3.2.3, used in §6.5.3.5

Cophon labels are properties of certain constituents like MStems, MWords, PStems. These labels encode the association of the constituent with a cophonology. These were introduced in Chapter §5: §5.3.2. The PStem cophonology is redundant because all PStems have this label (cf. Chapter 6: §6.6.2).

- Cophon:SLevel(x): x triggers the stem-level cophonology
- Cophon:WLevel(x): x triggers the word-level cophonology
- Cophon:PStem(x): x triggers the PStem-level cophonology

Cophon labels are used to determine what cophonology to apply in a given cycle. This information on active rule domains is encoded into Domain labels. These are properties of the SETTINGS constant (Chapter §5: §5.3.2). They are used in triggering the stem-level (Chapter 5: §5.5, Chapter 6: §6.6.1), word-level (Chapter 6: §6.6.3.2), and PStem-level cophonologies (Chapter 6: §6.6.2,6.6.3.3).

- Domain:Cophon:SLevel(SETTINGS): the SETTINGS has the domain of the SLevel cophonology
- Domain:Cophon:WLevel(SETTINGS): the SETTINGS has the domain of the WLevel cophonology
- Domain:Cophon:PStem(SETTINGS): the SETTINGS has the domain of the PStem cophonology

Because Armenian phonological processes vary by dialect, Chapter 6: §6.6.3.1 used labels for selecting the right dialect. These labels were on the SETTINGS constant.

- Western(SETTINGS) or Eastern(SETTINGS)

In Chapter 9: §9.3.1, Operation Nodes were introduced which encode the input's derivational history and its future morphological operations. These nodes utilize their own unary labels.

- Oper(x): the node x is an operation
- Oper:Root(x): we must generate the input
- Oper:n_zero(x): we must generate the covert nominalizer
- Oper:Def(x): we must generate the definite suffix -ə,-n
- Oper:CL_is(x): we must generate the clitic =e 'is'

4.A.3   Binary relations R

Binary relations encode the relationships between two nodes or elements in the domains. I defined binary relations based on segmental, morphological, prosodic, and derivational relationships.

4.A.3.1   Segmental relations

Segments are ordered in terms of the binary relation of immediate successor succ:seg(x, y).

4.A.3.2   Prosodic relations

The basic set of prosodic binary relations was introduced in Chapter 4: §4.5.2. These relations concerned the successor relationship among prosodic nodes, their dominance relations, and their association with morphological nodes.

I used the following successor relations for prosodic nodes.

- succ:syll(x, y): x, y are syllables and x precedes y
- succ:PStem(x, y): x, y are PStems and x precedes y

Prosodic nodes form a tree-like hierarchy based on the binary relation of prosodic dominance, which is divided into different types based on the identity of the mother and daughter nodes.

- PDom:syll_ons(x, y): a syllable x dominates a segment y as an onset
- PDom:syll_nuc(x, y): a syllable x dominates a segment y as a nucleus
- PDom:syll_coda1(x, y): a syllable x dominates a segment y as an inner coda
- PDom:syll_coda2(x, y): a syllable x dominates a segment y as an outer coda
- PDom:PStem_syll(x, y): a PStem x dominates a syllable y
- PDom:PWord_PStem(x, y): a PWord x dominates a PStem y
- PDom:PWord_syll(x, y): a PWord x dominates a syllable y
- PDom:PStem_PStem(x, y): a PStem x dominates a PStem y
- PDom:PWord_PWord(x, y): a PWord x dominates a PWord y

Prosodic nodes are associated with morphological nodes based on prosodic matching, prosodic wrapping, and prosodic subsumption. These were all introduced in Chapter 4.5.2 and used in different places. I didn't use Wrap:word(x, y) because I didn't formalize the prosody of words with multiple inflectional suffixes. This is nevertheless straightforward and analogous to the use of Wrap:stem(x, y).

- Match:stem(x, y): the MStem x is matched with (mapped to) the PStem y
            First used in Chapter 5: §5.4.3
- Match:word(x, y): the MWord x is matched with (mapped to) the PWord y
            First used in Chapter 6.3.2: §6.5.2.2
- Wrap:stem(x, y): the MStem x is wrapped into PStem y
            First used in Chapter 6.3.2: §6.5.1
- Wrap:word(x, y): the MWord x is wrapped into PWord y
            Never used

- Subsume:stem(x, y): the MStem x is subsumed into PStem y
First used in Chapter 6: §6.5.3.4

4.A.3.3 Morphological relations

Morphological nodes are primarily organized by morphological dominance $\mathsf{MDom}(x, y)$ which was introduced in Chapter 4: §4.5.1. In Chapter 4: §4.5.3.2, I decomposed this binary relation into four mutually exclusive types of morphological dominance based on the branching factor of the mother node.

- $\mathsf{MDom}(x, y)$: a morphological node x morphologically dominates some other morphological node or segment y
- $\mathsf{MDom:nary}(x, y) \stackrel{\text{def}}{=} \mathsf{MDom}(x, y) \wedge \mathsf{morph}(x) \wedge \mathsf{segment}(y)$
- $\mathsf{MDom:first}(x, y) \stackrel{\text{def}}{=} \mathsf{MDom}(x, y) \wedge \neg\mathsf{morph}(x) \wedge$ "y is the first daughter"
- $\mathsf{MDom:second}(x, y) \stackrel{\text{def}}{=} \mathsf{MDom}(x, y) \wedge \neg\mathsf{morph}(x) \wedge$ "y is the second daughter"
- $\mathsf{MDom:third}(x, y) \stackrel{\text{def}}{=} \mathsf{MDom}(x, y) \wedge \neg\mathsf{morph}(x) \wedge$ "y is the third daughter"

4.A.3.4 Derivational relations

In Chapter 9: §9.3.1, I introduced Operation Nodes. These participate in the following binary relations.

- $\mathsf{succ:Oper}(x, y)$: the operation x immediately precedes the operation y
- $\mathsf{operate\_at}(\textsc{settings}, y)$: the operation y is the current morphological operation that we must apply. It is linked with the $\textsc{settings}$ constant.

Chapter 5

Components of cyclic phonology

5.1 Architecture of morphophonology

The previous chapter established the basics of the notational system that I use. In this chapter, I apply the logical notation to formalize the basic aspects of an interactionist model for the morphology-phonology interface. The takeaway is that the interface is not only definable but also largely computationally local.

A cyclic interactionist model assumes that the derivation consists of three explicit components: the Morphology, Prosody, and Phonology. This chapter formalizes these three components. I likewise formalize an implicit step in the derivation which occurs between the Morphology and Prosody. This more explicit system is sketched out below.

(256)   *Sketch of an interactionist model with an explicit stage for the* SETTINGS
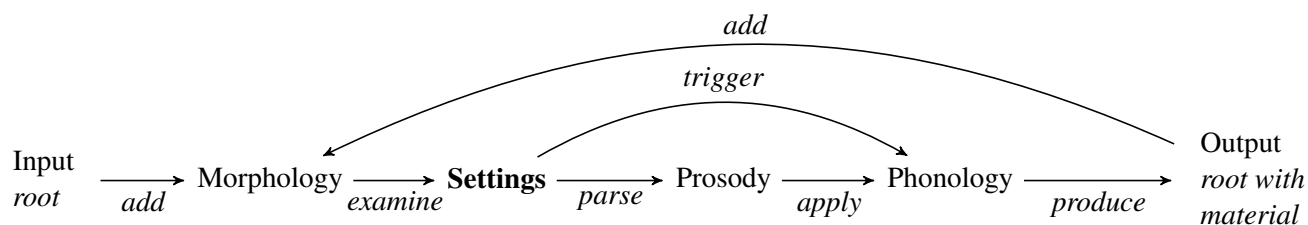
To illustrate, consider the Armenian simplex stem *amusin* 'husband' and its derivation table below. It undergoes two cycles, one stem-level for the covert nominalizer -∅ and one word-level for the covert nominative suffix -∅. First, given an input root $\sqrt{amusin}$, the **Morphology** adds or changes the input to create a larger morphological structure: i.e., add a covert category suffix -∅ to turn the root into a noun. The later Prosody and Phonology stages need to access certain information from the Morphology and derivation, i.e., that the input consists of a unparsed morphological stem. To get this information, we **examine** the settings of the derivation by examining the input's morphology. This information tells us to apply the right prosodic processes (parsing a stem) and the right phonological rule domains (stem-level stratum). With this information, the **Prosody** parses this structure into well-formed prosodic constituents with syllables and a prosodic stem *(a.mu.sin)$_s$*. Lastly, the **Phonology** is applied in the form of phonological rule domains.

Specifically, final stress is applied as part of the stem-level cophonology: *amusín*. These three components vacuously repeat in a second cycle to add covert inflection, form a prosodic word, and apply the word-level phonology.

(257)   *Derivation table for the stem* amusin *with explicit examination*

Input			/amusin -$\emptyset_S$ -$\emptyset_W$/
Cycle 1	MORPHO	Spell-out	/amusin -$\emptyset$/
	EXAMINE		*What should we parse and apply?*
	PROSODY	Syllabify	amu.sin
		Map PStem	(amu.sin)$_s$
	PHONO	*SLevel*	
		Stress	(amu.sín)$_s$
		DHR	
Cycle 2	MORPHO	Spell-out	(a.mu.sín)$_s$ - /-$\emptyset$/
	EXAMINE		*What should we parse and apply?*
	PROSODY	Syllabify	
		Map PWord	((a.mu.sín)$_s$)$_w$
	PHONO	*WLevel*	
		Stress	
Output			amusín

In this chapter, I formalize the three explicit components (Morphology, Prosody, Phonology). I show that they are computationally local. However, the implicit **Examination** or **Settings** step is *not* computationally local. Each of the four components is formalized as a *logical transduction* which feeds the next. Furthermore, for a single component, the logical transduction can be the composition of smaller transductions, e.g., the logical transduction for prosody consists of first syllabifying the input and then generating a PStem.

In §5.2, I first formalize the morphology and define how we can add a covert affix. In §5.3, I examine the settings of the derivation and encapsulate the relevant information into a constant called the SETTINGS. This constant is added into the domain D of our word signature from the previous chapter. With the SETTINGS in place, I then define syllabification and prosodic mapping in §5.4. I define cophonologies or rule domains in 5.5, and I use the SETTINGS to trigger final stress. In §5.6, I discuss the conceptual and computational significance of the SETTINGS constant and how it affects the locality of the interface. I finally conclude in §5.7.

Note that throughout this thesis, I *illustrate* the computation in an intuitively procedural or serial manner but this is only for illustration. Within a single component or logical transduction such as the Morphology, the output functions can apply in parallel or in any order as long as there is no circularity. I will use terms like 'generate' or 'create' often, but this is again just for illustration. Furthermore, within a single cycle, the four components of Morphology, Examination, Prosody, and Phonology are serially ordered. Each component is a logical transduction. Because of function composition, each component can be composed into a single transduction. And, an entire cycle can be composed into a single transduction. In Chapter §9, I further refine my interactionist model so that we can break down the recursive (potentially infinite) loop from the Phonology to the Morphology into a finite sequence of cycles.

5.2 Morphology: Adding a covert affix

When generating new morphological structure, there are three tasks involved. First, we **enlarge** the input by processing 'larger' chunks of morphological constructions. Second, we pick **allomorphs** for the morphemes in those chunks. Finally, we must **linearize** those chunks for the phonology. In this section, I demonstrate only the first task of 'enlarging' and some aspects of linearization. Allomorphy and linearization are further discussed in Chapter 6 where I illustrate the generation of *overt* affixes, and in Chapter 7 where I discuss the computation of allomorphy and linearization.

For ease of demonstration, I model morphology as a set of morphological processes, i.e., as a set of logical transductions which themselves are sets of output functions.[1] For example, adding the suffix *-agan* is a transduction, spelling out suppletive allomorphs for case is another transduction, and creating a compound is a separate transduction. This is similar to word-formation rules (cf. Aronoff 1976). Unifying all these possible morphological processes in the grammar requires some enrichment of the input (Chapter 9: §9.3).

For the simplex stem *amusin* 'husband', it consists of a free-standing root. It can form its own MStem by taking a covert category suffix *n*. Adding this zero suffix is an actual morphological process, shown below.

(258) *Desired output for creating an MStem over an MRoot in* amusin
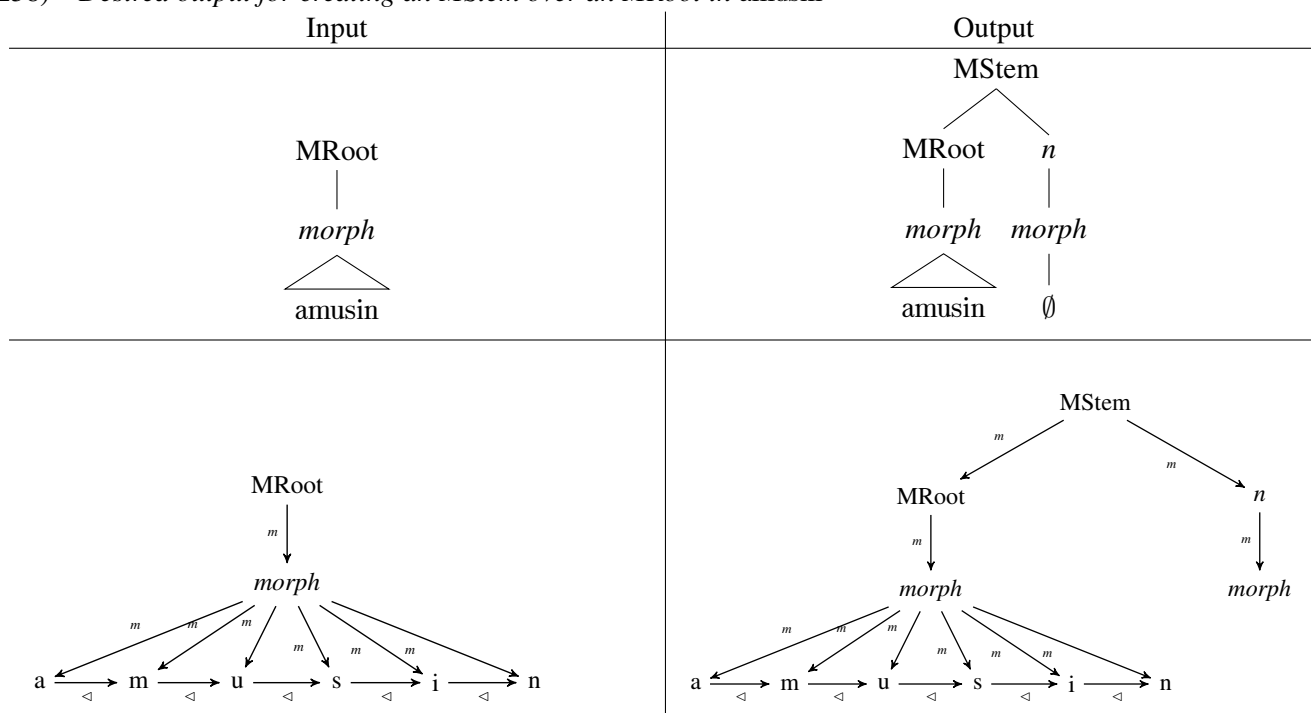
To generate this output, we define a morphological process which is *specific* to adding a zero *n*-suffix. We need to add <u>three</u> new morphological nodes (MNodes) for the morph, the morpheme *n*, and the MStem, each with their own index. No segments are added. Therefore, adding a zero *n*-suffix is a logical transduction that requires a copy set of size 4. In Copy 1, the input is faithfully outputted (259).

---

[1] I adopt an essentially item-and-process approach to morphological functions. However, this is only out of convenience. A fully item-and-arrangement approach is also feasible and in some ways computationally equivalent (Roark and Sproat 2007). I discuss the computability of these two approaches in Chapter 7: §7.2.

(259)   *QF output functions for vacuous identity in Copy 1*

- For every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$
- For every relation $\mathsf{rel} \in R$:
  $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

The new nodes for the morph, morpheme,[2] and MStem are generated in Copies 2-4. They are the output correspondents of the input's morphologically topmost node.[3] Via the user-defined predicate $\mathbf{MTopmost}(x)$ in (260a), some MNode x is the morphologically topmost node if it is an MNode and there is no other MNode y which dominates it. The predicate $\mathbf{MTompost}(x)$ is QF-definable. With $\mathsf{F_D}{:}\mathsf{MDom}(y)$, we can check that some MNode is topmost if it does not have a mother.

(260)   a. *FO user-defined predicate for finding the morphologically topmost MNode*
- $\mathbf{MTopmost}(x) \stackrel{\text{def}}{=} \mathsf{MNode}(x) \land \neg\exists y[\mathsf{MDom}(y, x)]$

b. *QF user-defined predicate for finding the morphologically topmost MNode*
- $\mathbf{MTopmost}(x) \stackrel{\text{def}}{=} \mathsf{MNode}(x) \land \mathsf{F_D}{:}\mathsf{MDom}(x) = \textsc{null}$

c. *QF output functions for creating new morphological nodes for the* n-suffix
- $\phi\mathtt{morph}(x^2) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{noun}(x^3) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{MStem}(x^4) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$

I leave implicit the fact the new nodes have various redundantly defined labels, e.g. if x is a morph then it is also an MNode.
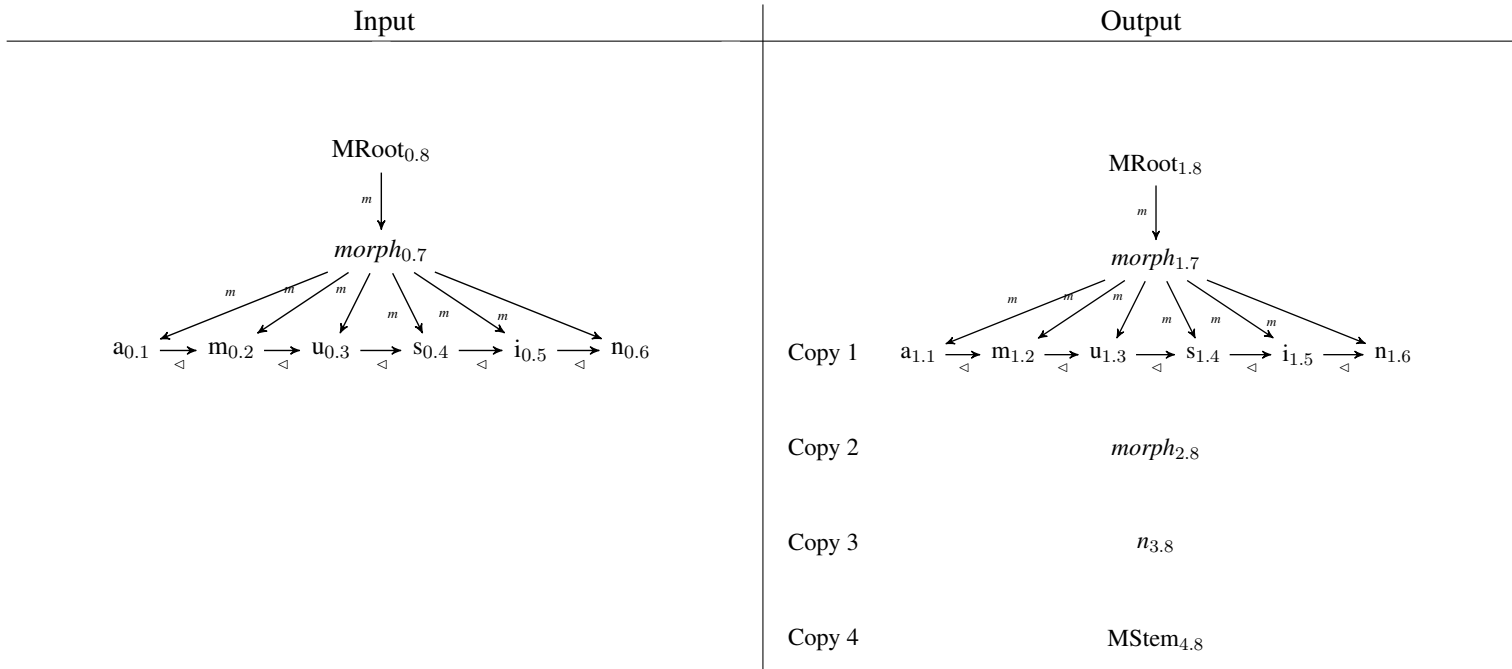
(261)   *Predictable labeling when creating new nodes*
- $\phi\mathtt{morph}(x^2) \rightarrow \phi\mathtt{MNode}(x^2)$
- $\phi\mathtt{noun}(x^3) \rightarrow \phi\mathtt{morpheme}(x^3)$
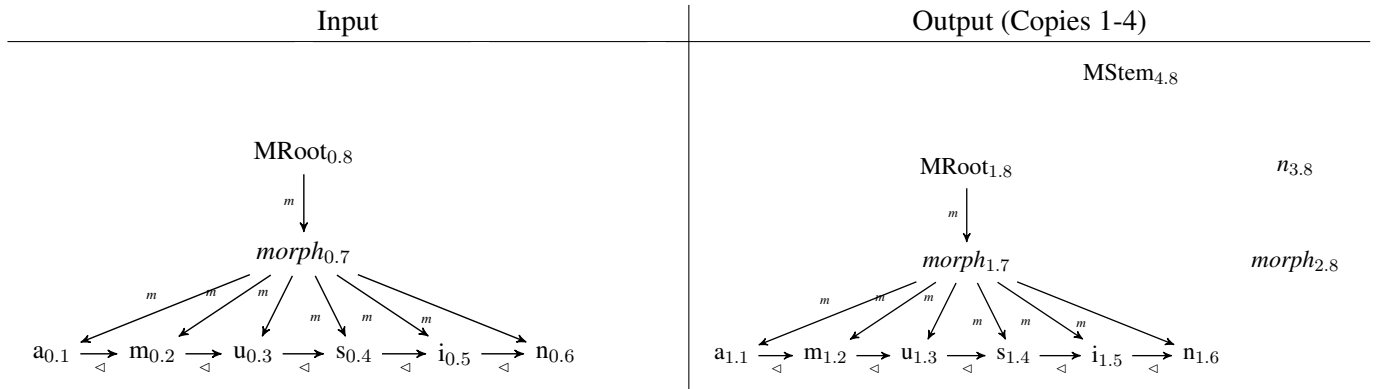 ...

I illustrate the output of the above functions below. The new nodes are all output correspondents for the input $\mathrm{MRoot}_{0.8}$ because it is the morphologically topmost node in the input.

---

[2]Note that the category feature has the label $\mathsf{noun}(x)$ but it is illustrated as *n* in the trees.

[3]New MNodes are defined in terms of the topmost node in order to ensure order-preservation. This is discussed in Chapter 7: §7.5.4.

(262) *Generating a zero suffix in* amusin – *generating morphology nodes*

Input	Output

Input:

$$\text{MRoot}_{0.8} \xrightarrow{m} morph_{0.7}$$

$$morph_{0.7} \text{ (with } m \text{ edges to) } a_{0.1} \xrightarrow{\triangleleft} m_{0.2} \xrightarrow{\triangleleft} u_{0.3} \xrightarrow{\triangleleft} s_{0.4} \xrightarrow{\triangleleft} i_{0.5} \xrightarrow{\triangleleft} n_{0.6}$$

Output:

$$\text{MRoot}_{1.8} \xrightarrow{m} morph_{1.7}$$

Copy 1: $a_{1.1} \xrightarrow{\triangleleft} m_{1.2} \xrightarrow{\triangleleft} u_{1.3} \xrightarrow{\triangleleft} s_{1.4} \xrightarrow{\triangleleft} i_{1.5} \xrightarrow{\triangleleft} n_{1.6}$

Copy 2: $morph_{2.8}$

Copy 3: $n_{3.8}$

Copy 4: $\text{MStem}_{4.8}$

The figure above is equivalent to the one below where I have shuffled nodes into a visually more appealing location.

(263) *Generating a zero suffix in* amusin – *rearranged copies*

Input	Output (Copies 1-4)

Input:

$$\text{MRoot}_{0.8} \xrightarrow{m} morph_{0.7}$$

$$a_{0.1} \xrightarrow{\triangleleft} m_{0.2} \xrightarrow{\triangleleft} u_{0.3} \xrightarrow{\triangleleft} s_{0.4} \xrightarrow{\triangleleft} i_{0.5} \xrightarrow{\triangleleft} n_{0.6}$$

Output (Copies 1-4):

$$\text{MStem}_{4.8}$$

$$\text{MRoot}_{1.8} \xrightarrow{m} morph_{1.7} \qquad n_{3.8}$$

$$a_{1.1} \xrightarrow{\triangleleft} m_{1.2} \xrightarrow{\triangleleft} u_{1.3} \xrightarrow{\triangleleft} s_{1.4} \xrightarrow{\triangleleft} i_{1.5} \xrightarrow{\triangleleft} n_{1.6} \qquad morph_{2.8}$$
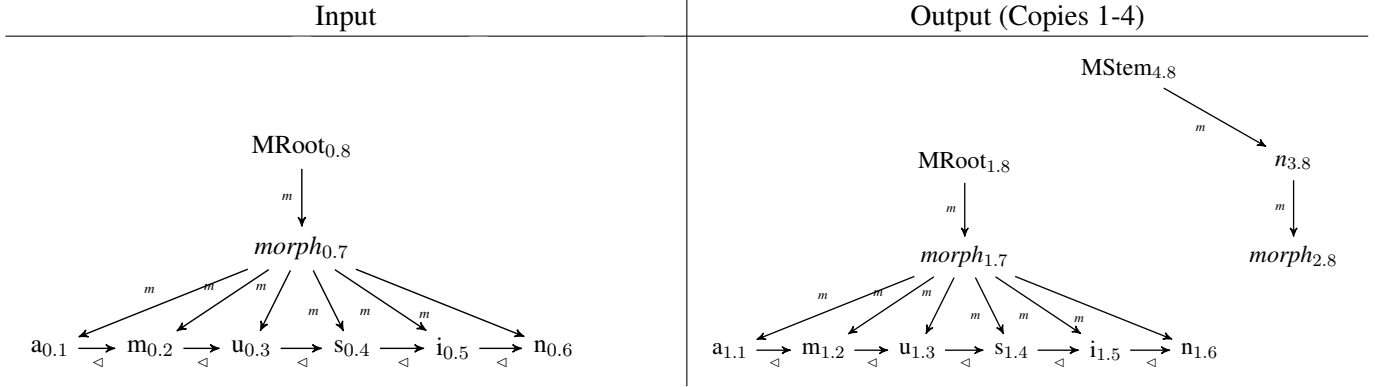
The new morphological structure must now be internally linearized or organized. Any new segments are internally linearized via immediate successor while morphological nodes are internally linearized via morphological dominance. The new suffix is a zero-suffix so we only internally linearize morphological nodes. This is done via the output functions below.

(264) *QF output functions for internally linearizing the zero-suffix* n *in* amusin

- $\phi\text{MD}\circ\text{m}(x^4, y^3) \stackrel{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$
- $\phi\text{MD}\circ\text{m}(x^3, y^2) \stackrel{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

This is shown below. For example, the new MStem is $MStem_{4.8}$, and the new morpheme n is $n_{3.8}$. Via the function $\phi\texttt{MDom}(x^4, y^3)$, the new MStem x^4 dominates the new morpheme n y^3. These are selected by this function because they are output correspondents of the morphologically topmost node in the input (the MRoot).

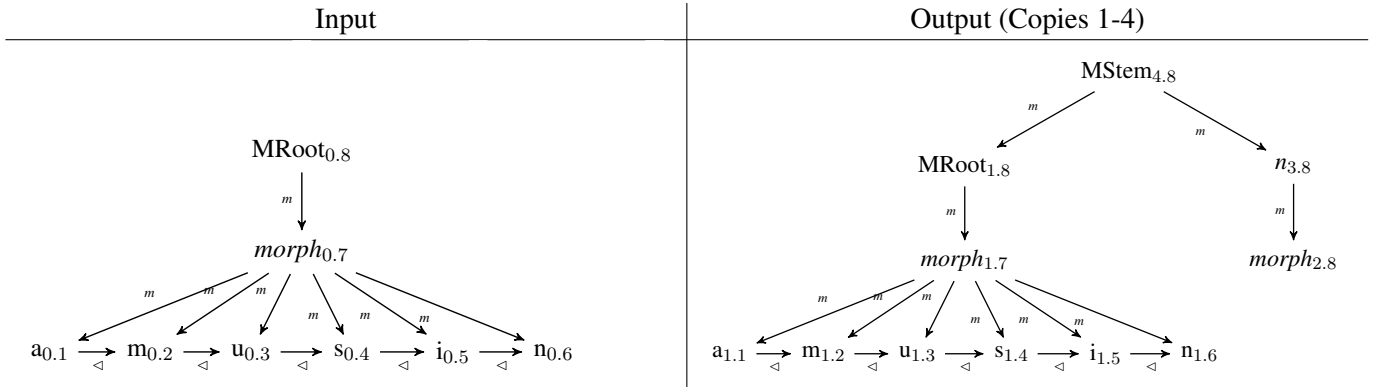(265)   *Generating a zero suffix in amusin – internal linearization*

Input	Output (Copies 1-4)

Input:

$MRoot_{0.8}$ —m→ $morph_{0.7}$ —m→ $a_{0.1}$ $\triangleleft$ $m_{0.2}$ $\triangleleft$ $u_{0.3}$ $\triangleleft$ $s_{0.4}$ $\triangleleft$ $i_{0.5}$ $\triangleleft$ $n_{0.6}$

Output (Copies 1-4):

$MStem_{4.8}$ —m→ $n_{3.8}$ —m→ $morph_{2.8}$

$MRoot_{1.8}$ —m→ $morph_{1.7}$ —m→ $a_{1.1}$ $\triangleleft$ $m_{1.2}$ $\triangleleft$ $u_{1.3}$ $\triangleleft$ $s_{1.4}$ $\triangleleft$ $i_{1.5}$ $\triangleleft$ $n_{1.6}$

The last step is to externally linearize the new morphology with the base. In the case of the zero suffix *n*, we only need to make the new MStem dominate the output MRoot via the output function below.

(266)   *QF output function for externally linearizing the zero-suffix* n *in* amusin

- $\phi\texttt{MDom}(x^8, y^1) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

This is illustrated below. The new $MStem_{4.8}$ x^4 dominates the output correspondent $MRoot_{1.8}$ y^1 because both are output correspondents of the morphologically topmost node, $MRoot_{0.8}$.

(267)   *Generating a zero suffix in amusin – external linearization*

Input	Output (Copies 1-4)

Input:

$MRoot_{0.8}$ —m→ $morph_{0.7}$ —m→ $a_{0.1}$ $\triangleleft$ $m_{0.2}$ $\triangleleft$ $u_{0.3}$ $\triangleleft$ $s_{0.4}$ $\triangleleft$ $i_{0.5}$ $\triangleleft$ $n_{0.6}$

Output (Copies 1-4):

$MStem_{4.8}$ —m→ $MRoot_{1.8}$ and —m→ $n_{3.8}$ —m→ $morph_{2.8}$

$MRoot_{1.8}$ —m→ $morph_{1.7}$ —m→ $a_{1.1}$ $\triangleleft$ $m_{1.2}$ $\triangleleft$ $u_{1.3}$ $\triangleleft$ $s_{1.4}$ $\triangleleft$ $i_{1.5}$ $\triangleleft$ $n_{1.6}$

The result is a complete new morphological item. All the output functions relied on only the predicate $\mathbf{MTopmost}(x)$ which is QF. Adding a covert affix is thus QF and computationally local. To define other possible morphological processes, the above functions must be slightly modified on a construction-by-construction basis, i.e., changing the segments or morphological labels, changing the direction of linearization, etc. These modifications are illustrated in Chapter 6: §6.2.1.

5.3   Implicitness in the derivation: Examining the morphology

Having created a new morphological item, the item must now be prosodically parsed and subject to phonological rules (a cophonology). However, in order to know what to parse and what rules to apply, we need to examine the input's morphophonological structure. The morphophonological structure is what **triggers** the prosodic parse and rule domains. Analyzing the structure is implicit in a traditional derivation. I make it into an explicit step in the table below. I only show the first cycle without covert inflection.

(268)   *Explicit derivation table for producing the first cycle of* amusín

			MS_n
			(tree: $\sqrt{}$ and n dominated by MS_n; $\sqrt{}$ over /amusin, n over -$\emptyset$)
Input			/amusin -$\emptyset_S$/
Cycle 1	MORPHO	Spell-out	/amusin -$\emptyset$/
	EXAMINE		*What should we parse and apply?*
	PROSODY	Syllabify	amu.sin
		Map PStem	(amu.sin)$_s$
	PHONO	*SLevel*	
		Stress	(amu.sín)$_s$
		DHR	
Output			amusín

Specifically, in order to parse or map some MStem into a PStem, we need to know 1) that the input contains an MStem, 2) if the input MStem has or has not been parsed before, and 3) what prosodic constituents already exist in the input. Similarly, in order to apply the stem-level cophonology, we need to know the topmost node in the morphological tree is an MStem. All of this information is implicitly calculated after applying the morphological processes. I call this step '**examining**' the settings of the derivation. This step will **determine** the *non-local morphological triggers* for the prosody and phonology.

In this section, I explain this intermediate step. I define a constant called the SETTINGS. This constant is added to the domain D of our word signature from the previous chapter. This constant **encapsulates** the above global information about the morphophonological derivation which will be used for mapping higher prosodic structure (§5.3.1) and for triggering phonological strata or rule domains (§5.3.2). Later in this chapter (§5.6), I discuss the ontological status of the SETTINGS, its utility, and its computational significance. Briefly, the SETTINGS encapsulates the non-local morphological information which triggers the (otherwise local) prosodic and phonological processes.

5.3.1   Encapsulating global information for prosody

In the case of *amusin*, the input is an unsyllabified morphological tree rooted by an MStem. This MStem non-recursively dominates an MRoot.[4] The desired output contains both syllable structure and a prosodic

---

[4]Recall that an MNode of label x is said to recursively dominate another MNode of the same label x.

stem (PStem); the PStem is mapped from and matched with the MStem. In this section, I omit indexes.

(269) *Desired output for an unprosodified* amusin

Input

Output
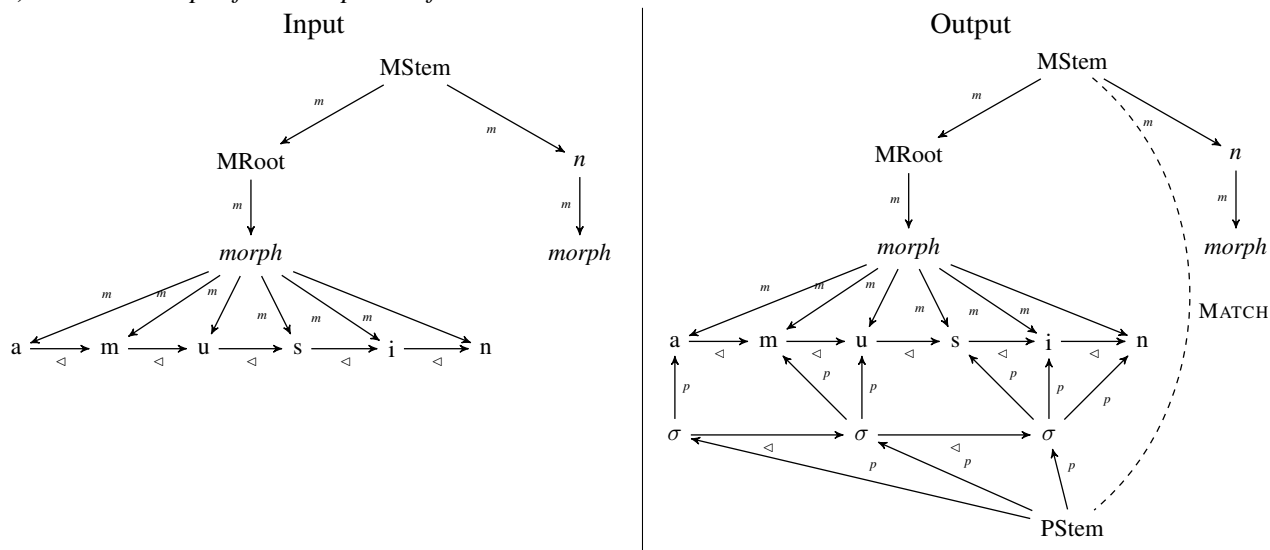
Generating the syllables does not need any morphological information. But in order to generate the PStem, we need to 'know' that the input contains an unparsed MStem. This information is encapsulated into our derivation's SETTINGS. I represent this constant as a floating box above the morphological tree.

(270) *Desired output for an unprosodified* amusin *using the* SETTINGS *variable*

Input

SETTINGS

Output
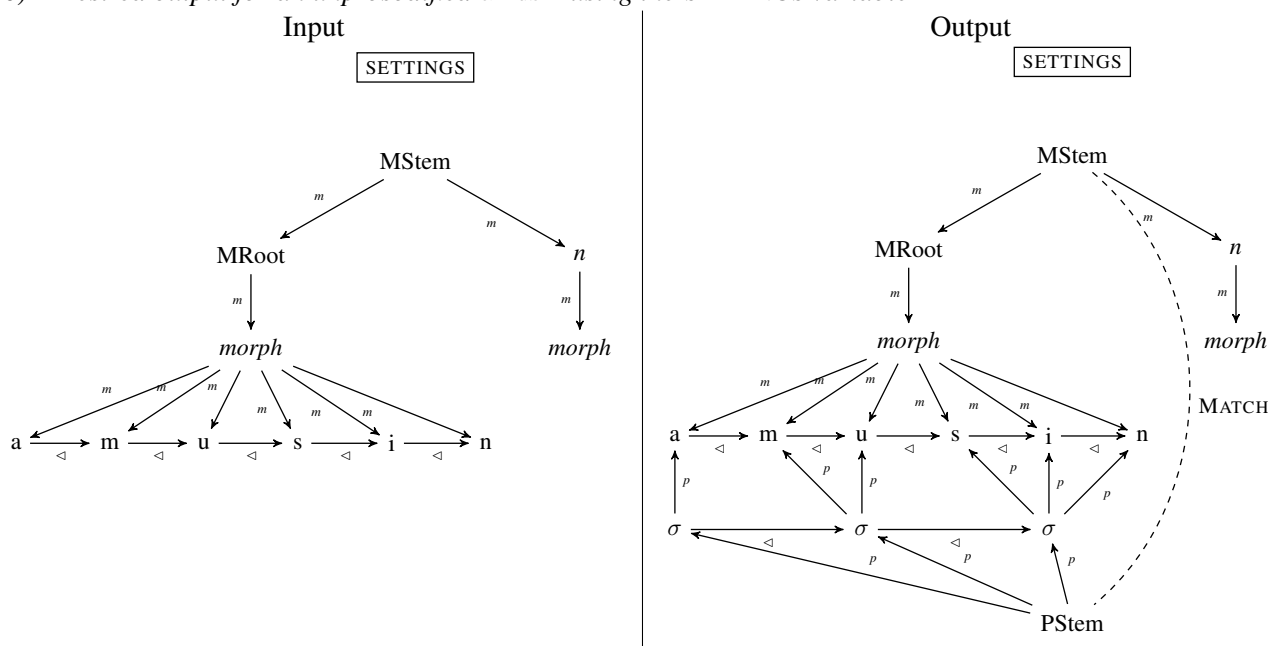
SETTINGS

To generate the right prosody, we need to update the SETTINGS with morphological information on what to parse. This information includes: 1) *What is the topmost morphological node?* and 2) *Has this MNode mapped to a prosodic node*? Possible answers to these questions are:

(271) *Possible settings for parsing:*

> The topmost MNode is...
> 1. an unparsed MStem over a root
> 2. an unparsed MStem over a (parsed) MStem
> 3. an unparsed MWord over an MStem
> 4. an unparsed MWord over a (parsed) MWord
> 5. a compound

We can redundantly include information about the topmost prosodic node in the input. This information is redundant because it is predictable from the morphology. For now, readers can ignore compounds.[5]

(272) *Possible redundant settings for parsing:*

The topmost MNode is...	+ The topmost PNode is...
1. an unparsed MStem over a root	+ a syllable
2. an unparsed MStem over a (parsed) MStem	+ a PStem
3. an unparsed MWord over an MStem	+ a PStem
4. an unparsed MWord over a (parsed) MWord	+ a PWord
5. a compound	+ a PStem

Determining the topmost morphological or prosodic node is done via the following user-defined predicates. The predicate $\mathbf{MTopmost}(x)$ picks the node x which is a morphological node and which is on the top of the tree, i.e., there is no other node that morphologically dominates it. As explained in §5.2, the predicate $\mathbf{MTopmost}(x)$ is QF-definable.

(273) *FO user-defined predicates for finding the topmost morphological and prosodic nodes*

- $\mathbf{MTopmost}(x) \overset{\text{def}}{=} \mathsf{MNode}(x) \wedge \neg\exists y[\mathsf{MDom}(y,x)]$
- $\mathbf{PTopmost}(x) \overset{\text{def}}{=} \mathsf{PNode}(x) \wedge \neg\exists y[\mathbf{PDom}(y,x)]$

Finding the topmost PNode is similarly defined.[6] Note how the relevant labels for $\mathbf{PTopmost}$ are all user-defined predicates. I generally do not reference the prosodically topmost node in this thesis.

In the case of the unsyllabified input *amusin*, the topmost MNode is an unparsed MStem which dominates an MRoot. This broken down into the following properties about the input:

1. The topmost MNode is an MStem
2. This MStem dominates an MRoot (= the MStem does not recursively dominate another MStem)
3. The MStem is unparsed (thus the topmost PNode is a syllable)

This information is what determines the prosodic parse. We encapsulate this information into the SETTINGS via a new type of **unary label**: Parse labels. Their role is to determine what MNode will be parsed. For *amusin*, the relevant Parse label is for parsing non-recursive MStems: Parse:MStem:nonrecursive(SETTINGS).

---

[5]This redundant information becomes useful if we no longer use a SETTINGS constant (Chapter 8; §8.3).

[6]Note that the prosodic 'tree' isn't always tree-like in having some unique node. If the word *amusin* is mapped to a sequence of syllables without a PStem *a.mu.sin*, then the 'topmost' PNode is a syllable even though there is no unique root for the tree.

(274)   *Unary labels for parsing PStems via the* SETTINGS *constant*

- Parse:MStem:nonrecursive(SETTINGS): is TRUE for SETTINGS iff we need to parse a non-recursive MStem into a PStem

Updating the SETTINGS is a distinct stage in the derivation. It is after a new morphological constituent is formed and before any prosody and rule application. This process is a logical transduction which uses a copy set of size 1. Everything in the input stays the same except for the labels on the SETTINGS.

(275)   *QF output functions for vacuous changes in* UPDATING THE SETTINGS

- For every label lab $\in L$ except labels for SETTINGS labels:
  $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$
- For every relation rel $\in R$:
  $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

The Parse status of the SETTINGS constant is determined via the output function below. It is set to TRUE for the SETTINGS iff the topmost morphological node in the tree is x, x is an MStem, it doesn't dominate another MStem y, and it is not matched with any PStem z. This output function is computationally non-local because it uses FO logic; we cannot replace the quantified variable $\exists x$ with unary functions. We need to do a global search over the input to find the topmost MNode.

(276)   *FO output function for parsing PStems via the* SETTINGS *constant*

- $\phi\texttt{Parse:MStem:nonrecursive}(\texttt{SETTINGS}^1) \stackrel{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge \mathsf{MStem}(x) \wedge$
  $\neg\exists y[\mathsf{MStem}(y) \wedge \mathsf{MDom}(x, y)]$
  $\neg\exists z[\mathsf{PStem}(z)) \wedge \mathsf{Match:stem}(x, z)]]$

For *amusin*, the output function above is true with the topmost node x as the $\text{MRoot}_{0.8}$. Thus, the SETTINGS constant is labelled TRUE for parsing a non-recursive MStem. This is shown below. I give new indexes to the recently generated suffix nodes.

(277)   *Setting parse status for* SETTINGS *for parsing a non-recursive MStem in* amusin
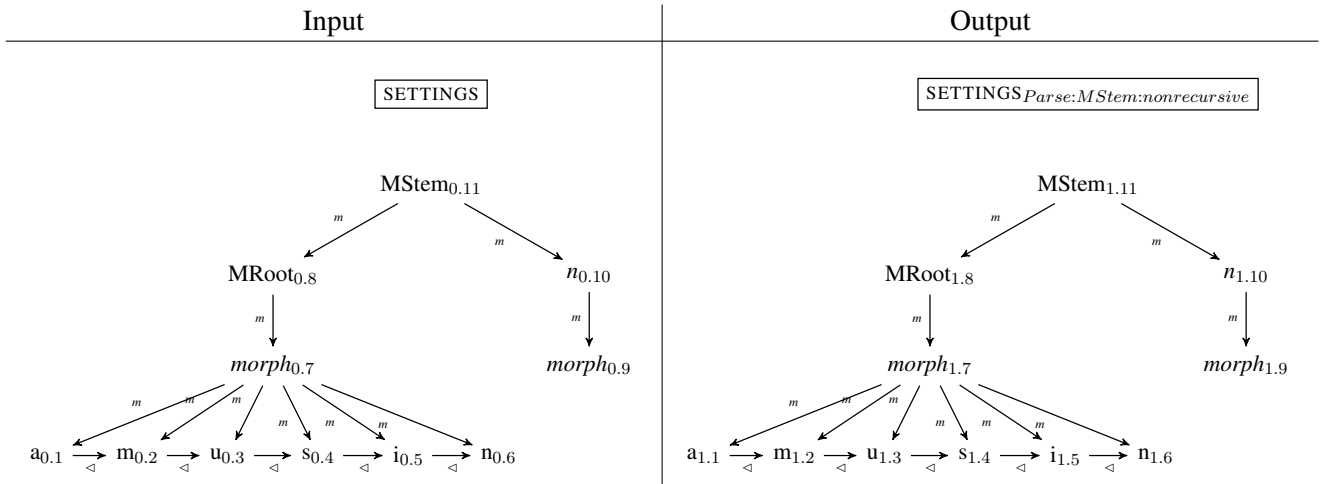
With the SETTINGS updated, the right MNode will parsed in a separate PROSODY step. This prosodic process is described in §5.4.3. I first formalize how the SETTINGS also determine the right cophonology.

5.3.2 Encapsulating global information for cophonologies and domains

Besides what to parse, the morphology tells us what set of **phonological alternations** to apply. These alternations are part of a **phonological rule domain**, also called a **cophonology** or **stratum**. I define these processes with logical output functions, but I use the term *rule* descriptively for easier illustration. In logical statements, I use the more specialized term 'cophonology'.

Rule domains are triggered by a certain morphological structure, whether the structure is a general morphological construction (MStem vs. MWord) or a specific morpheme (morpheme-specific rules or constraints (Inkelas 2008; Pater 2009)). In between, the domain can be a natural class of a small number of constructions, e.g., place names in Turkish (Inkelas and Orgun 2003). A cophonology can also be associated with a certain morphologically-derived prosodic constituent, e.g., the PStem (Downing 1999a) which triggers stress and (in Eastern Armenian) vowel reduction.

To apply rule domains, we need to logically formalize how to encode and select cophonologies. In order to encode the right stratum for a given derivation, two methods come to mind. The first is intuitively elegant and common but computationally awkward to define. I do not use it. The second alternative is computationally simple to define but (at first) counter-intuitive.

The first encoding mechanism is to bifurcate the set of rules or output functions into different subsets (rule strata) and to specify that each subset can only apply if the input's morphology is 'associated' with that stratum. It is unclear how this can be implemented because the prosaic description implies that the functions are being selected as variables for some meta-function for applying strata. The second approach is to set up a set of unary labels for possible *names* of cophonologies, and to place these labels onto the triggering MNodes/PNodes, as listed below. In the case of Armenian, MStems are labeled with the name of the stem-level (SLevel) cophonology, while MWords with that of the word-level (WLevel) cophonology.[7] PStems have the label of the PStem-level cophonology.

(278) *Unary labels for cophonologies for MNodes and PNodes*
- Cophon:SLevel(x): x triggers the stem-level cophonology
- Cophon:WLevel(x): x triggers the word-level cophonology
- Cophon:PStem(x): x triggers the PStem-level cophonology

I assume that MStems, MWords, and PStems are redundantly labeled with the right cophonology label.

(279) *Predictably associate certain morphological and prosodic constructions with certain cophonologies*
- MStem(x) → Cophon:SLevel(x)
- MWord(x) → Cophon:WLevel(x)
- PStem(x) → Cophon:PStem(x)

---

[7]Individual morphemes can have their own label for a morpheme-specific cophonology (passive cophonology) which may percolate higher in the tree. I set aside morpheme-specific phonology. Though in Chapter 8: §8.2.2, I sketch a morpheme-based alternative to constituent-based cophonologies.

In order to apply the right rules, we must examine the morphological structure in order to find the relevant morphological triggers. Alternatively, the right cophonology is selected by 'percolating' the cophonology label of the topmost MNode (or PNode) in the tree to the SETTINGS constant. Individual rules apply based on the SETTINGS's properties, specifically based on the following Domain labels.

(280)   *Unary labels for domains of cophonologies for* SETTINGS
  - Domain:Cophon:SLevel(SETTINGS): the SETTINGS has the domain of the SLevel cophonology
  - Domain:Cophon:WLevel(SETTINGS): the SETTINGS has the domain of the WLevel cophonology
  - Domain:Cophon:PStem(SETTINGS): the SETTINGS has the domain of the PStem cophonology

These labels are placed on the SETTINGS via the output functions below.[8] As with the prosody, these output functions require FO logic and are computationally non-local because of the quantifier $\exists x$. This quantifier finds the morphologically topmost MNode which is not connected to the SETTINGS constant.

(281)   *FO output functions for updating the* SETTINGS *for the right cophonology*
  - ϕDomain:Cophon:SLevel(SETTINGS^1) $\stackrel{\text{def}}{=} \exists x[\textbf{MTopmost}(x) \wedge \textsf{Cophon:SLevel}(x)]$
  - ϕDomain:Cophon:WLevel(SETTINGS^1) $\stackrel{\text{def}}{=} \exists x[\textbf{MTopmost}(x) \wedge \textsf{Cophon:WLevel}(x)]$

I illustrate this encapsulation or 'percolation' below. Note that the SETTINGS is simultaneously given the label for the right parsing and the right cophonology *before* we apply the prosody. Via the function ϕDomain:Cophon:SLevel(SETTINGS^1), the SETTINGS constant gets the SLevel domain label because the morphologically topmost MNode is an MStem x, and it has the SLevel cophonology label.

(282)   *Setting parse and domain status for* SETTINGS *for parsing a non-recursive MStem in* amusin
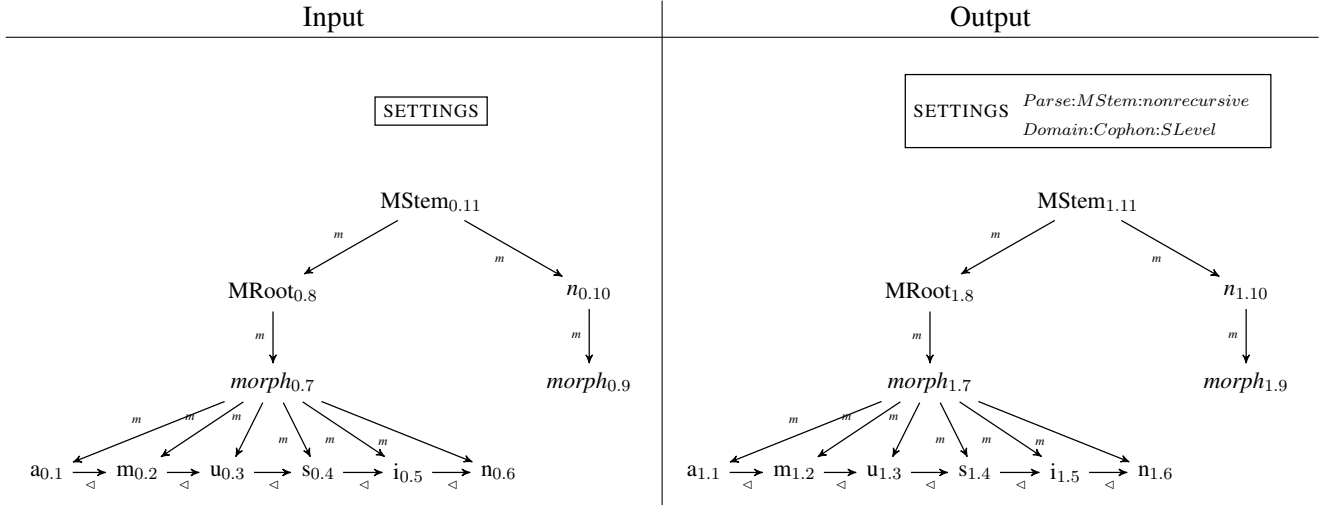
In order to give the SETTINGS constant the right unary labels, we need to use computationally non-local information via FO logic. However, once these right properties are encapsulated into the SETTINGS, we can then apply the prosodic parse and phonological rule domains. The latter two are computationally local because any potentially non-local trigger is encapsulated into the locally-accessible SETTINGS.

---

[8]For now, I set aside the PStem cophonology until Chapter 6: §6.6.2. The morphological cophonologies are determined after the Morphology, but this prosodic PStem cophonology is determined after the Prosody and before the Phonology.
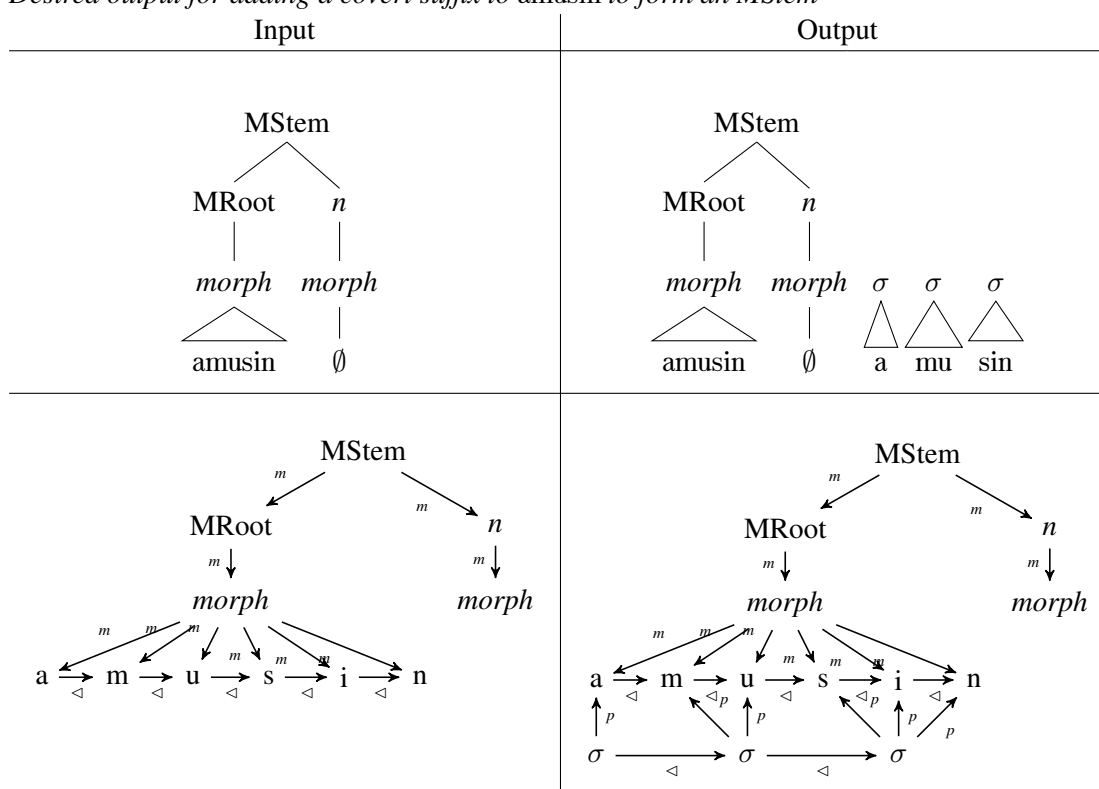
5.4 Prosody: Syllabification and prosodic mapping

Having updated the SETTINGS, we now generate the prosodic structure. I formalize this step as a transduction which consists of three smaller transductions: syllabification or the creation of new syllables (§5.4.1), ordering these syllables (§5.4.2), and the prosodic mapping of the PStem (§5.4.3). As I explain in Chapter 6: §6.5, I formalize prosodic mapping as a finite set of logical transductions, each for different possible parses.

5.4.1 Syllabification of an unsyllabified input

The first step in the prosody is **syllabification**. Syllabification takes as input an unparsed string of segments (and its morphological tree) and erects syllable structure. For example, the syllabification of the word *amusin* takes as input an unsyllabified morphological tree and generates a syllabified prosodic structure. The prosodic structure isn't exactly a 'tree' because it doesn't have a unique root node; it is instead a directed acyclic graph (DAG). The morphology and prosody are two DAGs that meet at the segments.

(283) *Desired output for adding a covert suffix to* amusin *to form an MStem*

Input	Output

MStem — MRoot, *n* — *morph* (amusin), *morph* (∅)

MStem — MRoot, *n* — *morph* (amusin), *morph* (∅), σ (a), σ (mu), σ (sin)

MStem → MRoot, *n* → *morph*, *morph* → a ⇄ m ⇄ u ⇄ s ⇄ i ⇄ n

MStem → MRoot, *n* → *morph*, *morph* → a ⇄ m ⇄ u ⇄ s ⇄ i ⇄ n, with σ → σ → σ

Inside words, initial syllabification is the same regardless of morphological structure, so it does not use any information from the SETTINGS of the derivation.[9] The maximal syllable is CVCC. Informally, vowels form the nucleus of some syllable. Any left-adjacent consonant is syllabified as the vowel's onset. Any

---

[9]Post-lexical syllabification is often affected by morphological structure. I do not discuss such cases in this thesis.

right-adjacent unsyllabified consonant is syllabified as a coda. If two unsyllabified consonants follow a vowel, these consonants can form a complex coda if they have falling sonority.[10] Below I formalize each of these steps. I set aside schwa epenthesis and appendix consonants (Vaux 1998b:ch2).

Syllabification requires a copy set of size 2. Copy 1 outputs any underlying segments and morphology; Copy 2 is where new syllable structure is created. In Copy 1, all relations surface faithfully *except* for any type of prosodic dominance between a syllable and a segment (onset, nucleus, inner coda, outer coda).

(284) *QF output functions for faithfully outputting labels and syllabification-independent relations in Copy 1*

- For every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$.
- For every relation $\mathsf{rel} \in R-\{\mathsf{PDom{:}syll\_ons}, \mathsf{PDom{:}syll\_nuc}, \mathsf{PDom{:}syll\_coda1}, \mathsf{PDom{:}syll\_coda2}\}$:
  $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

For *amusin*, the above functions generate the following faithful copy in Copy 1. I omit the morphological nodes and SETTINGS constant.

(285) *Syllabification of* amusin *– outputting segments*

Input $a_{0.1} \xrightarrow{\ \ } m_{0.2} \xrightarrow{\ \ } u_{0.3} \xrightarrow{\ \ } s_{0.4} \xrightarrow{\ \ } i_{0.5} \xrightarrow{\ \ } n_{0.6}$

Output   Copy 1 $a_{1.1} \xrightarrow{\ \ } m_{1.2} \xrightarrow{\ \ } u_{1.3} \xrightarrow{\ \ } s_{1.4} \xrightarrow{\ \ } i_{1.5} \xrightarrow{\ \ } n_{1.6}$

Copy 2

Unparsed segments are syllabified into the new syllables in Copy 2. The user-defined predicate below checks for any segments which are unsyllabified in the input or in Copy 1. An underlying segment x is unsyllabified if there is no syllable y in the input such that segment x is dominated by the syllable y as an onset, nucleus, or coda.

(286) a. *FO user-defined predicate for finding unsyllabified segments in the input*
- **unsyllabified**$(x) \stackrel{\text{def}}{=} \neg\exists y[\mathsf{syll}(y) \land [\mathsf{PDom{:}syll\_ons}(y, x)\lor$
  $\mathsf{PDom{:}syll\_nuc}(y, x)\land$
  $\mathsf{PDom{:}syll\_coda1}(y, x) \lor \mathsf{PDom{:}syll\_coda2}(y, x)]]$

b. *FO user-defined predicate for finding unsyllabified segments in Copy 1*
- ϕ**unsyllabified**$(x^1) \stackrel{\text{def}}{=} \neg\exists y[\phi\mathtt{syll}(y^1) \land [\phi\mathtt{PDom{:}syll\_ons}(y^1, x^1) \lor$
  $\phi\mathtt{PDom{:}syll\_nuc}(y^1, x^1)\lor$
  $\phi\mathtt{PDom{:}syll\_coda1}(y^1, x^1)\lor\phi\mathtt{PDom{:}syll\_coda2}(y^1, x^1)]]$

---

[10]My formalization is similar but not identical to Strother-Garcia (2019)'s. She provides multiple formalizations of syllable structure, including syllables with tree structure (Strother-Garcia 2019:37). In her tree model, she formalized prosodic concepts like onset and nucleus as individual nodes in the tree, whereas I formalize them as binary relations between segments and syllables. We are notationally equivalent.

The predicate $\phi\textbf{unsyllabified}(x^1)$ checks if some output correspondent x^1 in Copy 1 is syllabified with any syllables in Copy 1. The distinction between the predicates $\textbf{unsyllabified}(\textbf{x})$ and $\phi\textbf{unsyllabified}(x^1)$ is unneeded for now but is useful in resyllabification in Chapter 6: §6.4:. The two predicates are QF-definable. They use binary relations which can be converted to unary functions to find the syllable mother of some segment.

(287) a. *QF user-defined predicate for finding unsyllabified segments*
  - $\textbf{unsyllabified}(x) \overset{\text{def}}{=} \textsf{F}_\textsf{D}\text{:PDom:syll\_ons}(x) = \text{NULL} \wedge \textsf{F}_\textsf{D}\text{:PDom:syll\_nuc}(x) = \text{NULL} \wedge$
    $\textsf{F}_\textsf{D}\text{:PDom:syll\_coda1}(x) = \text{NULL} \wedge$
    $\textsf{F}_\textsf{D}\text{:PDom:syll\_coda2}(x) = \text{NULL}$

  b. *QF user-defined predicate for finding unsyllabified segments in Copy 1*
  - $\phi\textbf{unsyllabified}(x^1) \overset{\text{def}}{=} \phi\text{F}_D\text{:PDom:syll\_ons}(x^1) = \text{NULL} \wedge$
    $\phi\text{F}_D\text{:PDom:syll\_nuc}(x^1) = \text{NULL} \wedge$
    $\phi\text{F}_D\text{:PDom:syll\_coda1}(x^1) = \text{NULL} \wedge$
    $\phi\text{F}_D\text{:PDom:syll\_coda2}(x^1) = \text{NULL}$

Syllable creation and nucleus assignment proceeds as follows using the output functions below for Copy 2. Note that these output functions reference material generated in Copy 1.

Let x be an unparsed vowel in the input. It surfaces as an output vowel x^1 in Copy 1 (also called y^1 in the relation below). Via $\phi\text{syll}(x^2)$ in Copy 2, the output correspondent x^2 is generated and labeled as a syllable if the output vowel x^1 is a vowel and it is unsyllabified. Via $\phi\text{PDom:syll\_nuc}(x^2, y^1)$, this new syllable x^2 prosodically dominates the output vowel x^1 ($=y^1$) as its nucleus because x^2 is a syllable, y^1 is a vowel, y^1 and x^1 are the same vowel, and $x^1 = y^1$ is unsyllabified in Copy 1.

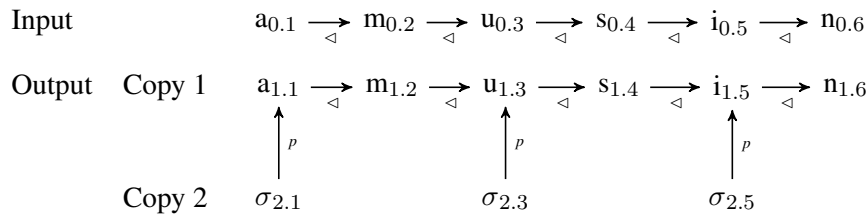(288) a. *QF output function for syllable creation in Copy 2*
  - $\phi\text{syll}(x^2) \overset{\text{def}}{=} \phi\text{vowel}(x^1) \wedge \phi\textbf{unsyllabified}(x^1)$

  b. *QF output function for nuclei assignment across Copy 1 and 2*
  - $\phi\text{PDom:syll\_nuc}(x^2, y^1) \overset{\text{def}}{=} \phi\text{syll}(x^2) \wedge \phi\text{vowel}(y^1) \wedge x = y \wedge \phi\textbf{unsyllabified}(y^1)$

To illustrate, the first, second, and third vowel in *amusin* are [a,u,i] with a subscript index $a_{0.1}, u_{0.3}, i_{0.5}$. In Copy 2, they project output correspondents that are labelled as syllables: $\sigma_{2.1}, \sigma_{2.3}, \sigma_{2.5}$.

(289) *Syllabification of* amusin *– syllable creation and nucleus assignment*

Input $\quad a_{0.1} \xrightarrow{\quad} m_{0.2} \xrightarrow{\quad} u_{0.3} \xrightarrow{\quad} s_{0.4} \xrightarrow{\quad} i_{0.5} \xrightarrow{\quad} n_{0.6}$

Output Copy 1 $\quad a_{1.1} \xrightarrow{\quad} m_{1.2} \xrightarrow{\quad} u_{1.3} \xrightarrow{\quad} s_{1.4} \xrightarrow{\quad} i_{1.5} \xrightarrow{\quad} n_{1.6}$

$\qquad\qquad\qquad \uparrow p \qquad\qquad\qquad \uparrow p \qquad\qquad\qquad \uparrow p$

Copy 2 $\quad \sigma_{2.1} \qquad\qquad\qquad \sigma_{2.3} \qquad\qquad\qquad \sigma_{2.5}$

Any surrounding consonants are then assigned as margins to these new syllables via the predicates and functions below. Let y be a consonant in the input. Let x be a vowel in the input which is newly parsed as

the nucleus of some syllable x^2 in Copy 2. In Copy 1, the output correspondent y^1 surfaces faithfully as a consonant. In Copy 2, the new syllable z^2 will dominate these consonants either as onsets or codas.

For illustration, I assume that only simplex onsets are allowed. Via $\phi\texttt{PDom:syll\_ons}(x^2, y^1)$, a consonant y^1 is a onset for x^2 if x^2 is a syllable, y^1 is a consonant, y^1 is unsyllabified, and there exists a vowel z^1 which surfaces as a vowel in Copy 1 such that z^1 immediately succeeds x^1, *and* z^1 is the nucleus of the syllable x^2.[11] This output function is made QF by using unary functions.

(290)   a. *FO output function for onset assignment across Copy 1 and 2*

- $\phi\texttt{PDom:syll\_ons}(x^2, y^1) \stackrel{\text{def}}{=} \phi\texttt{syll}(x^2) \wedge \phi\texttt{consonant}(y^1) \wedge \phi\textbf{unsyllabified}(y^1) \wedge$
$\exists z[\phi\texttt{vowel}(z^1) \wedge \phi\texttt{succ:seg}(y^1, z^1) \wedge$
$\phi\texttt{PDom:syll\_nuc}(x^2, z^1)]$

   b. *QF output function for onset assignment across Copy 1 and 2*

- $\phi\texttt{PDom:syll\_ons}(x^2, y^1) \stackrel{\text{def}}{=} \phi\texttt{syll}(x^2) \wedge \phi\texttt{consonant}(y^1) \wedge \phi\textbf{unsyllabified}(y^1) \wedge$
$\phi\texttt{vowel}(\phi\text{F}_L\texttt{:succ:seg}(y^1)) \wedge$
$\phi\text{F}_L\texttt{:succ:seg}(y^1) = \phi\text{F}_M\texttt{:PDom:syll\_nuc}(x^2)$

To illustrate in *amusin*, the consonants [m,s] are syllabified as the onset of the following vowel's syllable. For the consonant [m], x^2 is $\sigma_{2.3}$, y^1 is $m_{1.2}$, and z^1 is $u_{1.3}$. For the consonant [s], x^2 is $\sigma_{2.5}$, y^1 is $s_{1.3}$, and z^1 is $i_{1.5}$. For both consonants, $\phi\textbf{unsyllabified}(y^1)$ is TRUE because there is no syllable in Copy 1 which dominates these consonants.

(291)   *Syllabification of* amusin *– onset assignment*

Input $\quad\quad a_{0.1} \longrightarrow m_{0.2} \longrightarrow u_{0.3} \longrightarrow s_{0.4} \longrightarrow i_{0.5} \longrightarrow n_{0.6}$

Output   Copy 1 $\quad a_{1.1} \longrightarrow m_{1.2} \longrightarrow u_{1.3} \longrightarrow s_{1.4} \longrightarrow i_{1.5} \longrightarrow n_{1.6}$

Copy 2 $\quad \sigma_{2.1} \quad\quad\quad \sigma_{2.3} \quad\quad\quad \sigma_{2.5}$
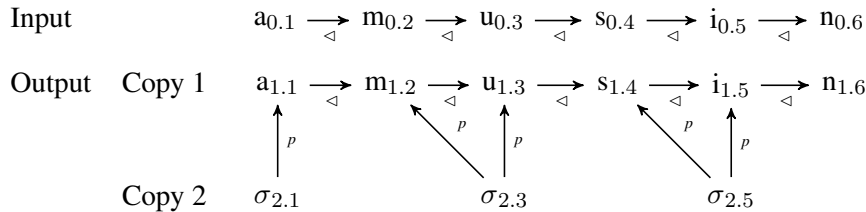
Coda assignment is similar. Via $\phi\texttt{PDom:syll\_coda1}(x^2, y^1)$, an outputted consonant y^1 is parsed as an inner coda for a syllable x^2 if x^2 is a syllable, y^1 is a consonant, y^1 is unsyllabified in Copy 1, and there is a vowel z^1 which *precedes* the consonant y^1 and is a nucleus for x^2. Additionally, there must not be any new syllable u^2 in Copy 2 such that u^2 parses y^1 as its onset. This last condition prevents syllables from losing their onsets to preceding syllables. In serial terms, onset assignment precedes coda assignment. In our logical framework, coda assignment is defined in terms of onset assignment.[12] This output function can be made QF.

---

[11]This formula is redundant because *z is x* because of the how syllables are defined.

[12]The distinction is subtle. An efficient compiler would implement onset assignment before coda assignment in order to to avoid repetitive tasks. But coda assignment can be implemented before onset assignment; the catch is that coda assignment would analyze its context while also looking for the context for onset assignment.
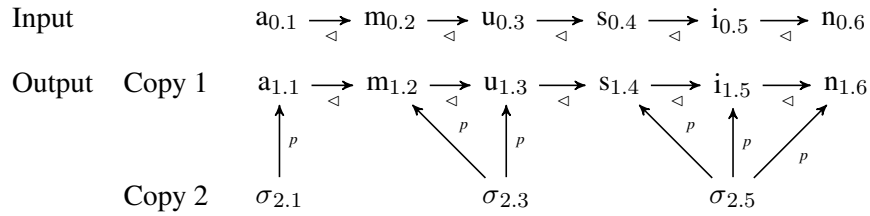
(292) a. *FO output function for inner coda assignment across Copy 1 and 2*
- $\phi\mathrm{PDom:syll\_coda1}(x^2, y^1) \overset{\text{def}}{=} \phi\mathrm{syll}(x^2) \wedge \phi\mathrm{consonant}(y^1) \wedge \phi\mathbf{unsyllabified}(y^1) \wedge$
$\exists z[\phi\mathrm{vowel}(z^1) \wedge \phi\mathrm{succ:seg}(z^1, y^1) \wedge] \wedge$
$\phi\mathrm{PDom:syll\_nuc}(x^2, z^1)] \wedge$
$\neg\exists u[\phi\mathrm{syll}(u^2) \wedge \phi\mathrm{PDom:syll\_ons}(u^2, y^1)]$

  b. *QF output function for inner coda assignment across Copy 1 and 2*
- $\phi\mathrm{PDom:syll\_coda1}(x^2, y^1) \overset{\text{def}}{=} \phi\mathrm{syll}(x^2) \wedge \phi\mathrm{consonant}(y^1) \wedge \phi\mathbf{unsyllabified}(y^1) \wedge$
$\phi\mathrm{vowel}(\phi\mathrm{F}_R\mathrm{:succ:seg}(y^1)) \wedge$
$\phi\mathrm{F}_R\mathrm{:succ:seg}(y^1) = \phi\mathrm{F}_M\mathrm{:PDom:syll\_nuc}(x^2) \wedge$
$\phi\mathrm{F}_D\mathrm{:PDom:syll\_ons}(y^1) = \mathrm{NULL}$

To illustrate, in *amusin*, only the consonant [n] is parsed as an inner coda. Here, x^2 is $\sigma_{2.5}$, y is $n_{0.6}$, y^1 is $n_{1.6}$, and z^1 is $i_{1.5}$. The condition $\phi\mathbf{unsyllabified}(y^1)$ is satisfied because $n_{1.6}$ is not syllabified in any syllable that's in Copy 1. The condition $\neg\exists u[\phi\mathrm{syll}(u^2) \wedge \phi\mathrm{PDom:syll\_ons}(u^2, y^1)]$ is satisfied because it is not parsed as an onset for any syllable in Copy 2. For other consonants like $s_{1.4}$, it is not parsed as a coda because it is parsable as an onset for the new syllable $\sigma_{2.5}$, thus violating $\phi\mathrm{PDom:syll\_ons}(u^2, y^1)$.[13]

(293) *Syllabification of* amusin – *inner coda assignment*

Input $\qquad a_{0.1} \longrightarrow m_{0.2} \longrightarrow u_{0.3} \longrightarrow s_{0.4} \longrightarrow i_{0.5} \longrightarrow n_{0.6}$

Output Copy 1 $\quad a_{1.1} \longrightarrow m_{1.2} \longrightarrow u_{1.3} \longrightarrow s_{1.4} \longrightarrow i_{1.5} \longrightarrow n_{1.6}$

Copy 2 $\qquad \sigma_{2.1} \qquad\qquad\quad \sigma_{2.3} \qquad\qquad\quad \sigma_{2.5}$

This completes the syllabification of *amusin*.

Note that *amusin* lacks any outer codas. To illustrate outer coda assignment, consider the base *surp*. For parsing outer codas, we need to reference what are possible complex codas. For illustration, the user-defined predicate **good_CC**(x, y) below lists all acceptable complex codas. They are defined over either the input or Copy 1.[14]

(294) *QF user-defined predicate for checking acceptable complex codas*
- **good_CC**$(x, y) \overset{\text{def}}{=} [\mathrm{r}(x) \wedge \mathrm{s}(y)] \vee [\mathrm{r}(x) \wedge \mathrm{t}(y)] \vee \ldots$
- ϕ**good_CC**$(x^1, y^1) \overset{\text{def}}{=} [\mathrm{r}(x^1) \wedge \mathrm{s}(y^1)] \vee [\mathrm{r}(x^1) \wedge \mathrm{t}(y^1)] \vee \ldots$

A consonant y^1 in Copy 1 is parsed as an outer coda for a syllable x^2 in Copy 2 in much the same way as an inner coda. x^2 must be a syllable, y^1 must be a consonant and specifically an unsyllabified consonant. There must exist a consonant w^1 in Copy 1 such that w^1 precedes y^1. w^1 is the inner coda of syllable x^2, and w^1, y^1 must form an acceptable complex coda. Finally, y^1 must not be parsed as some other syllable u^2's onset. This function can be made QF.

---

[13]Note that I say *parsable* and not *parsed* because coda assignment is not *temporarily* ordered after onset assignment in its abstract logical definition. If implemented, a compiler would have to do this ordering for efficiency.

[14]The predicate misses the generalization that the relevant property is sonority, but see Strother-Garcia (2018, 2019) on how to formalize sonority in formal logic by using binary predicates. Adopting sonority does not significantly change the computation.

(295)   a. *FO output function for outer coda assignment across Copy 1 and 2*

- ϕPDom:syll_coda2$(x^2, y^1) \overset{\text{def}}{=} \phi$syll$(x^2) \wedge \phi$consonant$(y^1) \wedge \phi$**unsyllabified**$(y^1) \wedge$
  $\exists w[\phi$consonant$(w^1) \wedge \phi$succ:seg$(w^1, y^1) \wedge$
  ϕPDom:syll_coda1$(x^2, w^1) \wedge$
  ϕ**good_CC**$(w^1, y^1)] \wedge$
  $\neg \exists u[\phi$syll$(u^2) \wedge \phi$PDom:syll_ons$(u^2, y^1)]$

   b. *QF output function for outer coda assignment across Copy 1 and 2*

- ϕPDom:syll_coda2$(x^2, y^1) \overset{\text{def}}{=} \phi$syll$(x^2) \wedge \phi$consonant$(y^1) \wedge \phi$**unsyllabified**$(y^1) \wedge$
  ϕconsonant$(\phi$F$_R$:succ:seg$(y^1)) \wedge$
  $x^2 = \phi$F$_D$:PDom:syll_coda1$(\phi$F$_R$:succ:seg$(y^1))$
  ϕ**good_CC**$(\phi$F$_R$:succ:seg$(y^1), y^1)$
  ϕF$_D$:PDom:syll_ons$(y^1) =$ NULL

To illustrate, consider the word *surp*. Its final consonant *s* is parsed as an outer coda. Here x^2 is $\sigma_{2.1}$, y^1 is $p_{1.4}$, z^1 is $u_{1.2}$, and w^1 is $r_{1.3}$. The consonant $s_{1.4}$ can form a complex coda with $r_{1.3}$ because the pair (r,p) have falling sonority and are in the list of acceptable complex codas in ϕ**good_CC**(w^1, y^1).

(296)   *Syllabification of* surp – *outer coda assignment*
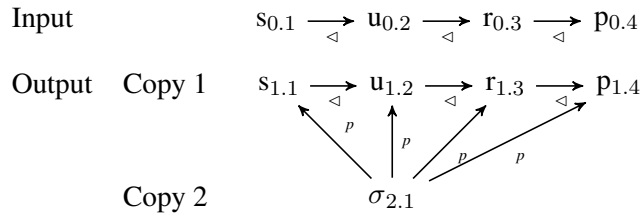
In sum, syllabification is logically definable with QF logic and computationally local.

5.4.2   Syllable ordering and tier projection

Any newly created syllables must be ordered amongst themselves. For ease of illustration, I treat syllable ordering as a separate step that follows syllabification. This factorization is trivial because syllabification and ordering can be composed into a single logical transduction. The ordering between syllables is determined by examining the distance between their nuclei. Doing so uses some long-distance computation in the form of a tier projection. In the string *amusin*, the vowels *a,u* are not linearly adjacent; *a* does not immediately precede *u* or vice-versa. The vowel *a* instead generally precedes *u*. But, the vowels are local on a tier of vowels. Formalizing tiers is the brunt of the work in this section. I formalize how this long-distance information can be extracted with transitive closure into a vowel tier. Doing so requires MSO logic. I then explain some alternatives which are computationally local and use QF logic.

5.4.2.1   Tier projection: Projecting long-distance information via transitive closure and tiers

The ordering transduction uses a copy set of size 1. In Copy 1, all underlying labels are faithfully outputted. All relations are faithfully outputted except for immediate successor among syllables. Note

how the syllables have new indexes in the input and output because they were added in the previous stage of the derivation. After a logical transduction has applied, i.e., after the Morphology or Syllabification has applied, the newly generated nodes get new indexes.

(297)   *Unary labels for syllable ordering over Copy 1*
- For every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$.
- For every relation $\mathsf{rel} \in R - \{\mathsf{succ{:}syll}(x,y)\}$:
  $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x,y)$

To illustrate, for the word *amusin*, the input is already syllabified but the syllables are unordered. The intermediate output is the same as the input because the input lacked any ordered syllables.

(298)   *Ordering syllables for* amusin - *outputting segments and syllables*
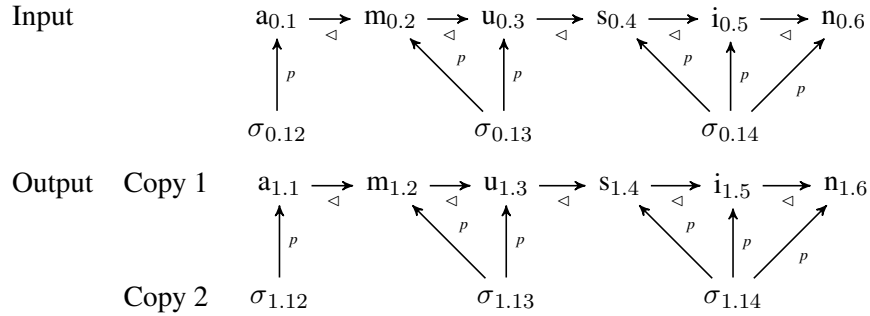
In Chapter 4: §4.4.1, I defined the predicate **gen_prec:seg**(x,y) which computes if x non-immediately precedes y. With this MSO predicate, we can check whether two vowels are *relatively* close to each other not. That is, we can generate a tier of vowels and check whether some vowel y is the first vowel after some other vowel x, i.e., if x, y are tier-local or x tier-precedes y. Given two vowels x, y, they are local on the tier of vowels if x, y are vowels, x generally precedes y, but there is no other vowel z which comes between them.

(299)   *MSO definition for tier-locality for vowels*

- **tier_local:vowel**$(x, y) \stackrel{\text{def}}{=} \mathsf{vowel}(x) \wedge \mathsf{vowel}(y) \wedge \textbf{gen\_prec:seg}(x,y) \wedge$
  $\neg \exists w [\mathsf{vowel}(w) \wedge \textbf{gen\_prec:seg}(x,w) \wedge \textbf{gen\_prec:seg}(w,y)]$

To illustrate, $a_{0.1}$ and $u_{0.3}$ in *amusin* are tier-local vowels because $u_{0.3}$ is part of the line X from $a_{0.1}$ to the end: *amusin*, and there is no other vowel between $a_{0.1}$ and $u_{0.3}$ on this line.

With this tier of vowels, we can now order syllables via $\phi\mathtt{succ{:}syll}(x^1, y^1)$. We know if some syllable is immediately before or after some other syllable by checking if their nuclei are tier-local or not. Given two unordered syllables x, y and their output correspondents x^1, y^1, x^1 immediately precedes y^1 if x, y are syllables, they have as nuclei the vowels u, v, and the vowel u tier-precedes v.

(300) *MSO output function for linearly ordering syllables*

- $\phi\texttt{succ:syll}(x^1, y^1) \stackrel{\text{def}}{=} \textsf{syll}(x) \wedge \textsf{syll}(y)\wedge$
  $\exists u, v[\textsf{vowel}(u) \wedge \textsf{vowel}(v)\wedge$
  $\textsf{PDom:syll\_nuc}(x, u)\wedge\textsf{PDom:syll\_nuc}(y, v)\wedge\textbf{tier\_local:vowel}(u, v)]$

I illustrate this output function below.

(301) *Ordering syllables for* amusin - *ordering syllables*
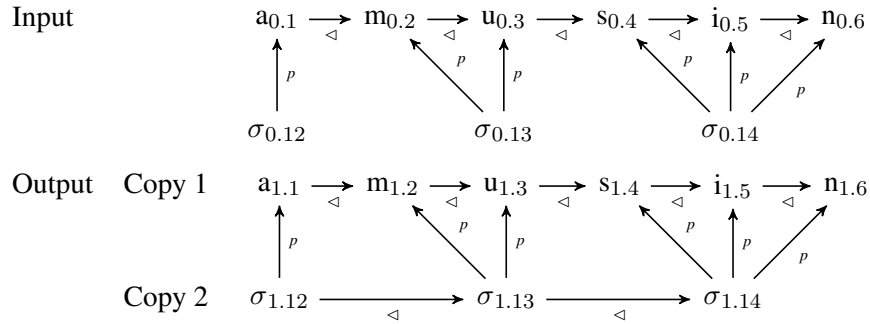
5.4.2.2   Local alternatives: precedence as a primitive and finite syllable size

The above formalization of syllable ordering utilizes a tier projection of vowels. Computing this projection requires MSO logic and is computationally non-local. It uses global information because general precedence **gen_prec:seg**(x, y) requires examining potentially unbounded portions of an input string. In this section, I describe two possible alternatives: enriching the input with general precedence, or imposing a bound on syllable size. The first reduces the computation to FO and the second to QF.

The first alternative formalization avoids the use of MSO logic by making the input directly encode general precedence (Heinz 2010; Rogers et al. 2013). Assume that the input is a string of segments which are ordered by general precedence. I illustrate this alternative representation for the word *æpəl* 'apple'.

(302) *Representing a string* æpəl 'apple' *in terms of general precedence instead of immediate successor*
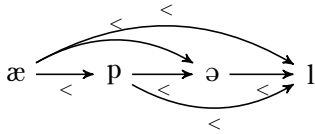
If general precedence is a primitive in our input representation, then the computation is easier and doesn't use MSO logic. It instead uses FO logic. In the predicate **tier_local:vowel**(x, y), the formula **gen_prec:seg**(x, y) is now an input binary relation gen_prec:seg(x, y).

(303) *FO definition for tier-locality for vowels with general precedence as an input binary relation*

- $\textbf{tier\_local:vowel}(x, y) \stackrel{\text{def}}{=} \textsf{vowel}(x) \wedge \textsf{vowel}(y) \wedge \textsf{gen\_prec:seg}(x, y) \wedge$
  $\neg\exists w[\textsf{vowel}(w) \wedge \textsf{gen\_prec:seg}(x, w) \wedge \textsf{gen\_prec:seg}(w, y)]$

The second alternative exploits the fact that syllables have bounded size. If we know that a syllable in a specific language is at most CVCC, then any two tier-local vowels x, y must be separated by at most 3 consonants. Thus, we can check if some vowel x tier-precedes a vowel y by checking if x, y are separated by 0, 1, 2, or 3 segments and none of them are vowels. This reduces the computation further down to computationally local QF logic. The predicate **tier_local:vowel**(x, y) can be made QF by replacing the relation $\text{succ:seg}(x, y)$ by the unary functions $\mathsf{F_L}\text{:succ:seg}(x)$ and $\mathsf{F_R}\text{:succ:seg}(y)$.

(304)   a. *FO user-defined predicate for tier-locality of vowels without general precedence*

- **tier_local:vowel**$(x, y) \stackrel{\text{def}}{=} \text{vowel}(x) \wedge \text{vowel}(y) \wedge$
  $[\,[\text{succ:seg}(x, y)] \vee$
  $\exists u[\text{succ:seg}(x, u) \wedge \text{succ:seg}(u, y) \wedge \neg\text{vowel}(u)] \vee$
  $\exists u, v[\text{succ:seg}(x, u) \wedge \text{succ:seg}(u, v) \wedge \text{succ:seg}(v, y) \wedge \neg\text{vowel}(u) \wedge$
  $\neg\text{vowel}(v)] \vee$
  $\exists u, v, w[\text{succ:seg}(x, u) \wedge \text{succ:seg}(u, v) \wedge \text{succ:seg}(v, w) \wedge$
  $\text{succ:seg}(w, y) \wedge \neg\text{vowel}(u) \wedge \neg\text{vowel}(v) \wedge \neg\text{vowel}(w)]\,]$

   b. *QF user-defined predicate for tier-locality of vowels without general precedence*

- **tier_local:vowel**$(x, y) \stackrel{\text{def}}{=} \text{vowel}(x) \wedge \text{vowel}(y) \wedge$
  $[y = \mathsf{F_L}\text{:succ:seg}(x)] \vee$
  $[y = \mathsf{F_L}\text{:succ:seg}^2(x) \wedge \neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}(x))] \vee$
  $[y = \mathsf{F_L}\text{:succ:seg}^3(x) \wedge \neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}(x)) \wedge$
  $\neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}^2(x))] \vee$
  $[y = \mathsf{F_L}\text{:succ:seg}^4(x) \wedge \neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}(x)) \wedge$
  $\neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}^2(x)) \wedge \neg\text{vowel}(\mathsf{F_L}\text{:succ:seg}^3(x))]$

In sum, tier projection in general uses MSO logic and is computationally complex. But tier projection for vowels requires the much less expressive QF logic because there are finite bounds on syllable size.

### 5.4.3 Prosodic mapping of a Prosodic Stem

In the previous section, the input *amusin* was partially prosodified by erecting the right syllable structure. We now parse the MStem into its own PStem via prosodic mapping.

(305)   *Desired output for a partially prosodified* amusin *without a PStem*
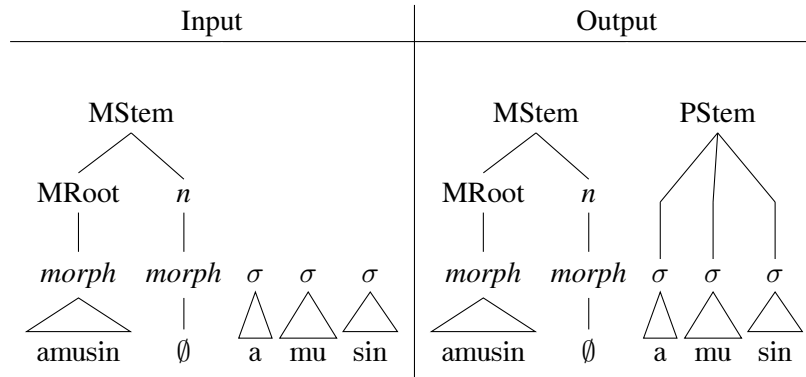
Although not clear in the above simple graph, the more explicit representation shows that the MStem is mapped to a PStem via a MATCH relation (cf. Selkirk 2011), represented by a dashed line. This PStem dominates the MStem's syllables.

(306)   *Explicit desired output for a partially prosodified* amusin *without a PStem*
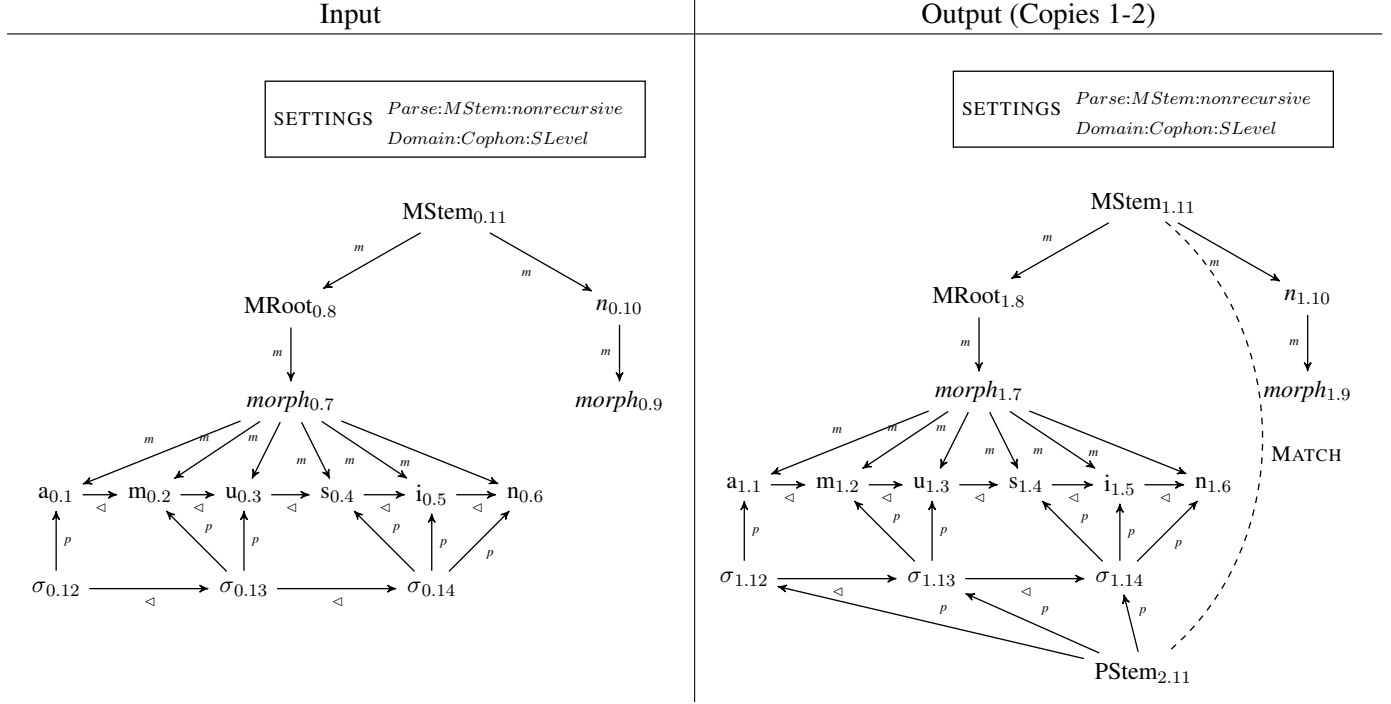
The relevant unary labels and binary relations are those below.

(307)   a.   *Unary labels involved in PStems formation*
- $\text{PStem}(x)$: x is a PStem
- $\text{MStem}(x)$: x is an MStem

       b.   *Binary relations involved in PStem formation*
- $\text{Match:stem}(x, y)$: the MStem x is matched with (mapped to) the PStem y
- $\text{PDom:PStem\_syll}(x, y)$: the PStem x prosodically dominates the syllable y

In order to generate the PStem, we need to 'know' that the input contains an unparsed non-recursive MStem. This information is encapsulated into our derivation's SETTINGS via the label Parse:MStem:nonrecursive. The right label was determined earlier in §5.3.1. We use this label here to generate a PStem. The prosodic transductions in this section are QF because any potential non-locality is encapsulated into the SETTINGS.

Parsing the MStem into a PStem is a transduction with a copy set of size 2. In Copy 1, the input is faithfully outputted. Note that relevant output functions have a condition for the right parsing setting: Parse:MStem:nonrecursive(SETTINGS).[15]

---

[15]As previewed in Chapter 4: §4.3.5, when other parse settings are defined, these output functions must be rewritten using helper predicates to handle disjunctions of different possible parse settings, i.e., redefine $\phi\text{lab}(x^1)$ for parsing a non-recursive MStem, recursive MStem, or MWord. This is further elaborated in §5.6.2 for the case of conflicting cophonologies.

(308) *QF output functions for the faithful output in Copy 1 when parsing a non-recursive MStem*
- For every label lab $\in L$:
  $\phi \texttt{lab}(x^1) \overset{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$ lab(x)
- For every relation rel $\in R$:
  $\phi \texttt{rel}(x^1, y^1) \overset{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$ rel(x, y)

In Copy 2, a PStem is generated for the unparsed non-recursive MStem in *amusin*. Again, we know that we are supposed to do this because of the labels on SETTINGS.

(309) *QF output functions for parsing a non-recursive unparsed MStem*
  a. *Generating a new PStem*
  - $\phi \texttt{PStem}(x^2) \overset{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$ MStem(x)
  b. *Matching the MStem with the new PStem*
  - $\phi \texttt{Match:stem}(x^1, y^2) \overset{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$
    $\phi \texttt{MStem}(x^1) \wedge \phi \texttt{PStem}(y^2) \wedge x = y$

(310) below illustrates. The MStem in *amusin* is mapped to a PStem in Copy 2. The input MStem x is MStem$_{0.11}$. It is mapped to a new PStem in Copy 2 via its output correspondent x^2 which is PStem$_{2.11}$. This PStem x^2 is created because the SETTINGS has the relevant label for parsing a non-recursive MStem and x is an MStem.

(310) *Parsing a non-recursive MStem in* amusin *into a PStem – mapping the PStem*
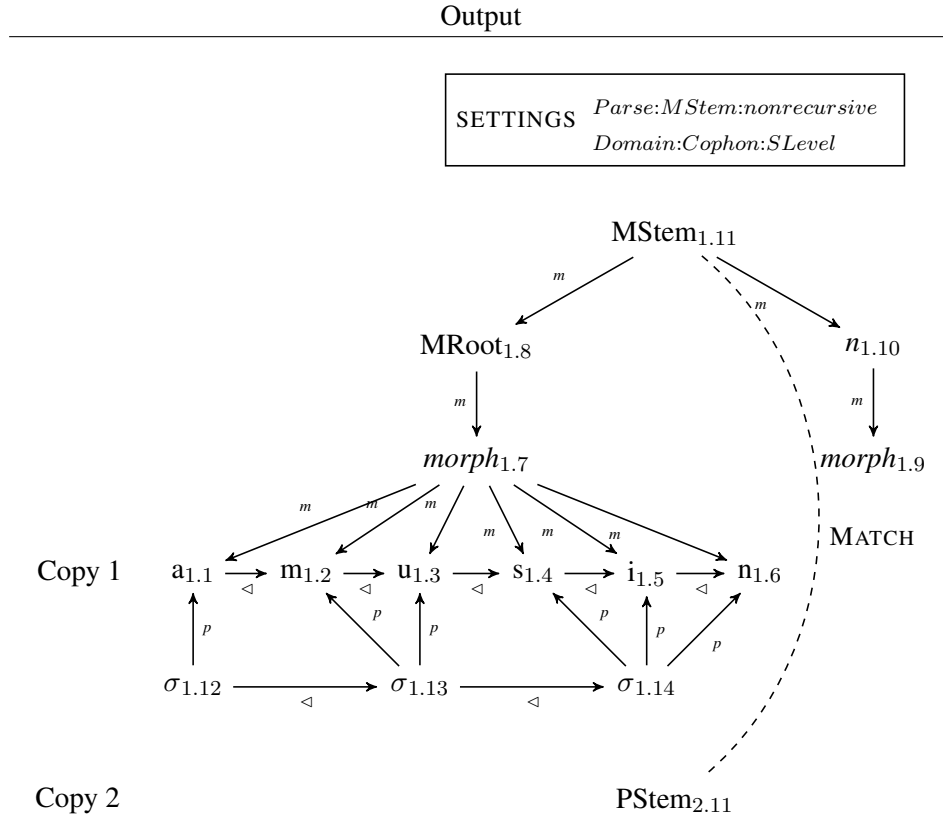
Once mapped, the PStem in Copy 2 (now y^2) is matched to the MStem in Copy 1 (x^1) via ϕMatch:stem(x^1, y^2). This output function is satisfied because x^1 is an MStem, y^2 is a PStem, and x^1, y^2 are output correspondents of the same underlying node ($x = y$).

This PStem must now prosodically dominate the MStem's syllables. This is illustrated below.

(311) *Parsing a non-recursive MStem in* amusin *into a PStem – dominating the MStem syllables*
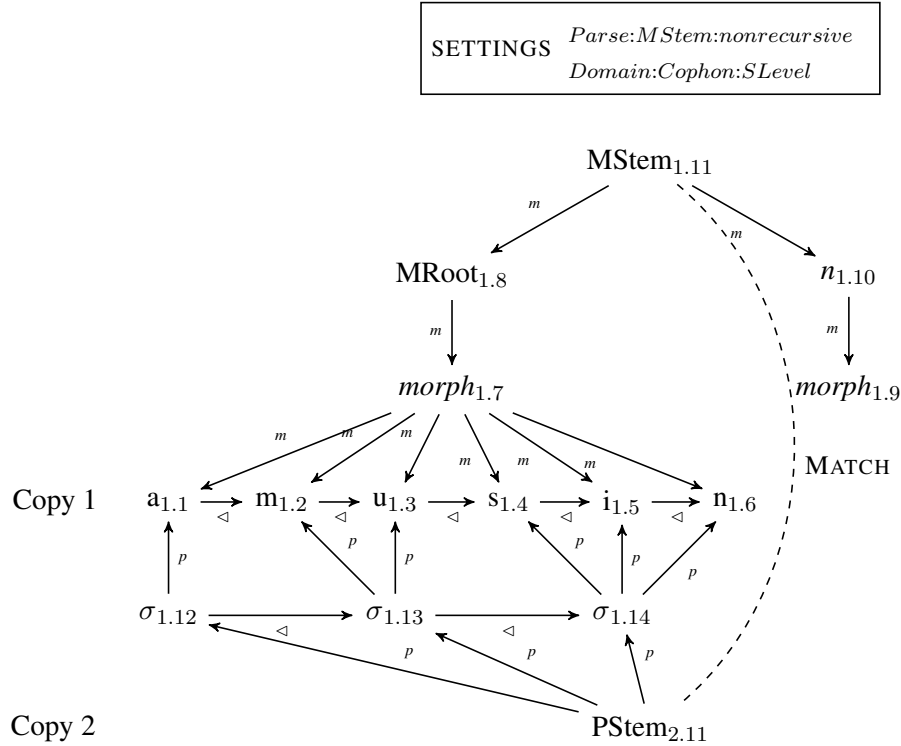
The relevant syllables are found using the user-defined predicate **syll_of_MStem**(x, y). This predicate is true for any x, y such that x is a syllable, y is an MStem, and the MStem y dominates the nucleus of the syllable x. Defining this last condition can potentially use long-distance information. In the case of a non-recursive MStem, no long-distance information is used because the MStem is separated from the segmental tier by only the morph and morpheme levels. Thus, we know if the syllable is part of the MStem if there exists a morpheme u, a morph v, and a segment w, such that the MStem y dominates the morpheme u which dominates the morph v which dominates the segment w, and the segment w is the nucleus of the syllable x.

(312) *FO user-defined predicate for finding syllables of a non-recursive MStem*

- **syll_of_MStem**$(x, y) \overset{\text{def}}{=}$ syll$(x) \wedge$ MStem$(y) \wedge$
  $\exists u, v, w[\text{morpheme}(u) \wedge \text{morph}(v) \wedge \text{seg}(w) \wedge$
  $\text{MDom}(y, u) \wedge \text{MDom}(u, v) \wedge \text{MDom}(v, w) \wedge \text{PDom:syll\_nuc}(x, w)]$

The above predicate is QF and uses local information. Given some syllable y, we find its unique

nucleus w. We then find the nucleus's MStem x via a finite line of daughter-to-mother dependencies in the morphological structure.

(313)   *QF user-defined predicate for finding syllables of a non-recursive MStem*

- **syll_of_MStem**$(x, y) \stackrel{\text{def}}{=}$ syll$(x) \wedge$ MStem$(y) \wedge$
  seg($\mathsf{F_M}$:PDom:syll_nuc$(x)) \wedge$
  morph($\mathsf{F_D}$:MDom($\mathsf{F_M}$:PDom:syll_nuc(x))) $\wedge$
  morpheme($\mathsf{F_D}$:MDom^2($\mathsf{F_M}$:PDom:syll_nuc(x))) $\wedge$
  $y = \mathsf{F}$:$_\mathsf{D}$:MDom^3($\mathsf{F_M}$:PDom:syll_nuc(x))

A more generalized predicate can check if a syllable is part of any type of MNode by using long-distance information. The predicate **gen_MDom**(x) uses MSO logic to compute long-distance morphological dominance (Chapter 4: §4.5.1). With this predicate, we know if a syllable x is part of some MNode y by checking if the syllable's nucleus is generally dominated by y, via the predicate **syll_of_MNode**(x, y).

(314)   *MSO user-defined predicates for finding the syllables of some MNode using long-distance information*

- **syll_of_MNode**$(x, y) \stackrel{\text{def}}{=}$ syll$(x) \wedge$ MNode$(y) \wedge$
  $\exists v[$seg$(v) \wedge$
  gen_MDom$(y, v) \wedge$ PDom:syll_nuc$(x, v)]$

For cyclic prosody, I do *not need* the more powerful MSO predicate that uses long-distance information. With the simpler QF predicate in (312), the PStem can prosodically dominate its MStem's syllables via the output function ϕPDom:PStem_syll(x^2, y^1). Again, the SETTINGS label must be added as a condition. The output function ϕPStem(x^2) causes some PStem x^2 in Copy 2 to prosodically dominate a syllable y^1 in Copy 1 if there exists an MStem w^1 in Copy 1 which is mapped to the PStem x^2 and crucially the MStem's input correspondent w generally dominates the input correspondent of the syllable y. This function can be made QF.[16]

(315)   a. *FO output function for creating prosodic dominance between a new PStem and its syllables*

- ϕPDom:PStem_syll$(x^2, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$
  ϕPStem$(x^2) \wedge \phi$syll$(y^1) \wedge$
  $\exists w[\phi$MStem$(w^1) \wedge \phi$Match:stem$(w^1, x^2) \wedge$
  syll_of_MStem$(y, w)]$

b. *QF output function for creating prosodic dominance between a new PStem and its syllables*

- ϕPDom:PStem_syll$(x^2, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:nonrecursive(SETTINGS) $\wedge$
  ϕPStem$(x^2) \wedge \phi$syll$(y^1) \wedge$
  ϕMStem$(\phi\mathsf{F}_R$:Match:stem$(x^2)) \wedge$
  syll_of_MStem$(y, \phi\mathsf{F}_R$:Match:stem$(x^2)^0)$

This completes the prosodic mapping of an MStem to a PStem. The entire computation is QF. Any potential non-locality from the morphology was factorized into the SETTINGS. In Chapter 6: §6.5, I show

---

[16]The unary function $\phi\mathsf{F}_R$:Match:stem(x^2) returns a node w^1 in Copy 1. To retrieve its input correspondent w, I use the superscript ^{0}.

how other areas of the prosodic phonology is likewise QF. in Chapter 8: §8.3, I show that the prosody is still QF if we do not use the SETTINGS factorization. Non-local prosody arises from postcylic parsing.

5.5   Phonological rule domains: Stress assignment in the stem-level cophonology

For *amusin*, the PStem has been parsed. The next step is to apply the right cophonology. Because the topmost MNode is an MStem, we should apply the stem-level cophonology. In Armenian, the relevant stem-level rules are stress assignment and destressed high vowel reduction. Because the input *amusin* is unstressed, vowel reduction vacuously applies. I only define stress assignment here: *amusín*. Reduction is defined in Chapter 6: §6.6.1. I treat both stress and reduction as logical transductions such that stress feeds reduction.

This simple process of final stress is complicated by the presence of both phonological and morphological triggers for the input *amusin*. The phonological triggers are all local to the phonological target *i*. Specifically, the vowel *i* must be the rightmost full vowel (316).
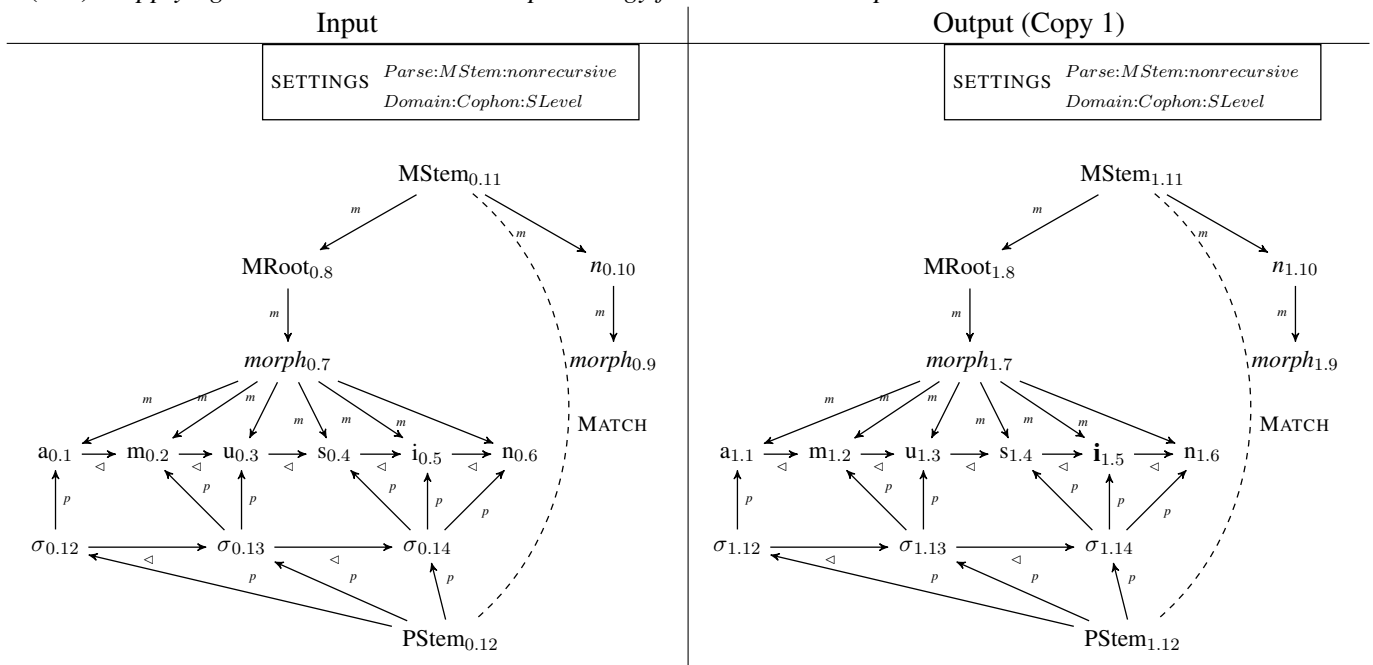
(316)   a.   amusín   'husband'
       b.   amusín-ə   'husband-DEF'

I illustrate the desired output below. The stressed vowel is enlarged and in bold.

(317)   *Applying stress in the stem-level cophonology for* amusin
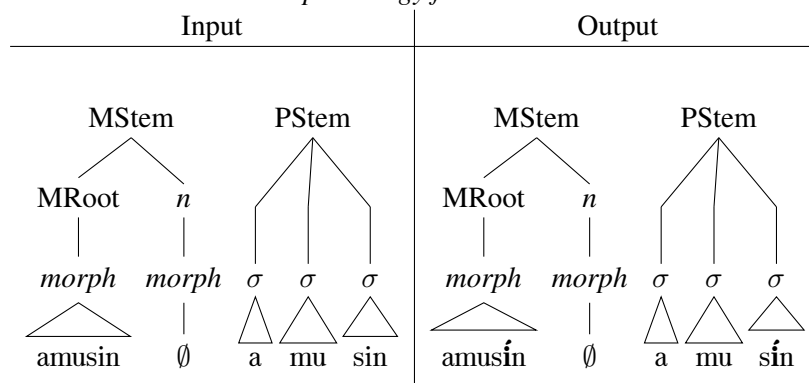
In the more explicit representation, we see that the cophonology is determined by the SETTINGS of the derivation.

(318) *Applying stress in the stem-level cophonology for* amusin *with explicit* SETTINGS

Input	Output (Copy 1)

SETTINGS $\quad$ *Parse:MStem:nonrecursive* / *Domain:Cophon:SLevel*

$\text{MStem}_{0.11}$ — m → $\text{MRoot}_{0.8}$, m → $n_{0.10}$

$\text{MRoot}_{0.8}$ — m → $morph_{0.7}$

$n_{0.10}$ — m → $morph_{0.9}$

$morph_{0.7}$ — m → $a_{0.1}$, $m_{0.2}$, $u_{0.3}$, $s_{0.4}$, $i_{0.5}$, $n_{0.6}$

$morph_{0.9}$ — MATCH

$\sigma_{0.12}$, $\sigma_{0.13}$, $\sigma_{0.14}$

$\text{PStem}_{0.12}$

SETTINGS $\quad$ *Parse:MStem:nonrecursive* / *Domain:Cophon:SLevel*

$\text{MStem}_{1.11}$ — m → $\text{MRoot}_{1.8}$, m → $n_{1.10}$

$\text{MRoot}_{1.8}$ — m → $morph_{1.7}$

$n_{1.10}$ — m → $morph_{1.9}$

$morph_{1.7}$ — m → $a_{1.1}$, $m_{1.2}$, $u_{1.3}$, $s_{1.4}$, $\mathbf{i}_{1.5}$, $n_{1.6}$

$morph_{1.9}$ — MATCH

$\sigma_{1.12}$, $\sigma_{1.13}$, $\sigma_{1.14}$

$\text{PStem}_{1.12}$

The morphological trigger for final stress is however long-distant. The trigger is the topmost MNode which can be at any distance from the rightmost vowel *i*. Computationally, finding the local phonological triggers is QF while finding the non-local morphological triggers is not QF. In order to factorize the computation, the non-locality of the morphological trigger is encapsulated into the constant SETTINGS as a unary label for the right SLevel cophonology.

With this in mind, stress assignment is a transduction with a copy set of size 1. The relevant unary labels are below. The labels stressed(x) and destressed(x) are new; they are defined for either vowels or syllables. Their interpretation is straightforward.

(319) *Unary labels involved in stress assignment*
- stressed:vowel(x): x is a stressed vowel
- stressed:syll(x): x is a stressed syllable
- destressed:vowel(x): x is a destressed vowel
- destressed:syll(x): x is a destressed syllable

In Armenian, stress assignment applies in multiple cophonologies: the stem-level, word-level, and PStem-level cophonology. The relevant morphophonological domain for stress-assignment is encapsulated into the user-defined predicate **StressDomain**(SETTINGS). This predicate is TRUE if the SETTINGS of the derivation has the domain label of the SLevel, WLevel, or PStem cophonology. The right SETTINGS label was determined earlier in the derivation in the SETTINGS step (§5.3.2).

(320) *QF user-defined predicate for morphophonological domain of stress assignment*
- **StressDomain**(SETTINGS) $\overset{\text{def}}{=}$ Domain:Cophon:SLevel(SETTINGS) $\lor$
  Domain:Cophon:WLevel(SETTINGS) $\lor$
  Domain:Cophon:PStem(SETTINGS)

In stress assignment, all binary relations are faithfully outputted in Copy 1. As for the unary labels, all labels are faithfully outputted except for stress-based labels. Note that these output functions check if the current cophonology *can* trigger stress assignment via the SETTINGS constant.

(321) *QF output functions for faithfully outputted binary relations in Copy 1 for stress assignment*
- For every relation rel $\in R$:
  $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \textbf{StressDomain}(\text{SETTINGS}) \wedge \mathsf{rel}(x, y)$
- For every label lab $\in L - \{$stressed:vowel, stressed:syll, destressed:vowel, destressed:syll$\}$:
  $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \textbf{StressDomain}(\text{SETTINGS}) \wedge \mathsf{lab}(x)$

Stress-assignment uses the set of user-defined predicates below to find the relevant positions and types of syllables/vowels. The predicates in (322a-i) find the final and penultimate syllables. The predicates in (322b-i) check if some vowel is a full vowel or not, if a syllable has a full-vowel nucleus, if a syllable has a schwa nucleus, or if a syllable precedes another syllable which has a schwa nucleus. Their definitions are straightforward. They can all be made QF.

(322) a. i. *FO user-defined predicates for finding relevant positions for stress assignment*
- $\textbf{final:syll}(x) \stackrel{\text{def}}{=} \neg\exists y[\mathsf{succ:syll}(x, y)]$
- $\textbf{penultimate:syll}(x) \stackrel{\text{def}}{=} \exists y[\mathsf{succ:syll}(x, y) \wedge \textbf{final:syll}(y)]$

      ii. *QF user-defined predicates for finding relevant positions for stress assignment*
- $\textbf{final:syll}(x) \stackrel{\text{def}}{=} \mathsf{F_L:succ:syll}(x) = \text{NULL}$
- $\textbf{penultimate:syll}(x) \stackrel{\text{def}}{=} \textbf{final:syll}(\mathsf{F_L:succ:syll}(x))$

   b. i. *FO user-defined predicates for finding syllables with full vowels vs. with schwas*
- $\textbf{vowel:full}(x) \stackrel{\text{def}}{=} \mathsf{vowel}(x) \wedge \neg\mathsf{schwa}(x)$
- $\textbf{syll:full}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \exists y[\textbf{vowel:full}(y) \wedge \mathsf{PDom:syll\_nuc}(x, y)]$
- $\textbf{syll:schwa}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \exists y[\mathsf{schwa}(y) \wedge \mathsf{PDom:syll\_nuc}(x, y)]$
- $\textbf{syll:pre-schwa}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \exists y[\mathsf{succ:syll}(x, y) \wedge \textbf{syll:schwa}(y)]$

      ii. *QF user-defined predicates for finding syllables with full vowels vs. with schwas*
- $\textbf{vowel:full}(x) \stackrel{\text{def}}{=} \mathsf{vowel}(x) \wedge \neg\mathsf{schwa}(x)$
- $\textbf{syll:full}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \textbf{vowel:full}(\mathsf{F_M:PDom:syll\_nuc}(x))$
- $\textbf{syll:schwa}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{schwa}(\mathsf{F_M:PDom:syll\_nuc}(x))$
- $\textbf{syll:pre-schwa}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \textbf{syll:schwa}(\mathsf{F_R:succ:syll}(x))$

With the above user-defined predicates, the predicates below pick which syllable and vowel *should* get stress. They specifically pick the rightmost syllable with a full vowel. These predicates can be made QF.

(323) a. *QF user-defined predicates for finding the rightmost full syllable*
- $\textbf{rightmost\_full\_syll}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \textbf{syll:full}(x) \wedge$
  $[\textbf{final:syll}(x) \vee [\textbf{penultimate:syll}(x) \wedge \textbf{syll:pre-schwa}(x)]]$

   b. *FO user-defined predicates for finding the rightmost full vowel*
- $\textbf{rightmost\_full\_vowel}(x) \stackrel{\text{def}}{=} \mathsf{vowel}(x) \wedge$
  $\exists y[\mathsf{PDom:syll\_nuc}(y, x) \wedge \textbf{rightmost\_full\_syll}(y)]$

   c. *QF user-defined predicates for finding the rightmost full vowel*
- **rightmost_full_vowel**$(x) \stackrel{\text{def}}{=}$ vowel$(x) \wedge$ **rightmost_full_syll**$(\mathsf{F_D}$:PDom:syll_nuc$(x))$

For *amusin*, the predicate **rightmost_full_syll**(x) picks out the input syllable $\sigma_{0.5}$ headed by the vowel $i_{0.5}$. This is because $\sigma_{0.5}$ is a syllable with a full vowel nucleus, and it is the final syllable.[17] The predicate **rightmost_full_vowel**(x) picks out the vowel $i_{0.5}$ because it is a vowel and is the nucleus of the syllable $y = \sigma_{0.5}$ which is the rightmost full-headed syllable.

With the predicates in place, we can now define output functions that will stress the correct syllable $\sigma_{0.5}$ and vowel $i_{0.5}$. In the case of syllables, the output function ϕstressed:syll(x^1) will stress some syllable x^1 in Copy 1 iff we have the right cophonology for stress, the input correspondent of x^1 is a syllable x, and x is the rightmost full syllable. Vowels in turn get stressed via ϕstressed:vowel(x^1) which is analogously defined.

(324) *QF output functions for assigning stress in Copy 1*
- ϕstressed:syll$(x^1) \stackrel{\text{def}}{=}$ **StressDomain**(SETTINGS) $\wedge$
                syll$(x) \wedge$ **rightmost_full_syll**(x)
- ϕstressed:vowel$(x^1) \stackrel{\text{def}}{=}$ **StressDomain**(SETTINGS) $\wedge$
                vowel$(x) \wedge$ **rightmost_full_vowel**(x)

As a last step, the label for *destresed* vowels and syllables is updated by checking if some item is stressed in the input but not in the output. Their interpretation is straightforward. For now, this label is unneeded. It will be useful once we model destressed vowel reduction in Chapter 6: §6.6.1.

(325) *QF output functions for destressing items in Copy 1*
- ϕdestressed:syll$(x^1) \stackrel{\text{def}}{=}$ stressed:syll$(x) \wedge \neg\phi$stressed:syll(x^1)
- ϕdestressed:vowel$(x^1) \stackrel{\text{def}}{=}$ stressed:vowel$(x) \wedge \neg\phi$stressed:vowel(x^1)

Later in the derivation for a simplex word like *amusin*, covert word-level morphology must be added. The larger MWord is then parsed as a PWord and the word-level cophonology applies with an extra round of final stress. These three changes are all vacuous so I don't formalize them here. I formalize them in Chapter 6: §6.2.1.2 in the context of overt inflection.

I emphasize that all the output functions above were QF *except* for the predicate **StressDoman**(SETTINGS). In the ultimate definition of this predicate, we need to determine if the SETTINGS constant had the right cophonology labels. Getting this information for the SETTINGS is not computationally local or QF. I explain the significance of this difference next.

---

[17]If the word had a final schwa like *amusín-ə*, then this predicate picks the penultimate syllable because it satisfies the disjunct [**penultimate:syll**$(x) \wedge$ **syll:pre-schwa**(x)] by being the penultimate syllable which precedes the final schwa-headed syllable
.

5.6   Significance and role of the SETTINGS

The previous section illustrated how I use the SETTINGS of the derivation to trigger the right prosodic parses and right phonological rule domains. On the one hand, we can in principle encode any type of global information into the SETTINGS. But on the other hand, in practice, we only ever encode the following types of global information:

1. What is the topmost MNode (and its properties)?
2. What dialect are we in?
3. Is the input an exception to any specific process?

I used the first property to determine what to prosodically parse in the form of Parse labels, and what rules to apply in the form of Domain labels. In Chapter 6: §6.6.3.1, I use the SETTINGS to encode if the input is being processed by one dialectal grammar or another, i.e., if we apply Eastern Armenian rules or Western Armenian rules. I do formalize any lexeme-specific variation or exceptions, but this can be done with the SETTINGS.

In this section, I discuss how the SETTINGS constant allows us to factorize the computation into local and non-local triggers. This has a practical utility in making grammar unification be easier to read.

5.6.1   Non-local triggers in rule domains

The utility of the SETTINGS constant comes from how it factorizes the triggers of prosodic and phonological processes into their local and non-local triggers. I first explain this schematically, and then I illustrate with logical formulas. The main idea is that the SETTINGS constant lets us factorize the derivation of a cycle into smaller components. This decomposition reduces the amount of non-local information that each module will need (cf. the role of factorization in computational syntax: Morawietz 2003).

This chapter divided a cycle into four components: Morphology, Examination/Settings, Prosody, and Phonology. The Morphology computes transductions or functions which are mostly local (QF), though some non-local functions exist depending on one's analysis (Chapter 7). In this 4-way factorization, the Examination or Settings step uses non-local computation (FO logic) in order to find the properties of the topmost node in the tree. Furthermore, the Prosody and Phonology only use QF logic or local computation. To apply a stem-level phonological process like final stress, we need to access morphological information that we're in the right stem-level cophonology. This information is potentially non-local and it is a property of the topmost node in the morphological tree. However the Examination step created a SETTINGS constant which already did the non-local computation of getting this information. Instead of getting this long-distance information from the Morphology, the Prosody and Phonology can locally accessible this information by examining the SETTINGS constant.

(326)   *Generative capacity of the different components with* SETTINGS *encapsulation*
Input ⟶ QF/FO Morphology → FO Settings ⟶ QF Prosody ⟶ QF Phonology ⟶ Output

In contrast, without an Examination or Settings step, a cycle consists of only three stages or modules. The Morphology computes QF or FO transductions. But in this 3-way factorization, the Prosody and Phonology

need to examine the properties of the topmost node in the morphological tree, which can be at any distance from the relevant prosodic nodes and phonological segments. As I explain in this section, this simple 3-way factorization makes the Phonology require at least FO logic.

(327)   *Generative capacity of the different components without* SETTINGS *encapsulation*
         Input $\longrightarrow$ QF/FO Morphology $\rightarrow$ QF/FO Prosody $\longrightarrow$ FO Phonology $\longrightarrow$ Output

The Prosody is more nuanced. As I explain later in Chapter 8: §8.3, we can use certain strategies in formalizing the Prosody so that the Prosody is still locally-computible with QF logic in this 3-way factorization. For easier illustration, this section focuses on the Phonology. Specifically, consider applying final stress in *amusín*. In the 4-way factorization, the SETTINGS allowed us to separate the non-local morphological trigger of stress assignment from its local phonological triggers. I repeat below the relevant output function.

(328)   *QF output function which places stress on the rightmost full vowel*
         • $\phi\texttt{stressed:vowel}(x^1) \overset{\text{def}}{=} \textbf{StressDomain}(\text{SETTINGS}) \wedge$
                                  $\text{vowel}(x) \wedge \textbf{rightmost\_full\_vowel}(x)$

This output function has two main conditions: $\textbf{StressDomain}(\text{SETTINGS})$ and $\textbf{rightmost\_full\_vowel}(x)$. The predicate $\textbf{rightmost\_full\_vowel}(x)$ encodes the phonological trigger for stress and is computationally local: *Is the vowel the rightmost full vowel*. In contrast, the predicate $\textbf{StressDomain}(\text{SETTINGS})$ encodes the morphological trigger for stress, i.e., that the topmost MNode is an MStem and that we should apply the stem-level cophonology. This predicate examines the SETTINGS and checks if the SETTINGS had the relevant cophonology label, e.g. Domain:Cophon:SLevel(SETTINGS).

(329)   *QF user-defined predicate for morphophonological domain of stress assignment*
         • $\textbf{StressDomain}(\text{SETTINGS}) \overset{\text{def}}{=}$ Domain:Cophon:SLevel(SETTINGS) $\vee$
                                  Domain:Cophon:WLevel(SETTINGS) $\vee$
                                  Domain:Cophon:PStem(SETTINGS)

At this stage of the derivation, the predicate $\textbf{StressDomain}(\text{SETTINGS})$ can be locally computed because the SETTINGS already has the relevant cophonology labels: Domain:Cophon:SLevel(SETTINGS). Because the SETTINGS is a constant, we do not need a quantifier to access the SETTINGS' properties.

However, the origins of this label Domain:Cophon:SLevel is non-local. It was generated earlier in the SETTINGS examination stage (§5.3) via the output function below, repeated from (281). This specific output function uses FO logic to do so. To generate this label on the SETTINGS, we examined the entire input and determined that the topmost MNode was an MStem which could trigger the stem-level cophonology.

(330)   *FO output functions for updating the* SETTINGS *with the stem-level cophonology*
         • $\phi\texttt{Domain:Cophon:SLevel}(\text{SETTINGS}^1) \overset{\text{def}}{=} \exists x[\textbf{MTopmost}(x) \wedge \text{Cophon:SLevel}(x)]$

In contrast in the 3-way factorization, there is no SETTINGS constant. The predicate Domain:Cophon:SLevel(SETTINGS) must be replaced by a predicate that uses an existential quantifier to compute this long-distance information.

Stress assignment then uses this predicate and is now a non-local process, i.e., it is now FO and no longer QF.

(331)   *Non-local computation in stress assignment*

    a. *FO logical statement for the non-local trigger of stress assignment*

       • **StressDomain** $\overset{\text{def}}{=} \exists x[\textbf{MTopmost}(x)\wedge$

$[\text{Cophon:SLevel}(x) \vee \text{Cophon:WLevel}(x) \vee \text{Cophon:PStem}(x)]$

    b. *FO output function which non-locally places stress on the rightmost full vowel*

       • $\phi\texttt{stressed:vowel}(x^1) \overset{\text{def}}{=} \textbf{StressDomain} \wedge$

$\text{vowel}(x) \wedge \textbf{rightmost\_full\_vowel}(x)$

In sum, the SETTINGS constant serves to factorize the local and non-local triggers of phonological processes in the morphology-phonology interface. Without the SETTINGS, entire phonological processes are computationally non-local. In Chapter 8, I explain that is non-locally can be partially removed in morpheme-based definitions for cophonologies. I likewise show that prosodic processes are still local and QF without the SETTINGS; non-locality arises from post-cyclic parsing.

5.6.2   Utility of SETTINGS in grammar unification

In addition to helping with factorizing the triggers of prosodic/phonological processes, the SETTINGS constant makes it easier to write logical formulas in a way that can be later unified together. Readers should reread §4.3.5 before continuing here.

To illustrate, assume a simple version of Armenian called Armenian-Prime which has a simple version of high vowel reduction: all high vowels reduce to schwa in the stem-level /amusin/→[aməsən], but not in the word-level: /amusin-ni/→//aməsən//→[aməsən-ni] where *-ni* is a word-level suffix. The table below omits the prosody.

(332)   *Derivation table for Armenian-Prime words* aməsən *and* aməsən-ni

Input			/amusin -$\emptyset_S$/	/amusin -$\emptyset_S$ -ni$_W$/
Cycle 1	MORPHO	Spell-out -$\emptyset$	amusin-$\emptyset$	amusin-$\emptyset$
	EXAMINE		*What should we parse and apply?*	
	PHONO	*SLevel*		
		reduction	aməsən	aməsən
Cycle 2	MORPHO	Spell-out *-ni*		aməsən-ni
	EXAMINE		*What should we parse and apply?*	
	PHONO	*WLevel*		
		No reduction		
Output			aməsən	aməsən-ni

For the stem-level cophonolgy in *aməsən*, the formula in (333a) assigns the stem-level cophonology to the SETTINGS at the examination step. Later in the phonology step, the output function $\phi\texttt{schwa}(x^1)$ is defined to change all high vowels and schwas to schwas as long as the stem-level cophonology is active.

(333) *Deriving schwas in stem-level phonology of Armenian-Prime*

    a. ϕDomain:Cophon:SLevel(SETTINGS^1) $\overset{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge$ Cophon:SLevel$(x)]$

    b. ϕschwa$(x^1) \overset{\text{def}}{=}$ Domain:Cophon:SLevel(SETTINGS) $\wedge$ [schwa$(x) \vee$ high$(x)]$

In contrast in the word-level phonology for *amǝsǝn-ni*, the formula in (334a) assigns the word-level cophonology to the SETTINGS at the examination step. Later, the output function ϕschwa(x^1) is now defined to change only schwas to schwas as long as the word-level cophonology is active.

(334) *Deriving schwas in word-level phonology of Armenian-Prime*

    a. ϕDomain:Cophon:WLevel(SETTINGS^1) $\overset{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge$ Cophon:WLevel$(x)]$

    b. ϕschwa$(x^1) \overset{\text{def}}{=}$ Domain:Cophon:WLevel(SETTINGS) $\wedge$ schwa(x)

In order to allow the same grammar to contain the two definitions of ϕschwa(x^1), the two versions of ϕschwa(x^1) must be combined.

(335) *Unified derivation of schwas in Armenian-Prime*

    a. ϕschwa$(x^1) \overset{\text{def}}{=}$ [Domain:Cophon:SLevel(SETTINGS) $\wedge$ [schwa$(x) \vee$ high$(x)]]\vee$
                 [Domain:Cophon:WLevel(SETTINGS) $\wedge$ schwa$(x)]$

The use of a SETTINGS constant allows us to better see that the unified function ϕschwa(x^1) encodes two different types of processes, one active in the stem-level cophonology and one in the word-level cophonology.

5.7 Conclusion

In this chapter, I formalized each of the four explicit and implicit steps in a cyclic interactionist derivation: the morphology, SETTINGS examination, prosody, and phonological rule domains. The morphology, prosody, and phonology were defined with respect to generating a simple stem *amusín*. These three processes were shown to be computationally local with QF logic.

However, the intermediate step of examining the settings of the derivation is computationally complicated. It requires using non-local information in the form of FO logic in order to find the right morphological triggers for the prosody and phonology, specifically the properties of the topmost morphological node. I encapsulated this non-local information into a constant called the SETTINGS. By storing the relevant non-local information into one specific part of the computation, we can access this information whenever needed and without recomputing the entire input. This encapsulation thus lets us separate the non-local morphological triggers from the local phonological triggers. Besides factorization, the encapsulation helps in creating a unified grammar later on.

In the next chapter, I go through a barrage of processes in the morphology-phonology interface in Armenian. By using a cyclic and interactionist model with SETTINGS encapsulation, I show that these processes are also computationally local and QF.

Chapter 6

Local generation of complex words

6.1 Overview

The previous chapter established the crux of formalization for the morphology-phonology interface, with each step illustrated with the simplex free-standing root *amusin* (336a). This chapter further illustrates this formalization by showing the logical computation of morphologically complex words, including derivatives, inflected items, and compounds. Some examples are shown in (336) below.

(336)	a.	Simple root		amusín	'husband'
	b.	Derivative		amusn-agán	'marital'
	c.	Inflected item	V-initial	amusn-óv	'husband-INST' (Eastern Arm.)
	d.			amusin-óv	'husband-INST' (Western Arm.)
	e.		C-initial	amusin-nér	'husband-PL'
	f.	Compound	Endocentric	t͡ʃúr + pós	'water + hole'
				t͡ʃər-a-pós	'water-hole'
	g.		Exocentric	t͡ʃúr + kújn	'water + color'
				t͡ʃər-a-kújn	'water-colored'

In a cyclic interactionist model (Chapter 5), each of these complex items is derived from a simpler base, e.g., *amusín* 'husband' or *t͡ʃúr* 'water', through a recursive sequence of morphological, prosodic, and phonological processes. The base has already undergone its own cycle of morphological, prosodic, and phonological processes in order to surface as some stressed item. The output of the first cycle is fed to the second cycle, where we generate generate new morphological, prosodic, and phonological structure. This chapter formalizes these new structures.

In the Introduction chapter, I briefly explained how the above 5 items (336) are derived in a cyclic framework. For example, for the derivative *amusn-agan* 'marital', it undergoes two cycles. The first cycle generates the base *amusín* through a sequence of morphological, prosodic, and phonological processes. In an intermediate stage in the first cycle, we examine the derivation's settings in order to determine what to parse and what rules to apply. In the second cycle, the **Morphology** adds an overt derivational suffix *-agan*.

With this new suffix, we **examine** the settings and determine that we have to parse a larger morphological stem and apply the stem-level cophonology. With this information in hand, the **Prosody** will resyllabify the derivative and restructure the existing PStem. The **Phonology** stage will apply the stem-level rules of stress and reduction.

(337)   *Cyclic generation of a derivative* amusn-agan

			MStem$_a$ MStem$_n$ √ n a /amusin -∅ /-agan/
Input			/amusin -∅$_S$ -agan$_S$/
Cycle 1	MORPHO	Spell-out	/amusin-/
	EXAMINE		*What should we parse and apply?*
	PROSODY	Syllabify	amu.sin
		Map PStem	(amu.sin)$_s$
	PHONO	*SLevel*	
		Stress	(amu.sín)$_s$
		DHR	
Cycle 2	MORPHO	Spell-out	(a.mu.sín)$_s$ /-agan/
	EXAMINE		*What should we parse and apply?*
	PROSODY	Resyllabify	(a.mu.sí.n)$_s$-a.gan
		Restructure PStem	(a.mu.sí.n-a.gan)$_s$
	PHONO	*SLevel*	
		Stress	(a.mu.sǐ.n-a.gán)$_s$
		DHR	(a.mus.n-a.gán)$_s$
Output			amusn-agán

In this chapter, I formalize what morphological, prosodic, and phonological processes apply in complex words. Two results from the formalization are that 1) all these processes are computationally definable, and 2) the bulk of these processes require *local* computation, not global computation. Non-locality is restricted to **Examining** the settings in order to find non-local morphological triggers for the Prosody and Phonology. I first formalize the relevant morphological processes (§6.2), how to update the SETTINGS (§6.3), resyllabification (§6.4), prosodic mappings and adjustment (§6.5), and cophonological rule domains (§6.6). All of these morphological, prosodic, and phonological processes are computationally local *if* we assume an interactionist model that encapsulates relevant information from the morphology into a constant SETTINGS.

For clarity, note that I use double slashes //. . .// throughout this chapter in order to represent intermediate representations or intermediate outputs. These are generated between any transductions within a cycle, e.g., as output of the Morphology, Examination, resyllabification, etc.

6.2 Overt morphology and compounding

In Chapter 5: §5.2, I formalized the generation of morphological structure by using construction-specific morphological transductions. The same is done for generating overt suffixes (§6.2.1). Compounding (§6.2.2) is more intricate because it uses function currying.

6.2.1 Generating and linearizing overt morphological structure

Generating an overt suffix is not computationally more complex than generating a covert suffix. When added to the base *amusin*, the derivational suffix *-agan* creates a larger MStem representation //amusín-agan//; the high vowel is later reduced during the stem-level phonology: *amusn-agán*. Similarly, the inflectional suffix *-ov* forms an MWord: *amusin-ov*. Adding an overt suffix is computationally local with QF logic. I go through the derivational suffix first and briefly discuss inflection.

6.2.1.1 Local computation in overt derivational morphology

The derivational suffix *-agan* is an overt adjectivizing suffix which creates a morphological stem. I show the input and output below

(338)   *Input and output for adding an overt derivational suffix*
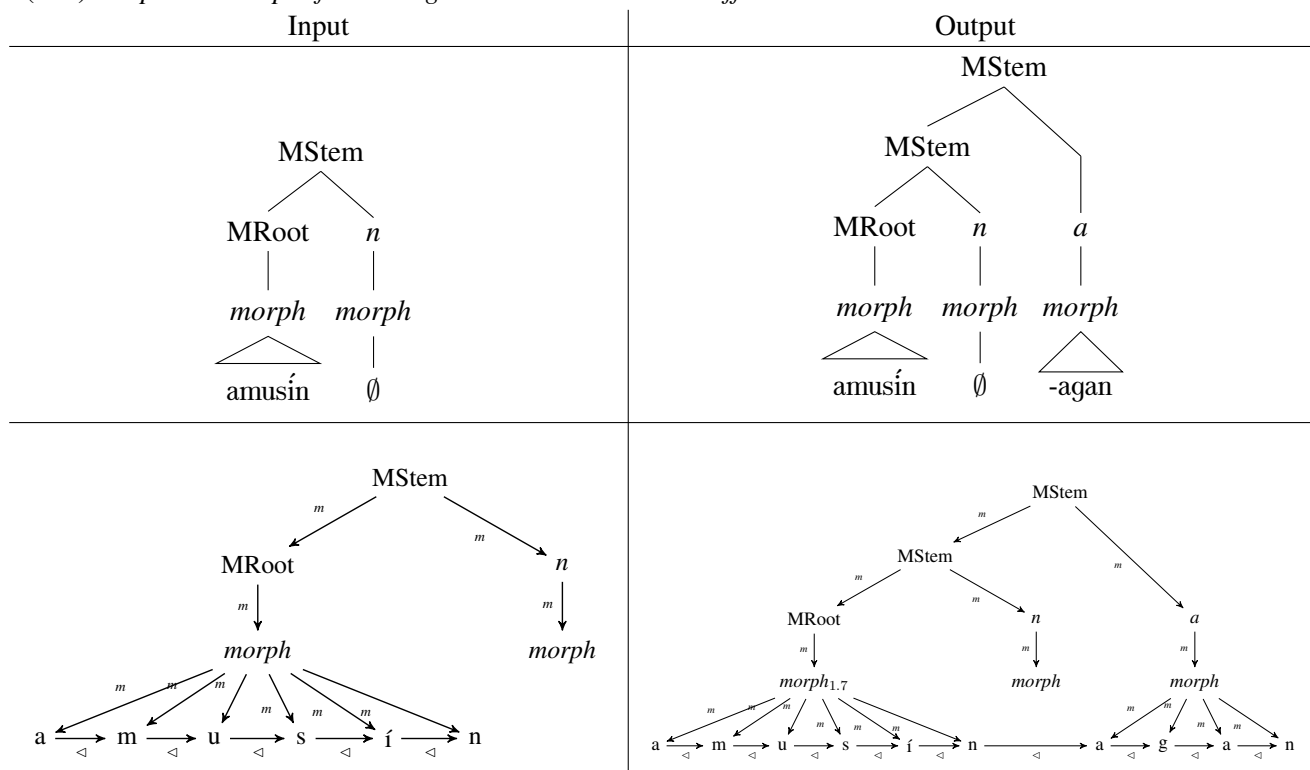

Adding this suffix can be thought of as occurring in the following steps.

1. Generate the...
   - 4 phonological segments: a,g,a,n
   - 3 morphological nodes: *morph*, *a*, and MStem
2. Internally linearize the affix's...
   - segments via immediate successor
   - morphology via morphological dominance
   - segments with the affix's morph via morphological dominance
3. Externally linearize the affix and base's...
   - segments via immediate successor: ...*n<a*...
   - morphology via morphological dominance: MStem < MStem

As with adding a zero suffix, I formalize each step below. I show that the process of simple suffixation is computationally local with QF logic. The above list is a basic template for constructing a morphological transduction for any affix. The relevant labels and directionality must be changed on an affix-by-affix basis. Since the number of morphological constructions (affixes) in a language is finite, this is feasible.

I repeat the input below. The input is the fully prosodified stem *amusín*. I show the indexes but omit the prosodic nodes and SETTINGS.

(339) *Input to generating the suffix* -agan
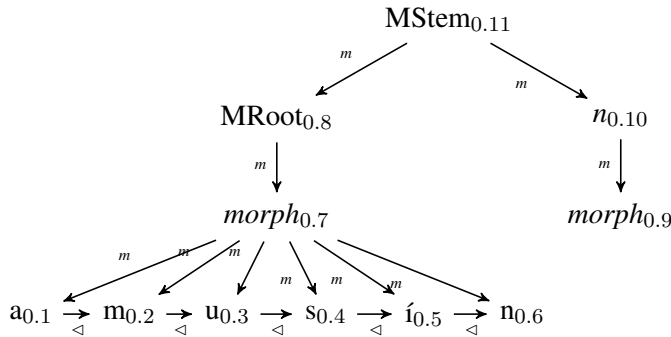
Because 7 new nodes must be generated, the transduction has a copy set of size 8. In Copy 1, the input is faithfully outputted.

(340) *QF output functions for vacuous identity in Copy 1*
   - For every lab $\in L$:
     $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \textsf{lab}(x)$
   - For every relation rel $\in R$:
     $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \textsf{rel}(x, y)$

Because this affix is a suffix, the affix's segments are output correspondents of the *final* segment in the input.[1] The final segment is picked via the predicate **final**:**seg**(x). Recall from Chapter 4: §4.4.2, that this

---

[1]If the affix were a prefix, then the segments would be output correspondents of the first segment in the input. The rationale of picking the input's final (initial) segment as the input correspondent for the affix's segment is to make affixation be an order-preserving transduction, as explained in Chapter 7: §7.5.

predicate can be made QF. Its meaning is straightforward. In Copies 2-5, the affix segments are generated as output correspondents for the final segment $n_{0.10}$.

(341) a. *FO user-defined predicate for finding the final segment*
- $\mathbf{final{:}seg}(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \neg\exists y[\mathsf{succ{:}seg}(x,y)]$

  b. *QF user-defined predicate for finding the final segment*
- $\mathbf{final{:}seg}(x) \stackrel{\text{def}}{=} \mathsf{F_L{:}succ{:}seg}(x) = \text{NULL}$

  c. *QF output functions for creating new segments for the suffix* -agan
- $\phi\mathsf{a}(x^2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x)$
- $\phi\mathsf{g}(x^3) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x)$
- $\phi\mathsf{a}(x^4) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x)$
- $\phi\mathsf{n}(x^5) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x)$

The new morphological nodes are defined in terms of the morphologically topmost node in the tree. The predicate $\mathbf{MTopmost}(x)$ below picks this topmost node. Recall from Chapter 5: §5.2 that this predicate can be made QF. In Copies 6-8, the new nodes are generated via the output functions below.

(342) a. *FO user-defined predicate for finding the morphologically topmost MNode*
- $\mathbf{MTopmost}(x) \stackrel{\text{def}}{=} \mathsf{MNode}(x) \wedge \neg\exists y[\mathsf{MDom}(y,x)]$

  b. *QF user-defined predicate for finding the morphologically topmost MNode*
- $\mathbf{MTopmost}(x) \stackrel{\text{def}}{=} \mathsf{MNode}(x) \wedge \mathsf{F_D{:}MDom}(x) = \text{NULL}$

  c. *QF output functions for creating new morphological nodes for the suffix* -agan
- $\phi\mathtt{morph}(x^6) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{adj}(x^7) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{MStem}(x^8) \stackrel{\text{def}}{=} \mathbf{MTopmost}(x)$

This is shown below. Note how the affix's segments and morphological nodes occupy different Copies. I have visually shuffled them to make the graph easier to read. For a new node, its index marks what copy it belongs to (2-8) and what is its input correspondent (the final segment $n_{0.6}$ or the topmost node $\text{MStem}_{0.11}$).

(343)   *Generating an overt derivational suffix in* /amusin-agan/ *– shuffled nodes*

$$\text{MStem}_{8.11}$$
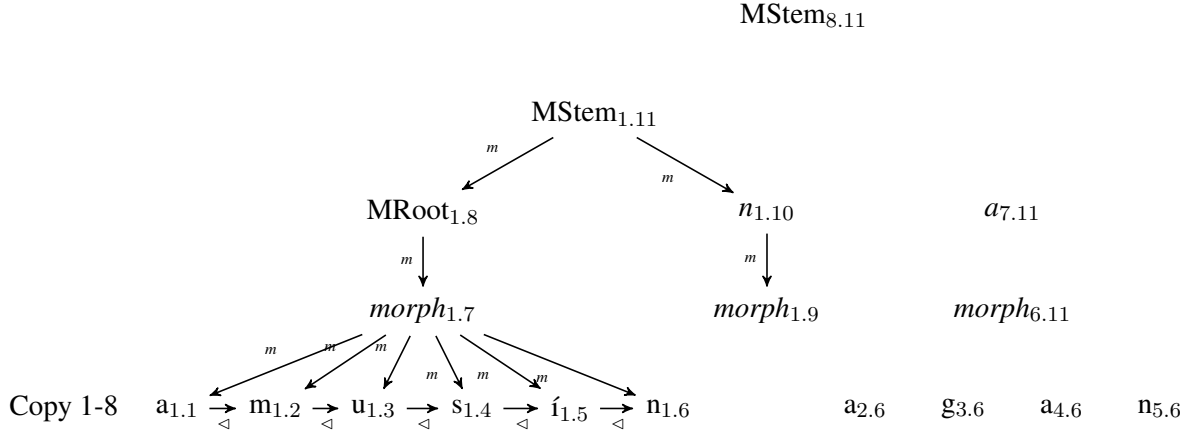
The output functions below internally linearize the affix's segments and the affix's morphological nodes.

(344)   a. *QF output functions for internally linearizing the segments of the suffix* -agan
   - $\phi\texttt{succ:seg}(x^2, y^3) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$
   - $\phi\texttt{succ:seg}(x^3, y^4) \overset{\text{def}}{=} \textbf{final:sg}(x) \wedge \textbf{final:seg}(y)$
   - $\phi\texttt{succ:seg}(x^4, y^5) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$

   b. *QF output functions for internally linearizing the MNodes of the suffix* -agan
   - $\phi\texttt{MDom}(x^8, y^7) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$
   - $\phi\texttt{MDom}(x^7, y^6) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

   c. *QF output functions for internally linearizing between the MNodes and segments of the suffix* -agan
   - $\phi\texttt{MDom}(x^6, y^2) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y)$
   - $\phi\texttt{MDom}(x^6, y^3) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y)$
   - $\phi\texttt{MDom}(x^6, y^4) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y)$
   - $\phi\texttt{MDom}(x^6, y^5) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y)$

The affix is externally linearized with the base via the output functions below. The function $\phi\texttt{succ:seg}(x^2, y^1)$ linearizes the affix's first segment $a_{2.6}$ with the base's final segment $n_{1.6}$ via immediate successor. They are picked because they are both output correspondents for the input's final segment $n_{0.6}$. Similarly, the function $\phi\texttt{MDom}(x^8, y^1)$ linearizes the affix's MStem$_{8.11}$ with the base's MStem$_{1.11}$ via morphological dominance because they are both output correspondents for the input's topmost MNode, MStem$_{0.11}$.

(345)   a. *QF output function for externally linearizing the suffix* -agan *with the base's final segment*
   - $\phi\texttt{succ:seg}(x^1, y^2) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$

   b. *QF output function for externally linearizing the suffix* -agan *with the base's MStem*
   - $\phi\texttt{MDom}(x^8, y^1) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

I illustrate linearization and the final output below. The main takeaway away is that all the output functions references only the predicates **MTopmost**(x) and **final**:**seg**(x). Because these predicates are QF, then the entire process is QF and computationally local.

(346) *Output of generating an overt derivational suffix in* $//\text{amusín-agan}//$
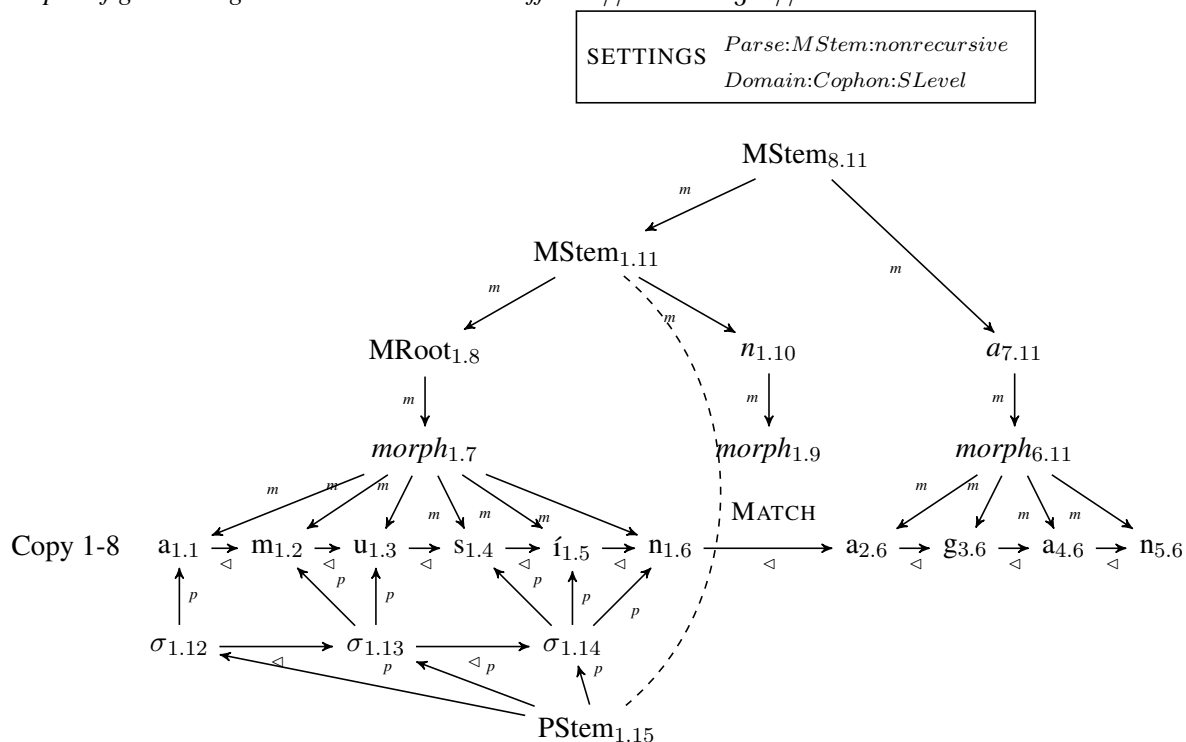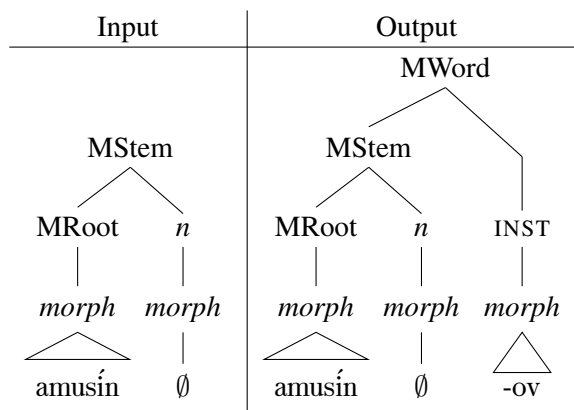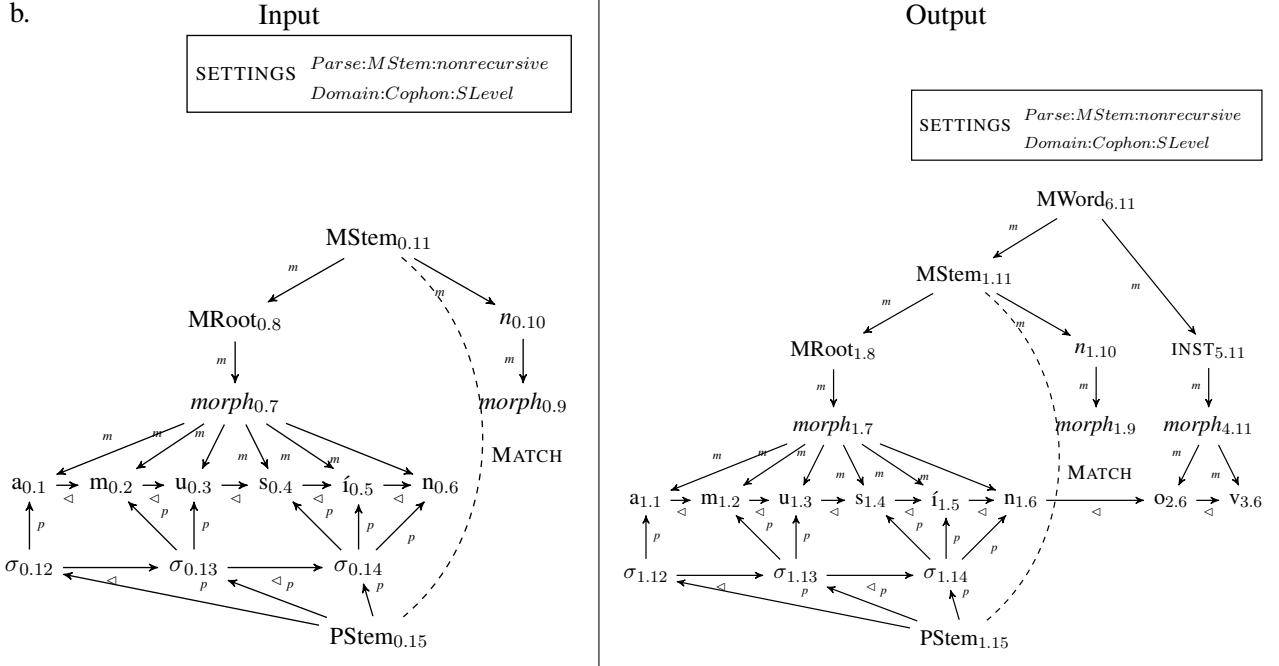

6.2.1.2 Local computation in overt inflectional morphology

Generating an inflectional suffix is just as complex, e.g., *-ov* in *amusin-ov*. Representationally, the main difference from derivational morphology is notation and labels.

(347) *Input and output for adding an overt inflectional suffix*

  a.

b.

Input

SETTINGS $Parse{:}MStem{:}nonrecursive$
            $Domain{:}Cophon{:}SLevel$

$\text{MStem}_{0.11}$

$\text{MRoot}_{0.8}$ $n_{0.10}$

$morph_{0.7}$ $morph_{0.9}$

MATCH

$a_{0.1} \rightarrow m_{0.2} \rightarrow u_{0.3} \rightarrow s_{0.4} \rightarrow í_{0.5} \rightarrow n_{0.6}$

$\sigma_{0.12} \longleftrightarrow \sigma_{0.13} \longleftrightarrow \sigma_{0.14}$

$\text{PStem}_{0.15}$

Output

SETTINGS $Parse{:}MStem{:}nonrecursive$
            $Domain{:}Cophon{:}SLevel$

$\text{MWord}_{6.11}$

$\text{MStem}_{1.11}$

$\text{MRoot}_{1.8}$ $n_{1.10}$ $\text{INST}_{5.11}$

$morph_{1.7}$ $morph_{1.9}$ $morph_{4.11}$

MATCH

$a_{1.1} \rightarrow m_{1.2} \rightarrow u_{1.3} \rightarrow s_{1.4} \rightarrow í_{1.5} \rightarrow n_{1.6} \longrightarrow o_{2.6} \rightarrow v_{3.6}$

$\sigma_{1.12} \longleftrightarrow \sigma_{1.13} \longleftrightarrow \sigma_{1.14}$

$\text{PStem}_{1.15}$

One difference is that we use a copy set of size 6 instead of 8, because the suffix has 2 segments *-ov* instead of 4 segments *-agan*. The labels of the segments and the morphological nodes are different, e.g., inflection generates an MWord, not an MStem.

(348)   *Different QF output functions for generating labels in an inflectional suffix* -ov 'INST'

   1.   Faithful output:
        - For every label lab $\in L$:
          $\phi\text{lab}(x^1) \stackrel{\text{def}}{=} \text{lab}(x)$
        - For every relation rel $\in R$:
          $\phi\text{rel}(x^1, y^1) \stackrel{\text{def}}{=} \text{rel}(x, y)$
   2.   Output the segments:
        - $\phi\text{o}(x^2) \stackrel{\text{def}}{=} \textbf{final:seg}(x)$
        - $\phi\text{v}(x^3) \stackrel{\text{def}}{=} \textbf{final:seg}(x)$
   3.   Output the new morphological nodes
        - $\phi\text{morph}(x^4) \stackrel{\text{def}}{=} \textbf{MTopmost}(x)$
        - $\phi\text{inst}(x^5) \stackrel{\text{def}}{=} \textbf{MTopmost}(x)$
        - $\phi\text{MWord}(x^6) \stackrel{\text{def}}{=} \textbf{MTopmost}(x)$

With the smaller copy set, we have fewer nodes to linearize.

(349)   *Different QF output functions for linearizing the binary relations in an inflectional suffix* -ov 'INST'

   1.   Internal linearization of segments:
        - $\phi\text{succ:seg}(x^2, y^3) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$
   2.   Internal linearization of morphology

- $\phi\texttt{MDom}(x^4, y^2) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{final{:}seg}(y)$
- $\phi\texttt{MDom}(x^4, y^3) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{final{:}seg}(y)$
- $\phi\texttt{MDom}(x^5, y^4) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$
- $\phi\texttt{MDom}(x^6, y^5) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

3. External linearization of morphology
   - $\phi\texttt{MDom}(x^6, y^1) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

4. External linearization of segments
   - $\phi\texttt{succ:seg}(x^1, y^2) \overset{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \mathbf{final{:}seg}(y)$

Interested readers can work out the individual steps of the generation themselves. They will find that the computation works and is computationally local, just like for the larger derivational suffix *-agan*.

6.2.2 Generating the morphology of a compound

Formalizing compound morphology is complicated. I first describe the basic structure of compounds (§6.2.2.1), and I introduce the relevant unary labels and binary relations (§6.2.2.2). I partition compound formation into three steps, each of which is formalized in its own section: Linking (§6.2.2.3), Concatenation (§6.2.2.4), and MStem formation (§6.2.2.5). The formalization focuses on endocentric compounds, but they can be analogously defined for exocentric compounds (§6.2.2.6). Again, the entire computation is local. The brunt of the work is done by the predicates $\mathbf{final{:}seg}(x)$ and $\mathbf{MTopmost}(x)$ which are QF-definable (see Chapter 4: §4.4.2 and Chapter 5: §5.2).

6.2.2.1 Morphology of compounds

In this section, I discuss two possible constituencies for compound morphology. One is based on simple concatenation, while the other on more abstract operations. I ultimately use the concatenation-based system.

In Armenian, the three most common types of compounds are nominal, possessive, and deverbal compounds. Nominal compounds are endocentric; possessive and deverbal compounds are exocentric. Recall from Chapter 1 that the plural suffix is *-er* for monosyllabic bases, *-ner* after polysyllabic bases. When the second stem of the compound is monosyllabic, exocentric compounds are transparently pluralized as polysyllabic bases with *-ner*, while endocentric compounds are paradoxically pluralized as monosyllabic bases with *-er*. I explain this paradox in terms of prosodic structure in §6.5.3. The takeaway for now is that endocentricity is an important property of compounds.

(350) *Types of compounds*

Endocentric Compound	Exocentric Compound	
NOMINAL COMPOUND	POSSESSIVE COMPOUND	DEVERBAL COMPOUND
t͡ʃúr + pós 'water + hole'	t͡ʃúr + kújn 'water + color'	t͡ʃúr + per-él 'water + to bring'
t͡ʃər-a-pós 'water-hole'	t͡ʃər-a-kújn 'water-colored'	t͡ʃər-a-pér 'water-bearer'
t͡ʃər-a-pos-ér 'water-holes'	t͡ʃər-a-kujn-nér 'water-colored (PL)'	t͡ʃər-a-per-nér 'water-bearers'

To model compound morphology, one approach is simple concatenation (Allen 1979; Selkirk 1982; Ackema and Neeleman 2004; Ralli 2012). In this framework, a nominal compound is just two nouns which are combined to form a larger noun: N + N → N. Other compound types are similarly defined.

(351)   *Compound morphology as simple concatenation*

Endocentric Compound	Exocentric Compound	
NOMINAL COMPOUND	POSSESSIVE COMPOUND	DEVERBAL COMPOUND
ant͡srev-a-t͡ʃúr 'rain-water'	t͡ʃar-a-sírd 'evil-hearted'	hat͡s-a-kórd͡z 'bread-maker'
$\begin{array}{c} N \\ \overbrace{N\ LV\ N} \\ \text{t͡ʃər -a- pós} \end{array}$	$\begin{array}{c} N \\ \overbrace{A\ LV\ N} \\ \text{t͡ʃər -a- kújn} \end{array}$	$\begin{array}{c} N \\ \overbrace{N\ LV\ V} \\ \text{t͡ʃər -a- pér} \end{array}$

Because I assume that parts-of-speech are separate into their own covert morpheme (Giegerich 1999; Marantz 2007), I need to a more refined concatenation-based approach to compounds. Compounds consist of two morphological stems *MStem$_L$* and *MStem$_R$* which are connected via a linking vowel LV. There's a special morphological node which is the concatenation of MStem$_L$, MStem$_R$, and the LV. This node is called the **morphological concatenation** or the MConc. The MConc takes a covert category suffix and forms a larger MStem called MStem$_C$. This MStem$_C$ can be endocentirc $_{En}$ or exocentric $_{Ex}$.

(352)   *Compound morphology as refined concatenation*

Endocentric Compound	Exocentric Compound	
NOMINAL COMPOUND	POSSESSIVE COMPOUND	DEVERBAL COMPOUND
ant͡srev-a-t͡ʃúr 'rain-water'	t͡ʃar-a-sírd 'evil-hearted'	hat͡s-a-kórd͡z 'bread-maker'
MStem$_{C,En}$	MStem$_{C,Ex}$	MStem$_{C,Ex}$

MStem$_R$ and MStem$_C$ are nouns in nominal compounds. In deverbal compounds, MStem$_R$ is a verbal root while MStem$_C$ is a noun or adjective; in possessive compounds, MStem$_R$ is a noun while MStem$_C$ is an adjective. With this representation, the meaning of the compound is not predictable, nor is its endocentricity. There are no covert nodes or geometric positions (complements, specifiers, etc.) that can encode the semantic relationship between the stems.

A more nuanced analysis is that a compound contains covert morphosyntactic structure which encodes its semantics (Ten Hacken 2009; Harley 2009). In this analysis, a nominal compound still consists of two stems

each with its own category affix (353a). But, STEM1 is now adjoined to STEM2. Adjunction is represented by making the label of STEM2 percolate up the tree (I use indexes on the *n*'s for disambiguation). The linking vowel is later added as a zero morph (not shown). Possessive compounds have a similar internal structure (353b). The difference is that an additional covert affix *a* is added on top of the tree.

(353)   *Compound morphology with articulated structure*

Endocentric Compound	Exocentric Compound		
NOMINAL COMPOUND	POSSESSIVE COMPOUND	DEVERBAL COMPOUND	
an͡tsrev-a-t͡ʃúr 'rain-water'	t͡ʃar-a-sírd 'evil-hearted'	ha͡ts-a-kórd͡z 'bread-maker'	
a.	b.	c.i	c.ii

Modeling a deverbal compound is more complicated because of its argument structure. If we assume that roots cannot select arguments, then a deverbal compound's structure looks isomorphic to a possessive compound (353c.i). The *n* node of STEM2 is replaced by *v*. The features on the node *v* can determine if the deverbal compound should be interpreted as transitive, intransitive, or passive. In contrast, if we assume that roots *can* select arguments (Harley 2014), then a deverbal compound can consist of a category-less root selecting an argument (353c.ii). This constituent is then nominalized. At no point is a verbal *v* affix inserted.

In this chapter, I represent compounds using refined concatenation. I have not found positive evidence for more enriched covert structure (cf. Padrosa Trias 2011). In any case, the analysis does not depend on the choice of morphological formalization.

6.2.2.2   Logical formalization for compounding

Compound formation is complicated and requires the use of some new types of unary labels. I introduce and define them all here. The labels MStem:Comp:Left is visualized by a subscript $_L$, MStem:Comp:Right by $_R$, MStem:Comp by $_C$, MStem:Comp:Endo by $_{En}$, and MStem:Comp:Endo by $_{Ex}$.

(354)   *Unary labels used in compounding*
- MConc(x): a morphological node x is the morphological concatenation of stems and a linking vowel
- LV_morpheme(x): a morpheme x is the linking vowel used in compounding
- MStem:Comp:Left(x): the MStem x is the left stem or MStem$_L$ of a compound
- MStem:Comp:Right(x): the MStem x is the right stem or MStem$_R$ of a compound
- MStem:Comp(x): the MStem x is a compound

- MStem:Comp:Endo(x): the MStem x is a hyponymic or endocentric compound
- MStem:Comp:Exo(x): the MStem x is a non-hyponymic or exocentric compound

Here I formalize only the formation of endocentric nominal and exocentric possessive compounds. Extending the formalization to other compound types is trivial.

Intuitively, compound formation is not like affixation. In affixation, the input is a single item (the base) while the output is the same base but 'larger'. In the case of compounds, the input is two separate items and the output is their combination, as illustrated with the nominal compound *tʃər-a-pos* 'water-hole'.

(355)   *Stems that make up a nominal endocentric compound* t͡ʃər-a-pos *'water-hole'*
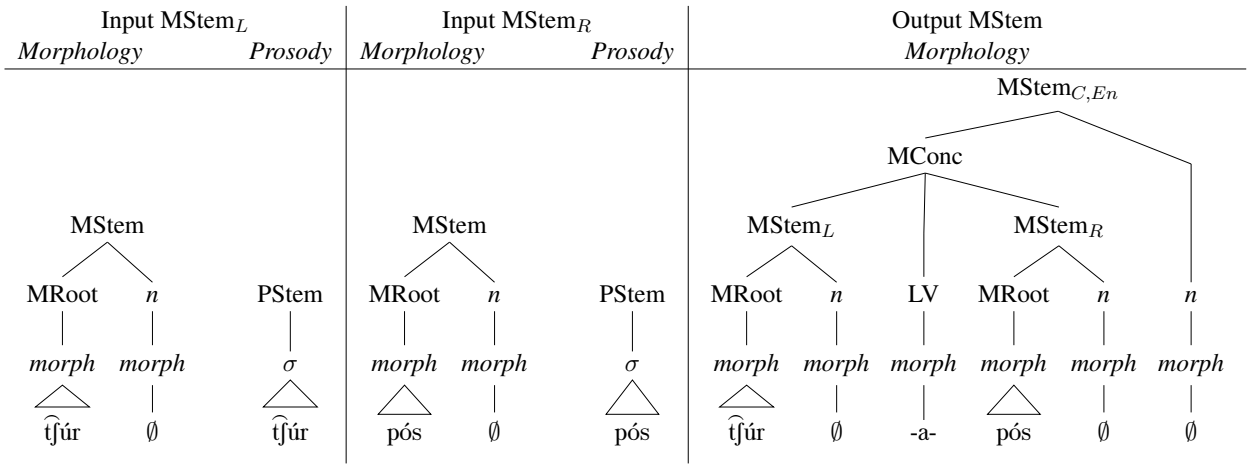
But in order to formalize the intuition that compound formation takes multiple inputs, we either need to 1) enrich our logical formalism to capture multi-input functions[2] or, 2) find some way to encode these multiple 'inputs' as distinct single-input functions. I pursue the second approach and it involves function currying. Let the input be MStem$_L$, e.g., t͡ʃúr 'water'. Given this input, compound formation consists of the following steps.

1. **Linking**: Concatenate MStem$_L$ with a linking vowel (LV) to form an MConc
2. **Concatenation**: Concatenate MStem$_L$ and the LV with MStem$_R$
3. **MStem formation**: Generate an MStem over the concatenation and determine if the compound is endocentric or not

Below, I formalize these 3 steps as separate transductions which apply in order.[3] Each of these steps is computationally local and uses QF logic. This because all the output functions reference only the simple predicates **MTopmost**(x) and **final:seg**(x) which are locally-computed and QF-definable.

---

[2]The first approach *could* be done using multi-tape finite-state transducers (MT-FSTs) (Kay 1987; Kiraz 2001). But it is unclear what the logical equivalent of MT-FSTs would be.

[3]I assume that the input stems are not compounds themselves. Without this assumption, many of the output functions here would need to be rewritten in order to faithfully preserve the morphology and prosody of old underlying compounds. I specify such reformulations when needed. Furthermore, the details of any of the three steps is lexeme specific. A given MStem$_L$ forms compounds with only some MStem$_R$'s. Some compounds don't use linking vowels. Knowing whether the compound will be interpreted as hyponymic or not (thus endocentric or exocentric) can be unpredictable. I abstract away from these idiosyncrasies here.
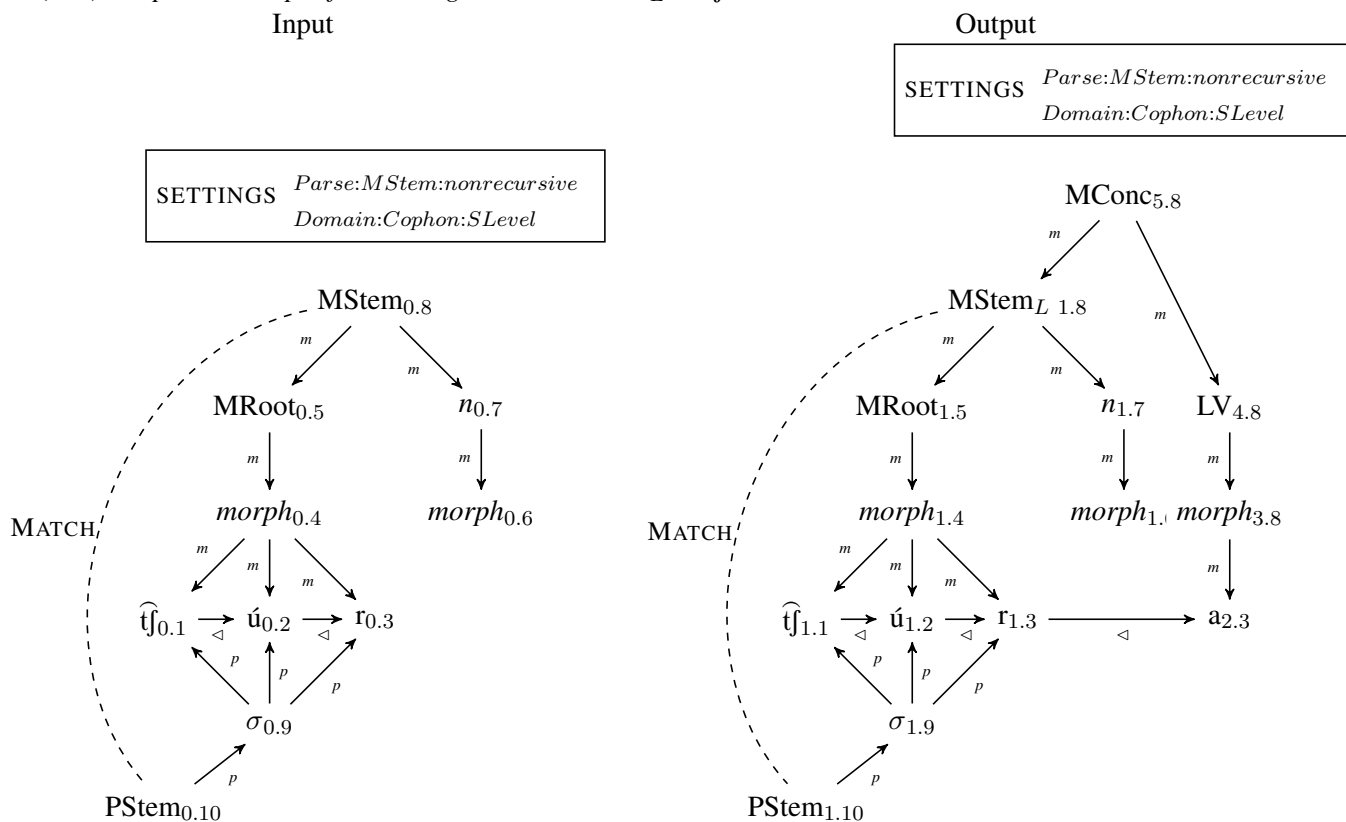
6.2.2.3 Formalizing Linking

In the Linking step, a linking vowel is added to MStem_L. Their concatenation is under the scope of a morphological concatenation node or an MConc. I show a simplified input and output.

(356) *Input and output for Linking*
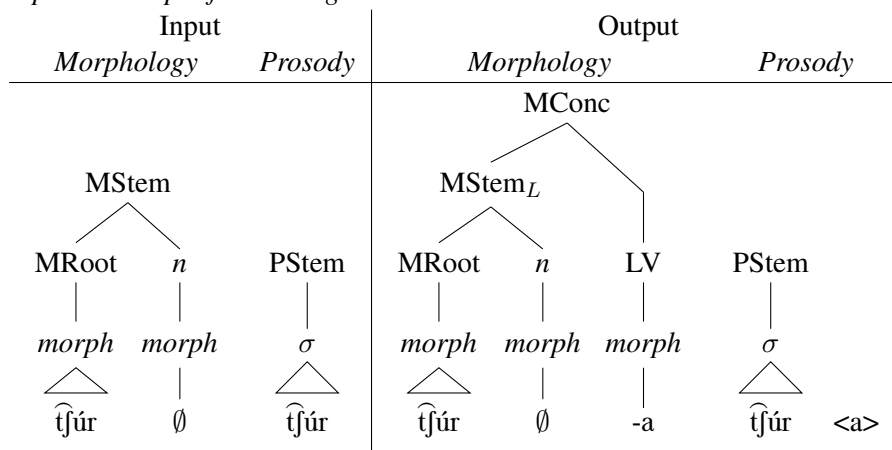
I show the full input and output below.

(357) *Input and output for Linking with the MStem_L as* t͡ʃúr

Generating a linked MConc is a transduction with a copy set of size 5. Defining this transduction is analogous to defining the previous other morphological processes. The only difference is in the label of segments and MNodes. In Copy 1, all relations are faithfully outputted.

(358)   *QF output functions to faithfully output relations in MStem$_L$ in Copy 1*
- For every relation rel $\in R$:
  $\phi\mathtt{rel}(x^1, y^1) \overset{\mathsf{def}}{=} \mathsf{rel}(x, y)$

The morphologically topmost node in the input MStem$_L$ will become the left stem of a compound. It will have the label MStem:Comp:Left(x). This is done with the output function below. Recall that $\mathbf{MTopmost}(x)$ is QF-definable.[4]

(359)   *QF output function for labeling the input as the MStem$_L$ of a compound*
- $\phi\mathtt{MStem:Comp:Left}(x^1) \overset{\mathsf{def}}{=} \mathbf{MTopmost}(x) \vee \mathsf{MStem{:}Comp{:}Left}(x)$

All other labels are faithfully outputted.

(360)   *QF output function for faithfully outputting other labels in Copy 1*
- For every label lab $\in L - \{$MStem:Comp:Left$\}$:
  $\phi\mathtt{lab}(x^1) \overset{\mathsf{def}}{=} \mathsf{lab}(x)$

In Copies 2-5, the new material is generated. In Copy 2, the linking vowel *a* is generated as the output correspondent of the final segment *r*. The final segment is picked by the predicate **final:seg**(x) which is QF-definable.

(361)   *QF output function for generating the linking vowel's segment in Copy 2*
- $\phi\mathtt{a}(x^2) \overset{\mathsf{def}}{=} \mathbf{final}{:}\mathbf{seg}(x)$

In Copies 3-5, the morphological nodes are generated as output correspondents for the morphologically topmost node, the MStem$_{0.11}$. The morpheme node for the linking vowel at Copy 4 is labeled as a linking vowel.

(362)   *QF output functions for generating morphological nodes in Copies 2-4*
- $\phi\mathtt{morph}(x^3) \overset{\mathsf{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{LV\_morpheme}(x^4) \overset{\mathsf{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{MConc}(x^5) \overset{\mathsf{def}}{=} \mathbf{MTopmost}(x)$

Linearization is formalized the same as in previous morphology sections. The new nodes are internally linearized via MDom. Likewise, the new morphology is externally linearized with the base by making the MConc dominate the base's MStem. The linking vowel *a* is externally linearized with the base's final segment via immediate successor. This is done with the output functions below.

---

[4]The disjunct $\vee$MStem:Comp:Left(x) is to preserve any underlying left stems of compounds if the input MStem$_L$ itself is a compound.

(363)   a. *QF output functions for internally linearizing the new material in Linking*

- $\phi\texttt{MDom}(x^3, y^2) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y)$
- $\phi\texttt{MDom}(x^4, y^3) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$
- $\phi\texttt{MDom}(x^5, y^4) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

   b. *QF output functions for externally linearizing the base with the new morphology*

- $\phi\texttt{MDom}(x^5, y^1) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$
- $\phi\texttt{succ:seg}(x^1, y^2) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$

6.2.2.4 Formalizing Concatenation

The input is now MStem_L concatenated with the linking vowel. The entire construction is an MConc. This section will concatenate this construction with MStem_R. I show the input and output below. Note that the subscript $_R$ on some stem means that it is the second stem of a compound. As before, the entire process is computationally local. The main strategy is converting MStem_R into a transduction.

(364)   *Input and output of Concatenation with MStem_L $\widehat{\text{t\textesh úr}}$*
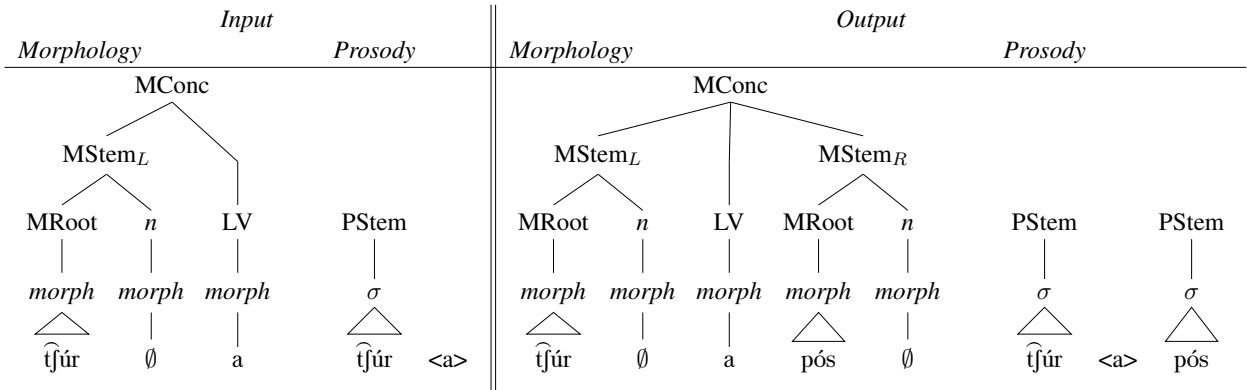
The full structure of the input $\widehat{\text{t\textesh úr}}$-*a* is shown below. I discard the SETTINGS constant. I likewise show the full structure of the second stem *pós*.

(365)   *Input MConc* $\widehat{\text{t}\int}$*úr-a for Concatenation withe the second stem* pós
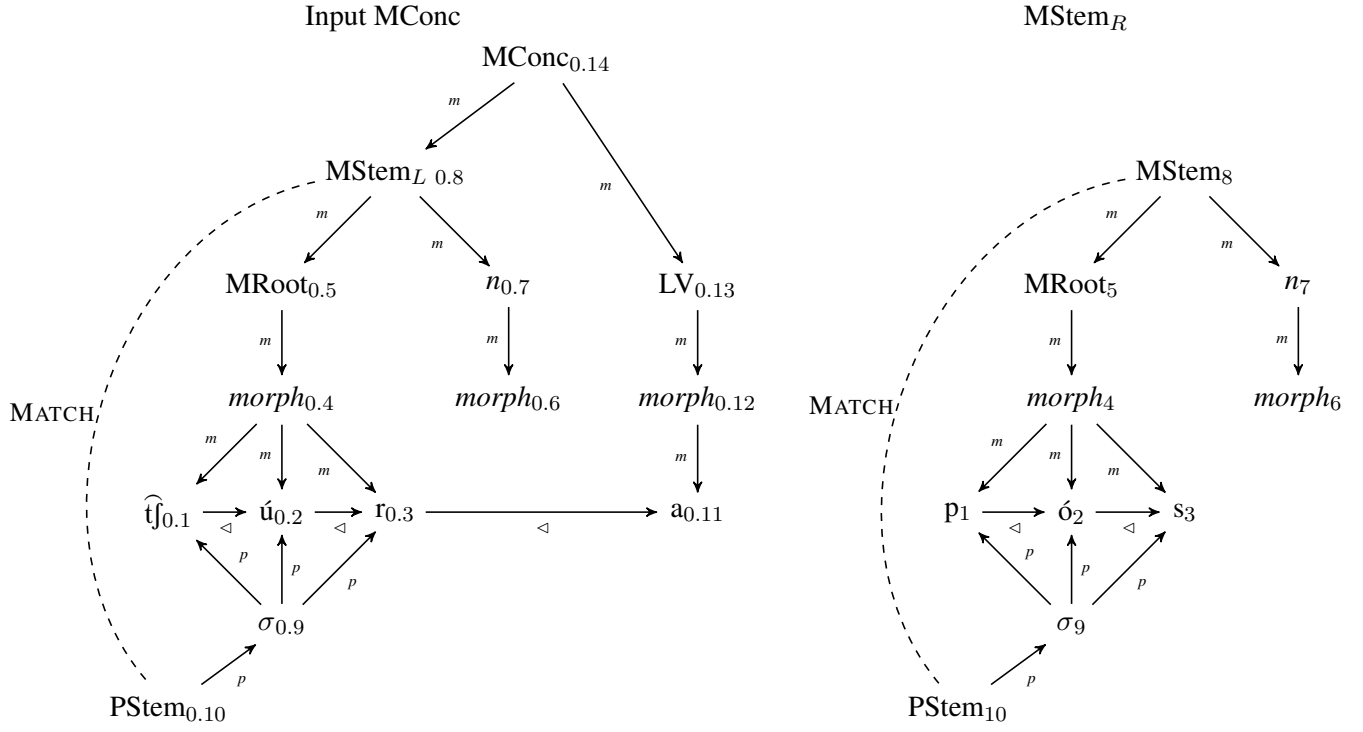
The above structure for MStem_R encodes the fact that the MStem_R *pos* is a set of nodes (the domain D), their labels L, and the relations R between them. This information is made explicit in the list of logical statements below.

(366)   *For the word* pós *with* **word signature** $\langle D, L, R \rangle$

    a. **Domain** D: {1,2,3,4,5,6,7,8,9,10}

    b. **Unary labels** L:

        1. *for segments*
- $p(x)$=TRUE for {1}
- $o(x)$=TRUE for {2}
- $s(x)$=TRUE for {3}
- $a(x)$=TRUE for $\emptyset$
- $b(x)$=TRUE for $\emptyset$

          . . .

        2. *for prosodic nodes*
- $\text{syll}(x)$=TRUE for {9}
- $\text{PStem}(x)$=TRUE for {10}

          . . .

        3. *for morphological nodes*
- $\text{morph}(x)$=TRUE for {4,6}
- $\text{MRoot}(x)$=TRUE for {5}
- $\text{noun}(x)$=TRUE for {7}

- $\text{MStem}(x)=$TRUE for $\{8\}$

  $\cdots$

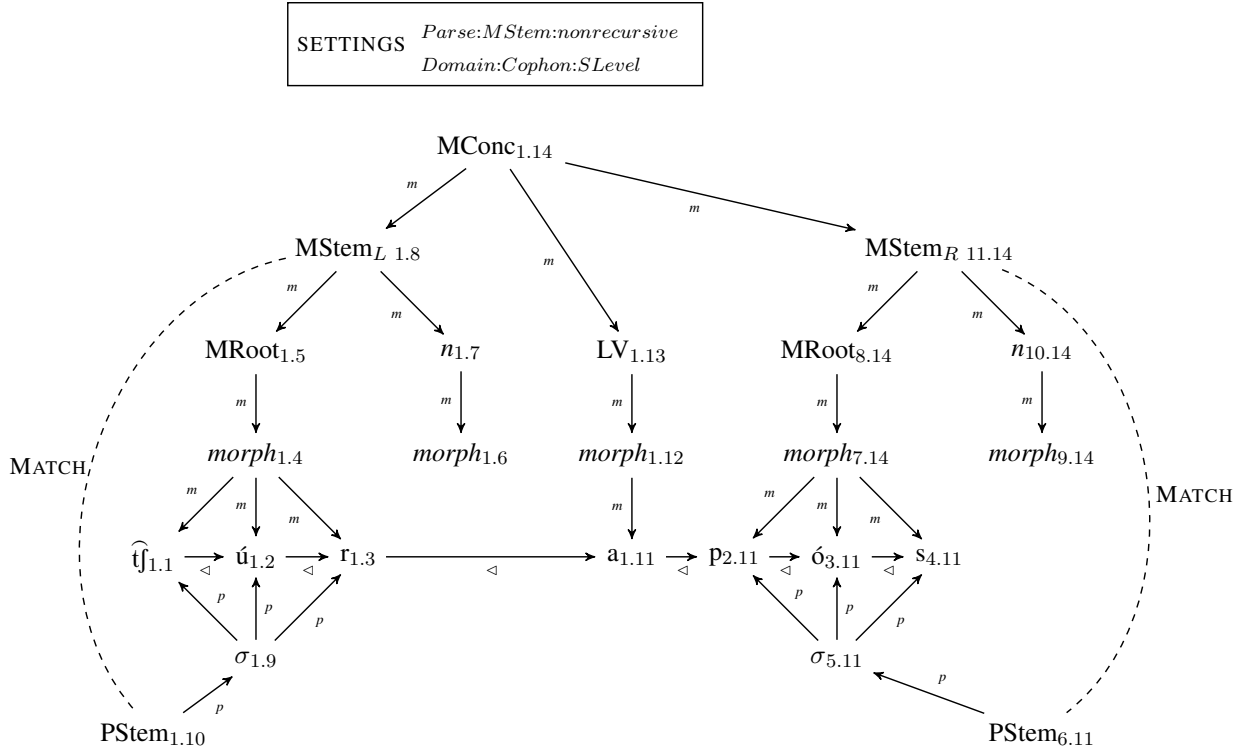c. **Binary relations** R:

- $\text{succ:seg}(x,y)=$TRUE for $\{(1,2),(2,3)\}$
- $\text{MDom}(x,y)=$TRUE for $\{(4,1),(4,2),(4,3),(5,4),(7,6),(8,5),(8,7)\}$
- $\text{PDom:syll\_ons}(x,y)=$TRUE for $\{(9,1)\}$
- $\text{PDom:syll\_nuc}(x,y)=$TRUE for $\{(9,2)\}$
- $\text{PDom:syll\_coda1}(x,y)=$TRUE for $\{(9,3)\}$
- $\text{PDom:PStem\_syll}(x,y)=$TRUE for $\{(10,9)\}$
- $\text{Match:stem}(x,y)=$TRUE for $\{(8,10)\}$

  $\cdots$

Any possible stem can be represented as a set of such statements. For any given MStem_R, it consists of s number of segments, p number of prosodic nodes, and m number of morphological nodes. This set of statements can be used to simulate a transduction for adding a stem to a compound. This transduction has a copy set of size $1+s+p+m$.

In the case of adding the stem *pos*, s is 3, p is 2, and m is 5. Thus we use a transduction of size 11. I show the full desired output below.

(367) *Output of Concatenation the MStem$_L$ $\widehat{\text{t\textesh}}$úr-a with the MStem$_R$ pós*

In Copy 1, the input MConc is faithfully outputted.

(368)   *QF output function for generating a faithful copy of the base in Copy 1*
- For every label $\text{lab} \in L$:
  $\phi\texttt{lab}(x^1) \overset{\text{def}}{=} \text{lab}(x)$ for every label in input
- For every relation $\text{rel} \in R$:
  $\phi\texttt{rel}(x^1, y^1) \overset{\text{def}}{=} \text{rel}(x, y)$ for every binary relation in input

In Copies 2 to $1 + s + p + m$, new material is added. Lining up the domain of nodes of MStem_R to individual Copies is done via a 1-to-1 homomorphism. Each of the segments of MStem_R gets its own copy from Copies 2 to $1 + s$. Each of the prosodic nodes of MStem_R gets its own copy from Copies $1 + s + 1$ to $1 + s + p$, and each of the morphological nodes of MStem_R gets its own copy from Copies $1 + s + p + 1$ to $1 + s + p + m$. Two caveats for the homomorphism are that the initial segment of MStem_R gets Copy 2, while the morphologically topmost node in MStem_R gets Copy $1 + s + p + m$.

An example homomorphism for *pós* is shown below. Let this homomorphism function be called *h*.

(369)   *Mapping domain nodes to Copies*
1.   Segment node to Copy   (2 to $1 + s$)
     - Index $1 \rightarrow$ Copy 2 for the segment *p*
     - Index $2 \rightarrow$ Copy 3 for the segment *o*
     - Index $3 \rightarrow$ Copy 4 for the segment *s*
2.   Prosodic node to Copy   ($1 + s + 1$ to $1 + s + p$)
     - Index $9 \rightarrow$ Copy 5 for the syllable *.pos.*
     - Index $10 \rightarrow$ Copy 6 for the PStem *(.pos.)*
3.   Morphological node to Copy   ($1+s+p+1$ to $1+s+p+m$)
     - Index $4 \rightarrow$ Copy 7 for the root morph *pos*
     - Index $5 \rightarrow$ Copy 8 for the MRoot *pos*
     - Index $6 \rightarrow$ Copy 9 for the covert morph $\emptyset$
     - Index $7 \rightarrow$ Copy 10 for the nominalizer morpheme *n*
     - Index $8 \rightarrow$ Copy 11 for the MStem *pos-$\emptyset$*

In order to generate the segments in Copies 2 to $1 + s$, the segments are defined as output correspondents of the final segment in the input. In Copies $1 + s + 1$ to $1 + s + p$, the prosodic nodes are defined as output correspondents of the final segment in the input.[5] And for the morphological nodes in Copies $1 + s + p + 1$ to $1 + s + p + m$, they are defined as output correspondents of the morphologically topmost node in the input.

(370)   a. *Template of QF output functions for generating the segments of MStem_R in Copies* $c \in [2, \dots, 1+$
        $s]$
        - $\phi\texttt{lab}(x^c) \overset{\text{def}}{=} \textbf{final:seg}(x)$

---

[5]The generated PNodes are defined in terms of the final segment in the input instead of the 'topmost' prosodic node in the input because it is unclear what would count as the 'topmost' node in prosodic structure. The input could contain a string of syllables without any PStem, thus the input lacks a root in the prosodic tree. The input could also contain a string of syllables such that all but that last syllable is dominated by a PStem. In this scenario, it is unclear if the PStem or the final syllable is any more 'topmost' than the other.

    b. *Template of QF output functions for generating the prosodic nodes of MStem$_R$ in Copies* c $\in [1 + s + 1, \ldots, 1 + s + p]$
- $\phi\texttt{lab}(x^c) \overset{\text{def}}{=} \textbf{final:seg}(x)$

    c. *Template of QF output functions for generating the morphological nodes of MStem$_R$ in Copies* c $\in [1 + s + p + 1, \ldots, 1 + s + p + m]$
- $\phi\texttt{lab}(x^c) \overset{\text{def}}{=} \textbf{MTopmost}(x)$

The topmost node of MStem$_R$ is the MStem in Copy $1 + s + p + m$. It is labeled as the second stem in the compound via the label MStem:Comp:Right. This is marked by a subscript $_R$ on the graph.

(371) *QF output function for labeling the MStem in Copy* $c = 1 + s + p + m$ *as the right stem of a compound*
- $\phi\texttt{MStem:Comp:Right}(x^c) \overset{\text{def}}{=} \textbf{MTopmost}(x)$

All underlying relations are faithfully outputted thanks to the homomorphism. Let *a,b* be any two nodes in MStem$_R$, let *c* and *d* be the Copies which are assigned to these nodes.

(372) *Template of QF output functions for faithfully outputting any underlying relations in MStem$_R$ for any two node*
- For every relation rel $\in R$:
  $\phi\texttt{rel}(x^c, y^d)$ is TRUE iff the nodes *a,b* satisfy the relation rel in MStem$_R$

The new MStem$_R$ is linearized with the input by the two functions below. The first segment of MStem$_R$ at Copy 2 immediately succeeds the final segment of the input at Copy 1. The topmost morphological node of MStem$_R$ at Copy $1 + s + p + m$ is morphologically dominated by the input's MConc at Copy 1.

(373) *QF output functions for externally linearizing MStem$_R$ with the base*
- For Copy $c = [2]$
  $\phi\texttt{succ:seg}(x^1, y^c) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$
- For Copy $c = 1 + s + p + m$
  $\phi\texttt{MDom}(x^1, y^c) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

For a MStem$_R$ *pós*, the output of all these functions is shown in (367). Interested readers are encouraged to work out the individual steps of the generation themselves.

6.2.2.5   Formalizing MStem formation

The final step in generating a compound consists of 1) adding a new MStem layer over the MConc, 2) adding a covert category suffix, and 3) determining if the compound is endocentric or not. The first two steps are analogous to generating a covert category suffix for a simplex root (Chapter 5: §5.2). The last one is compound-specific. As before, all steps are computationally local.

The input and output for this section are shown below. Note that the subscript $_C$ marks that an MStem is a compound. The subscript $_{En}$ marks that a compound MStem is endocentric.

(374) *Input and output for MStem formation*
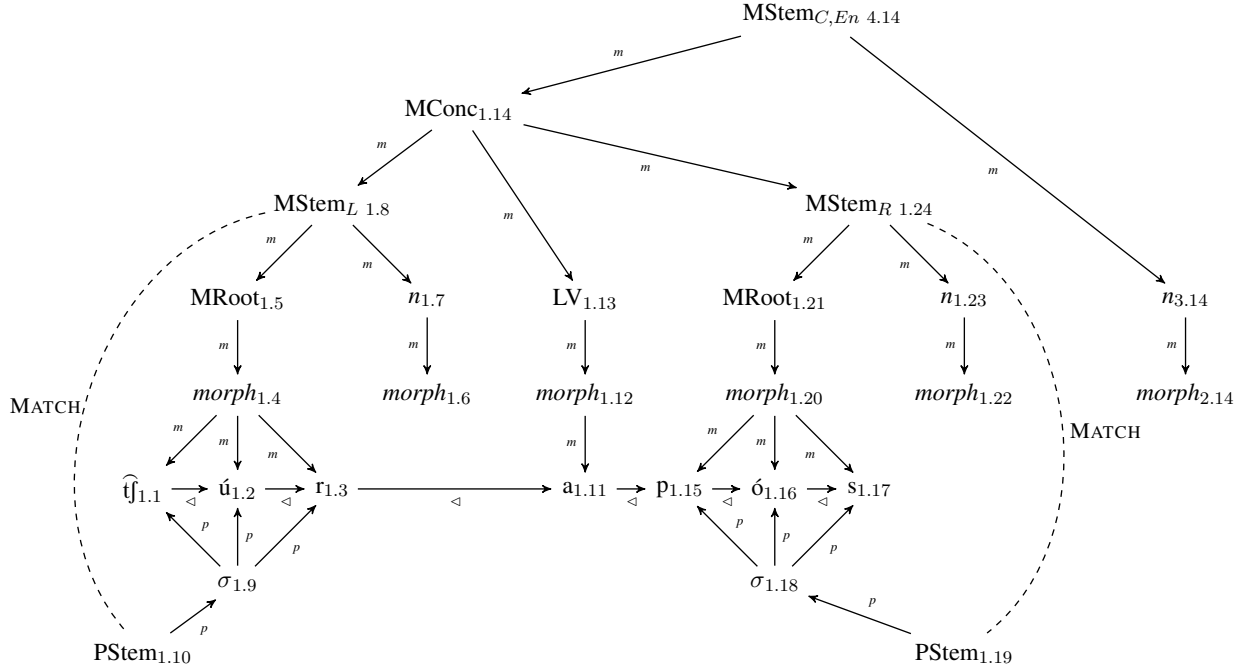
	Input	*Output*
Morphology		

MStem$_{C,En}$

MConc (Input)
- MStem$_L$
  - MRoot — *morph* — $\widehat{t\int}$úr
  - n — *morph* — $\emptyset$
- LV — *morph* — a
- MStem$_R$
  - MRoot — *morph* — pós
  - n — *morph* — $\emptyset$

MConc (Output)
- MStem$_L$
  - MRoot — *morph* — $\widehat{t\int}$úr
  - n — *morph* — $\emptyset$
- LV — *morph* — a
- MStem$_R$
  - MRoot — *morph* — pós
  - n — *morph* — $\emptyset$
- n — *morph* — $\emptyset$

Prosody:

Input:
- PStem — σ — $\widehat{t\int}$úr
- <a>
- PStem — σ — pós

Output:
- PStem — σ — $\widehat{t\int}$úr
- <a>
- PStem — σ — pós

These steps require a transduction with a copy set of size 4. I show the full input and output below, without the SETTINGS node.

(375) a. *Input to MStem formation*

MConc$_{0.14}$
- m → MStem$_{L\ 0.8}$
  - m → MRoot$_{0.5}$
    - m → *morph*$_{0.4}$
      - m → $\widehat{t\int}_{0.1}$
      - m → ú$_{1.2}$
      - m → r$_{0.3}$
  - m → $n_{0.7}$
    - m → *morph*$_{0.6}$
- m → LV$_{0.13}$
  - m → *morph*$_{0.12}$
    - m → a$_{0.11}$
- m → MStem$_{R\ 0.24}$
  - m → MRoot$_{0.21}$
    - m → *morph*$_{0.20}$
      - m → p$_{0.15}$
      - m → ó$_{0.16}$
      - m → s$_{0.17}$
  - m → $n_{0.23}$
    - m → *morph*$_{0.22}$

$\widehat{t\int}_{0.1}$ ◁ ú$_{1.2}$ ◁ r$_{0.3}$ ◁ a$_{0.11}$ ◁ p$_{0.15}$ → ó$_{0.16}$ ◁ s$_{0.17}$

MATCH, $\sigma_{0.9}$, PStem$_{0.10}$ (p)
MATCH, $\sigma_{0.18}$, PStem$_{0.19}$ (p)
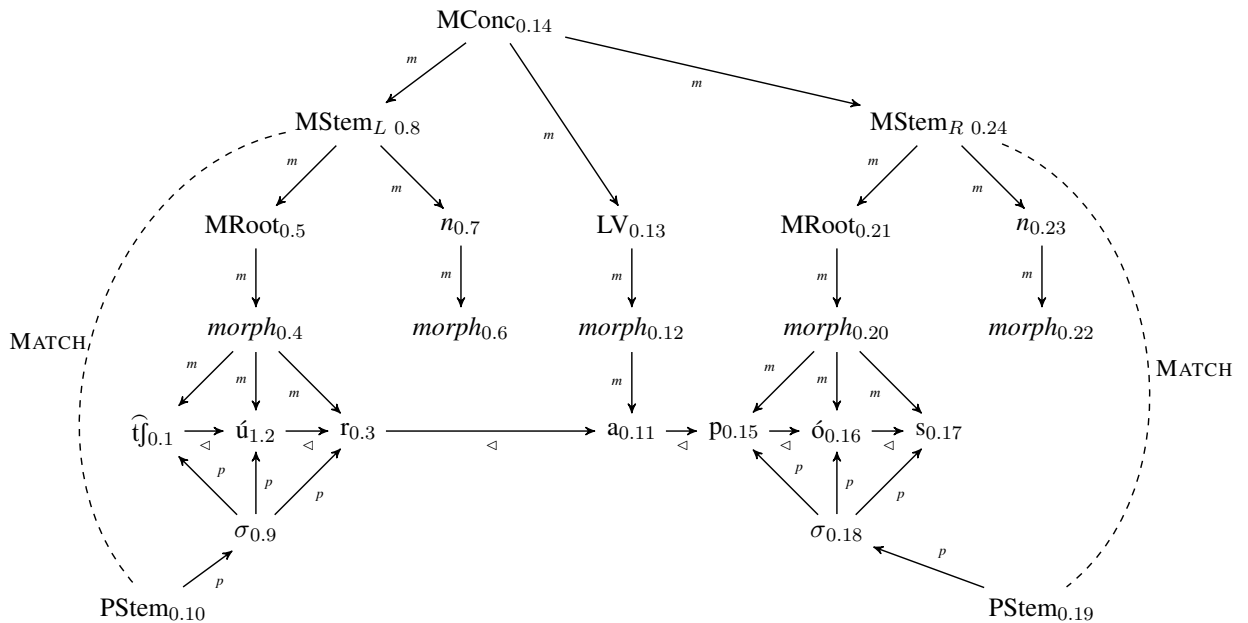
b. *Output of MStem formation for an endocentric compound*

In Copy 1, the input is faithfully outputted.

(376) *QF output functions for faithfully outputting the input in Copy 1*
- For every label lab $\in L$:
  $\phi\mathtt{lab}(x^1) \overset{\text{def}}{=} \mathsf{lab}(x)$
- For every relation rel $\in R$:
  $\phi\mathtt{rel}(x^1, y^1) \overset{\text{def}}{=} \mathsf{rel}(x, y)$

In Copies 2-4, the new morphological nodes are generated as output correspondents of the morphologically topmost node in the input, the MConc at '0.14'.

(377) *QF output functions for generating the morphological nodes in MStem formation*
- $\phi\mathtt{morph}(x^2) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{noun}(x^3) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$
- $\phi\mathtt{MStem}(x^4) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$

Note the new MStem in Copy 4 must be specified as a compound via the label MStem:Comp. I visualize this as a subscript $_C$.

(378) *QF output function for labeling the new MStem as a compound MStem*
- $\phi\mathtt{MStem{:}Comp}(x^4) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$

The new morphological nodes are internally linearized with themselves and externally linearized with the input via morphological dominance.

(379)   a. *QF output functions for internally linearizing the new morphology*
- $\phi\text{MDom}(x^4, y^3) \stackrel{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$
- $\phi\text{MDom}(x^3, y^2) \stackrel{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

b. *QF output functions for externally linearizing the new morphology*
- $\phi\text{MDom}(x^4, y^1) \stackrel{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y)$

For the third step, it is a lexeme-specific property whether the compound will be interpreted as hyponymic and thus endocentric or non-hyponymic and thus exocentric. I abstract from this arbitrariness by using the output functions below. One of the two functions will be arbitrarily chosen. I visualize endocentric and exocentric compounds with the subscripts $_{En}$ and $_{Ex}$.

(380)   *Output functions for labeling the compound as endocentric or exocentric*
- $\phi\text{MStem:Comp:Endo}(x^4) \stackrel{\text{def}}{=} \textbf{MTopmost}(x)$ and the compound is interpreted as hyponymic
- $\phi\text{MStem:Comp:Exo}(x^4) \stackrel{\text{def}}{=} \textbf{MTopmost}(x)$ and the compound is interpreted as non-hyponymic

6.2.2.6   Excursus: Exocentric compounds

An exocentric compound like //t͡ʃúr-a-kújn//→t͡ʃər-a-kújn 'water-colored' is constructed in essentially the same way except that a) MStem$_R$ has more segments, b) the outer MStem is labeled as exocentric, and c) its part of speech is an adjective. I show the input and output of the three stages of compound formation for an exocentric compound. I omit the morph level. Interested readers can work through the derivation for themselves using the previous illustrations for endocentric compounds as a guide. The computation is still local.

(381)   *Generating an exocentric compound* t͡ʃúr-a-kújn→t͡ʃər-a-kújn

	Morphology	*Prosody*
Input	MStem ⟨MRoot n⟩ t͡ʃúr ∅	PStem σ t͡ʃúr
Output of Linking	MConc MStem LV MRoot n t͡ʃúr ∅ a	PStem σ t͡ʃúr
Output of Concatenation	MConc MStem$_L$ MStem$_R$ MRoot n LV MRoot n t͡ʃúr ∅ a kújn ∅	PStem PStem σ σ t͡ʃúr <a> kújn
Output for MStem formation	MStem$_{C,Ex}$ MConc MStem$_L$ MStem$_R$ MRoot n LV MRoot n a t͡ʃúr ∅ a kújn ∅ ∅	PStem PStem σ σ t͡ʃúr <a> kújn

## 6.3 Examining the Settings of complex words

With the new morphology in place, the SETTINGS is updated in order to encode 1) what to parse, and 2) what cophonology to apply. In this section, I show what information is examined for a derivative, inflected item, and a compound. This information is non-local because it references the morphologically topmost MNode. As explained before in §6.2.1.1, given a node x, we can locally determine if it is topmost or not, i.e., **MTopmost**(x) is a QF predicate. But, finding this node x is not QF definable because we need to search through the tree in order to find the topmost node, i.e., $\exists x[\textbf{MTopmost}(x)]$ is not QF.

The transduction must use a copy set of size 1. In Copy 1, all the input items, labels, and relations are faithfully outputted except for any parse or domain labels on SETTINGS. The only changes are in the SETTINGS constant. I first go over updates over what phonological rule domains to apply (§6.3.1) and what to prosodically parse (§6.3.2).

6.3.1 Phonological Rule Domains

In terms of phonological rule domains, the derivative and compound should trigger the stem-level cophonology because they form an MStem. In contrast, an inflected item should trigger the word-level cophonology because it forms an MWord. I show below the intermediate and simple representation of these three items, after the morphology has applied.

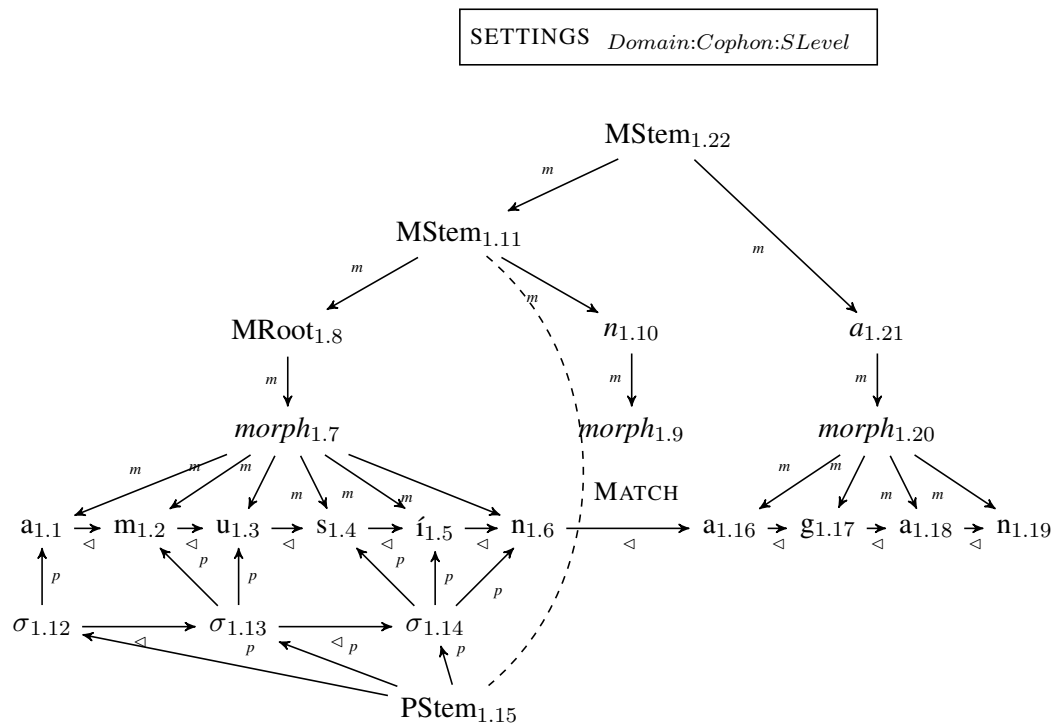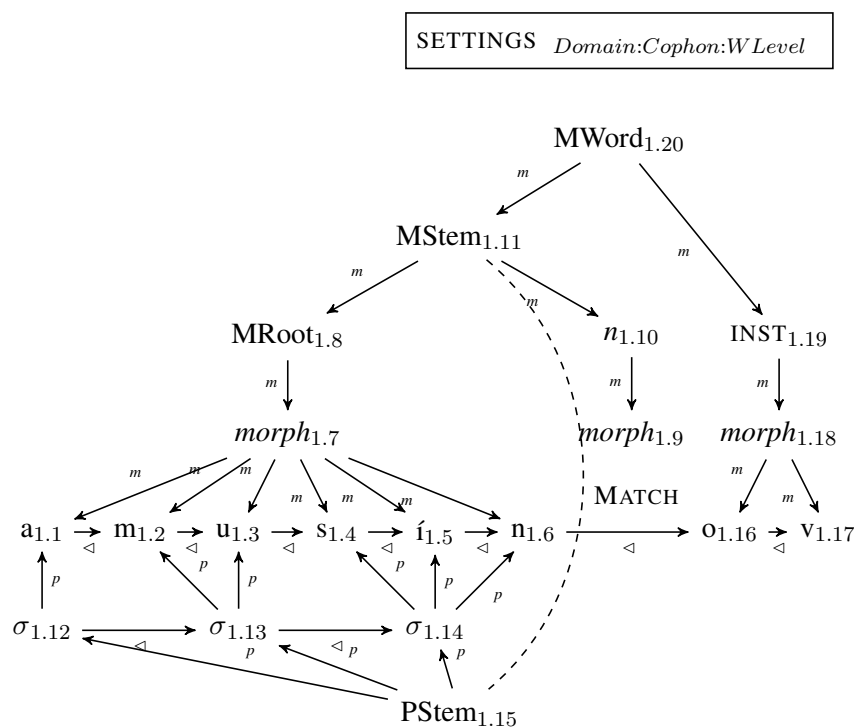(382)   *Morphological structure of complex items*
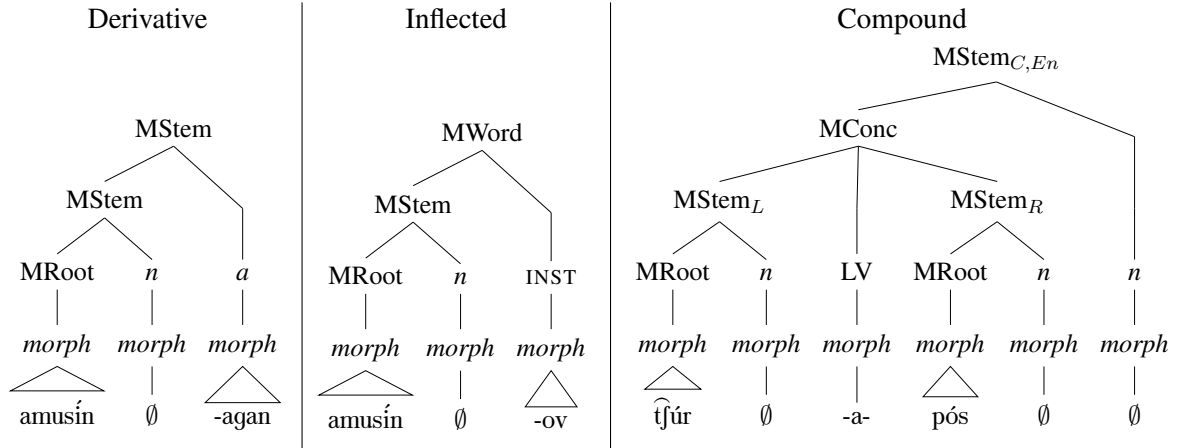
The functions below will update the SETTINGS with the right cophonology label. They examine the topmost MNode and check its cophonology label. These are the topmost MStem and its Cophon:SLevel property for the derivative and compound, while they're the topmost MWord and its Cophon:WLevel property for the inflected item.

(383)   *FO output functions for updating the* SETTINGS *for the right cophonology*
   - $\phi\texttt{Domain:Cophon:SLevel}(\text{SETTINGS}^1) \stackrel{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge \mathsf{Cophon:SLevel}(x)]$
   - $\phi\texttt{Domain:Cophon:WLevel}(\text{SETTINGS}^1) \stackrel{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge \mathsf{Cophon:WLevel}(x)]$

I show below the output for the derivative and inflected item; I omit the compounds for brevity. For the derivative //amusín-agan// and compound //t͡ʃúr-a-pós//, the SETTINGS has the label Domain:Cophon:SLevel. For the inflected item //amusín-ov//, the SETTINGS has the label Domain:Cophon:WLevel.

(384)   *Updating the* SETTINGS *for the cophonology domain*

a.   *In a derivative* //amusín-agan//

SETTINGS   *Domain:Cophon:SLevel*

$\text{MStem}_{1.22}$

$\text{MStem}_{1.11}$ $n_{1.10}$ $a_{1.21}$

$\text{MRoot}_{1.8}$

morph$_{1.7}$ *morph*$_{1.9}$ *morph*$_{1.20}$

$a_{1.1}$ $m_{1.2}$ $u_{1.3}$ $s_{1.4}$ $í_{1.5}$ $n_{1.6}$ MATCH $a_{1.16}$ $g_{1.17}$ $a_{1.18}$ $n_{1.19}$

$\sigma_{1.12}$ $\sigma_{1.13}$ $\sigma_{1.14}$

$\text{PStem}_{1.15}$

b.   *In an inflected item* //amusín-ov//

SETTINGS   *Domain:Cophon:WLevel*

$\text{MWord}_{1.20}$

$\text{MStem}_{1.11}$

$\text{MRoot}_{1.8}$ $n_{1.10}$ $\text{INST}_{1.19}$

morph$_{1.7}$ *morph*$_{1.9}$ *morph*$_{1.18}$

$a_{1.1}$ $m_{1.2}$ $u_{1.3}$ $s_{1.4}$ $í_{1.5}$ $n_{1.6}$ MATCH $o_{1.16}$ $v_{1.17}$

$\sigma_{1.12}$ $\sigma_{1.13}$ $\sigma_{1.14}$

$\text{PStem}_{1.15}$

6.3.2   Instructions for Prosody

With the right cophonology determined, we now determine the right prosodic parse. This is more complicated than determining the cophonology; I go through each item individually. I do not show their more explicit input form. In every case, we examine the morphologically topmost MNode, what it immediately dominates, and what is the prosodic status of its dominee. In order to find the topmost MNode, we need to access global information which is not computationally local (not QF) but requires FO logic.

6.3.2.1   Derivation

For the derivative //amusín-agan//, I show the input morphology and prosody below.

(385)   *Morphology and prosody of an unparsed derivative* //amusín-agan//
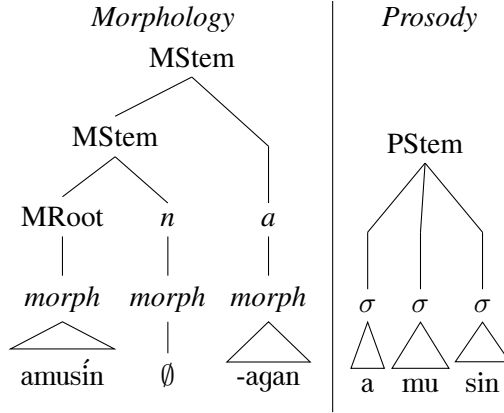
Prosodically, the base has a matched MStem and the derivative now has an unparsed *recursive* MStem. It is recursive because the affix's MStem dominates another MStem. We encode this configuration into the new parse label below that is only defined for SETTINGS.

(386)   *Unary label for parsing a recursive MStem into a PStem via the* SETTINGS *constant*
- Parse:MStem:recursive(SETTINGS): is TRUE for SETTINGS iff we need to parse an unparsed
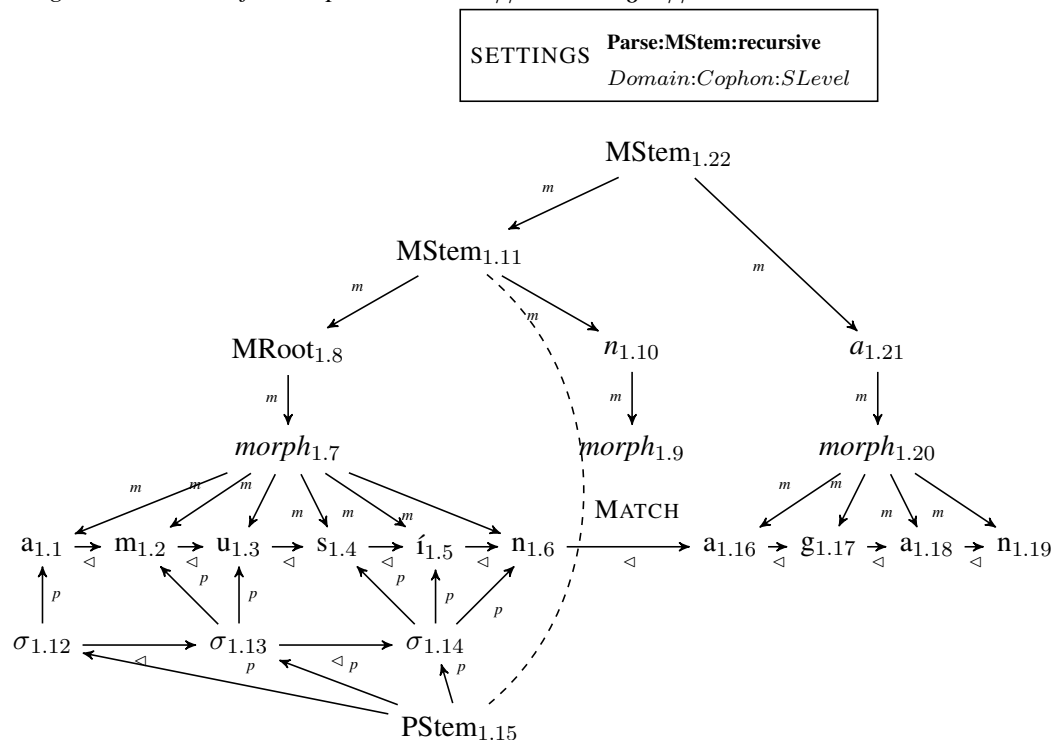   MStem which dominates a matched (parsed) MStem

The output function below generates this new label on the SETTINGS. It is true because the morphologically topmost node is an $\text{MStem}_{0.22}$ (x) which dominates a matched $\text{MStem}_{0.15}$ (y), but it itself is not matched to a PStem (w).

(387)   *FO output function for parsing a recursive MStem via the* SETTINGS *constant*
- $\phi\texttt{Parse:MStem:recursive}(\text{SETTINGS}^1) \stackrel{\text{def}}{=}$
$$\exists x[\mathbf{MTopmost}(x) \wedge \mathsf{MStem}(x)\wedge$$
$$\exists y, z[\mathsf{MStem}(y) \wedge \mathsf{MDom}(x,y) \wedge \mathsf{PStem}(z) \wedge \mathsf{Match:stem}(y,z)]]\wedge$$
$$\neg\exists w[\mathsf{PStem}(w) \wedge \mathsf{Match:stem}(x,w)]]$$

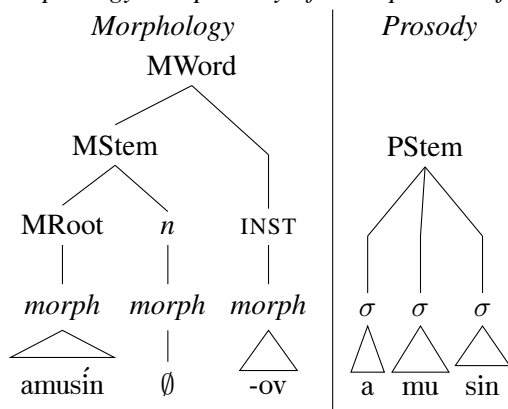I apply this function below. I mark the changed parse label in bold.

(388) *Updating the* SETTINGS *for the parse label in* //amusín-agan//

SETTINGS	**Parse:MStem:recursive**
	Domain:Cophon:SLevel

$\text{MStem}_{1.22}$

$\text{MStem}_{1.11}$ $n_{1.10}$ $a_{1.21}$

$\text{MRoot}_{1.8}$ *morph*$_{1.9}$ *morph*$_{1.20}$

morph$_{1.7}$ MATCH

$a_{1.1} \rightarrow m_{1.2} \rightarrow u_{1.3} \rightarrow s_{1.4} \rightarrow í_{1.5} \rightarrow n_{1.6} \longrightarrow a_{1.16} \rightarrow g_{1.17} \rightarrow a_{1.18} \rightarrow n_{1.19}$

$\sigma_{1.12} \longleftrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.14}$

$\text{PStem}_{1.15}$

6.3.2.2 Inflection

With the right SETTINGS determined for the derivative, we now update the SETTINGS to prosodify an inflected item //amusín-ov//. The input morphology and prosody is shown below.

(389) *Morphology and prosody of an unparsed inflected item* //amusín-ov//

Morphology

MWord

MStem — INST

MRoot — *n* — INST

morph *morph* *morph*

amusín ∅ -ov

Prosody

PStem

σ σ σ

a mu sin

The cophonology SETTINGS will be updated to Domain:Cophon:WLevel. As for the prosody, we need

to encode the fact that the input contains an unparsed MWord which is topmost and dominates a matched (parsed) MStem.[6] The MWord must be mapped to a PWord.

(390)   *Unary label for parsing a non-recursive MWord via the* SETTINGS *constant*
   - Parse:MWord:nonrecursive(SETTINGS): is TRUE for SETTINGS iff we need to parse a non-recursive MWord into a PStem

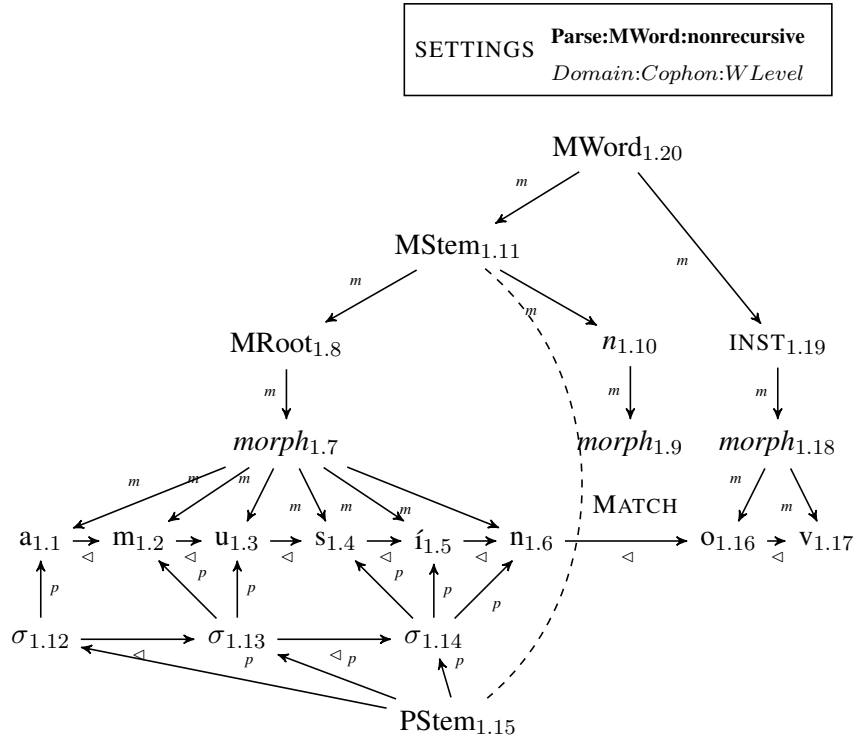This parsing fact is encoded into the SETTINGS. The SETTINGS get the label Parse:MWord:nonrecursive because there exists an $\text{MWord}_{0.20}$ (x) which is topmost, dominates an $\text{MStem}_{0.11}$ (y) which is matched into a $\text{PStem}_{0.15}$ (z), while x itself is unparsed, i.e., not matched into a PWord w.

(391)   *FO output function to update the* SETTINGS *with the parsing instructions to parse a non-recursive MWord*
   - $\phi\texttt{Parse:MWord:nonrecursive}(\texttt{SETTINGS}^1) \overset{\text{def}}{=}$
$$\exists x[\mathbf{MTopmost}(x) \wedge \mathsf{MWord}(x) \wedge$$
$$\exists y, z[\mathsf{MStem}(y) \wedge \mathsf{MDom}(x, y) \wedge \mathsf{PStem}(z) \wedge \mathsf{Match:stem}(y, z)] \wedge$$
$$\neg\exists w[\mathsf{PWord}(w) \wedge \mathsf{Match:word}(x, w)]]$$

I show the output below.

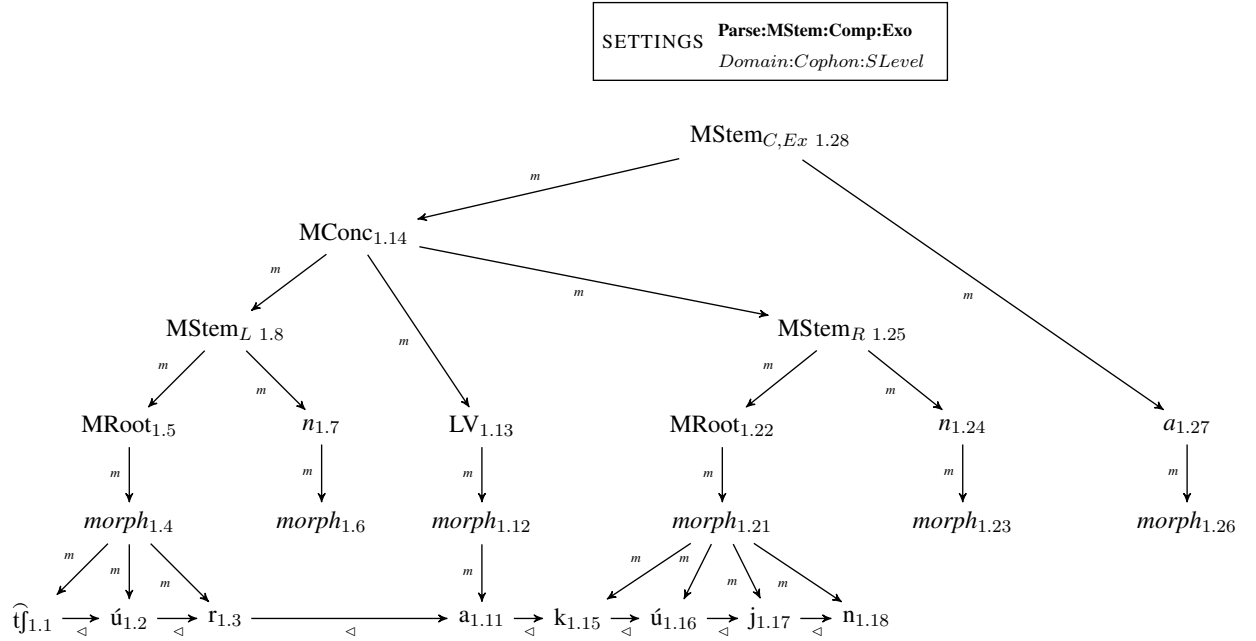(392)   *Output of updating the* SETTINGS *for an inflected item with the suffix* -ov

---

6.3.2.3 Compounds

For compounds, endocentric and exocentric compounds have different prosodic parses. Consider their input morphology below.

(393) *Morphological structure of different compounds*

Endocentric nominal	Exocentric possessive

As explained in the Chapter 1: §1.1.4, endocentric compounds are parsed into two PStems, while exocentric compounds are parsed into one PStem. These different representations are discussed later in §6.5.3 where we generate the right prosody. For now, the main task is to update the SETTINGS with the relevant information. These instructions are modeled with two new types of parse labels. They merely remember if the input's topmost node is an endocentric vs. exocentric compound.

(394) *Unary label for parsing compounding via the* SETTINGS *constant*
- Parse:MStem:Comp:Endo(SETTINGS): is TRUE for SETTINGS iff we need to parse an endocentric compound
- Parse:MStem:Comp:Exo(SETTINGS): is TRUE for SETTINGS iff we need to parse an exocentric compound

The SETTINGS is updated with the right parse label via the output functions below.

(395) *FO output functions for updating the* SETTINGS *with the right parse label for compounds*
- $\phi \texttt{Parse:MStem:Comp:Endo}(\texttt{SETTINGS}^1) \stackrel{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge \text{MStem:Comp:Endo}(x)]$
- $\phi \texttt{Parse:MStem:Comp:Exo}(\texttt{SETTINGS}^1) \stackrel{\text{def}}{=} \exists x[\mathbf{MTopmost}(x) \wedge \text{MStem:Comp:Exo}(x)]$

For the endocentric compound *tʃúr-a-pós* and exocentric compound *tʃúr-a-kújn*, the input and output are largely the same. I show the output below for both endocentric and exocentric compounds. Note the difference in subscripts for the large MStem in both, and the difference in parse labels in **bold**. I omit the prosodic nodes.

(396)   a. *Updating the* SETTINGS *on the prosodic parse for endocentric*

SETTINGS	**Parse:MStem:Comp:Endo**
	Domain:Cophon:SLevel

$\text{MStem}_{C,En\ 1.27}$

$\text{MConc}_{1.14}$

$\text{MStem}_{L\ 1.8}$

$\text{MStem}_{R\ 1.24}$

$\text{MRoot}_{1.5}$ $n_{1.7}$ $\text{LV}_{1.13}$ $\text{MRoot}_{1.21}$ $n_{1.23}$ $n_{1.26}$

$morph_{1.4}$ $morph_{1.6}$ $morph_{1.12}$ $morph_{1.20}$ $morph_{1.22}$ $morph_{1.25}$

$\widehat{\text{tʃ}}_{1.1} \rightarrow \text{ú}_{1.2} \rightarrow \text{r}_{1.3} \longrightarrow \text{a}_{1.11} \rightarrow \text{p}_{1.15} \rightarrow \text{ó}_{1.16} \rightarrow \text{s}_{1.17}$
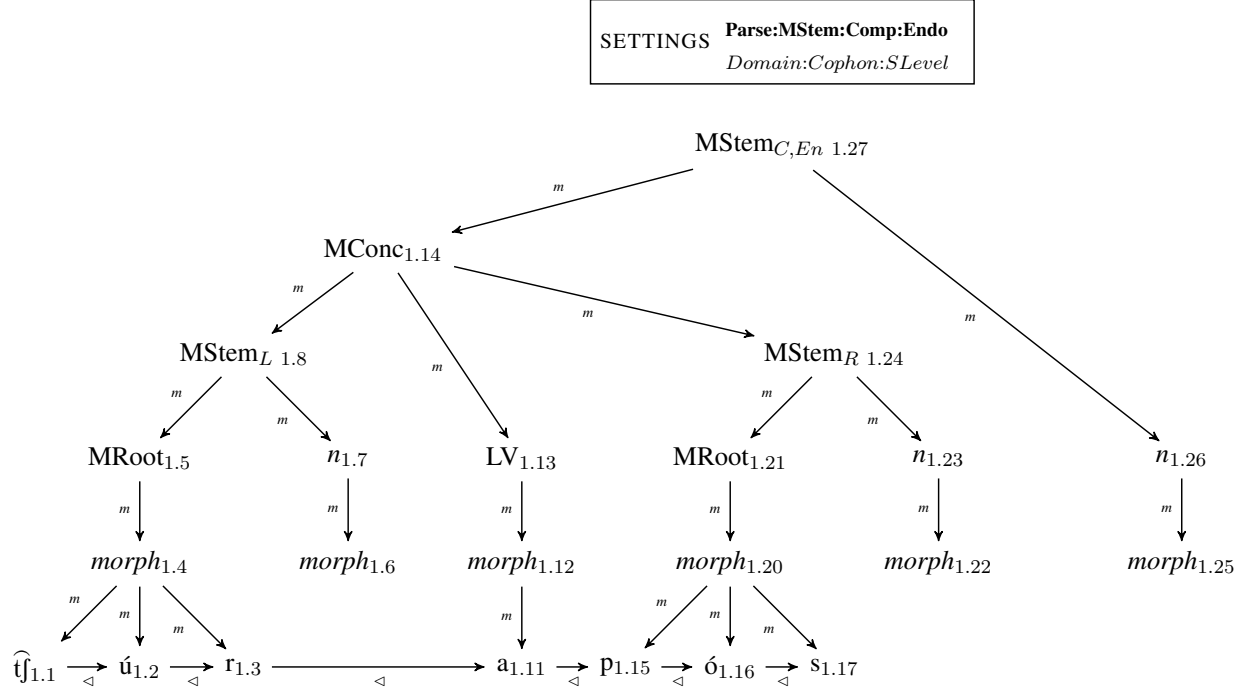
b. *Updating the* SETTINGS *on the prosodic parse for exocentric compound*

SETTINGS	**Parse:MStem:Comp:Exo**
	Domain:Cophon:SLevel

$\text{MStem}_{C,Ex\ 1.28}$

$\text{MConc}_{1.14}$

$\text{MStem}_{L\ 1.8}$

$\text{MStem}_{R\ 1.25}$

$\text{MRoot}_{1.5}$ $n_{1.7}$ $\text{LV}_{1.13}$ $\text{MRoot}_{1.22}$ $n_{1.24}$ $a_{1.27}$

$morph_{1.4}$ $morph_{1.6}$ $morph_{1.12}$ $morph_{1.21}$ $morph_{1.23}$ $morph_{1.26}$

$\widehat{\text{tʃ}}_{1.1} \rightarrow \text{ú}_{1.2} \rightarrow \text{r}_{1.3} \longrightarrow \text{a}_{1.11} \rightarrow \text{k}_{1.15} \rightarrow \text{ú}_{1.16} \rightarrow \text{j}_{1.17} \rightarrow \text{n}_{1.18}$

6.4 Locality in resyllabification

Having updated the SETTINGS, the five items can be prosodically parsed. I first formalize resyllabification, and then generate higher prosodic structure. For better organization, I treat prosodic mapping in a separate section §6.5. Syllabification is a cornerstone of computational formalizations of phonology (Scobbie 1993a; Bird 1995; Coleman 1996, 1998). A less discussed topic is resyllabification because of its inherently derivational or trans-derivational nature which cannot be faithfully expressed by one-level formalisms. However, just as locality is a property of syllabification (Strother-Garcia 2019), this section shows that resyllabification is likewise computationally local.

6.4.1 Resyllabification patterns in Armenian

The five main examples of this chapter undergo similar resyllabification patterns. The table below shows the intermediate output of resyllabification, before higher prosodic structure (PStems, PWords) is readjusted or generated. I omit the exocentric compound $//\widehat{t\int}$úr-<a>-kújn//.

(397) *Resyllabification of complex items in Armenian*

	Derivation	V-initial Inflection	C-initial Inflection	Endocentric compound
Input	PStem: σ σ σ — a mu sín -<-agan>	PStem: σ σ σ — a mu sín -<-ov>	PStem: σ σ σ — a mu sín -<-ner>	PStem: σ — t͡ʃúr <a> PStem: σ — pós
Output	PStem: σ σ σ σ σ — a mu sí n-a gan	PStem: σ σ σ σ — a mu sí n-ov	PStem: σ σ σ σ — a mu sín ner	PStem: σ — t͡ʃú PStem: σ — r-a PStem: σ — pós

Before formalizing resyllabification, I go over the range of resyllabification patterns which exist in Armenian. Most resyllabification effects are **coda loss** and **onset creation** wherein input codas become onsets. Relevant contexts are summarized below. The brackets <> represent unsyllabified material, and double slashes //...// represent an intermediate representation before suffixes, clitics, and compounding have triggered resyllabification. Base-final codas become onsets before suffixes and enclitics: [.kar.] but [.ka.r-er.], [.ka.r=al.]. In compounding, a linking vowel (LV) -a- is often added and this triggers coda loss: [.ka.r-a-dun.]. If the second stem of a compound is V-initial, then no linking vowel is used and coda loss applies in the first stem: [.ka.r-a.d͡zux.].

(398)   *Resyllabification contexts for coda loss and onset creation in Armenian*

	Morphology	Input syllabification	Resyllabification	
Base	*kar*	/<kar>/	[<kar>]	'rock'
Base + suffix	*kar-er*	//.kar. <er>//	[.ka.rér]	'rocks'
Base + clitic	*kar=al*	//.kar. <al>//	[.ká.ral]	'rock also'
Compound with LV	*kar <a> dun*	//.kar. <a> .dun.//	[.ka.ra.dún.]	'stone-house'
Compound without LV	*kar ad͡zux*	//.kar. .a.d͡zux.//	[.ka.ra.d͡zúx.]	'pit-coal'

Some morphological contexts can trigger **coda creation**. Among suffixes, consider a lone-consonant suffix like the nominalizer *-k* or possessive suffix *-s,-t*. When added to a V-final base, they become a coda: [.da.ri.] and [.da.ri-k.], [.da.ri-s.], [.da.ri-t.]. If *-k* is added after a consonant with higher sonority, it forms a complex coda: [.kir-k.]; otherwise it forms an appendix marked by parentheses: [.ba.hant͡ʃ-(k).]. The possessive suffixes never form complex codas but instead trigger schwa epenthesis after consonants: [.ki.r-əs.], [.ki.r-ət.] and [.ba.han.t͡ʃ-əs.], [.ba.han.t͡ʃ-ət.].[7] Prefixes never resyllabify a base-initial onset into a coda.[8]

(399)   *Resyllabification of new codas*

Base-final segment	Nominalizer *-k*	Possessive *-s*	Possessive *t*
V	*dari-k* 'age' //.da.ri. <-k>// [.da.rik.]	*dari-s* 'my year' //.da.ri. <-s>/ [.da.ris.]	*dari-t* 'your year' //.da.ri. <-t>// [.da.rit.]
Higher-sonority C	*kir-k* 'book' //.kir. <-k>// [.kirk.]	*kir-s* 'my writing' //.kir. <-s>// [.ki.rəs.]	*kir-t* 'your writing' //.kir. <-t>// [.ki.rət.]
Equal- or lower-sonority C	*bahant͡ʃ-k* 'credit' //.ba.hant͡ʃ. <-k>// [.ba.hant͡ʃ(k)]	*bahant͡ʃ-s* 'my demand' //.ba.hant͡ʃ. <-s>// [.ba.han.t͡ʃəs.]	*bahant͡ʃ* 'your demand' //.ba.hant͡ʃ. <-t>// [.ba.han.t͡ʃət.]

There is no case of **onset loss** whereby some syllable loses its underlying onset. This is because Armenian lacks infixes. There is no case of **complex onset creation** because Armenian is generally a CVCC language.

6.4.2   Resyllabification of base syllables

Having described the existing resyllabification patterns in Armenian, I formalize those patterns here. In the following two sections, I illustrate the formalization with //a.mu.sín-<agan>//. The formulas also

---

[7]I do not formalize the behavior of possessive suffixes, but they behave similarly to clitic formation in Romance. Their behavior is likely due to a prosodic constraint against parsing the possessive suffix as part of subconstituent in a syllable (cf. Bonet and Lloret 2005), or restrictions on forming phrase-level syllables in the post-lexical phrasal phonology (cf. Cardinaletti and Repetti 2009). This behavior is likely morpheme- or construction-specific (Baronian 2017).

[8]Except for linking vowels after learned prefixes like *ham-a-t͡sajn* 'same-voice (as in *unanimous*)', there are no V-final prefixes. The closest case involves root-initial sibilant-stop clusters like /steɣd͡z-/ or /sk-/ which generally surface with a prothetic schwa when word- or stem-initial: *əsteɣd͡z-el* 'to create' or *əsk-al* 'to feel', even after prefixes *an-əsk-a* 'unconscious'. The schwa is lost in some exocentric compounds, *pan-a-steɣd͡z* [..nas.teɣ...] 'poet'. I set these cases aside because data on these compounds is limited.

work for the words discussed in the previous section. In Chapter 5: §5.4.1, I formalized the basic formula for initial syllabification. Those formula were carefully defined so that they can also handle resyllabification. The main idea is that we first output all underlyingly syllabified material which will survive resyllabification, e.g., for the input //a.mu.sín.-<agan>//, we form the intermediate output //a.mu.si.<n-agan>// with coda loss.

Syllabification is a transduction with a copy set of size 2. In Copy 1, all input nodes and labels surface faithfully. All relations surface faithfully *except* for any type of prosodic dominance between a syllable and a segment (onset, nucleus, inner coda, outer coda). The reason is because of possible resyllabification.

(400) *QF output functions for faithfully outputting labels and syllabification-independent relations in Copy 1*

- For every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$.
- For every relation $\mathsf{rel} \in R-\{\mathsf{PDom:syll\_ons}, \mathsf{PDom:syll\_nuc}, \mathsf{PDom:syll\_coda1}, \mathsf{PDom:syll\_coda2}\}$:
  $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

I illustrate this 'intermediate' output below. I omit the morphological nodes, PStem, and the SETTINGS.

(401) *Resyllabifying a derivative* //amusín-agan// – *outputting underlying syllables in Copy 1*
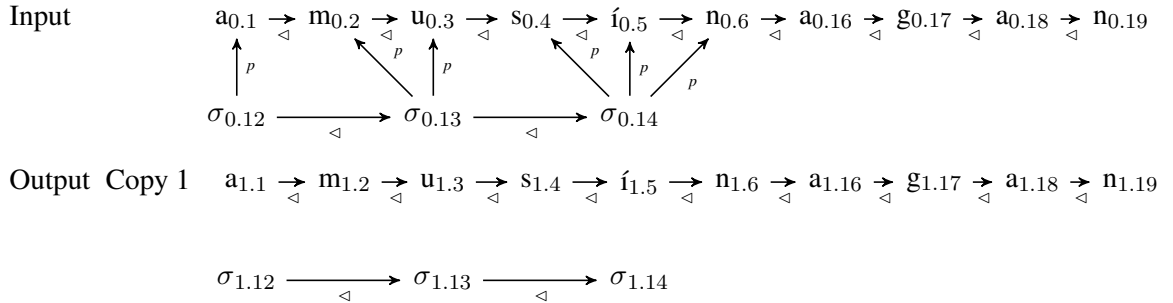
As said, the main strategy is that in Copy 1 we faithfully output only those prosodic dominances which will *survive* resyllabification. Recall from Chapter 5: §5.4.1, I use the following user-defined predicate to find unsyllabified segments in the input. This predicate is QF-definable and locally computed.

(402) a. *QF user-defined predicate for finding unsyllabified segments*

- $\phi\textbf{unsyllabified}(x^1) \stackrel{\text{def}}{=} \phi\mathrm{F}_D\mathtt{:PDom:syll\_ons}(x^1) = \textsc{null} \wedge$
  $\phi\mathrm{F}_D\mathtt{:PDom:syll\_nuc}(x^1) = \textsc{null} \wedge$
  $\phi\mathrm{F}_D\mathtt{:PDom:syll\_coda1}(x^1) = \textsc{null} \wedge$
  $\phi\mathrm{F}_D\mathtt{:PDom:syll\_coda2}(x^1) = \textsc{null}$

Underlying syllable-to-nucleus dominances are faithfully outputted because it does not get changed during resyllabification.[9] This is done via the output function below.

---

[9] The exception is when vowel deletion is used as a vowel hiatus repair rule. I do not formalize those rules here. For efficiency, we could treat vowel hiatus repair as a post-syllabification transduction.

(403) *QF output function for faithfully outputting underlying nuclei*

- $\phi\texttt{PDom:syll\_nuc}(x^1, y^1) \overset{\text{def}}{=} \textsf{PDom:syll\_nuc}(x, y)$

This is illustrated below.

(404) *Resyllabifying a derivative* //amusín-agan// *– outputting underlying nuclei in Copy 1*
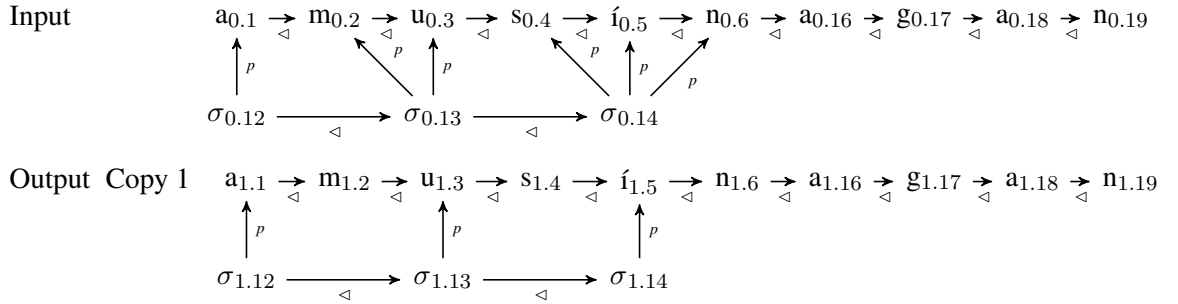
Onsets in the output come from two sources in the input: old input onsets vs. old input codas that are resyllabified. This distinction is formalized by two helper predicates for PDom:syll_ons. First, input onsets survive in resyllabification because an input onset is never changed into a coda. This 'default' behavior is formalized in the helper predicate **should__PDom:onset__old**(x, y).[10]

(405) *QF helper predicate for faithfully outputting underlying onsets*

- **should__PDom:onset__old**$(x, y) \overset{\text{def}}{=} \textsf{PDom:syll\_ons}(x, y)$

However, in certain cases, an *input* syllable can acquire a new onset. This occurs in compounds without a linking vowel where the second stem is V-initial: //.kar. .a.d͡zux.// → [.ka.ra.d͡zúx.] 'pit-coal'. This condition is formalized in the helper predicate **should__PDom:onset__new**(x, y). Some input consonant y *should be* parsed as an onset for an input syllable x iff y is underlyingly some syllable's coda, y is not x's onset, but the nucleus of x is a vowel z which follows y. I don't illustrate this formula. It is QF-definable.

(406) a. *FO helper predicate for outputting new onsets for underlying syllables*

- **should__PDom:onset__new**$(x, y) \overset{\text{def}}{=} \textsf{syll}(x) \wedge \textsf{consonant}(y) \wedge$
  $\exists w[\textsf{PDom:syll\_coda1}(w, y) \vee \textsf{PDom:syll\_coda2}(w, y)] \wedge$
  $\neg\textsf{PDom:syll\_ons}(x, y) \wedge$
  $\exists z[\textsf{vowel}(z) \wedge \textsf{PDom:syll\_nuc}(x, z) \wedge \textsf{succ:seg}(y, z)]$

b. *QF helper predicate for outputting new onsets for underlying syllables*

- **should__PDom:onset__new**$(x, y) \overset{\text{def}}{=} \textsf{syll}(x) \wedge \textsf{consonant}(y) \wedge$
  $[\textsf{F}_\textsf{D}\textsf{:PDom:syll\_coda1}(y) \neq \textsc{null} \vee \textsf{F}_\textsf{D}\textsf{:PDom:syll\_coda2}(y) \neq \textsc{null}] \wedge$
  $\textsf{F}_\textsf{D}\textsf{:PDom:syll\_ons}(x) \neq y \wedge$
  $[\textsf{vowel}(\textsf{F}_\textsf{M}\textsf{:PDom\_nuc}(x)) \wedge \textsf{F}_\textsf{M}\textsf{:PDom\_nuc}(x) = \textsf{F}_\textsf{L}\textsf{:succ:seg}(y)]$

---

[10]Even though the predicate uses a binary relation $\textsf{PDom:syll\_ons}(x, y)$, this predicate is QF because no quantifiers are used to find new variables. The unary function-version of this predicate would use $\textsf{F}_\textsf{M}\textsf{:PDom:syll\_ons}(x) = y$.

Together, the two helper predicates are used to license onsets for syllables in Copy 1 via the output function $\phi\texttt{PDom:syll\_ons}(x^1, y^1)$.

(407) *QF output function for outputting onsets for underlying syllables*

- $\phi\texttt{PDom:syll\_ons}(x, y) \stackrel{\text{def}}{=} \textbf{should\_\_PDom:onset\_\_old}(x, y) \vee$
  $\textbf{should\_\_PDom:onset\_\_new}(x, y)$

This is illustrated below. Note how all underlying onsets surface because they satisfy the helper predicate **should__PDom:onset__old**(x, y).

(408) *Resyllabifying a derivative* //amusín-agan// *– outputting onsets for underlying syllables in Copy 1*
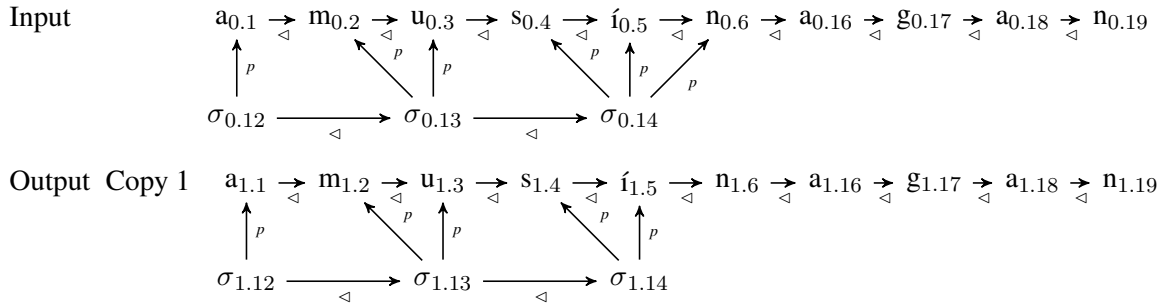
Output codas likewise come from two input sources: old input codas which are not resyllabified vs. unsyllabified segments which become codas. This distinction is captured with two sets of *old* vs. *new* helper predicates. First for the old codas, input inner and outer codas *should be* outputted faithfully in Copy 1 iff the underlying coda consonant y is not followed by a vowel z. This is formalized with the helper predicates below, which are QF-definable.

(409) a. *FO helper predicates for faithfully outputting underlying codas*

- $\textbf{should\_\_PDom:coda1\_\_old}(x, y) \stackrel{\text{def}}{=} \textsf{PDom:syll\_coda1}(x, y) \wedge \neg \exists z[\textsf{vowel}(z) \wedge \textsf{succ:seg}(y, z)]$
- $\textbf{should\_\_PDom:coda2\_\_old}(x, y) \stackrel{\text{def}}{=} \textsf{PDom:syll\_coda2}(x, y) \wedge \neg \exists z[\textsf{vowel}(z) \wedge \textsf{succ:seg}(y, z)]$

b. *QF helper predicates for faithfully outputting underlying codas*

- $\textbf{should\_\_PDom:coda1\_\_old}(x, y) \stackrel{\text{def}}{=} \textsf{PDom:syll\_coda1}(x, y) \wedge \neg \textsf{vowel}(\textsf{F}_\textsf{L}{:}\textsf{succ:seg}(y))$
- $\textbf{should\_\_PDom:coda2\_\_old}(x, y) \stackrel{\text{def}}{=} \textsf{PDom:syll\_coda2}(x, y) \wedge \neg \textsf{vowel}(\textsf{F}_\textsf{L}{:}\textsf{succ:seg}(y))$

As for new codas, an input unsyllabified consonant y should become the *new* inner coda of an input syllable x in Copy 1 only when that consonant follows the syllable's nucleus z: //.pa.ri. -<k>// $\rightarrow$ [.pa.rik.]. The helper predicate **should__PDom:coda1__new**(x, y) defines this context in Copy 1. Forming a new outer coda is similar but requires checking if y is preceded by a consonant z, such that z precedes y, z is inner coda, and zy can form an acceptable complex coda: //.kir.<k>// $\rightarrow$ [.kirk.].

(410) a. *FO helper predicates for outputting codas for underlying syllables*

- $\textbf{should\_\_PDom:coda1\_\_new}(x, y) \stackrel{\text{def}}{=} \textsf{consonant}(y) \wedge \textbf{unsyllabified}(y) \wedge$
  $\exists z[\textsf{vowel}(z) \wedge \textsf{succ:seg}(z, y) \wedge \textsf{PDom:syll\_nuc}(x, z)]$

- **should__PDom:coda2__new**$(x,y) \stackrel{\text{def}}{=}$ consonant$(y) \wedge$ **unsyllabified**$(y)\wedge$
  $\exists z[$consonant$(z) \wedge$ succ:seg$(z,y)\wedge$
  PDom:syll_coda1$(x,z) \wedge$ **good_CC**$(z,y)]$

b. *QF helper predicates for outputting codas for underlying syllables*
- **should__PDom:coda1__new**$(x,y) \stackrel{\text{def}}{=}$ consonant$(y) \wedge$ **unsyllabified**$(y)\wedge$
  vowel$(\mathsf{F_R}$:succ:seg$(y))\wedge\mathsf{F_R}$:succ:seg$(y) = \mathsf{F_M}$:PDom:syll_nuc(x)
- **should__PDom:coda2__new**$(x,y) \stackrel{\text{def}}{=}$ consonant$(y)\wedge$
  consonant$(\mathsf{F_R}$:succ:seg$(y))\wedge$
  $\mathsf{F_R}$:succ:seg$(y) = \mathsf{F_M}$:PDom:syll_coda1$(x)\wedge$
  good_CC$(\mathsf{F_R}$:succ:seg$(y),y)$

All these helper predicates for codas are used for the output functions below.

(411)   *QF output function for outputting old and new codas for underlying syllables*
- ϕPDom:syll_coda1$(x^1,y^1) \stackrel{\text{def}}{=}$ **PDom:coda1__old**$(x,y)\vee$**PDom:coda1__new**(x,y)
- ϕPDom:syll_coda2$(x^1,y^1) \stackrel{\text{def}}{=}$ **PDom:coda2__old**$(x,y)\vee$**PDom:coda2__new**(x,y)

In the case of $//$.a.mu.sín.- $<$agan$>//$, the only underlying coda *n* does not surface as a coda because it precedes a vowel; it does not satisfy **should__PDom:coda1__old**(x,y). These output functions produce the same result as in (408) above.

6.4.3   Syllabification of new material

The previous step in resyllabification was concerned in altering the structure of existing syllables from the input to Copy 1. With the right underlying syllable structure outputted, we can now syllabify new material. All formulas in this section are repeated from Chapter 5: §5.4.1 and are QF-definable. I do not repeat their QF versions.

Copy 1 includes all the prosodic dominances which will *surface* in the output. We use the user-defined predicate below to pick out any remaining unsyllabified material in Copy 1.

(412)   a. *FO user-defined predicate for finding unsyllabified segments in Copy 1*
- ϕ**unsyllabified**$(x^1) \stackrel{\text{def}}{=} \neg\exists y[\phi$syll$(y^1) \wedge [\phi$PDom:syll_ons$(y^1,x^1) \vee$
  ϕPDom:syll_nuc$(y^1,x^1)\vee$
  ϕPDom:syll_coda1$(y^1,x^1)\vee\phi$PDom:syll_coda2$(y^1,x^1)]]$

b. *QF user-defined predicate for finding unsyllabified segments in Copy 1*
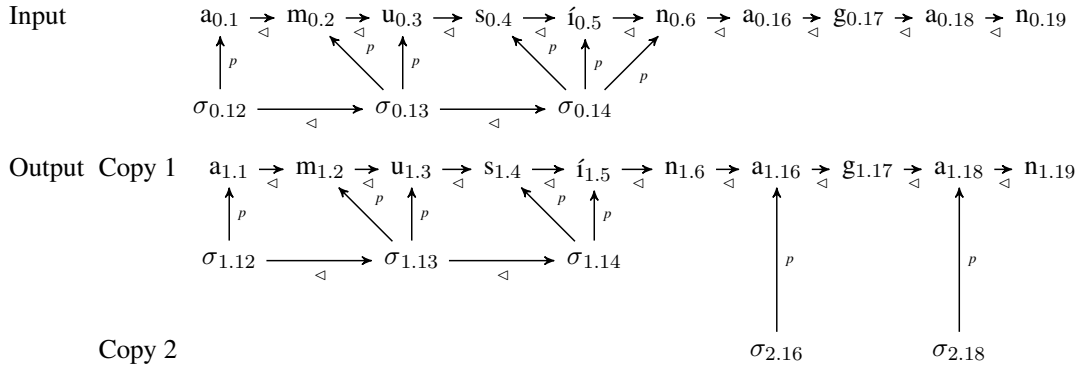- ϕ**unsyllabified**$(x^1) \stackrel{\text{def}}{=} \phi\mathsf{F}_D$:PDom:syll_ons$(x^1) =$ NULL $\wedge$
  $\phi\mathsf{F}_D$:PDom:syll_nuc$(x^1) =$ NULL $\wedge$
  $\phi\mathsf{F}_D$:PDom:syll_coda1$(x^1) =$ NULL $\wedge$
  $\phi\mathsf{F}_D$:PDom:syll_coda2$(x^1) =$ NULL

In Copy 2, new syllables are created for the suffix using the exact **same** functions as from Chapter 5: §5.4.1, repeated below. The suffix vowels erect syllables because they lack underlying syllables, while the base's vowels do not erect new syllables because they are syllabified in Copy 1.

(413)   a. *QF output function for syllable creation in Copy 2*
- $\phi\texttt{syll}(x^2) \stackrel{\text{def}}{=} \phi\texttt{vowel}(x^1) \wedge \phi\textbf{unsyllabified}(x^1)$

   b. *QF output function for nuclei assignment across Copy 1 and 2*
- $\phi\texttt{PDom:syll\_nuc}(x^2, x^1) \stackrel{\text{def}}{=} \phi\texttt{syll}(x^2) \wedge \phi\texttt{vowel}(x^1) \wedge \phi\textbf{unsyllabified}(x^1)$

This is illustrated below.

(414)   *Resyllabifying a derivative //amusín-agan// – creating new syllables and nuclei in Copy 2*

Input $a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to í_{0.5} \to n_{0.6} \to a_{0.16} \to g_{0.17} \to a_{0.18} \to n_{0.19}$

$\sigma_{0.12} \to \sigma_{0.13} \to \sigma_{0.14}$

Output   Copy 1 $a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \to í_{1.5} \to n_{1.6} \to a_{1.16} \to g_{1.17} \to a_{1.18} \to n_{1.19}$

$\sigma_{1.12} \to \sigma_{1.13} \to \sigma_{1.14}$

Copy 2 $\sigma_{2.16}$ $\sigma_{2.18}$

New syllables in Copy 2 take the right consonants as onsets by using the **same** formula from Chapter 5: §5.4.1, repeated below.

(415)   *FO output function for onset assignment across Copy 1 and 2*
- $\phi\texttt{PDom:syll\_ons}(x^2, y^1) \stackrel{\text{def}}{=} \phi\texttt{syll}(x^2) \wedge \phi\texttt{consonant}(y^1) \wedge \phi\textbf{unsyllabified}(y^1) \wedge$
  $\exists z[\phi\texttt{vowel}(z^1) \wedge \phi\texttt{succ:seg}(y^1, z^1) \wedge$
  $\phi\texttt{PDom:syll\_nuc}(x^2, z^1)]$

This is illustrated below. The input coda *n* of the root *a.mu.si<u>n</u>* is now an onset for the vowel *a* in *a.mu.si.<u>n-a</u>.gan*. The relevant nodes and indexes are $n_{1.6}$ $(=y^1)$, $a_{1.6}$ $(=z^1)$, and $\sigma_{2.16}$ $(=x^2)$. This is because the consonant $n_{1.6}$ is unsyllabified in Copy 1, i.e., it is not part of a syllable in Copy 1, and it is followed by a vowel $a_{1.6}$.

(416)   *Resyllabifying a derivative //amusín-agan// – creating new onsets for the new syllables in Copy 2*

Input $a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to í_{0.5} \to n_{0.6} \to a_{0.16} \to g_{0.17} \to a_{0.18} \to n_{0.19}$

$\sigma_{0.12} \to \sigma_{0.13} \to \sigma_{0.14}$

Output   Copy 1 $a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \to í_{1.5} \to n_{1.6} \to a_{1.16} \to g_{1.17} \to a_{1.18} \to n_{1.19}$

$\sigma_{1.12} \to \sigma_{1.13} \to \sigma_{1.14}$

Copy 2 $\sigma_{2.16}$ $\sigma_{2.18}$
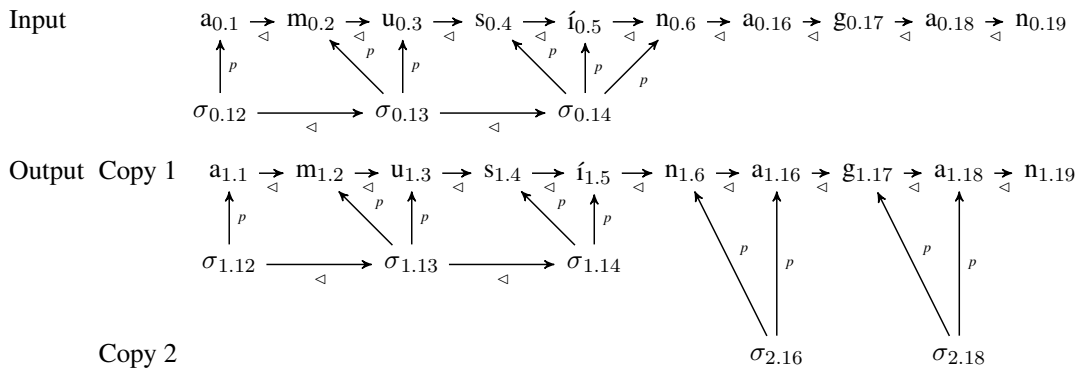
For codas, the only relevant consonant is *g*. It is syllabified as a coda with the same coda-assigning functions from Chapter 5: §5.4.1.

(417)   a. *QF user-defined predicate for checking acceptable complex codas*
- **good_CC**$(x, y) \overset{\text{def}}{=} [\mathsf{r}(x) \land \mathsf{s}(y)] \lor [\mathsf{r}(x) \land \mathsf{t}(y)] \lor \dots$
- ϕ**good_CC**$(x^1, y^1) \overset{\text{def}}{=} [\mathsf{r}(x^1) \land \mathsf{s}(y^1)] \lor [\mathsf{r}(x^1) \land \mathsf{t}(y^1)] \lor \dots$

   b. *FO output function for inner coda assignment across Copy 1 and 2*
- $\phi\mathtt{PDom:syll\_coda1}(x^2, y^1) \overset{\text{def}}{=} \phi\mathtt{syll}(x^2) \land \phi\mathtt{consonant}(y^1) \land \phi\textbf{unsyllabified}(y^1) \land$
$\exists z[\phi\mathtt{vowel}(z^1) \land \phi\mathtt{succ:seg}(z^1, y^1) \land] \land$
$\phi\mathtt{PDom:syll\_nuc}(x^2, z^1)] \land$
$\neg \exists u[\phi\mathtt{syll}(u^2) \land \phi\mathtt{PDom:syll\_ons}(u^2, y^1)]$

This is illustrated below. I do not go into detail about why the above output functions generate the right prosody. This was done in Chapter 5: §5.4.1 where I also showed that $\phi\mathtt{PDom:syll\_coda1}(x^2, y^1)$ is QF-definable. Interested readers are encouraged to review Chapter 5: §5.4.1 and work out the correctness for themselves.

(418)   *Resyllabifying a derivative* //amusín-agan// *– creating new codas for the new syllables in Copy 2*
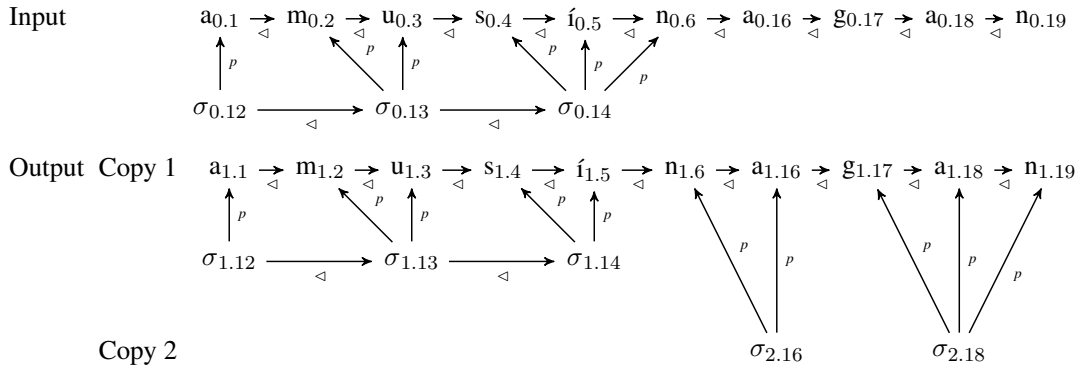
The final step is ordering all the syllables via $\mathsf{succ:syll}(x, y)$. The syllable-ordering transduction was defined and illustrated in Chapter 5: §5.4.2. I do not repeat the formula and explanation here. I show below the formalized input-output pair.

(419)   *Ordering syllables in a resyllabified derivative* //amusín-agan//
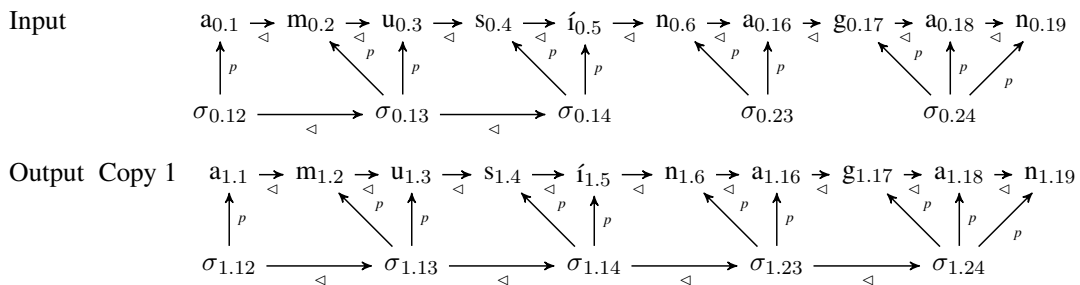
The take-away is that both syllabification and resyllabification are computationally local processes.

6.5   Locality in prosodic mapping

The previous section formalized the resyllabification of complex items, and established the computational locality of resyllabification. This section describes the range of possible prosodic parses for complex items, the formalization of such prosodic parses, and their computational locality.

I discuss the following types of prosodic parses. These concern the mapping and behavior of morphologically-derived prosodic constituents which are above the level of the syllable or foot. These different types of prosodic parses are manifest in the prosodification of derivational, inflectional, and compound morphology

(420)   *Aspects of prosodic mapping in complex words*

a. *Prosody of derivation*
   1. **Prosodic Restructuring**: a prosodic constituent expands and incorporates more syllables
   2. **Prosodic Recursion**: a prosodic constituent of level x dominates another constituent with the same label x
   3. **Prosodic Flattening**: a pair of recursive prosodic constituents of the same label x are flattened into a single constituent

b. *Prosody of inflection*
   1. **Prosodic Misalignment**: a prosodic constituent is associated with a morphological constituent, but the two do not dominate the same set of segments
   2. **Prosodic Layering**: a prosodic constituent of level x dominates constituents of lower levels $x - 1$ and/or $x - 2$

c. *Prosody of compounding*
   1. **Prosodic Linearization**: multiple prosodic constituents of the same label x are ordered together via immediate succession
   2. **Prosodic Subsumption**: a morphological constituent is not matched or wrapped into a prosodic constituent, but is subsumed (= broken up) into one.
   3. **Prosodic Fusion**: a sequence of multiple prosodic constituents of the same label x are fused into a single constituent of the same label x

I formalize prosodic mapping as a set of logical transductions. Each transduction formalizes one type of prosodic mapping, e.g., one transduction for PStem restructuring, another for PWord generation, etc. Some of these transductions feed other ones, i.e., prosodic recursion can feed prosodic flattening.

The morphological trigger for these prosodic parses is the properties of the topmost morphological node. The SETTINGS encapsulates such potentially global information that is relevant for the prosodic parse. By using both the SETTINGS and cyclic interactionist model, all the above are locally computed. As I explain later in Chapter 8: §8.3, without the SETTINGS, the prosody is still local; but post-cyclic non-interactionist prosody is not local.

6.5.1   Prosody of derivation

Derivational morphology triggers prosodic restructuring (§6.5.1.1) and the possibility of recursive prosody (§6.5.1.2). I show that these mappings are logically-definable and computationally local.
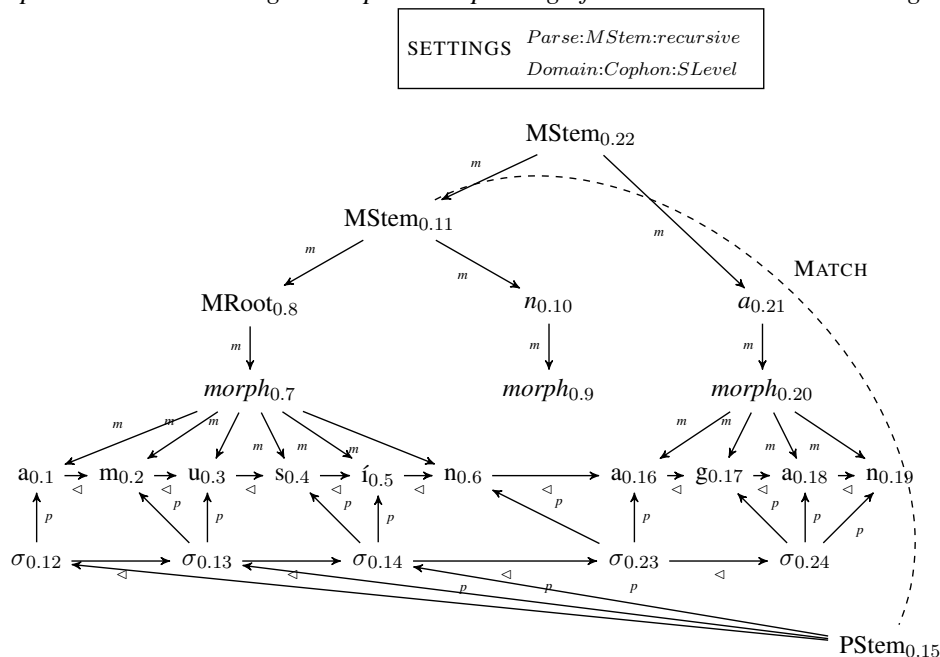
6.5.1.1   Prosodic restructuring

In prosodic restructuring, a prosodic constituent expands and incorporates more syllables. This is illustrated in the derivative //amusín-agan//. The input PStem is associated with root's MStem: //(amusí)$_s$n-agan//. The input's morphology contains an additional MStem layer. This MStem contains the suffix -*agan*, dominates the root's MStem, and is prosodically unparsed. In the output, the PStem should expand into the suffix and dominate its syllables: //(amusín-agan)$_s$//. I show that this process is computationally local and QF-definable.

(421)   *Prosodic restructuring in a derivative* //amusín-agan//
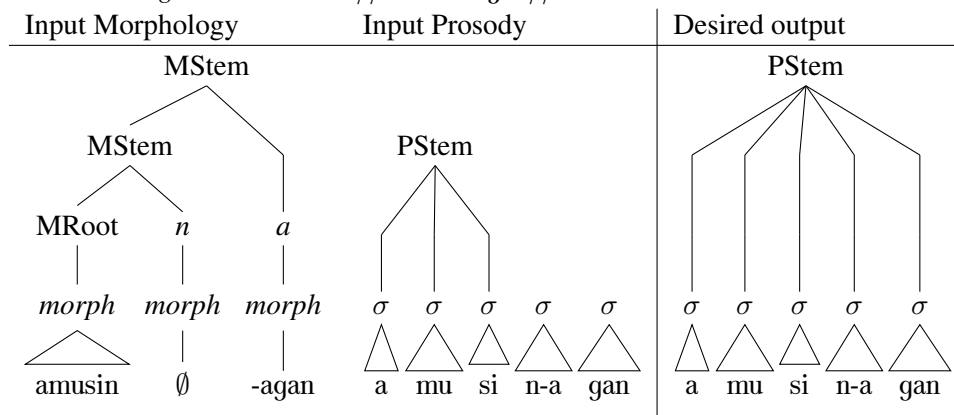
In the more explicit representation of the input, the PStem is matched with the root's MStem$_{0.11}$. The SETTINGS has the label Parse:MStem:recursive. This encodes the fact that the topmost node is an MStem$_{0.22}$ which is unparsed and recursively dominates a parsed MStem$_{0.11}$.

(422)   *Input to prosodic restructuring or the prosodic parsing of a recursive MStem into large PStem*

Prosodic restructuring is a transduction with a copy set of size 1. For brevity, I show the final output of restructuring below. The smaller $MStem_{1.11}$ is *wrapped* into the $PStem_{1.15}$, while the larger $MStem_{1.22}$ is *matched* with the $PStem_{1.15}$. These two types of prosodic binary relations were described in Chapter 4: §4.5.2.

(423) *Output of prosodic restructuring:* $//(\text{amusín-agan})_s//$
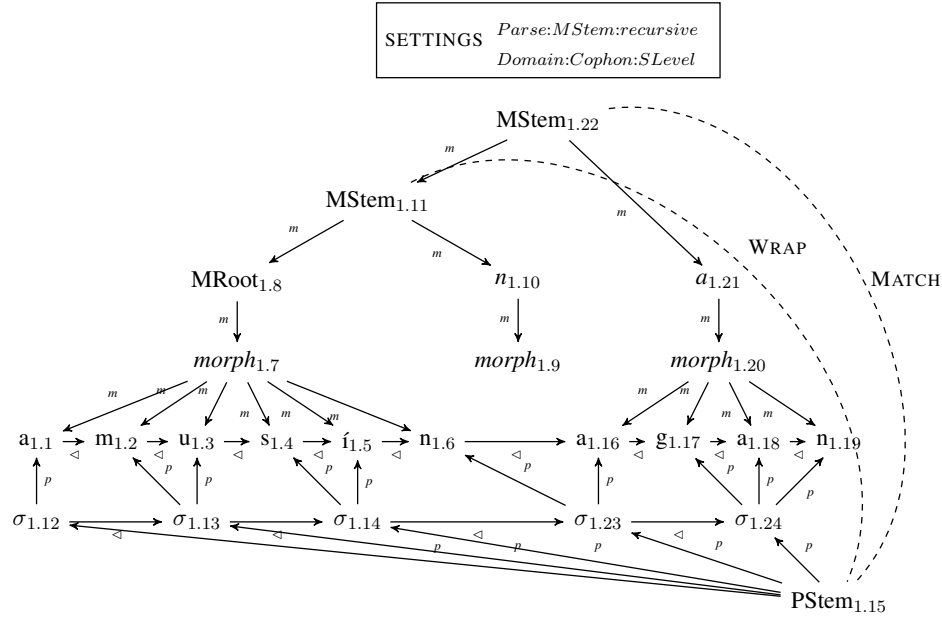
The following predicates pick out the topmost MStem and the PStem. The first predicate **MStem:unparsed**(x) picks the topmost unparsed $MStem_{0.22}$ (x) which dominates a parsed $MStem_{0.11}$ (u). The second predicate **PStem:expanding**(x) finds the 'expanding' $PStem_{0.15}$ (x) whose $MStem_{0.11}$ (y) is dominated by an unparsed topmost $MStem_{0.22}$ (z).[11] These predicates are computationally local and QF-definable.

(424) a. *FO user-defined predicates for finding the new larger MStem and the target expanding PStem*
   - **MStem:unparsed**$(x) \overset{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathbf{MTopmost}(x) \wedge \neg \exists y[\mathsf{PStem}(y) \wedge \mathsf{Match{:}stem}(x,y)]$
     $$\exists u,v[\mathsf{MStem}(u) \wedge \mathsf{MDom}(x,u) \wedge \mathsf{PStem}(v) \wedge \mathsf{Match{:}stem}(u,v)]$$
   - **PStem:expanding**$(x) \overset{\text{def}}{=} \mathsf{PStem}(x) \wedge$
     $$\exists y,z[\mathsf{Match{:}stem}(y,x) \wedge \mathbf{MStem{:}unparsed}(z) \wedge$$
     $$\mathsf{MDom}(z,y)] \wedge$$
     $$\neg \exists y[\mathsf{PStem}(y) \wedge \mathsf{succ{:}PStem}(x,y)]$$

   b. *QF user-defined predicates for finding the new larger MStem and the target expanding PStem*
   - **MStem:unparsed**$(x) \overset{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathbf{MTopmost}(x) \wedge \mathsf{F_L{:}Match{:}stem}(x) = \textsc{null}$
     $$\mathsf{MStem}(\mathsf{F_M{:}MDom}(x)) \wedge$$
     $$\mathsf{F_L{:}Match{:}stem}(\mathsf{F_M{:}MDom}(x)) \neq \textsc{null}$$

---

[11]PStem restructuring or expansion is largely due to suffixation; I assume an extra condition that a PStem is expanding if it is the rightmost PStem (= does not precede a PStem): $\neg \exists y[\mathsf{PStem}(y) \wedge \mathsf{succ{:}PStem}(x,y)]$. This condition is redundant here but may be useful if a future formalization looks at compounds before derivational suffixes.

- **PStem:expanding**$(x) \overset{\text{def}}{=} \mathsf{PStem}(x) \wedge$
  $$\mathbf{MStem:unparsed}(\mathsf{F_D:MDom}(\mathsf{F_R:Match:stem}(x)))$$
  $$\mathsf{F_L:succ:PStem}(x) = \text{NULL}$$

Generating the right output requires multiple steps. First, all node labels are faithfully outputted in Copy 1. All relations are likewise faithfully outputted *except* for any relations involving PStems. Throughout this section, we check that we have the right Parse label; this is locally accessed, thanks to the constant SETTINGS.

(425) *QF output functions to faithfully outputting labels and relations except for those involving PStems*

- For every label lab $\in L$:
  $\phi\mathtt{lab}(x^1) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \mathsf{lab}(x)$
- For every relation rel $\in R - \{\mathsf{Match:stem}, \mathsf{Wrap:stem}, \mathsf{PDom:PStem\_syll},$
  $$\mathsf{PDom:PStem\_PStem}, \mathsf{PDom:PWord\_PStem}\}:$$
  $\phi\mathtt{rel}(x^1, y^1) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \mathsf{rel}(x, y)$

In Copy 1, the expanding PStem y^1 (y is '0.15') should be matched with the larger unparsed MStem x^1 (x is '0.22'). This is encoded in the following output function.[12]

(426) *QF output function for matching the expanding PStem with the larger unparsed MStem*

- $\phi\mathtt{Match:stem}(x^1, y^1) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \mathsf{MStem}(x) \wedge \mathsf{PStem}(y) \wedge$
  $$\mathbf{MStem:unparsed}(x) \wedge \mathbf{PStem:expanding}(y)$$

The smaller MStem x^1 (x is '0.11') is associated with the expanding PStem y^1 (y is '0.15') via a wrapping relationship. Intuitively, the MStem is part of the larger PStem.[13]

---

[12]If the input contains multiple PStems, such that one PStem was expanding but the others were not, then we need a different version of the output function $\phi\mathtt{Match:stem}(x^1, y^1)$ to make sure that they should get associated with their MStems. This definition is useful if want to formalize cases where endocentric compounds undergo derivational morphology: $(abc)_s + (def)_s\text{-}ghi \rightarrow (abc)_s(def\text{-}ghi)_s$.

(1) *QF output function for matching a non-expanding PStem with its underlying MStem*

- $\phi\mathtt{Match:stem}(x, y) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \mathsf{MStem}(x) \wedge \mathsf{PStem}(y) \wedge$
  $$\mathsf{Match:stem}(x, y) \wedge \neg\mathbf{PStem:expanding}(y)$$

To make a unified grammar, the two definitions of $\phi\mathsf{Match:stem}(x^1, y^1)$ must be replaced with helper predicates: **should__Match:stem__new**(x, y) and **should__Match:stem__old**(x, y), and then unified with disjunction:

(2) *QF output function for matching the expanding PStem with the larger unparsed MStem*

- $\phi\mathtt{Match:stem}(x^1, y^1) \overset{\text{def}}{=} \mathbf{should\_\_Match:stem\_\_old}(x, y) \vee \mathbf{should\_\_Match:stem\_\_new}(x, y)$

[13]If the input contains a wrapped MStem, then this should faithfully surface as wrapped. This is the case for words with multiple derivational suffixes: $(abc\text{-}def)_s\text{-}ghi \rightarrow (abc\text{-}def\text{-}ghi)_s$ where each morpheme is separated by dashes and introduces a new MStem layer. This is formalized with the helper predicate below.

(1) *QF helper predicate for faithfully outputting underlying wrap relations*

- **should__Wrap:stem__old**$(x^1, y^1) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \mathsf{MStem}(x) \wedge \mathsf{PStem}(y) \wedge$
  $$\mathsf{Wrap:stem}(x, y)$$

(427)   *QF output function for wrapping the smaller MStem into the expanding PStem*

- $\phi\mathtt{Wrap:stem}(x^1, y^1) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\textsc{settings}) \wedge \mathsf{MStem}(x) \wedge \mathsf{PStem}(y) \wedge$
  $\mathbf{PStem:expanding}(y) \wedge \mathsf{Match:stem}(x, y)$

The expanding PStem must incorporate its old input syllables and newly incorporate the suffix's syllables. The following predicate finds the 'closest' syllables of a given MStem. This predicate is computationally local and QF-definable; see Chapter 5: §5.4.3 for the QF version of $\mathbf{syll\_of\_MStem}(x, y)$.[14]

(428)   *FO user-defined predicates for finding an MStem's closest syllables*

- $\mathbf{syll\_of\_MStem}(x, y) \overset{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{MStem}(y) \wedge$
  $\exists u, v, w[\mathsf{morpheme}(u) \wedge \mathsf{morph}(v) \wedge \mathsf{seg}(w) \wedge$
  $\mathsf{MDom}(y, u) \wedge \mathsf{MDom}(u, v) \wedge \mathsf{MDom}(v, w) \wedge \mathsf{PDom:syll\_nuc}(x, w)]$

As for syllable incorporation, this is facilitated by the use of helper predicates. The first predicate finds the syllables y which are underlyingly dominated by a PStem x; these prosodic dominances *should* be faithfully outputted. This helper predicates picks out the syllables $\sigma_{0.12}, \sigma_{0.13}, \sigma_{0.14}$. The second predicate finds the syllables y of the larger unparsed MStem z and it *should* give these syllables to the expanding PStem x. This helper predicates picks out the syllables $\sigma_{0.23}, \sigma_{0.24}$; it is QF-definable.

(429)   a. *QF helper predicate for letting a PStem dominates its underlying syllables*

- $\mathbf{should\_PDom:PStem\_syll\_old}(x, y) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\textsc{settings}) \wedge$
  $\mathsf{PStem}(x) \wedge \mathsf{syll}(y) \wedge \mathsf{PDom:PStem\_syll}(x, y)$

b. *FO helper predicate for letting the expanding PStem dominate the syllables of the new larger MStem's suffixes*

- $\mathbf{should\_PDom:PStem\_syll\_new}(x, y) \overset{\text{def}}{=} \mathsf{Parse:MStem:recursive}(\textsc{settings}) \wedge$
  $\mathsf{PStem}(x) \wedge \mathsf{syll}(y) \wedge \mathbf{PStem:expanding}(x) \wedge$
  $\exists z[\mathbf{MStem:unparsed}(z) \wedge \mathbf{syll\_of\_MStem}(y, z)]$

c. *QF helper predicate for letting the expanding PStem dominate the syllables of the new larger MStem's suffixes*

- $\mathbf{should\_PDom:PStem\_syll\_new}(x, y) \overset{\text{def}}{=}$
  $\mathsf{Parse:MStem:recursive}(\textsc{settings}) \wedge \mathsf{PStem}(x) \wedge \mathsf{syll}(y) \wedge$
  $\mathbf{PStem:expanding}(x) \wedge$
  $\mathbf{MStem:unparsed}(\mathsf{F_D:MDom}(\mathsf{F_R:Match:stem}(x))) \wedge$
  $\mathbf{syll\_of\_MStem}(y, \mathsf{F_D:MDom}(\mathsf{F_R:Match:stem}(x)))$

The above definition of $\phi\mathtt{Wrap:stem}(x^1, y^1)$ should be reformulated as a helper predicate $\mathbf{should\_Wrap:stem\_new}(x, y)$. The work of these two helper predicates is summarized in the output function below.

(2)   *QF output function for wrapping the right MStems into PStems*

- $\phi\mathtt{Wrap:stem}(x^1, y^1) \overset{\text{def}}{=} \mathbf{should\_Wrap:stem\_old}(x, y) \vee \mathbf{should\_Wrap:stem\_new}(x, y)$

[14]Intuitively, the MStem consists of a base MStem *amusin* and a suffixal morpheme *-agan*. The predicate $\mathbf{syll\_of\_MStem}(x, y)$ does not do a global examination of the larger MStem to find all of its syllables. It only picks the syllables whose nuclei are introduced by the MStem's principal morpheme, the suffix *-agan*. These suffixes are locally close to the larger MStem node; the base MStem node's syllables are too 'far away' from the topmost MStem node.

These helper predicates are used for the output function below which outputs all correct prosodic dominances from PStems to syllables.

(430)   *QF output function to make the larger PStem dominate the syllables of its two MStems*

- $\phi\texttt{PDom:PStem\_syll}(x^1, y^1) \stackrel{\text{def}}{=} \textbf{should\_\_PDom:PStem\_syll\_\_old}(x, y) \vee$
$$\textbf{should\_\_PDom:PStem\_syll\_\_new}(x, y)$$

This completes the *local* generation of a larger PStem from a recursive MStem via prosodic restructuring.

6.5.1.2   Prosodic recursion and flattening

When given multiple layers of recursive MNodes, this layering is prosodifed via prosodic restructuring in Armenian. The output contains a single PStem for multiple MStems. In contrast, some languages display prosodic recursion whereby multiple layers of recursive MNode are mapped to multiple recursive layers of PNodes (Ito and Mester 2009, 2012, 2013). For example in ChiChewa, multiple layers of MStems map to multiple recursive PStems (Downing 2016). However, Armenian does not display evidence for the prosodic recursion of PStems. Instead, PStems are flat and non-recursive.

(431)   *Parsing complex derivatives as prosodically flat, not prosodically recursive*
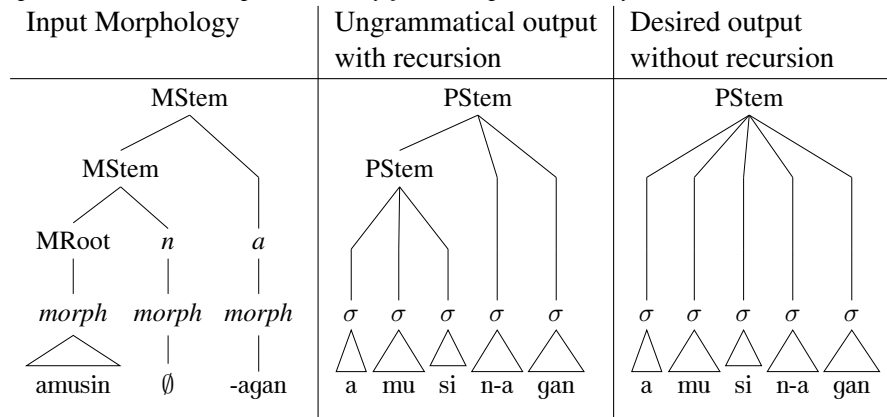
Computationally, recursive prosody is not necessarily more complex than non-recursive prosody. In fact, as an alternative to the previous section's formalization of prosodic restructuring, we can derive the output of prosodic restructuring from two steps: generating a recursive PStem, and flattening a recursive PStem. Each step is its own logical transduction.

(432) *Recursion+Flattening approach for prosodic restructuring in* //amusín-agan//
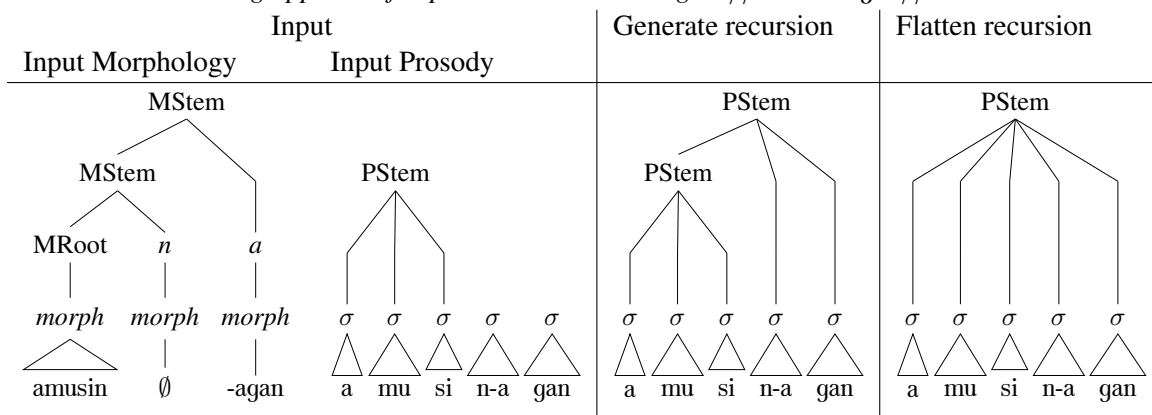
The composition of these two steps is prosodic restructuring. Each of these two steps is computationally local and QF-definable.[15] I define them separately. Both processes reference the prosodic parse label Parse:MStem:recursive(SETTINGS) which is true when the input contains a topmost recursive MStem.

6.5.1.2.1 Generating a recursive PStem

To generate a recursive PStem, we use a transduction with a copy set of size 2. I illustrate the input below.

(433) *Input for prosodic parsing of a recursive MStem into large PStem*

---
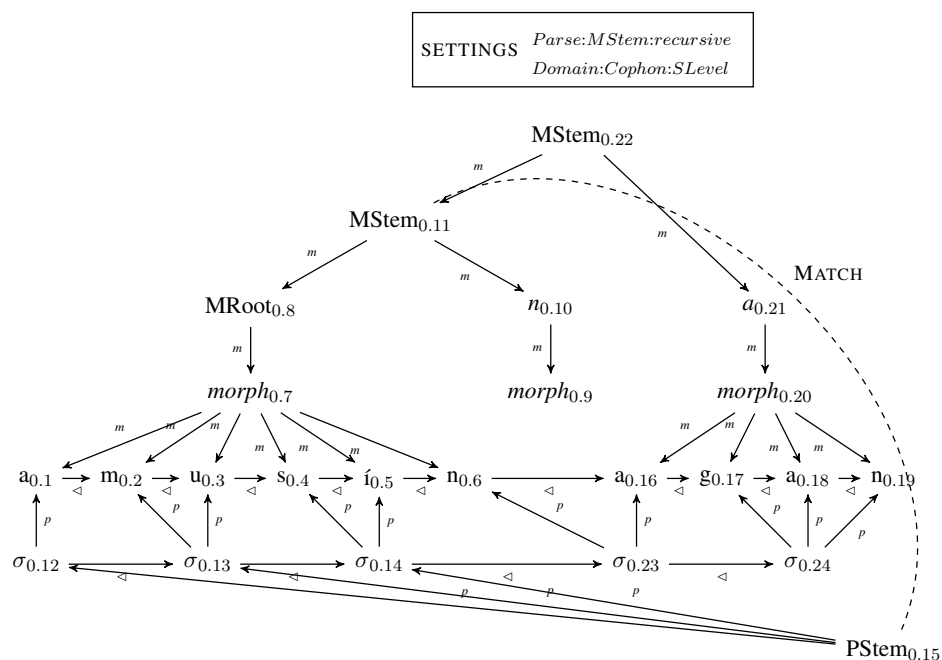
[15]Non-locality can arise when flattening potentially unbounded number of prosodic nodes in a compound, see Chapter 8: §8.3.2.2.

I show the output of prosodic recursion below. In the output, the topmost $\text{MStem}_{1.22}$ is matched with a new $\text{PStem}_{2.22}$. This $\text{PStem}_{2.22}$ dominates the old $\text{PStem}_{1.15}$ and the suffix's syllables $\sigma_{1.23}, \sigma_{1.24}$. I show that generating prosodic recursion is computationally local and QF-definable.

(434)   *Output of generating a recursive PStem in* $//((\text{amusín})_s\text{-agan})_s//$
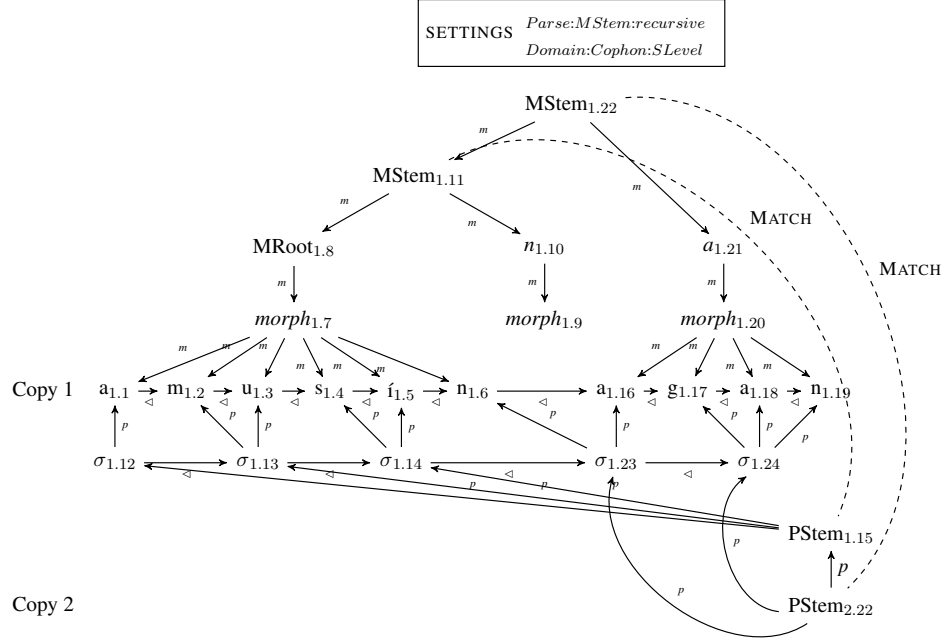
In Copy 1, all labels and relations are faithfully outputted. As before, all the functions here *explicitly* require that the derivation's SETTINGS has the right parse label: Parse:MStem:recursive(SETTINGS).

(435)   *QF output functions for vacuous identity in Copy 1*
- For every label $\text{lab} \in L$:
  $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{lab}(x)$
- For every relation $\text{rel} \in R$:
  $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{lab}(x)$

In Copy 2, a new $\text{PStem}_{2.22}$ is generated as an output correspondent for the topmost $\text{MStem}_{0.22}$ in the input. This is done via the output function below which selects the underlying $\text{MStem}_{0.22}$ which is morphologically topmost node; it is QF-definable. I redundantly specify that the relevant MStem must be underlyingly unparsed; this is redundant because that is implied from the SETTINGS's parse label.

(436)   a. *FO output function for generating a new PStem in Copy 2*
- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{MStem}(x) \wedge \textbf{MTopmost}(x) \wedge$
  $\neg\exists y[\text{PStem}(y) \wedge \text{Match:stem}(x, y)]$

b. *QF output function for generating a new PStem in Copy 2*

- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{MStem}(x) \wedge \textbf{MTopmost}(x) \wedge$
  $\mathsf{F_L}\text{:Match:stem} = \text{NULL}$

The large MStem and the new PStem must become associated via a MATCH relation. The relevant nodes are the larger input $\text{MStem}_{0.22}$ (x), its output correspondent $\text{MStem}_{1.22}$ in Copy 1 (x^1), and its output correspondent $\text{PStem}_{2.22}$ in Copy 2 $(x^2 = y^2)$. We prosodically match the output $\text{MStem}_{1.22}$ if it is an underlying MStem. It is matched with the $\text{PStem}_{2.22}$ in Copy 2, as long as they're both output correspondents of the same underlying $\text{MStem}_{0.22}$ $(x = y)$.

(437)　*QF output function for associating the new PStem in Copy 2 with the unparsed MStem*

- $\phi\texttt{Match:stem}(x^1, y^2) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge$
  $\text{MStem}(x) \wedge \phi\texttt{PStem}(y^2) \wedge x = y$

The larger PStem in Copy 2 prosodically dominates the syllables of the larger MStem's suffix in Copy 1 via the function below. These syllables $\sigma_{0.23}, \sigma_{0.24}$ are selected by using the predicate $\textbf{syll\_of\_MStem}(x, y)$. I described this predicate in Chapter 5: §5.4.3 and I showed that it is QF-definable.

(438)　a. *FO user-defined predicate for finding syllables of a non-recursive MStem*

- $\textbf{syll\_of\_MStem}(x, y) \stackrel{\text{def}}{=} \text{syll}(x) \wedge \text{MStem}(y) \wedge$
  $\exists u, v, w[\text{morpheme}(u) \wedge \text{morph}(v) \wedge \text{seg}(w) \wedge$
  $\text{MDom}(y, u) \wedge \text{MDom}(u, v) \wedge \text{MDom}(v, w) \wedge \text{PDom:syll\_nuc}(x, w)]$

b. *QF output function for making the larger PStem prosodically dominate the syllables of its MStem's suffix*

- $\phi\texttt{PDom:PStem\_syll}(x^2, y^1) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge$
  $\phi\texttt{PStem}(x^2) \wedge \phi\texttt{syll}(y^1) \wedge \textbf{syll\_of\_MStem}(y, x)$

The larger $\text{PStem}_{2.22}$ recursively dominates the smaller $\text{PStem}_{1.15}$. This requires a type of binary relation for recursive prosody: $\text{PDom:PStem\_PStem}(x, y)$ (cf. Chapter 4: §4.5.2). Recursive dominance is generated via the output function below. I assume that the two PStems are 'close' to each other, in that the larger $\text{PStem}_{2.22}$ (x^2) is associated with an $\text{MStem}_{1.22}$ (u^1) which immediately morphologically dominates the $\text{MStem}_{1.11}$ (v^1) of the smaller $\text{PStem}_{1.15}$ (y^1). Because of this assumption, the output function is QF-definable.[16]

(439)　a. *FO output function for making the larger PStem in Copy 2 prosodically dominate the smaller PStem in Copy 1*

- $\phi\texttt{PDom:PStem\_PStem}(x^2, y^1) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge$
  $\phi\texttt{PStem}(x^2) \wedge \phi\texttt{PStem}(y^1) \wedge$
  $\exists u, v[\phi\texttt{MStem}(u^1) \wedge \phi\texttt{MStem}(v^1) \wedge \phi\texttt{MDom}(u^1, v^1) \wedge$
  $\phi\texttt{Match:stem}(u^1, x^2) \wedge \phi\texttt{Match:stem}(v^1, y^1)]$

---

[16]If there is no guarantee of this relative proximity between the PStems and their associated MStems, then we may need to use global information in the morphological and prosodic tree in order to find the topmost MStem/PStem in the input (cf. Chapter 8: §8.3.2.2).

b. *QF output function for making the larger PStem in Copy 2 prosodically dominate the smaller PStem in Copy 1*

- $\phi\text{PDom:PStem\_PStem}(x^2, y^1) \stackrel{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge$
$$\phi\text{PStem}(x^2) \wedge \phi\text{PStem}(y^1) \wedge$$
$$\phi\text{F}_D\text{:MDom}(\phi\text{F}_R\text{:Match:stem}(y^1)) = \phi\text{F}_R\text{:Match:stem}(x^2)$$

The recursive structure is now properly generated, and it is locally generated. The next step is to flatten the pair of recursive PStems into a single PStem.

6.5.1.2.2   Flattening a recursive PStem

Flattening is a transduction with a copy set of size 1. The input is shown below. It is the recursive output of the previous section, but with updated indexes. Throughout this section, I assume that there are at most 2 layers of recursive PStems: $*(((abc)_{s1}\,def)_{s2}\,ghi)_{s3}$.[17] This ensures the locality of prosodic flattening.

(440)   *Input to flattening recursive PStems*
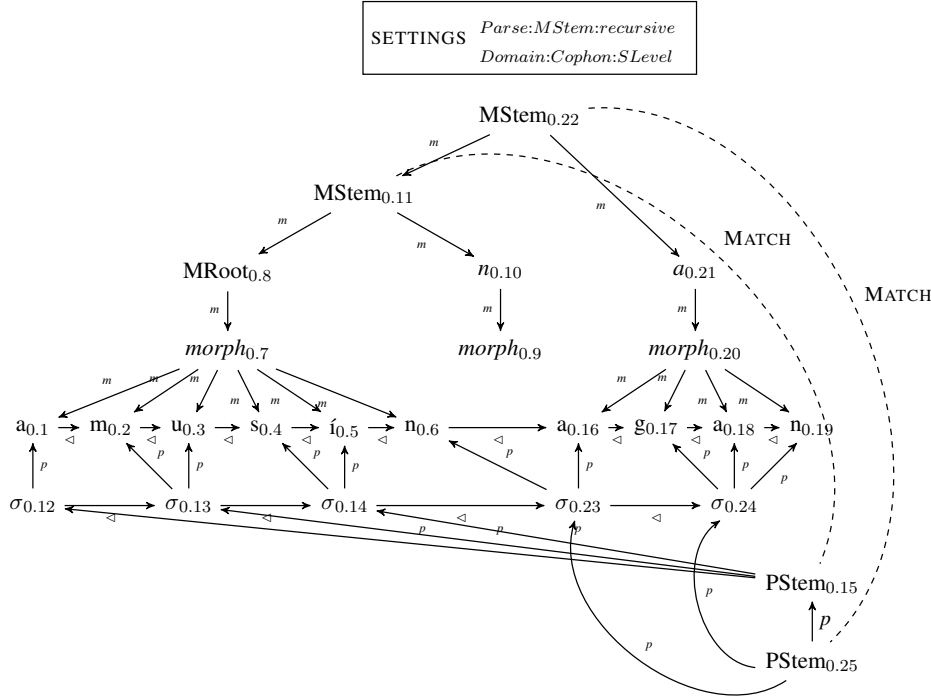
The output of flattening is shown below. The desired output has the smaller $\text{PStem}_{0.15}$ disappear. Its syllables ($\sigma_{1.12}, \sigma_{1.13}, \sigma_{1.14}$) and $\text{MStem}_{0.11}$ are re-associated with the larger $\text{PStem}_{1.25}$.

---

[17]If there are multiple layers of recursive PStems that we want to flatten, then the predicate **PStem:topmost_dominator_of**(x, y) must be redefined to find the topmost undominated PStem, *s3*. Doing so requires long-distance information.   It specifically requires defining a transitive closure for prosodic dominance among PStems PDom:PStem_PStem(x, y). This is discussed in Chapter 8: §8.3.2.2.

(441)   *Flattening a recursive PStem in* $//(\text{amusín-agan})_s//$
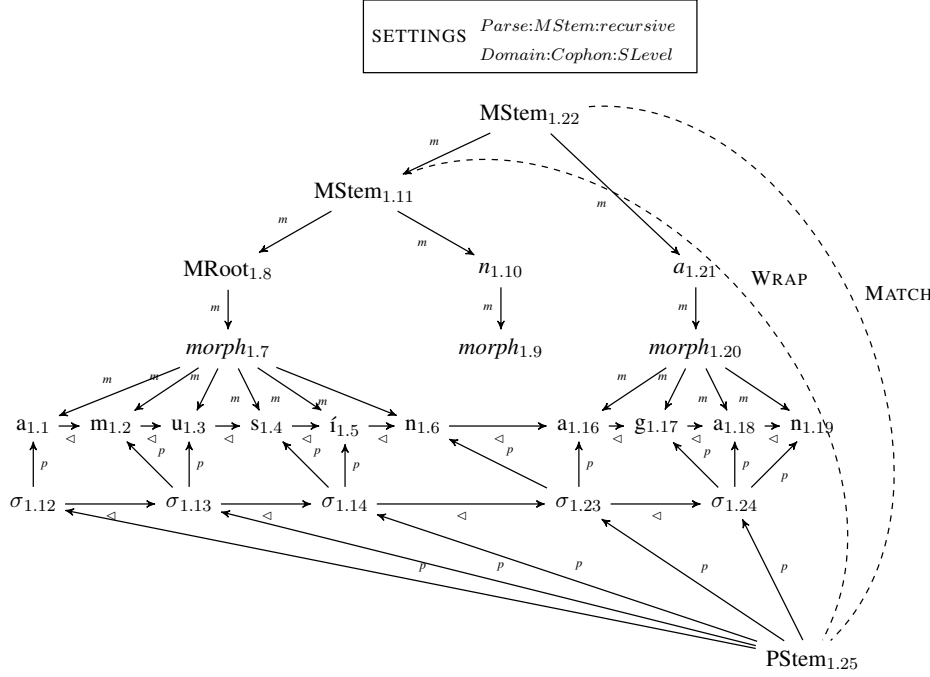
The relevant PStem nodes are picked out with the user-defined predicates below.[18]   These predicates are locally-computed and QF-definable. The predicate **PStem**:**dominated**(x) picks out PStems which recursively dominated by other PStems: $\text{PStem}_{0.15}$. The predicate **PStem**:**undominated**(x) picks out a PStem which is not recursively dominated, e.g., $\text{PStem}_{0.25}$. The predicate **PStem**:**topmost_dominator**(x, y) checks if two PStems x, y are in recursive dominance such that x is undominated.

(442)   a. *FO user-defined predicates to finding a recursively dominated PStem*
  - **PStem**:**dominated**$(x) \stackrel{\text{def}}{=} \text{PStem} \wedge \exists y[\text{PStem}(y) \wedge \text{PDom:PStem\_PStem}(y, x)]$

b. *QF user-defined predicates to finding a recursively dominated PStem*
  - **PStem**:**dominated**$(x) \stackrel{\text{def}}{=} \text{PStem} \wedge \mathsf{F_D}\text{:PDom:PStem\_PStem}(x) \neq \text{NULL}$

c. *QF user-defined predicates to finding the dominator x of a recursively undominated PStem y*
  - **PStem**:**topmost_dominator_of**$(x, y) \stackrel{\text{def}}{=} $ **PStem**:**undominated**$(x) \wedge$
  PStem:**dominated**$(y) \wedge$
  $\text{PDom:PStem\_PStem}(x, y)$

d. *QF user-defined predicates to finding any undominated PStems*
  - **PStem**:**undominated**$(x) \stackrel{\text{def}}{=} \text{PStem} \wedge \neg$**PStem**:**dominated**$(x)$

In Copy 1, all labels and relations are faithfully outputted except for those which involve PStems.[19] Note that again we specify that we have the right parse condition throughout the derivation.

---

[18]The predicates and functions have been carefully designed to also work when the input contains multiple non-recursive PStems, e.g., with endocentric compounds $(abc)_s(def)_s$, §6.5.3. However, I have not worked on any data or formalizations where compounds undergo further derivational morphology. This section would need to be tweaked if such data were to be found and formalized.

[19]For easier illustration, I ignore WRAP relations in PStems, prosodic dominances between PStems and segments, and prosodic dominances between PWords and PStems.

(443)   *QF output functions for faithfully outputting labels and relations except for those involving PStems*

- For every label lab $\in L - \{\text{PStem}\}$:
  $\phi\texttt{lab}(x^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{lab}(x)$
- For every relation rel $\in R - \{\text{Match:stem}, \text{Wrap:stem}, \text{PDom:PStem\_syll},$
  $\qquad\qquad\qquad\qquad\qquad \text{PDom:PStem\_PStem}, \text{PDom:PWord\_PStem}\}$:
  $\phi\texttt{rel}(x^1, y^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{rel}(x, y)$

Undominated PStems like $\text{PStem}_{1.25}$ are faithfully outputted via the output function below. The small $\text{PStem}_{1.15}$ is not outputted because it is recursively dominated in the input by $\text{PStem}_{0.25}$.

(444)   *QF output function for outputting undominated PStems in Copy 1*

- $\phi\texttt{PStem}(x^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \textbf{PStem:undominated}(x)$

An undominated $\text{PStem}_{1.25}$ is matched with its underlying $\text{MStem}_{1.22}$ via the output function below.

(445)   *QF output function for matching undominated PStems with their MStems in Copy 1*

- $\phi\texttt{Match:stem}(x^1, y^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge$
  $\qquad\qquad \textbf{PStem:undominated}(y) \wedge \text{Match:stem}(x, y)$

Given some dominated $\text{PStem}_{0.15}$, its $\text{MStem}_{1.11}$ is 'taken' by the dominator $\text{PStem}_{1.25}$ via wrapping the MStem into this dominator PStem. This function is locally-computed and QF-definable.[20]

(446)   a. *FO output function for wrapping a dominated PStem's MStems into the dominating PStem*

- $\phi\texttt{Wrap:stem}(x^1, y^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{MStem}(x) \wedge \text{PStem}(y) \wedge$
  $\qquad\qquad \exists z[\textbf{PStem:dominated}(z) \wedge \textbf{PStem:topmost\_dominator\_of}(y, z) \wedge$
  $\qquad\qquad [\text{Match:stem}(x, z) \vee \text{Wrap:stem}(x, z)]]$

   b. *QF output function for wrapping a dominated PStem's MStems into the dominating PStem*

- $\phi\texttt{Wrap:stem}(x^1, y^1) \overset{\text{def}}{=} \text{Parse:MStem:recursive}(\text{SETTINGS}) \wedge \text{MStem}(x) \wedge \text{PStem}(y) \wedge$
  $\qquad\qquad [[\textbf{PStem:dominated}(\text{F}_\text{L}:\text{Match:stem}(x)) \wedge$
  $\qquad\qquad\qquad \textbf{PStem:topmost\_dominator\_of}(\text{F}_\text{L}:\text{Match:stem}(x))] \vee$
  $\qquad\qquad [\textbf{PStem:dominated}(\text{F}_\text{L}:\text{Wrap:stem}(x)) \wedge$
  $\qquad\qquad\qquad \textbf{PStem:topmost\_dominator\_of}(\text{F}_\text{L}:\text{Wrap:stem}(x))]]$

An undominated $\text{PStem}_{1.25}$ will dominate its own syllables ($\sigma_{1.23}, \sigma_{1.24}$) and the syllables of its dominated PStem ($\sigma_{1.12}, \sigma_{1.13}, \sigma_{1.14}$). The different conditions for incorporation are facilitated by using the following helper predicates. The first predicate finds the syllables y dominated by the undominated PStem x; these prosodic dominances *should* be faithfully outputted. The second predicate finds the syllables y of a dominated PStem z and it *should* give them to z's dominating PStem x. This predicate is locally-computed and QF-definable.

---

[20]In the input, these input MStems can either be matched or wrapped into the dominated PStem. That is why we have the disjunct Wrap:stem(x, z).

(447)   a. *QF helper predicates for letting undominated PStems dominate their own syllables*
- **should__PDom:PStem_syll__old**$(x, y) \stackrel{\text{def}}{=}$
  Parse:MStem:recursive(SETTINGS) $\land$ PStem$(x) \land$ syll$(y) \land$
  PStem:undominated$(x) \land$ PDom:PStem_syll(x, y)

b. *FO helper predicates for letting undominated PStems dominate the syllables of their dominated PStems*
- **should__PDom:PStem_syll__new**$(x, y) \stackrel{\text{def}}{=}$
  Parse:MStem:recursive(SETTINGS) $\land$ PStem$(x) \land$ syll$(y) \land$
  PStem:undominated$(x) \land$
  $\exists z[$**PStem:dominated**$(z) \land$ PDom:PStem_syll(z, y)
  PStem:topmost_dominator_of$(x, z)]$

c. *QF helper predicates for letting undominated PStems dominate the syllables of their dominated PStems*
- **should__PDom:PStem_syll__new**$(x, y) \stackrel{\text{def}}{=}$
  Parse:MStem:recursive(SETTINGS) $\land$ PStem$(x) \land$ syll$(y) \land$
  PStem:undominated$(x) \land$
  PStem:dominated$($F$_\text{D}$:PDom:PStem_syll$(y)) \land$
  PStem:topmost_dominator_of$(x,$ F$_\text{D}$:PDom:PStem_syll$(y))$

These helper predicates are used for the output function below which outputs all correct prosodic dominances from PStems to syllables. This completes the flattening process.

(448)   *QF output function to make the larger PStem dominate the syllables of its two MStems*
- ϕPDom:PStem_syll$(x^1, y^1) \stackrel{\text{def}}{=}$ **should__PDom:PStem_syll__old**$(x, y) \lor$
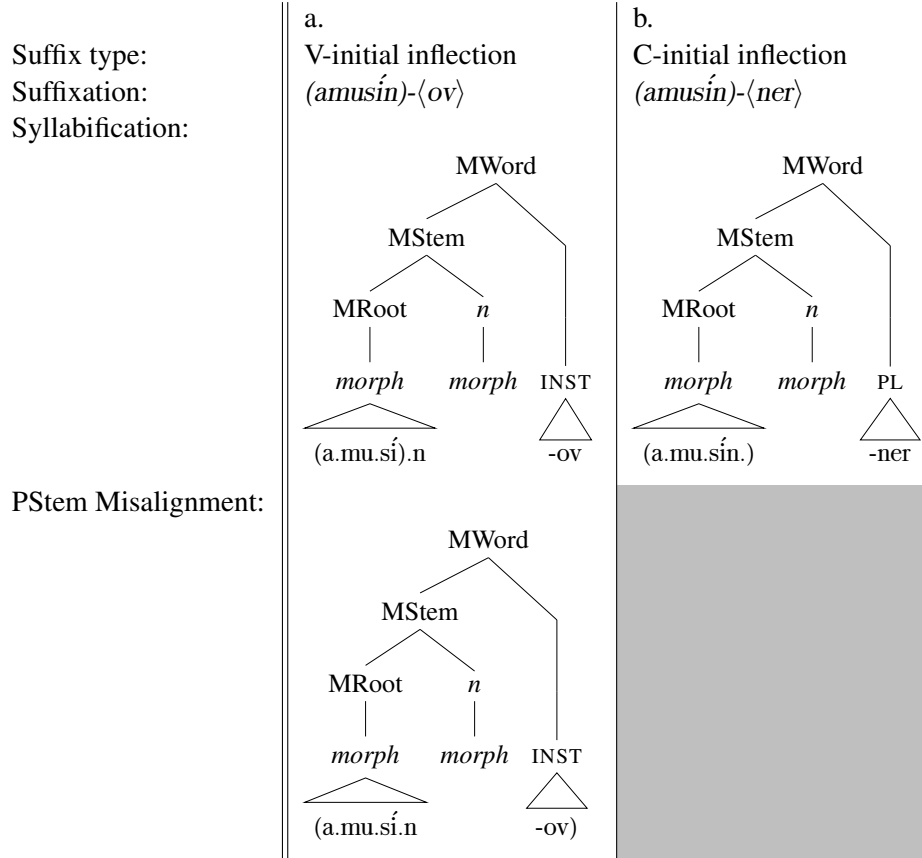 **should__PDom:PStem_syll__new**(x, y)

This completes prosodic flattening of two layers of recursive PStems. The computation was local.

6.5.2   Prosody of inflection

The previous sections looked at prosodic expansion and recursion, as they would appear in derivational morphology. But before inflectional morphology, we see prosodic misalignment and the generation of new prosodic constituents. I formalize both of these processes and show they are locally-computed.

6.5.2.1   Prosodic misalignment and overparsing

Prosodic misalignment is when a pair of matched prosodic and morphological constituents are non-isomorphic, i.e., the two constituents dominate different sets of segments (Nespor and Vogel 1986; Selkirk 1986, 1996, 2011; McCarthy and Prince 1993; Truckenbrodt 1995, 1999). I formalize prosodic alignment as it appears in Armenian. I show the morphological tree, and the prosodic stem is shown in parentheses.

(449)   *Illustrating different PStem-MStem alignments in inflection*

	a.	b.
Suffix type:	V-initial inflection	C-initial inflection
Suffixation:	*(amusín)-⟨ov⟩*	*(amusín)-⟨ner⟩*
Syllabification:		

PStem Misalignment:

Misalignment is triggered by the need to keep prosodic constituents well-aligned with syllable boundaries. Before C-initial inflection (449b), the PStem and MStem are well-aligned: $//(\text{amusín})_s\text{-ner}//$. One case of misalignment is prosodic underparsing (cf. prosodic undermatch in Guekguezian 2017a), whereby the PStem does not contain all the segments of its MStem. Before V-initial inflectional morphology (449a), the PStem is at first misaligned from its MStem and underparses it: $//(\text{amusí})_s\text{n-ov}//$. The final segment *n* of the MStem is resyllabified into the subsequent PStem-external syllable. The underparsed input is repaired by prosodic overparsing whereby the PStem expands into the inflectional suffix: $//(\text{amusín-ov})_s//$. The PStem overparses the MStem by containing segments *-ov* which aren't in its MStem.

I focus on the misalignment case in V-initial inflection $//(\text{amusí})\text{n}_s\text{-ov}//$. The input is explicitly shown below with all the relevant morphological and prosodic information. Because prosodic overparsing occurs at the juncture between derivation and inflection, it is limited to cases where the input contains a topmost MWord (inflection) which dominates an MStem (derivation). This context is encapsulated into the SETTINGS as the Parse label: Parse:MWord:nonrecursive.

(450)   *Underparsed input to PStem overparsing before V-initial inflection:* $//(\text{amusi})_s\text{n-ov}//$
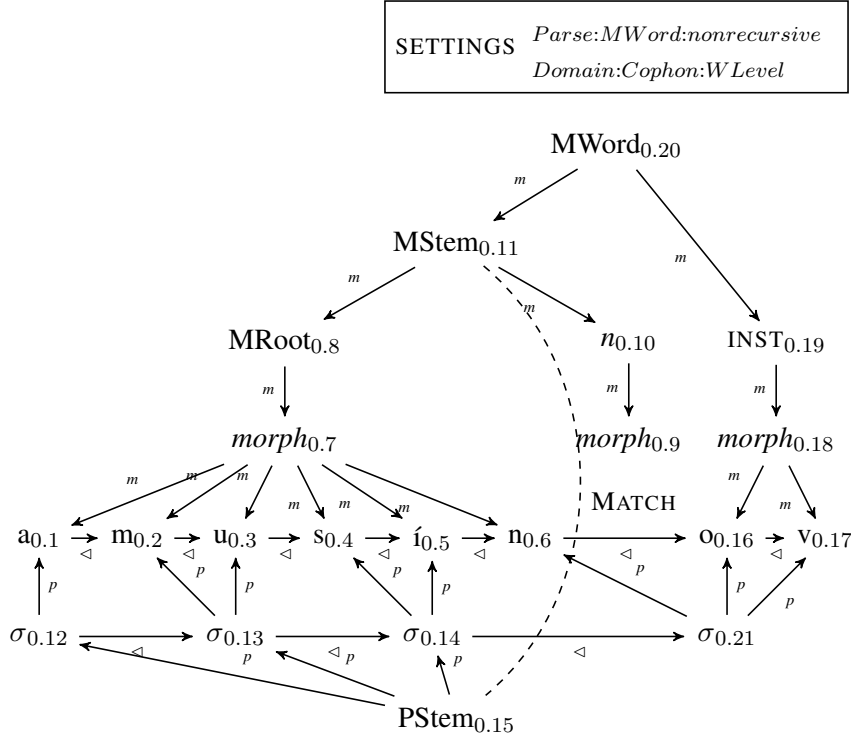

For easier illustration, I use the following predicates to pick out the relevant MStem and PStem which are involved in PStem misalignment: the rightmost $\text{PStem}_{0.15}$ and its $\text{MStem}_{0.11}$.[21] These predicates are computationally local and QF-definable.

(451)   a. *FO user-defined predicate to select the target PStem and target MStem*
   - $\textbf{TargetPStem}(x) \overset{\text{def}}{=} \text{PStem} \wedge \neg\exists y[\text{succ:PStem}(x,y)]$
   - $\textbf{TargetMStem}(x) \overset{\text{def}}{=} \text{MStem} \wedge \exists y[\textbf{TargetPStem}(y) \wedge \text{Match:stem}(x,y)]$

   b. *QF user-defined predicate to select the target PStem and target MStem*
   - $\textbf{TargetPStem}(x) \overset{\text{def}}{=} \text{PStem} \wedge \mathsf{F_L}\text{:succ:PStem}(x) = \text{NULL}$
   - $\textbf{TargetMStem}(x) \overset{\text{def}}{=} \text{MStem} \wedge \textbf{TargetPStem}(\mathsf{F_L}\text{:Match:stem}(x))$

PStem overparsing is a transduction with a copy set of size 1. I show the output below. The only change is that the target $\text{PStem}_{1.15}$ dominates the syllables of its $\text{MStem}_{1.11}$ and the syllable $\sigma_{1.21}$ of the suffix *-ov*.

---

[21]The condition on 'rightmost' is because of endocentric or hyponymic compounds which contains two PStems. Only the rightmost PStem is affected in inflection. In the case of non-compound words, there is always only one PStem.

(452)   *Output of PStem overparsing in V-initial inflection* $//(\text{amusín-ov})_s//$
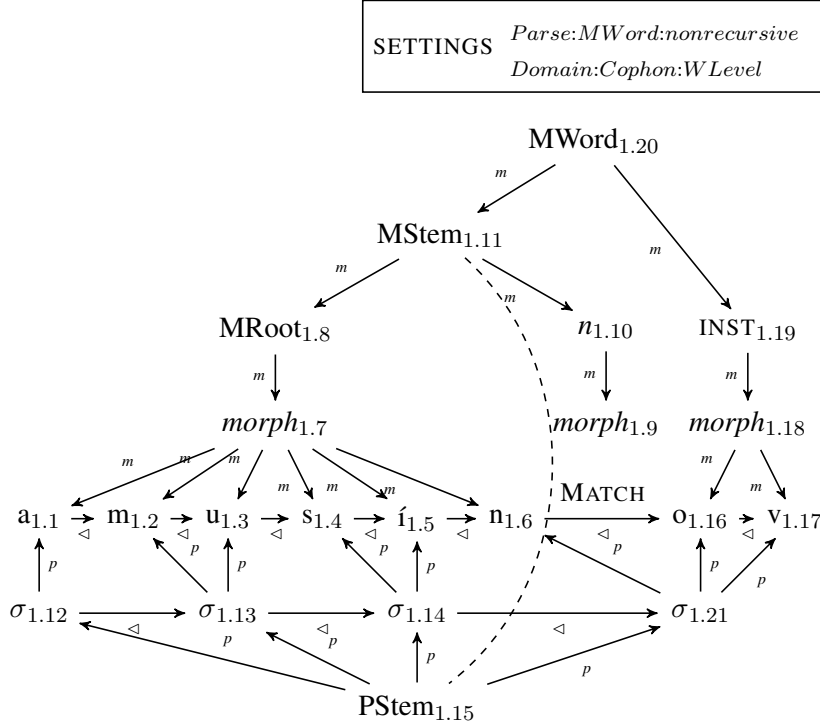
$$\boxed{\begin{array}{ll} \text{SETTINGS} & \textit{Parse:MWord:nonrecursive} \\ & \textit{Domain:Cophon:WLevel} \end{array}}$$

In Copy 1, all labels are faithfully outputted. In Copy 1, all relations are faithfully outputted except for the prosodic dominance between PStems and syllables. We need to check that we have the right parse SETTINGS, i.e., parsing a non-recursive MWord.

(453)   a. *QF output function to faithfully output every label during PStem misalignment*
   - For every label lab $\in L$:
     $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathsf{Parse:MWord:nonrecursive}(\text{SETTINGS})$

   b. *QF output function to faithfully output every relation except for the prosodic dominance of PStems to syllables*
   - For every relation rel $\in R - \{\mathsf{PDom:PStem\_syll}\}$:
     $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathsf{Parse:MWord:nonrecursive}(\text{SETTINGS})$

In the input, the syllable of the pre-inflectional segment is outside the PStem in V-initial inflection $//(\text{amusi.})_s\text{n-ov}//$, and inside the PStem in C-initial inflection $//(\text{amusin})_s\text{-ner}//$. But in the output, the PStem dominates the syllable of the pre-inflectional segment regardless if the segment is underlyingly outside the PStem (in V-initial inflection) or underlyingly inside the PStem (in C-initial inflection): *(amusi.n-ov)$_s$, (amusin)$_s$-ner*. The basic generalization in prosodic overparsing is that the PStem *must* contain its MStem's final segment $n_{0.6}$. There are two ways to find this segment. One requires global computation by referencing the target MStem and PStem; another uses local information because it references the morpheme boundary between derivation and inflection. I illustrate both approaches and use the latter.

The most intuitive way to find this segment requires global computation. Given some segment $n_{0.6}$ (x), the predicate **seg_in_TargetMStem**(x) checks if $n_{0.6}$ is in the target MStem$_{0.11}$ (y). It does so by checking

if the MStem generally morphologically-dominates x. General morphological dominance was defined in Chapter 4: §4.5.1. There can be an unbounded number of MNodes which separate the segment x and the target MStem y.[22] The predicate **final_seg_in_TargetMStem**(x) then checks if x is the final segment of the target MStem$_{0.11}$.

(454)   a. *MSO user-defined predicate for non-locally checking if a segment is in the target MStem*
   - **seg_in_TargetMStem**$(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \exists y[\textbf{TargetMStem}(y) \wedge \textbf{gen\_MDom}(y, x)$

   b. *MSO user-defined predicate for non-locally checking if a segment is the final segment in the target MStem*
   - **final_seg_in_TargetMStem**$(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \textbf{seg\_in\_TargetMStem}(x) \wedge$
     $$\neg \exists y[\mathsf{seg}(y) \wedge \mathsf{succ:seg}(x, y) \wedge \textbf{seg\_in\_TargetMStem}(y)]$$

The above two predicates are computationally non-local because they reference long-distance morphological dominance. Any formulas which reference these predicates are likewise non-local. An alternative local definition for **final_seg_in_MStem**(x) is to exploit the fact that MStems are always inside MWords, and that the target MStem is the rightmost MStem. A priori, a segment $n_{0.6}$ (x) is the rightmost segment in the target MStem if:

1. x is part of an MStem
2. x precedes an inflectional suffix y which is part of an MWord

I will redefine the predicate **final_seg_in_TargetMStem**(x) using this information. These conditions

---

[22]This definition uses general dominance because it is possible that the input has multiple layers of MNodes separating the MStem-final segment from the topmost MStem. These layers could be triggered by zero suffixes or by prefixes. In this case, global information is needed. Consider a nonce word with multiple nonce prefixes *un-* and zero suffixes -$\emptyset$: un*-(amusi)$_\mathrm{s}$-n-$\emptyset^*$-ov. The target MStem is MStem$_z$. The target MStem$_z$ and its final segment n are underlined. The PStem is in parentheses.

(1)   *Tree with unbounded number of nodes between the target MStem and its final segment*
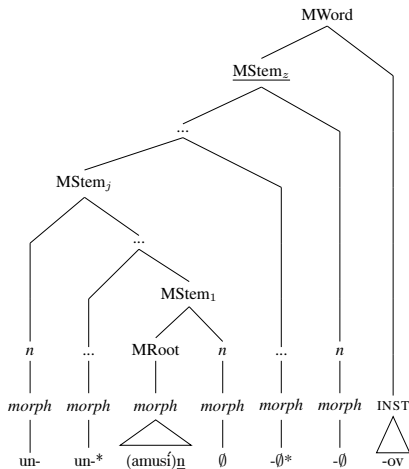
The deeply embedded MStem is MStem$_1$ and it has the root and PStem. This MStem is preceded by a sequence of prefixes. These prefixes form MStem$_2$ to MStem$_j$ and they dominate the root. The root is followed by a sequence of zero suffixes from MStem$_{j+1}$ to MStem$_z$. These MStem dominate MStem$_j$. Thus, the zero suffixes scope over the prefixes.

In order to detect check that the segment n is dominated by MStem$_z$, we need to traverse all the MStem nodes from MStem$_2$ to MStem$_z$. This requires some form of long-distance computation such as with **gen_MDom**(x, y).

are formalized with the following predicates. The predicate $\mathbf{seg\_in\_MStem}(x)$ is satisfied by the root segments $a_{0.1}$–$n_{0.6}$. The predicate $\mathbf{seg\_in\_MWord}(x)$ is satisfied by the suffix segments $o_{0.16}, v_{0.17}$. Similar to the QF predicate $\mathbf{syll\_of\_MStem}(x, y)$ These two predicates check for the 'closest' MNode which dominates the segments, i.e., the MNode of the segment's morpheme. The predicate $\mathbf{seg\_pre\_infl}(x)$ is satisfied by the MStem-final segment $n_{0.6}$ because it precedes an inflectional segment $o_{0.16}$.

(455) a. *FO user-defined predicate for checking if a segment is part of an MStem*
- $\mathbf{seg\_in\_MStem}(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \exists u, v, w[\mathsf{MStem}(u) \wedge \mathsf{morpheme}(v) \wedge \mathsf{morph}(w) \wedge$
  $\mathsf{MDom}(u, v) \wedge \mathsf{MDom}(v, w) \wedge \mathsf{MDom}(w, x)]$

b. *FO user-defined predicate for checking if a segment is part of an inflectional morpheme in an MWord*
- $\mathbf{seg\_in\_MWord}(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \exists u, v, w[\mathsf{MWord}(u) \wedge \mathsf{morpheme}(v) \wedge \mathsf{morph}(w) \wedge$
  $\mathsf{MDom}(u, v) \wedge \mathsf{MDom}(v, w) \wedge \mathsf{MDom}(w, x)]$

c. *FO user-defined predicate for checking if a segment is pre-inflectional*
- $\mathbf{seg\_pre\_infl}(x) \stackrel{\text{def}}{=} \mathbf{seg\_in\_MStem}(x) \wedge \exists y[\mathsf{seg}(y) \wedge \mathsf{succ{:}seg}(x, y) \wedge \mathbf{seg\_in\_MWord}(y)]$

All of these predicates can be locally computed and are QF-definable.

(456) a. *QF user-defined predicate for checking if a segment is part of an MStem*
- $\mathbf{seg\_in\_MStem}(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \mathsf{morph}(\mathsf{F_D{:}MDom}(x)) \wedge \mathsf{morpheme}(\mathsf{F_D{:}MDom^2}(x)) \wedge$
  $\mathsf{MStem}(\mathsf{F_D{:}MDom^3}(x))$

b. *QF user-defined predicate for checking if a segment is part of an inflectional morpheme in an MWord*
- $\mathbf{seg\_in\_MWord}(x) \stackrel{\text{def}}{=} \mathsf{seg}(x) \wedge \mathsf{morph}(\mathsf{F_D{:}MDom}(x)) \wedge \mathsf{morpheme}(\mathsf{F_D{:}MDom^2}(x)) \wedge$
  $\mathsf{MWord}(\mathsf{F_D{:}MDom^3}(x))$

c. *QF user-defined predicate for checking if a segment is pre-inflectional*
- $\mathbf{seg\_pre\_infl}(x) \stackrel{\text{def}}{=} \mathbf{seg\_in\_MStem}(x) \wedge \mathbf{seg\_in\_MWord}(\mathsf{F_L{:}succ{:}seg}(x))$

The PStem must dominate whatever syllable $\sigma_{0.21}$ contains this pre-inflectional segment $n_{0.6}$. The predicate $\mathbf{syll\_of\_pre\_infl\_seg}(x)$ picks out this syllable x which dominates some pre-inflectional segment y. It checks if the syllable x dominates some segment y, such that y is pre-inflectional. The predicate is locally computable.

(457) a. *FO user-defined predicate for picking syllable of the pre-inflectional segment*
- $\mathbf{syll\_of\_pre\_infl\_seg}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \exists y[\mathsf{seg}(y) \wedge \mathbf{seg\_pre\_infl}(y) \wedge$
  $[\mathsf{PDom{:}syll\_ons}(x, y) \vee \mathsf{PDom{:}syll\_nuc}(x, y) \vee$
  $\mathsf{PDom{:}syll\_coda1}(x, y) \vee \mathsf{PDom{:}syll\_coda2}(x, y)]$

b. *QF user-defined predicate for picking syllable of the pre-inflectional segment*
- $\mathbf{syll\_of\_pre\_infl\_seg}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge [\mathbf{seg\_pre\_infl}(\mathsf{F_M{:}PDom{:}syll\_ons}(x)) \vee$
  $\mathbf{seg\_pre\_infl}(\mathsf{F_M{:}PDom{:}syll\_nuc}(x)) \vee$
  $\mathbf{seg\_pre\_infl}(\mathsf{F_M{:}PDom{:}syll\_coda1}(x)) \vee$
  $\mathbf{seg\_pre\_infl}(\mathsf{F_M{:}PDom{:}syll\_coda2}(x))]$

In terms of syllable incorporation, there are two sources for the PStem's syllables. In one condition, the PStem dominates all (old) syllables which it did dominate in the input. In the other condition, the PStem dominates the (new) syllable of the suffix. These conditions are encoded via two helper predicates. For the first condition, the PStem should faithfully dominate all of its underlying syllables.

(458)   *QF helper predicate for letting PStems faithfully dominate their underlying syllables*
- **should_PDom:PStem_syll__old**$(x, y) \overset{\text{def}}{=}$ Parse:MWord:nonrecursive(SETTINGS)$\wedge$ PDom:PStem_syll(x, y)

For the second condition, the target PStem$_{0.15}$ (x) must dominate the syllable $\sigma_{0.21}$ (y) of the pre-inflectional segment $n_{0.6}$.

(459)   *QF helper predicate for making the target PStem dominate the syllable of the pre-inflectional segment*
- **should_PDom:PStem_syll__new**$(x, y) \overset{\text{def}}{=}$ Parse:MWord:nonrecursive(SETTINGS)$\wedge$ **TargetPStem**$(x) \wedge$ **syll_of_pre_infl_seg**(y)

The work of these two helper predicates is summarized in the output function below.

(460)   *QF helper predicate to summarize what the PStem should dominate*
- ϕPDom:PStem_syll$(x, y) \overset{\text{def}}{=}$ **should__PDom:PStem_syll__old**$(x, y) \vee$ **should__PDom:PStem_syll__new**(x, y)

This is visualized below. I omit the morphological nodes and SETTINGS for easier illustration.

(461)   *PStem overparsing – PStem dominates its underlying syllables*
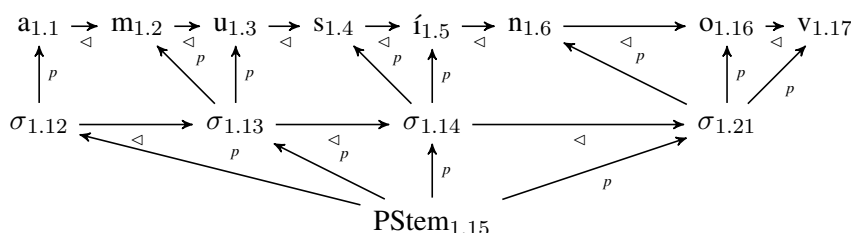
In sum, prosodic misalignment and prosodic overparsing is computationally local.

6.5.2.2   Prosodic layering and generation of a prosodic word

All the previous sections concerned the generation of prosodic stems. Prosodic layering is when different types of prosodic constituents are present in the same word. Above PStems, prosodic structure can contain multiple types of distinct morphosyntactically-derived constituents, such as prosodic words or PWords. Here, I formalize the generation of prosodic words and how they dominate smaller prosodic constituents.

All words in Armenian contain a PWord. The PWord overlaps with the PStem when the word is uninflected $((amusin)_s)_w$ (462a) or has V-initial inflection $((amusin\text{-}ov)_s)_w$ (462b). If the word has C-initial inflection, the PWord is larger than the PStem because C-initial inflection is PStem-external: $((amusin)_s\text{-}ner)_w$ (462c). I illustrate PWord generation in $((amusin)_s\text{-}ner)_w$, and I show that it is a computationally local process.

(462)   *Input and output of PWord generation*

	Input Morphology	*Input Prosody*	*Output Prosody*
a. Uninflected 'husband'	MWord — MStem (MRoot, *n*), NOM; MRoot → *morph* → amusín; *n* → *morph* → ∅; NOM → −∅	PStem (σ σ σ): a mu sín	PWord — PStem (σ σ σ): a mu sín
b. V-initial inflection 'husband-INST'	MWord — MStem (MRoot, *n*), INST; MRoot → *morph* → amusín; *n* → *morph* → ∅; INST → -ov	PStem (σ σ σ σ): a mu sí n-ov	PWord — PStem (σ σ σ σ): a mu sí n-ov
c. C-initial inflection 'husband-PL	MWord — MStem (MRoot, *n*), INST; MRoot → *morph* → amusín; *n* → *morph* → ∅; INST → -ner	PStem (σ σ σ σ): a mu sín ner	PWord — PStem (σ σ σ) ner: a mu sín ner

I am agnostic over whether PWord generation occurs simultaneously or right after PStem overparsing. For illustration, I assume PWord generation happens after PStem overparsing. PWords are first generated when the input contains an MWord. The main trigger for PWord generation is the SETTINGS label Parse:MWord:nonrecursive. A PWord is generated if the topmost MNode in the input is a (recently added) MWord which dominates an MStem. I show the explicit input and output for the C-inflected word *amusin-ner*.

(463)   a. *Input of PWord generation:* $//(\text{amus}\acute{\text{i}}\text{n})_s\text{-ner}//$

b. *Output of PWord generation:* $//((\text{amus}\acute{\text{i}}\text{n})_s\text{-ner})_w//$

Erecting the PWord is a transduction with a copy set of size 2. The main task is mapping the $\text{MWord}_{0.21}$ into a $\text{PWord}_{2.21}$, matching them, and letting this PWord dominate the right prosodic constituents. The predicates and formula used to erect a PWord are analogous to the ones used for PStems (Chapter 5: §5.4.3).

Only the name 'PStem' and 'MStem' are replaced with 'PWord' and 'MWord'.

In Copy 1, all labels and relations are faithfully outputted. We need to check the SETTINGS to make sure that we are parsing a non-recursive MWord.

(464)   *QF output functions for faithfully outputting node labels and relations in Copy 1*

- For every label lab $\in L$:
  $\phi\texttt{lab}(x^1) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge \textsf{lab}(x)$
- For every relation rel $\in R$:
  $\phi\texttt{rel}(x^1, y^1) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge \textsf{rel}(x, y)$

In Copy 2, a $\text{PWord}_{2.21}$ is generated as an output correspondent for the input $\text{MWord}_{0.21}$. Redundantly, this MWord should be unparsed in the input. This is redundant because the parse label on the SETTINGS ensures that. This is all locally computible.

(465)   a. *FO output function for generating a PWord in Copy 1*
- $\phi\texttt{PWord}(x^2) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge$
  $\textsf{MWord}(x) \wedge \neg\exists y[\textsf{PWord}(y) \wedge \textsf{Match:word}(x, y)]$

b. *QF output function for generating a PWord in Copy 1*
- $\phi\texttt{PWord}(x^2) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge$
  $\textsf{MWord}(x) \wedge \textsf{F}_\textsf{L}\textsf{:Match:word}(x) = \textsc{null}$

The output $\text{MWord}_{1.21}$ and $\text{PWord}_{2.21}$ are associated via the $\textsf{Match:word}(x, y)$ relation, which is locally computible.

(466)   a. *FO output function for associating the output PWord and MWord via a* MATCH *relation*
- $\phi\texttt{Match:word}(x^1, y^2) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge$
  $\phi\texttt{MWord}(x^1) \wedge \phi\texttt{PWord}(y^2) \wedge x = y \wedge$
  $\neg\exists z[\textsf{PWord}(z) \wedge \textsf{Match:word}(x, z)]$

b. *QF output function for associating the output PWord and MWord via a* MATCH *relation*
- $\phi\texttt{Match:word}(x^1, y^2) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge$
  $\phi\texttt{MWord}(x^1) \wedge \phi\texttt{PWord}(y^2) \wedge x = y \wedge$
  $\textsf{F}_\textsf{L}\textsf{:Match:word}(x) = \textsc{null}$

The $\text{PWord}_{2.21}$ must dominate any PStems ($\text{PStem}_{1.15}$) which are part of its MWord. For simplicity, let us assume that the derivation is word-bounded; meaning that the input is never something larger like a phrase. Thus, any input PStems are part of the MWord.

(467)   *QF output function for letting the PWord dominate the PStem*

- $\phi\texttt{PDom:PWord\_PStem}(x^2, y^1) \overset{\text{def}}{=} \textsf{Parse:MWord:nonrecursive}(\textsc{settings}) \wedge$
  $\phi\texttt{PWord}(x^2) \wedge \phi\texttt{PStem}(y^1)$

Some syllables are dominated by the PStem ($\sigma_{1.12}, \sigma_{1.13}, \sigma_{1.14}$) while some syllables are dominated by the PWord ($\sigma_{1.22}$). The new PWord$_{2.21}$ must dominate the syllables of the MWord if these syllables are not already parsed into the PStem, i.e., the syllable of the inflectional suffix *-ner*. This is facilitated by the predicates below. The predicate **syll_of_MWord(x, y)** picks out the syllables x whose nuclei w are part of an MWord y. Similar predicates were used to find syllables in an MStem (Chapter 5: §5.4.3). As with MStems, the predicate below will pick out the syllables which are the 'closest' to the MWord, i.e., separated by only a morpheme, morph, and nucleus segment. This predicate is locally computed and QF-definable.

(468) a. *FO user-defined predicate for selecting the syllables introduced by an MWord*
- **syll_of_MWord**$(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{MWord}(y) \wedge$
  $\exists u, v, w[\mathsf{morpheme}(u) \wedge \mathsf{morph}(v) \wedge \mathsf{seg}(w) \wedge$
  $\mathsf{MDom}(y, u) \wedge \mathsf{MDom}(u, v) \wedge \mathsf{MDom}(v, w) \wedge \mathsf{PDom\!:\!syll\_nuc}(x, w)]$

b. *QF user-defined predicate for selecting the syllables introduced by an MWord*
- **syll_of_MWord**$(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{MWord}(y) \wedge$
  $\mathsf{seg}(\mathsf{F_M\!:\!PDom\!:\!syll\_nuc}(x)) \wedge$
  $\mathsf{morph}(\mathsf{F_D\!:\!MDom}(\mathsf{F_M\!:\!PDom\!:\!syll\_nuc}(x))) \wedge$
  $\mathsf{morpheme}(\mathsf{F_D\!:\!MDom}^2(\mathsf{F_M\!:\!PDom\!:\!syll\_nuc}(x))) \wedge$
  $y = \mathsf{F\!:\!_D\!:\!MDom}^3(\mathsf{F_M\!:\!PDom\!:\!syll\_nuc}(x))$

The predicate below picks out syllables which are not dominated by a PStem. It is QF-definable.

(469) a. *FO user-defined predicate for selecting syllables which are not dominated by a PStem*
- **LoneSyll**$(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \neg\exists y[\mathsf{PDom\!:\!PStem\_syll}(y, x)]$

b. *QF user-defined predicate for selecting syllables which are not dominated by a PStem*
- **LoneSyll**$(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{F_D\!:\!PDom\!:\!PStem\_syll}(x) = \textsc{null}$

The new PWord will prosodically dominate any syllables which 1) are *lone syllables* that not already dominated by a PStem, and 2) which belong to the MWord which is associated with the new PWord. This is all locally computible and QF-definable.[23]

(470) a. *FO output function to let the PWord dominate the inflectional suffix's syllable*
- $\phi\mathtt{PDom\!:\!PWord\_syll}(x^2, y^1) \stackrel{\text{def}}{=} \mathsf{Parse\!:\!MWord\!:\!nonrecursive}(\textsc{settings}) \wedge$
  $\phi\mathtt{PWord}(x^2) \wedge \phi\mathtt{syll}(y^1) \wedge \mathbf{LoneSyll}(y)$
  $\exists w[\phi\mathtt{MWord}(w^1) \wedge \phi\mathtt{Match\!:\!word}(w^1, x^2) \wedge$
  $\mathbf{syll\_of\_MWord}(y, w)]$

b. *QF output function to let the PWord dominate the inflectional suffix's syllable*
- $\phi\mathtt{PDom\!:\!PWord\_syll}(x^2, y^1) \stackrel{\text{def}}{=} \mathsf{Parse\!:\!MWord\!:\!nonrecursive}(\textsc{settings}) \wedge$
  $\phi\mathtt{PWord}(x^2) \wedge \phi\mathtt{syll}(y^1) \wedge \mathbf{LoneSyll}(y)$
  $\phi\mathtt{MWord}(\phi\mathsf{F}_R\!:\!\mathtt{Match\!:\!word}(x^2)) \wedge$
  $\mathbf{syll\_of\_MWord}(y, \phi\mathsf{F}_R\!:\!\mathtt{Match\!:\!word}(x^2)^0)$

---

[23]The unary function $\phi\mathsf{F}_R\!:\!\mathtt{Match\!:\!word}(x^2)$ returns a node w^1 in Copy 1. To retrieve its input correspondent w, I use the superscript ^{0}.

This completes the generation of the PWord. All the above output functions were computationally local and used QF logic. Any non-locality was factorized into the SETTINGS as a locally-accessible label.

6.5.3   Prosody of compounding

Compound prosody is more complicated than the prosody of derivation and inflection. In a two-member compound, each stem or item first forms its own prosodic constituent. These two prosodic constituents can get restructured, linearized, and fused. Some of these processes differ between endocentric and exocentric compounds. Specifically, endocentric compounds like 'water-hole' (471a) form two PStems while exocentric compounds like 'water-colored' (471b) form one PStem. The evidence is their different plural forms.

(471)   a. t͡ʃúr + pós   'water + hole'   b. t͡ʃúr + kújn   'water + color'
            t͡ʃər-a-pós   'water-hole'   t͡ʃər-a-kújn   'water-colored'
            (t͡ʃər-a)$_s$-(pós)$_s$ (t͡ʃər-a-kújn)$_s$
            t͡ʃər-a-pos-ér   'water-holes'   t͡ʃər-a-kujn-nér   'water-colored (objects)'
            (t͡ʃər-a)$_s$-(pos)$_s$-ér [24]   (t͡ʃər-a-kujn)$_s$-nér

In simplex words, the plural suffix is *-er* after a monosyllabic base, *-ner* after a polysyllabic base. But in compounds, the suffix counts the number of syllables in the rightmost PStem. The endocentric compound is paradoxically pluralized as a monosyllabic base *t͡ʃər-a-pos-er* with *-er* because its rightmost PStem is monosyllabic *(pós)$_s$*. The exocentric compound is transparently pluralized as polysyllabic with *-ner* because it forms a single PStem: *(t͡ʃər-a-kujn)$_s$-ner*. Pluralization thus shows distinct prosodic constituencies.

I focus on generating the singular forms. A fragment of their derivation is in (472). Consider an endocentric compound like *t͡ʃər-a-pós* 'water-hole'. Before this final output, the intermediate output of the morphology is //t͡ʃúr-a-pós// where the two component MStems (MStem$_L$ and MStem$_R$) form two stressed PStems (PStem$_L$ and PStem$_R$). The compound eventually undergoes the stem-level phonology of reduction to form *t͡ʃər-a-pós*. But before that step, the prosodic structure is modified. Resyllabification applies: //(t͡ʃú)$_s$r-a-(pós)//. The first PStem$_L$ expands to incorporate the linking vowel -a-: //(t͡ʃúr-a-)$_s$-(pós)$_s$// (472i.a). The two PStems are linearized or ordered (472i.b, not explicit in the table). Finally, because the compound is endocentric, then the two component PStems are not fused together. Instead, the entire compound's MStem$_C$ is prosodically subsumed into the second PStem (472i.c, not explicit in the table).

Contrast this with an exocentric compound like *t͡ʃər-a-kújn* 'water-colored'. The output of compounding is also two stressed PStems: //t͡ʃúr-a-kújn//. The first PStem$_L$ is restructured to incorporate the linking vowel (472ii.a), and the two PStems are linearized. However, the two PStems are then fused into a single PStem$_C$: //(t͡ʃúr-a-kújn)$_s$//. This larger PStem$_C$ is matched with the entire compound MStem$_C$.

---

[24]Technically, the suffix *-er* is later incorporated into the rightmost PStem. I do not show this for clearer illustration.

(472)   *Stages in the prosodic parse of compounds*

|	i. Endocentric 'water-hole'	ii. Exocentric 'water-colored'
Input Morphology	$\text{MStem}_{C,En}$ MConc MStem_L LV MStem_R n t͡ʃúr a pós ∅	$\text{MStem}_{C,Ex}$ MConc MStem_L LV MStem_R a t͡ʃúr a kújn ∅
Input Prosody before resyllabification	PStem_L PStem_R σ σ t͡ʃúr ⟨a⟩ pós	PStem_L PStem_R σ σ t͡ʃúr ⟨a⟩ kújn
Input Prosody after resyllabification	PStem_L PStem_R σ σ σ t͡ʃú r-a pós	PStem_L PStem_R σ σ σ t͡ʃú r-a kújn
a. Prosodic restructuring	PStem_L PStem_R σ σ σ t͡ʃú r-a pós	PStem_L PStem_R σ σ σ t͡ʃú r-a kújn
b. Prosodic linearization		
c. Prosodic subsumption		
d. Prosodic fusion		PStem_C σ σ σ t͡ʃú r-a kújn
Eventual output from vowel reduction	t͡ʃər-a-pós	t͡ʃər-a-kújn

I formalize the four prosodic steps which apply after resyllabification. I formalize each of them as a logical transduction. Each is shown to be computationally local.[25] I first go over some preliminary notation.

---

[25]Interestingly in Chapter 8: §8.3, I show that the locality is due to the cyclicity of our model; a post-cyclic parse of compounds is not computationally local.

6.5.3.1   Preliminary notation

Throughout the formalization, I regularly reference the same morphological and prosodic nodes. I first show the explicit input to parsing an endocentric compound $//(\widehat{t\int}\text{ú})_s\text{r-a-}(\text{pós})_s//$.

(473)   *Input to prosodically parsing an endocentric compound* $//\widehat{t\int}\text{úr-a-pós}//$
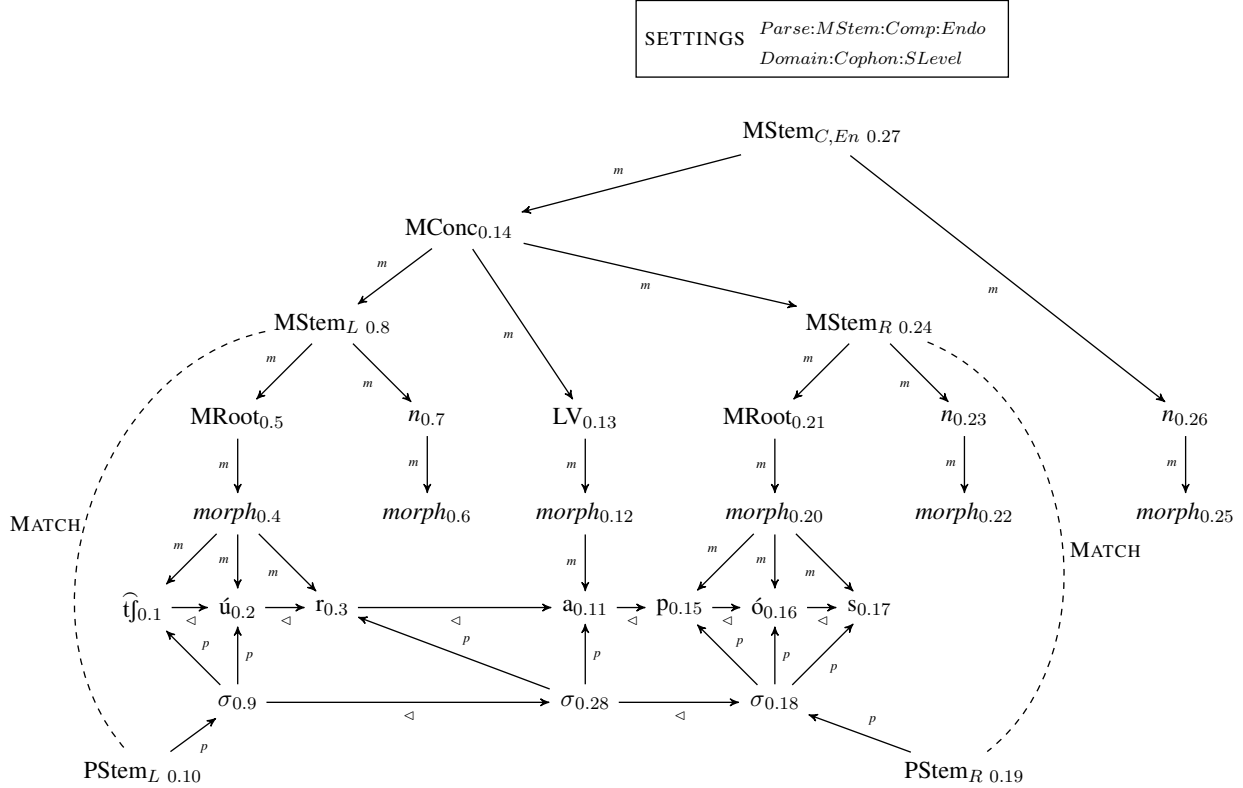
The following predicates pick commonly-used morpho-prosodic constituents in the following order: the current compound or MStem_C as $\text{MStem}_{0.27}$, $\text{MConc}_{0.14}$, the linking vowel's segment as $a_{0.11}$, the left MStem or MStem_L as $\text{MStem}_{0.8}$, and the right MStem or MStem_R as $\text{MStem}_{0.24}$.

(474)   *FO user-defined predicates for picking out the MStems and segments involved in compound prosody*
    1. *The compound MStem*
        $\mathbf{CurrentComp}(x) \stackrel{\text{def}}{=} \text{MStem:Comp}(x) \wedge \mathbf{MTopmost}(x)$
    2. *The compound's MConc*
        $\mathbf{CurrentConc}(x) \stackrel{\text{def}}{=} \text{MConc}(x) \wedge \exists y[\mathbf{CurrentComp}(y) \wedge \text{MDom}(y, x)]$
    3. *The compound's linking vowel -a-*
        $\mathbf{LV\_segment}(x) \stackrel{\text{def}}{=} \text{vowel}(x) \wedge \exists u, v, w[\mathbf{CurrentConc}(u) \wedge \text{LV\_morpheme}(v) \wedge$
                      $\text{morph}(w) \wedge \text{MDom}(u, v) \wedge \text{MDom}(v, w) \wedge \text{MDom}(w, x)]$
    4. *The compound's left MStem or MStem_L*
        $\mathbf{LeftMStem}(x) \stackrel{\text{def}}{=} \text{MStem:Comp:Left}(x) \wedge \exists y[\mathbf{CurrentConc}(y) \wedge \text{MDom}(y, x)]$

5. *The compound's right MStem or MStem$_R$*
   $\textbf{RightMStem}(x) \stackrel{\text{def}}{=} \text{MStem:Comp:Right}(x) \wedge \exists y[\textbf{CurrentConc}(y) \wedge \text{MDom}(y, x)]$

These predicates compute local information and are thus QF-definable.

(475) *QF user-defined predicates for picking out the MStems and segments involved in compound prosody*
   1. $\textbf{CurrentComp}(x) \stackrel{\text{def}}{=} \text{MStem:Comp}(x) \wedge \textbf{MTopmost}(x)$
   2. $\textbf{CurrentConc}(x) \stackrel{\text{def}}{=} \text{MConc}(x) \wedge \textbf{CurrentComp}(\mathsf{F_D}\text{:MDom}(x))$
   3. $\textbf{LV\_segment}(x) \stackrel{\text{def}}{=} \text{vowel}(x) \wedge \text{morph}(\mathsf{F_D}\text{:MDom}(x)) \wedge$
      $\qquad\qquad \text{LV\_morpheme}(\mathsf{F_D}\text{:MDom}^2(x)) \wedge \textbf{CurrentConc}(\mathsf{F_D}\text{:MDom}^3(x))$
   4. $\textbf{LeftMStem}(x) \stackrel{\text{def}}{=} \text{MStem:Comp:Left}(x) \wedge \textbf{CurrentConc}(\mathsf{F_D}\text{:MDom}(x))$
   5. $\textbf{RightMStem}(x) \stackrel{\text{def}}{=} \text{MStem:Comp:Right}(x) \wedge \textbf{CurrentConc}(\mathsf{F_D}\text{:MDom}(x))$

Prosodically, each component MStem contains at least one PStem: $//(\widehat{\text{t\textesh ú}})_s\text{r-a-(pós)}//$. One PStem$_L$ precedes the linking vowel , and one PStem$_R$ follows the linking vowel. These two PStems match with MStem$_L$ and MStem$_R$ respectively. The helper predicates below pick out these PStems.[26] For the endocentric compound, the left PStem$_L$ is PStem$_{0.10}$, the right PStem$_R$ is PStem$_{0.19}$. These predicates are locally-computible.

(476) a. *FO user-defined predicates for picking out the PStems involved in compound prosody*
   1. *The left PStem$_L$ of the compound's left MStem or MStem$_L$*
      $\textbf{LeftPStem}(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge \exists y[\textbf{LeftMStem}(y) \wedge \text{Match:stem}(y, x)]$
   2. *The right PStem$_R$ of the compound's right MStem or MStem$_R$*
      $\textbf{RightPStem}(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge \exists y[\textbf{RightMStem}(y) \wedge \text{Match:stem}(y, x)]$

   b. *QF user-defined predicates for picking out the PStems involved in compound prosody*
   1. *The left PStem$_L$ of the compound's left MStem or MStem$_L$*
      $\textbf{LeftPStem}(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge \textbf{LeftMStem}(\mathsf{F_R}\text{:Match:stem}(x))$
   2. *The right PStem$_R$ of the compound's right MStem or MStem$_R$*
      $\textbf{RightPStem}(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge \textbf{RightMStem}(\mathsf{F_R}\text{:Match:stem}(x))$

Some of the prosodic processes in compounds are the same for both endocentric and exocentric compounds. The predicate below checks if the parsing instruction is that of a compound, either endocentric or exocentric.

(477) *QF user-defined predicate for checking that the input is a compound*

- $\textbf{Parse:MStem:Comp}(\textsc{settings}) \stackrel{\text{def}}{=} \text{Parse:MStem:Comp:Endo}(\textsc{settings}) \wedge$
  $\qquad\qquad\qquad\qquad \text{Parse:MStem:Comp:Exo}(\textsc{settings})$

With all the basic locally-computed predicates set up, we can now parse the compounds. The computation will be local because it references simple local predicates such as the ones above.

---

[26]If the component MStems are actually compounds themselves, then picking out the right PStems is more complicated. MStem$_R$ could be its own endocentric compounds with multiple PStems.

### 6.5.3.2   Prosodic restructuring and incorporation of linking vowels

The first step in parsing a compound is restructuring the left PStem_L in order to incorporate the linking vowel. Like PStem restructuring in derivation, restructuring in compounding is computationally local.

(478)   *Prosodic restructuring in compounds*

The input to this step was shown before in §6.5.3.1. For illustration, I repeat a simplified version of the input without any morphological nodes.

(479)   *Input to prosodic restructuring and linking vowel incorporation in an endocentric compound* $//(\widehat{\text{t\textipa{S}}}\text{ú})_s\text{r-a-}(\text{pós})_s//$
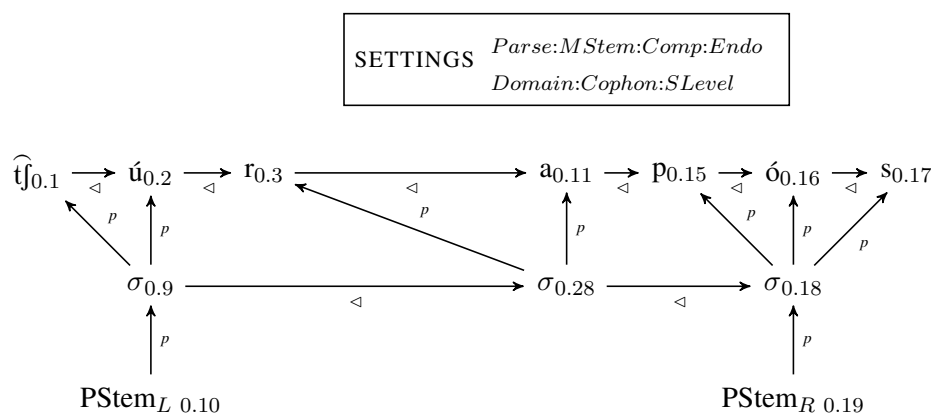
Expansion of the left PStem_L into the linking vowel is a transduction with a copy set of size 1. I show the output below. The necessary change is that $\text{PStem}_{1.10}$ now dominates the linking vowel's syllable $\sigma_{1.28}$

(480)   *Output of incorporating the linking vowel's syllable*
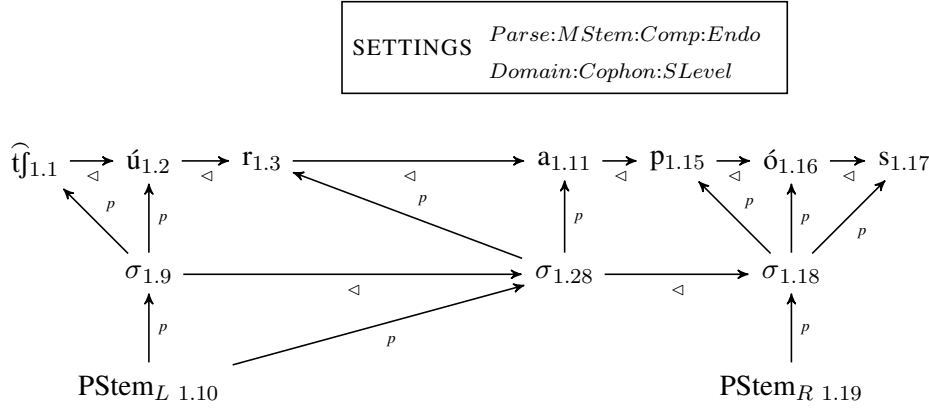
In Copy 1, all labels and nodes are faithfully outputted. Note that we check the SETTINGS to make sure that we are supposed to parse a compound. All relations are faithfully outputted except for the prosodic dominance between PStems and syllables.

(481)   a. *QF output function for faithfully outputting labels in Copy 1*
   - For every label $\mathsf{lab} \in L$
     $$\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathbf{Parse{:}MStem{:}Comp}(\text{SETTINGS})$$
   b. *QF output function for faithfully outputting relations in Copy 1 except for PStem-to-syllable dominance*
   - For every relation $\mathsf{rel} \in R - \{\mathsf{PDom{:}PStem\_syll}\}$:
     $$\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x,y) \wedge \mathbf{Parse{:}MStem{:}Comp}(\text{SETTINGS})$$

Restructuring affects the prosodic dominance between PStems and syllables. This relation is affected in two separate ways which I encode with helper predicates. In one case, the left $\text{PStem}_{1.10}$ (x) should incorporate the syllable $\sigma_{1.28}$ (y) of the linking vowel $a_{1.11}$ (u) via the helper predicate below. This predicate is QF-definable.

(482)   a. *FO helper predicate for making the left PStem dominate the linking vowel's syllable*
   - $\mathbf{should\_PDom{:}PStem\_syll\_new}(x,y) \stackrel{\text{def}}{=} \mathbf{Parse{:}MStem{:}Comp}(\text{SETTINGS})\wedge$
     $\mathbf{LeftPStem}(x) \wedge \mathsf{syll}(y)\wedge$
     $\exists u[\mathbf{LV\_segment}(u) \wedge \mathsf{PDom{:}syll\_nuc}(y,u)]$
   b. *QF helper predicate for making the left PStem dominate the linking vowel's syllable*
   - $\mathbf{should\_PDom{:}PStem\_syll\_new}(x,y) \stackrel{\text{def}}{=} \mathbf{Parse{:}MStem{:}Comp}(\text{SETTINGS})\wedge$
     $\mathbf{LeftPStem}(x) \wedge \mathsf{syll}(y)\wedge$
     $\mathbf{LV\_segment}(\mathsf{F_M{:}PDom{:}syll\_nuc}(y))$

All other underlying prosodic dominance between PStems and syllables should be faithfully outputted by the helper predicate below. I.e., $\text{PStem}_{1.10}$ should dominate $\sigma_{1.9}$, and $\text{PStem}_{1.19}$ should dominate $\sigma_{1.18}$.

(483)   *QF helper predicate for faithfully outputting underlying prosodic dominances of PStems and syllables*

- **should__PDom:PStem_syll__old**$(x^1, y^1) \stackrel{\text{def}}{=}$ **Parse:MStem:Comp**(SETTINGS)$\wedge$
  PDom:PStem_syll(x, y)

The work of the two helper predicates is summarized by the output function below.

(484)   *QF output function for prosodic dominance of PStem and syllables*

- ϕPDom:PStem_syll$(x^1, y^1) \stackrel{\text{def}}{=}$ **should__PDom:PStem_syll__old**$(x, y)\vee$
  should__PDom:PStem_syll__new(x, y)

All the above output functions reference QF user-defined predicates. Thus, the computation is local.

6.5.3.3   Prosodic linearization and ordering

The second step in parsing a compound is to linearize the two PStems in a successor relationship. This is only visible in the more explicit input and output representation for compounds. I omit the morphological nodes. The PStems of the input must be linearized as adjacent because the final syllable of the left PStem$_{0.10}$ now precedes the initial syllable of the right PStem$_{0.19}$. This process is local.

(485)   a. *Input of linearizing PStems in an endocentric compound* $//(\widehat{\text{t\textipa{S}}}\text{úr-a})_s\text{-(pós)}_s//$
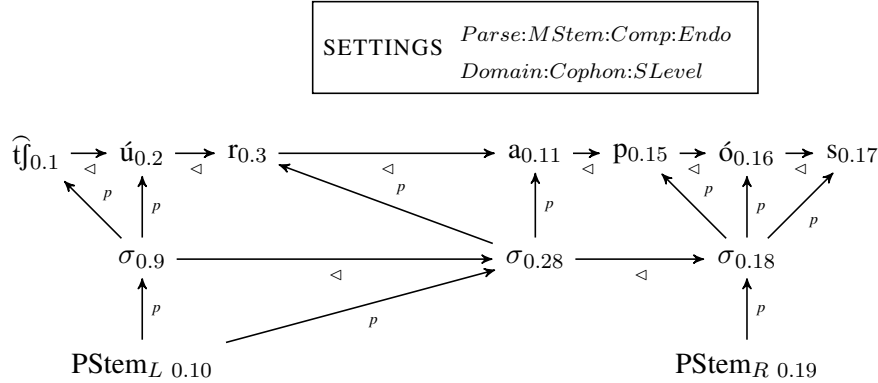
b. *Output of linearizing PStems in an endocentric compound* $//(\widehat{\text{t\textipa{S}}}\text{úr-a})_s\text{-(pós)}_s//$
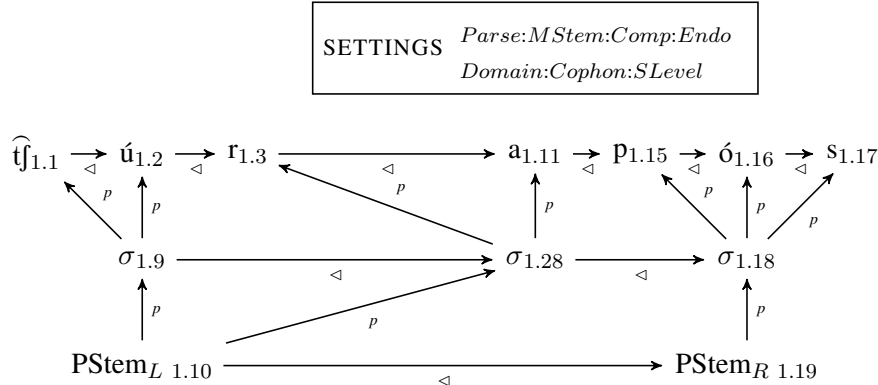
Linearization is a transduction with a copy set of size 1. In Copy 1, all labels are faithfully outputted.

(486) *QF output function for faithfully outputting the labels of the input*
- For every label lab $\in L$:
  $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathbf{Parse:MStem:Comp}(\text{SETTINGS})$

The two PStems are ordered together via a new type of immediate precedence that is specialized for PStems: succ:PStem(x, y). I assume all relations are faithfully outputted except for succ:PStem(x, y).

(487) *QF output function for faithfully outputting the relations of the input*
- For every label lab $\in R - \{\mathsf{succ:PStem}\}$:
  $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathbf{Parse:MStem:Comp}(\text{SETTINGS})$

Compounds are generally made of two component stems, each with their own PStem: the left PStem$_{0.9}$ (x) and the right PStem$_{0.19}$ (y). If we assume that the input only has two PStems, then we can linearize the output PStems via the output function below.[27]

(488) *QF output function for ordering PStems*
- $\phi\texttt{succ:PStem}(x^1, y^1) \stackrel{\text{def}}{=} \mathbf{Parse:MStem:Comp}(\text{SETTINGS}) \wedge \mathbf{LeftPStem}(x) \wedge \mathbf{RightPStem}(y)$

In sum, PStem linearization is also local because it only uses local information, i.e., QF-definable predicates.

---

[27]The above technique does not work when a compound has more than two PStems. A more general way to order any two or more PStems requires some non-local information by referencing the final and initial syllables of a PStem. The predicate **initial_syll_in_PStem**(x, y) finds the first syllable x of a PStem y, while the analogous predicate **final_syll_in_Pstem**(x, y) finds the last syllable. Both of these predicates are locally-computible with QF logic (not shown).

(1) a. *FO user-defined predicates to find the initial syllable of a PStem*
- **initial_syll_in_PStem**$(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{PStem}(y) \wedge \mathsf{PDom:PStem\_syll}(y, x) \wedge$
  $\neg \exists z[\mathsf{syll}(x) \wedge \mathsf{succ:syll}(z, x) \wedge \mathsf{PDom:PStem\_syll}(y, z)]$

b. *FO user-defined predicates to find the final syllable of a PStem*
- **final_syll_in_PStem**$(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{PStem}(y) \wedge \mathsf{PDom:PStem\_syll}(y, x) \wedge$
  $\neg \exists z[\mathsf{syll}(x) \wedge \mathsf{succ:syll}(x, z) \wedge \mathsf{PDom:PStem\_syll}(y, z)]$

Two PStems x, y are then ordered if the final syllable of one PStem x precedes the initial syllable of the other PStem y.

(2) *FO output function for ordering PStems based on the initial and final syllables*
- $\phi\texttt{succ:PStem}(x, y) \stackrel{\text{def}}{=} \mathsf{PStem}(x) \wedge \mathsf{PStem}(y) \wedge \exists u, v[\mathsf{syll}(u) \wedge \mathsf{syll}(v) \wedge$
  final_syll_in_PStem$(u, x) \wedge$ **initial_syll_in_PStem**$(v, y) \wedge \mathsf{succ:syll}(u, v)]$

The above function works but it requires non-local information. Given a PStem, finding its daughter syllables is not local or QF-definable because a PStem can have an unbounded number of daughters. The same non-locality problem occurs for ordering higher levels of prosodic structure (PWords) because PWords dominate an unbounded number of syllables.

To make this process be local, we would need to enrich the set of successor relations to let PStems precede syllables succ:PStem_syll(x, y) or succeed syllables succ:syll_PStem(x, y). Thus, the left PStem$_{0.10}$ would precede the initial syllable $\sigma_{0.18}$ of the right PStem$_{0.19}$, and the right PStem$_{0.19}$ would succeed the final syllable $\sigma_{0.28}$ of the left PStem$_{0.10}$.
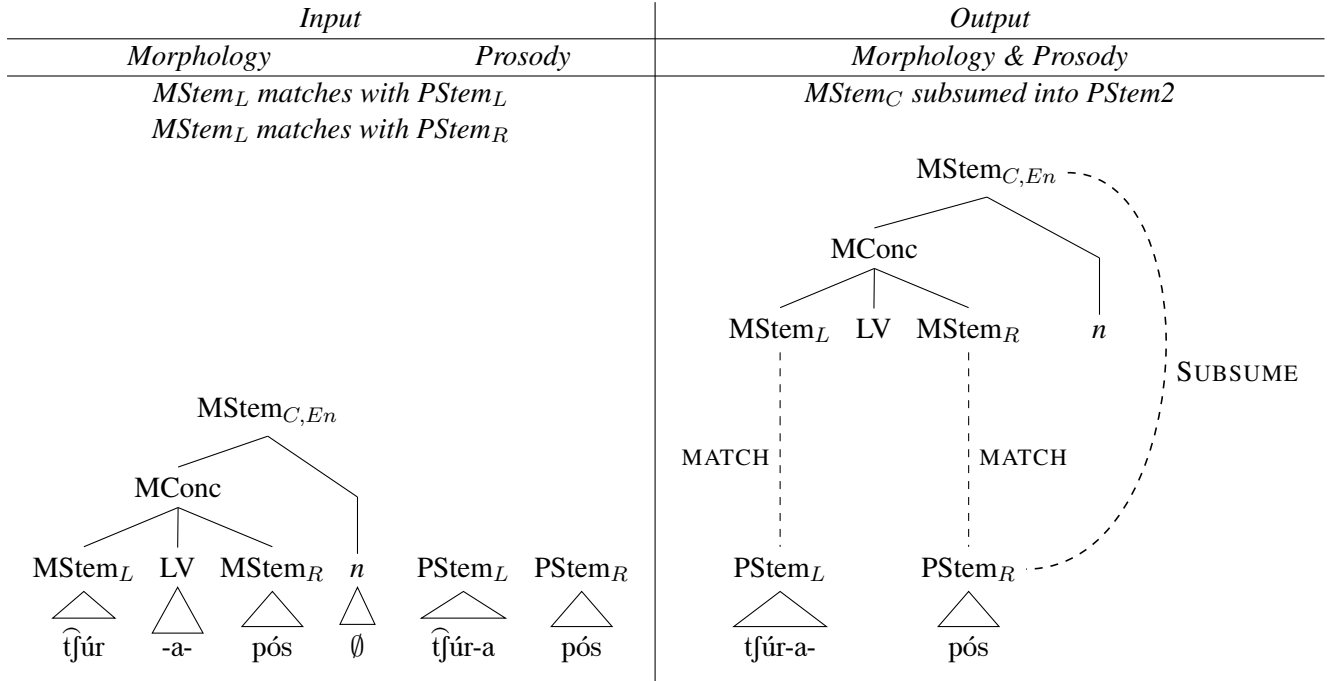
6.5.3.4 Prosodic subsumption in endocentric compounds

Having linearized the PStems, the next step is to determine if the two PStems will be fused or not. In the case of an endocentric compound like 'water-hole' (489a), the two PStems stay intact. The evidence for intact PStem is that the plural suffix counts the number of syllables in $MStem_R$, i.e., $PStem_R$: $\widehat{t\int}$ər-a-$(\underline{pos})_s$-er.

(489)   a. $\widehat{t\int}$úr + pós   'water + hole'   b. $\widehat{t\int}$úr + kújn   'water + color'
            $\widehat{t\int}$ər-a-pós   'water-hole' $\widehat{t\int}$ər-a-kújn   'water-colored'
            $\widehat{t\int}$ər-a-$\underline{pos}$-ér   'water-holes' $\widehat{t\int}$ər-a-kujn-nér   'water-colored (objects)'
            $(\widehat{t\int}$ər-a$)_s$-$(\underline{pos})_s$-ér $(\widehat{t\int}$ər-a-kujn$)_s$-nér

The entire endocentric compound $MStem_C$ is neither matched with either PStem nor mapped to a separate PStem; it instead is subsumed into the right PStem of $MStem_R$. I draw a simplified sketch of this below.

(490)   *Sketch of input and output prosodic associations in prosodic subsumption of an endocentric compound*

Input		*Output*
Morphology	*Prosody*	*Morphology & Prosody*
$MStem_L$ matches with $PStem_L$		$MStem_C$ subsumed into PStem2
$MStem_L$ matches with $PStem_R$		

As explained in Chapter 4: §4.5.2, prosodic subsuming is a special type of prosodic association. Prosodic subsuming is restricted to compound MStems in endocentric compounds. I formalize the relation as $\mathsf{Subsume{:}stem}(x, y)$.

(491)   *Binary relations for specialized prosodic mappings like* Subsuming
   • $\mathsf{Subsume{:}stem}(x, y)$: the MStem x is subsumed into PStem y

Explicitly, the endocentric compound contains the two PStems: $PStem_L$ is $PStem_{0.10}$, and $PStem_R$ is $PStem_{0.19}$. They are each matched with some component MStem: $MStem_L$ is $MStem_{0.8}$, and $MStem_R$ is $MStem_{0.24}$. In the input, the compound $MStem_C$ is $MStem_{0.27}$ and it is not prosodically associated with any PStem. In the output, the only change is that the $MStem_C$ is prosodically subsumed into the right $PStem_{1.19}$.

(492) a. *Input to PStem subsumption in an endocentric* $//(\widehat{t\int}\acute{u}r\text{-}a)_s\text{-}(p\acute{o}s)_s//$

SETTINGS *Parse:MStem:Comp:Endo*
Domain:Cophon:SLevel

$\text{MStem}_{C,En\ 0.27}$

$\text{MConc}_{0.14}$

$\text{MStem}_{L\ 0.8}$ $\text{MStem}_{R\ 0.24}$

$\text{MRoot}_{0.5}$ $n_{0.7}$ $\text{LV}_{0.13}$ $\text{MRoot}_{0.21}$ $n_{0.23}$ $n_{0.26}$

$morph_{0.4}$ $morph_{0.6}$ $morph_{0.12}$ $morph_{0.20}$ $morph_{0.22}$ $morph_{0.25}$

MATCH

$\widehat{t\int}_{0.1}$ $\acute{u}_{0.2}$ $r_{0.3}$ $a_{0.11}$ $p_{0.15}$ $\acute{o}_{0.16}$ $s_{0.17}$

MATCH

$\sigma_{0.9}$ $\sigma_{0.28}$ $\sigma_{0.18}$

$\text{PStem}_{L\ 0.10}$ $\text{PStem}_{R\ 0.19}$
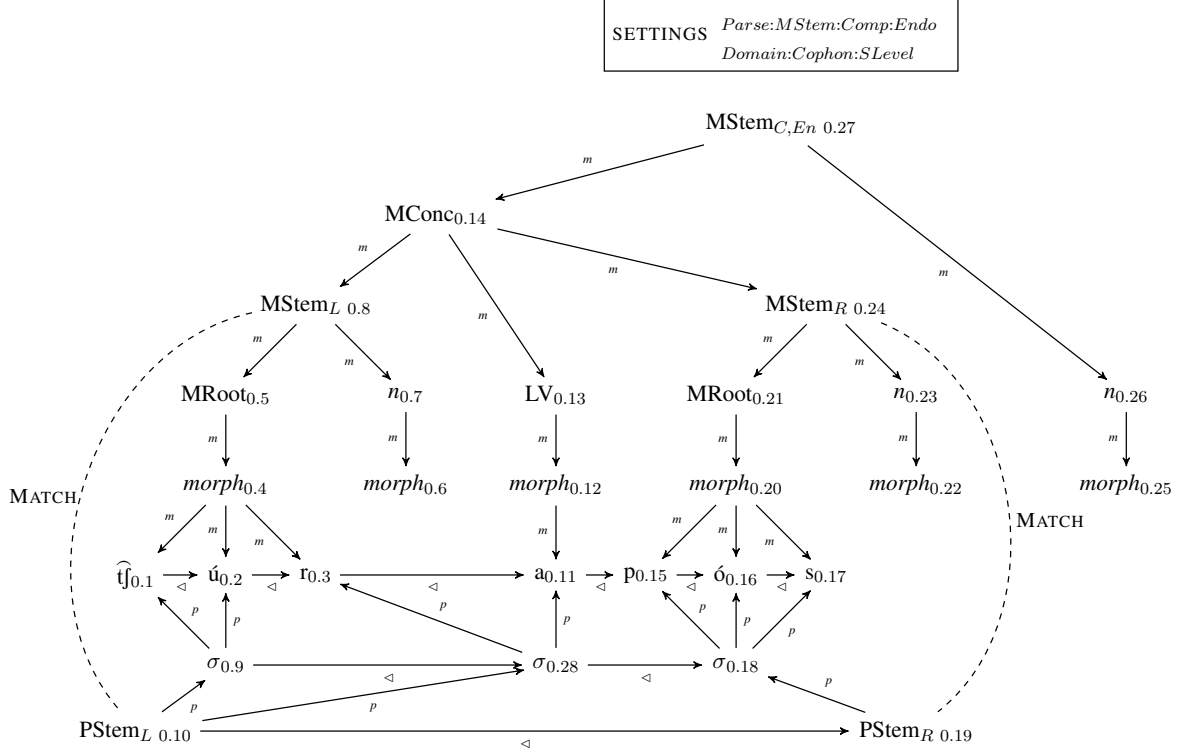
b. *Output of PStem subsumption in an endocentric compound* $//(\widehat{t\int}\acute{u}r\text{-}a)_s\text{-}(p\acute{o}s)_s//$

SETTINGS *Parse:MStem:Comp:Endo*
Domain:Cophon:SLevel

$\text{MStem}_{C,En\ 1.27}$

$\text{MConc}_{1.14}$

$\text{MStem}_{L\ 1.8}$ $\text{MStem}_{R\ 1.24}$

SUBSUME

$\text{MRoot}_{1.5}$ $n_{1.7}$ $\text{LV}_{1.13}$ $\text{MRoot}_{1.21}$ $n_{1.23}$ $n_{1.26}$

$morph_{1.4}$ $morph_{1.6}$ $morph_{1.12}$ $morph_{1.20}$ $morph_{1.22}$ $morph_{1.25}$

MATCH

$\widehat{t\int}_{1.1}$ $\acute{u}_{1.2}$ $r_{1.3}$ $a_{1.11}$ $p_{1.15}$ $\acute{o}_{1.16}$ $s_{1.17}$

MATCH

$\sigma_{1.9}$ $\sigma_{1.28}$ $\sigma_{1.18}$

$\text{PStem}_{L\ 1.10}$ $\text{PStem}_{R\ 1.19}$
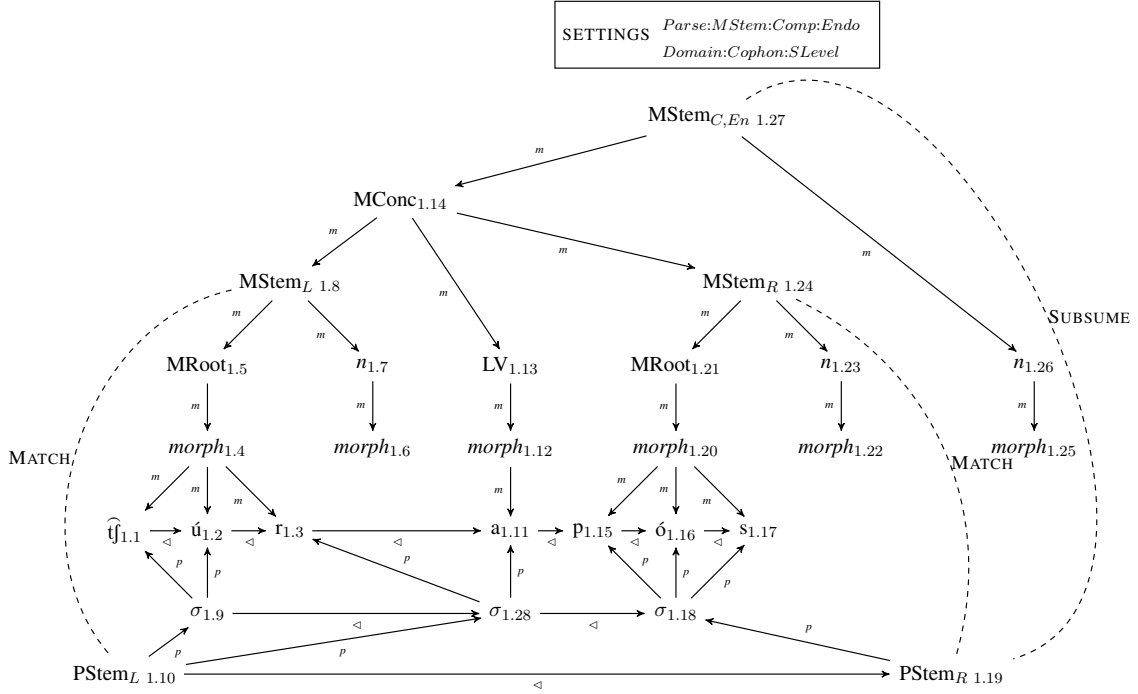
Generating prosodic subsumption is a transduction with a copy set of size 1. The main morphological

trigger is encapsulated into the SETTINGS as the parse label Parse:MStem:Comp:Endo. This process is computationally local. In Copy 1, all nodes and labels are faithfully outputted.

(493)   *QF output function for faithfully outputting the input in Copy 1 for an endocentric compound*
   - $\phi\mathtt{lab}(x) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathsf{Parse:MStem:Comp:Endo}(\text{SETTINGS})$

All relations are the same in the input and the output *except* for the prosodic relationship between the compound $\mathrm{MStem}_{C,1.27}$ and the right $\mathrm{PStem}_{1.19}$. The output function below faithfully outputs all underlying relations except for prosodic subsumption.

(494)   *QF output function for faithfully outputting underlying relations of an endocentric compound except for prosodic subsuming*
   - For every relation $\mathrm{rel} \in R - \{\mathsf{Subsume:stem}\}$
     $\phi\mathtt{rel}(x, y) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathsf{Parse:MStem:Comp:Endo}(\text{SETTINGS})$

In Copy 1, the compound $\mathrm{MStem}_{C,1.27}$ (x) should be prosodically subsumed into the right $\mathrm{PStem}_{1.19}$ (y) via the output function below.[28]

(495)   *QF output function for prosodically subsuming the compound MStem into the right PStem in an endocentric compound*
   - $\mathbf{Subsume:stem}(x, y) \stackrel{\text{def}}{=} \mathsf{Parse:MStem:Comp:Endo}(\text{SETTINGS}) \wedge$
     $\mathbf{CurrentComp}(x) \wedge \mathbf{RightPStem}(y)$

This completes the prosodic parsing of an endocentric compound. All formula were QF-definable, thus the computation is local.

6.5.3.5   Prosodic fusion in exocentric compounds

In contrast to an endocentric compound, exocentric compounds like 'water-colored' (496b) are prosodically parsed into a single large PStem. The evidence for fused PStems is that the plural suffix counts the number of syllables in entire compound MStem_C: *(t͡ʃər-a-kujn)$_s$-ner*.

---

[28]If the grammar allows endocentric compounds made out of other endocentric compounds, then $\phi\mathtt{Subsume:stem}(x^1, y^1)$ must be reformulated as a helper predicate $\mathbf{should\_Subsume:stem\_new}(x, y)$. In large compounds, any underlying prosodic subsumption relations should be faithfully outputted via the helper predicate below.

(1)   *QF helper predicate for faithfully outputting underlying prosodic subsuming relations in an endocentric compound*
   - $\mathbf{should\_Subsume:stem\_old}(x, y) \stackrel{\text{def}}{=} \mathsf{Subsume:stem}(x, y) \wedge \mathsf{Parse:MStem:Comp:Endo}(\text{SETTINGS})$

These two helper predicates are summarized into a single output function for endocentric compounds.

(2)   *QF output function for outputting prosodic subsumption in an endocentric compound*
   - $\phi\mathtt{Subsume:stem}(x^1, y^1) \stackrel{\text{def}}{=} \mathbf{should\_Subsume:stem\_old}(x, y) \vee \mathbf{should\_Subsume:stem\_new}(x, y)$

(496) a. t͡ʃúr + pós   'water + hole'   b. t͡ʃúr + kújn   'water + color'
           t͡ʃər-a-pós   'water-hole'   t͡ʃər-a-kújn   'water-colored'
           t͡ʃər-a-pos-ér   'water-holes'   t͡ʃər-a-kujn-nér   'water-colored (objects)'
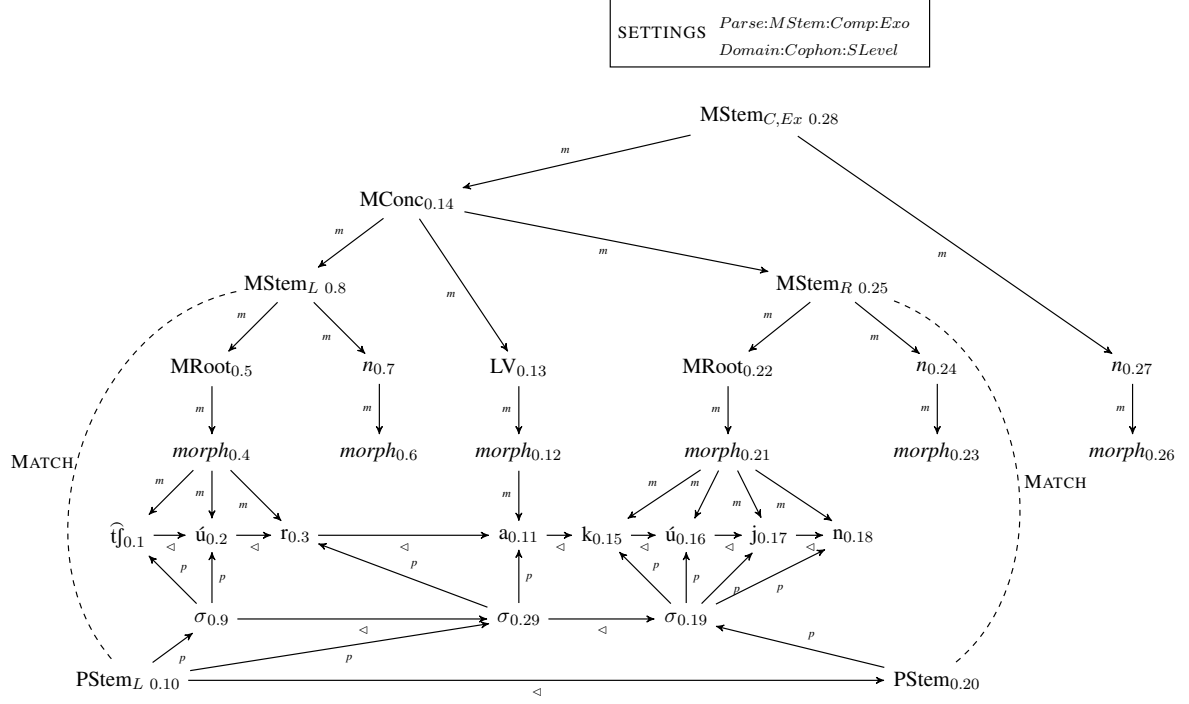   (t͡ʃər-a)$_s$-(pos)$_s$-ér   (t͡ʃər-a-kujn)$_s$-nér

The two input PStems are merged into a single PStem$_C$ which dominates the syllables of the two original PStems. The entire compound MStem$_C$ is matched with this larger PStem$_C$. The component MStems (MStem$_L$, MStem$_R$) are wrapped into PStem$_C$.

(497) *Sketch of input and output prosodic associations in prosodic fusion of an exocentric compound*
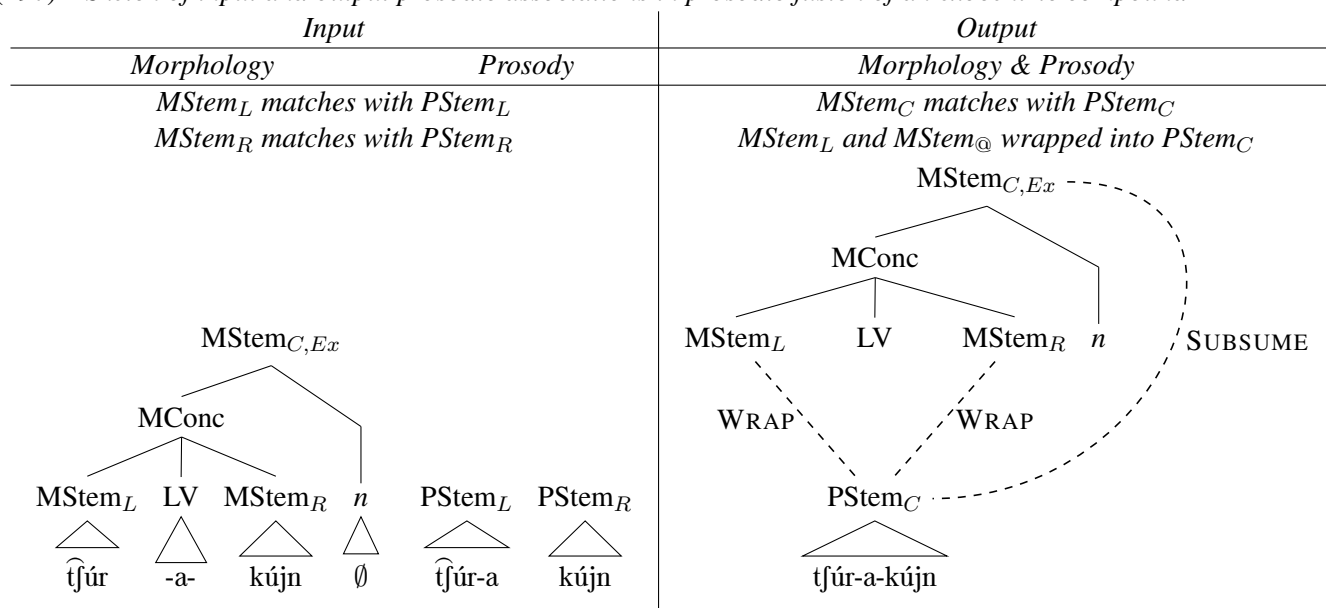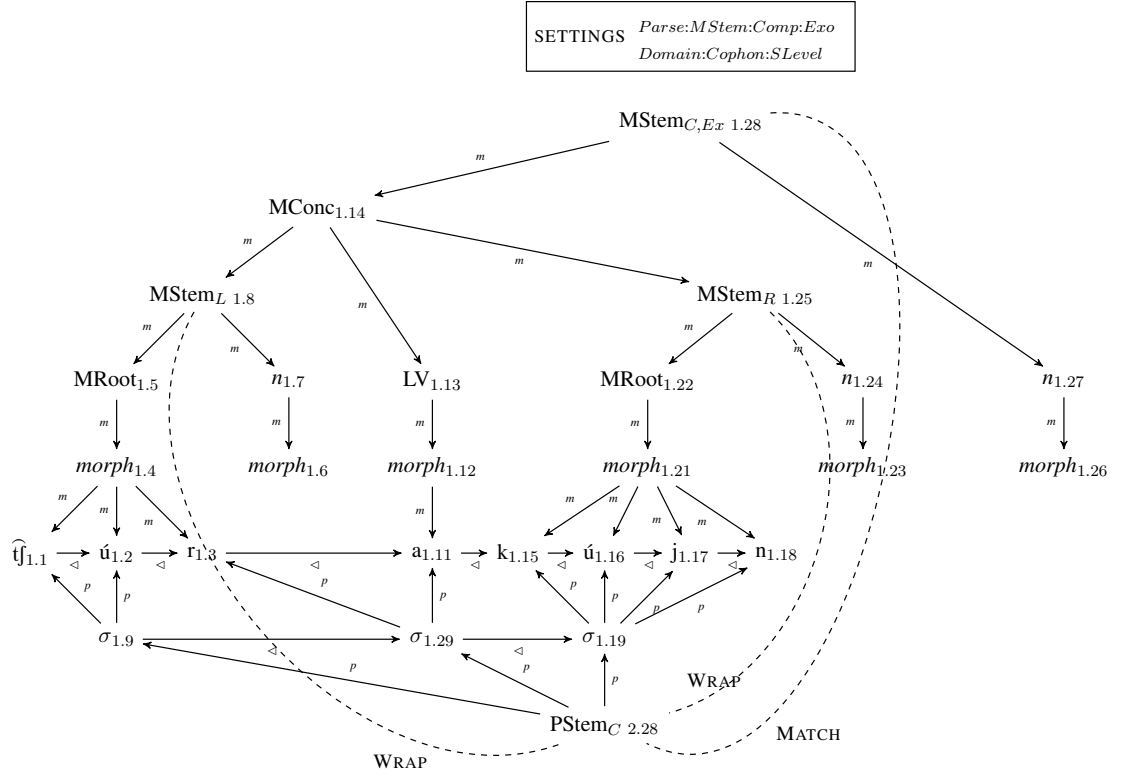

As with endocentric compounds, the explicit representation for exocentric compounds shows all of these abstract associations. In the input, MStem$_L$ is MStem$_{0.8}$ and it is mapped to PStem$_L$ which is PStem$_{0.10}$; in contrast, MStem$_R$ is MStem$_{0.25}$ and it is parsed to PStem$_R$ which is PStem$_{0.20}$. In the input, MStem$_C$ is MStem$_{0.29}$ and it is not parsed into any PStem. In the output, we generate a new PStem$_C$ as PStem$_{2.28}$. All the MStems are re-associated with this PStem via WRAP and MATCH relations. This process is computationally local.

(498)   a. *Input to PStem fusion in an exocentric* $//(\widehat{\text{t}\int}\text{úr-a})_s\text{-(kújn)}_s//$

SETTINGS *Parse:MStem:Comp:Exo*
Domain:Cophon:SLevel

MStem$_{C,Ex\ 0.28}$

MConc$_{0.14}$

MStem$_{L\ 0.8}$ MStem$_{R\ 0.25}$

MRoot$_{0.5}$ $n_{0.7}$ LV$_{0.13}$ MRoot$_{0.22}$ $n_{0.24}$ $n_{0.27}$

$morph_{0.4}$ $morph_{0.6}$ $morph_{0.12}$ $morph_{0.21}$ $morph_{0.23}$ $morph_{0.26}$

$\widehat{\text{t}\int}_{0.1}$ $\text{ú}_{0.2}$ $\text{r}_{0.3}$ $\text{a}_{0.11}$ $\text{k}_{0.15}$ $\text{ú}_{0.16}$ $\text{j}_{0.17}$ $\text{n}_{0.18}$

MATCH   MATCH

$\sigma_{0.9}$ $\sigma_{0.29}$ $\sigma_{0.19}$

PStem$_{L\ 0.10}$ PStem$_{0.20}$

b. *Output of PStem fusion in an exocentric* $//(\widehat{\text{t}\int}\text{úr-a-kújn})_s//$

SETTINGS *Parse:MStem:Comp:Exo*
Domain:Cophon:SLevel

MStem$_{C,Ex\ 1.28}$

MConc$_{1.14}$

MStem$_{L\ 1.8}$ MStem$_{R\ 1.25}$

MRoot$_{1.5}$ $n_{1.7}$ LV$_{1.13}$ MRoot$_{1.22}$ $n_{1.24}$ $n_{1.27}$

$morph_{1.4}$ $morph_{1.6}$ $morph_{1.12}$ $morph_{1.21}$ $morph_{1.23}$ $morph_{1.26}$

$\widehat{\text{t}\int}_{1.1}$ $\text{ú}_{1.2}$ $\text{r}_{1.3}$ $\text{a}_{1.11}$ $\text{k}_{1.15}$ $\text{ú}_{1.16}$ $\text{j}_{1.17}$ $\text{n}_{1.18}$

$\sigma_{1.9}$ $\sigma_{1.29}$ $\sigma_{1.19}$

PStem$_{C\ 2.28}$

WRAP   WRAP   MATCH   WRAP

Generating PStem fusion is a transduction with a copy set of size 2. The main morphological trigger is encapsulated into the SETTINGS as the parse label Parse:MStem:Comp:Exo. In Copy 1, all underlying labels should be faithfully outputted except for PStems. Similarly for relations in Copy 1, all relations are faithfully outputted except for those involving PStems.[29]

(499) a. *QF output function for faithfully outputting the input in Copy 1 for an exocentric compound except* for PStems
- For every label lab $\in L - \{PStem\}$:
  $\phi\texttt{lab}(x) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings})$

b. *QF output function for faithfully outputting the input in Copy 1 for an exocentric compound except* for those involving PStems
- For every relation rel $\in R - \{\mathsf{Match:stem}, \mathsf{Wrap:stem}, \mathsf{PDom:PStem\_syll}, \ldots\}$:
  $\phi\texttt{rel}(x, y) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings})$

In Copy 2, a $\mathsf{PStem}_{2.28}$ (x^2) is generated as an output correspondent for the compound $\mathsf{MStem}_{C,0.28}$ (x).

(500) *QF output function for generating a PStem in Copy 2 for an exocentric compound*
- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \mathbf{CurrentComp}(x) \wedge \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings})$

This new $\mathsf{PStem}_{2.28}$ (x^2) in Copy 2 dominates the syllables (y^1) of the input left and right PStems (u) in Copy 1. The output function which does this is locally-computed and QF-definable.

(501) a. *FO output function for assigning the syllables of the underlying left/right PStem to the new PStem*
- $\phi\texttt{PDom:PStem\_syll}(x^2, y^1) \stackrel{\text{def}}{=} \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings}) \wedge$
  $\mathsf{PStem}(x^2) \wedge \mathsf{syll}(y) \wedge$
  $\exists u[\mathsf{PStem}(u) \wedge \mathsf{PDom:PStem\_syll}(u, y) \wedge$
  $[\mathbf{LeftPStem}(u) \vee \mathbf{RightPStem}(u)]]$

---

[29]Exocentric compounds with three or more members show are more complicated to parse. In Copy 1, any PStems which are not the left and right PStems of the compound are faithfully outputted. All PStem-relations that *don't* involve the left and right PStem are faithfully outputted.

(1) a. *QF output function for faithfully outputting PStems* outside *outside of the compound's left and right PStems*
- For every label lab $\in \{PStem\}$:
  $\phi\texttt{lab}(x) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings}) \wedge \neg[\mathbf{LeftPStem}(x) \vee \mathbf{RightPStem}(x)]$

b. *QF output function for faithfully outputting relations that involve PStems* outside *of the compound's left and right PStems*
- For every relation rel $\in \{\mathsf{Match:stem}, \mathsf{Wrap:stem}, \mathsf{PDom:PStem\_syll}, \ldots\}$:
  $\phi\texttt{rel}(x, y) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathsf{Parse:MStem:Comp:Exo}(\textsc{settings}) \wedge$
  $\neg[\mathbf{LeftPStem}(x) \vee \mathbf{RightPStem}(x)]$

b. *QF output function for assigning the syllables of the underlying left/right PStem to the new PStem*

- ϕPDom:PStem_syll$(x^2, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:Comp:Exo(SETTINGS)$\wedge$
  PStem$(x^2) \wedge$ syll$(y) \wedge$
  $[\textbf{LeftPStem}(\text{F}_\text{D}:\text{PDom:PStem\_syll}(y)) \vee$
  $\textbf{RightPStem}(\text{F}_\text{D}:\text{PDom:PStem\_syll}(y))]$

The compound MStem$_{C,1.28}$ (x^1) from Copy 1 is matched with the new PStem$_{2.28}$ (y^2).

(502)   *QF output function to make the new PStem match with the compound MStem*

- ϕMatch:stem$(x^1, y^2) \stackrel{\text{def}}{=}$ Parse:MStem:Comp:Exo(SETTINGS)$\wedge$
  $\textbf{CurrentComp}(x) \wedge \phi$PStem$(y^2)$

Any underlying MStems (x^1) which were prosodically associated with the underlying PStems (z) are now associated with the new PStem$_C$ (y^2). This means that MStem$_L$ and MStem$_R$ (MStem$_{1.8}$ and MStem$_{1.25}$) are now wrapped into the new PStem$_{2.28}$. In fact, if any underlying MStems were even *wrapped* or matched with the old PStems, they are now wrapped into the new PStem.[30] This reassocation is locally-computed.

(503)   a. *FO output function for transferring any wrapped or matched MStem of the underlying left-right PStems to the new compound PStem*

- ϕWrap:stem$(x^1, y^2) \stackrel{\text{def}}{=}$ Parse:MStem:Comp:Exo(SETTINGS)$\wedge$
  MStem$(x) \wedge \phi$PStem$(y^2) \wedge$
  $\exists z[\text{PStem}(z) \wedge [\textbf{LeftPStem}(z) \vee \textbf{RightPStem}(z)] \wedge$
  $[\text{Wrap:stem}(x, z) \vee \text{Match:stem}(x, z)]]$

b. *QF output function for transferring any wrapped or matched MStem of the underlying left-right PStems to the new compound PStem*

- ϕWrap:stem$(x^1, y^2) \stackrel{\text{def}}{=}$ Parse:MStem:Comp:Exo(SETTINGS)$\wedge$
  MStem$(x) \wedge \phi$PStem$(y^2) \wedge$
  $[\textbf{LeftPStem}(\text{F}_\text{L}:\text{Match:stem}(x)) \vee$
  $\textbf{RightPStem}(\text{F}_\text{L}:\text{Match:stem}(x)) \vee$
  $\textbf{LeftPStem}(\text{F}_\text{L}:\text{Wrap:stem}(x)) \vee$
  $\textbf{RightPStem}(\text{F}_\text{L}:\text{Wrap:stem}(x))]$

This completes the generation of a fused PStem in an exocentric compound. The computation was local because quantifiers were not needed. k

---

[30]In exocentric compounds made with three or more members, there may be MStems which are subsumed into some PStem in the input. They should now be subsumed into the new PStem.

(1)   *FO output function for transferring any wrapped or matched MStem of the underlying left-right PStems to the new compound PStem*

- ϕSubsume:stem$(x^1, y^2) \stackrel{\text{def}}{=}$ Parse:MStem:Comp:Exo(SETTINGS) $\wedge$ MStem$(x) \wedge \phi$PStem$(y^2) \wedge$
  $\exists z[\text{PStem}(z) \wedge [\textbf{LeftPStem}(z) \vee \textbf{RightPStem}(z)] \wedge \text{Subsume:stem}(x, z)]$

6.6 Locality in stratal phonological rules

Having parsed the relevant prosodic structure, the right phonological rule domains must apply. These rules differ between derivation and inflection, V-initial and C-initial inflection, and between dialects.

For the derivative //amusín-agan//, the stem-level phonology applies with stress shift and reduction: *amusn-agán*. The word-level phonology is later vacuously applied in a vacuous third cycle. For the inflected items, we see different cophonologies apply. C-initial inflection like //amusín-ner// triggers the word-level cophonology, where we have stress but no reduction: *amusin-nér*. V-initial inflection like //(amusín-ov)$_s$// has more complications. It triggers PStem misalignment which iteslf triggers the PStem-level cophonology. Reduction is active in the PStem-level cophonology for Eastern Armenian *amusn-óv* but not Western Armenian *amusin-óv*. The word-level cophonology then vacuously applies.

I formalize the selection of these cophonologies, specifically how to formalize

1. structure-changing processes like reduction,
2. prosodically triggered cophonologies or rule-domains,
3. encoding of dialectal variation
4. application of the PStem-level vs. word-level cophonology

6.6.1 Stem-level cophonology: Locality of reduction

Overt derivational morphology in //amusín-agan// triggers the stem-level cophonology or phonological rule domain, including stress shift and reduction. I treat stress and reduction as two logical transductions which apply in a sequence. In Chapter 5: §5.5, I formalized stress assignment. The main task of this section is to formalize vowel reduction, a complicated process which references strata, syllabification, past stress, local contexts, and different possible repairs. Helper predicates will be heavily used in this section.[31]
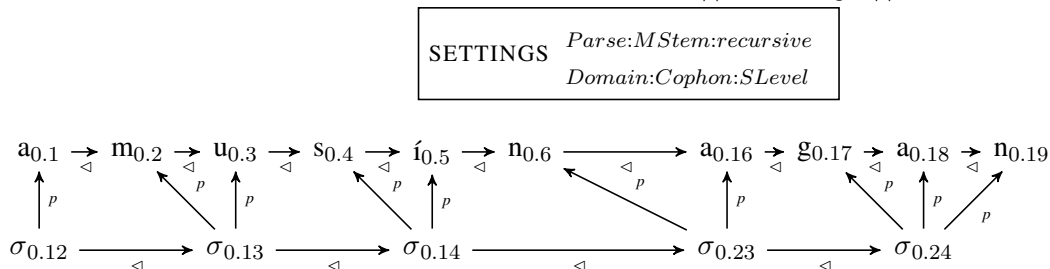
(504) *Partial stratal derivation of the derivative* amusn-agan

		MStem
		MStem — MRoot n n
		/amusin/ -∅ /-agan/
Input		/amusin -∅ -agan/
Cycle 1		a.mu.sín
Cycle 2		...
	PHONO *SLevel*	
	Stress	a.mu.sĭn-a.gán
	DHR (reduction)	a.mus.n-a.gán
Output		amusn-agán

---

[31]I do not discuss how the stem-level phonology applies in compounds. An intermediate input like *t͡ʃúr-a-pós* 'water-hole' will undergo the stem-level phonology of stress shift and reduction *t͡ʃər-a-pós*, just like a derivative. The formulas in this section are illustrated with derivatives but they also handle compounds.

Throughout this section, the formulas reference the SETTINGS constant. The constant has the relevant Domain label to trigger the stem-level cophonology: Domain:Cophon:SLevel(SETTINGS). I show the explicit input below; I omit the morphological nodes and PStem.

(505)   *Input for applying the stem-level phonology for a derivative* $//$amusín-agan$//$

$$\boxed{\begin{array}{ll} \text{SETTINGS} & \textit{Parse:MStem:recursive} \\ & \textit{Domain:Cophon:SLevel} \end{array}}$$

$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to í_{0.5} \to n_{0.6} \longrightarrow a_{0.16} \to g_{0.17} \to a_{0.18} \to n_{0.19}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.23} \longrightarrow \sigma_{0.24}$
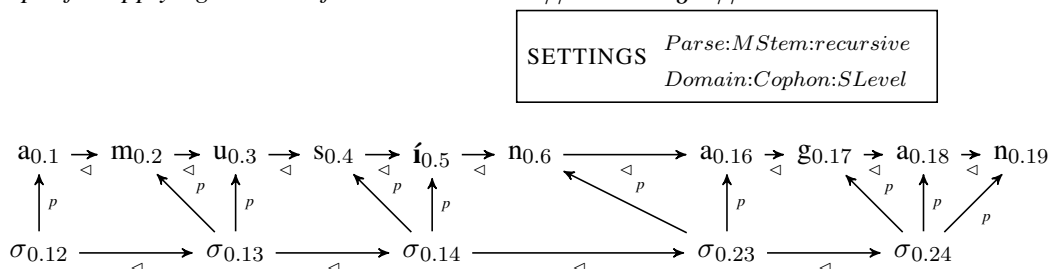
I first go quickly go through stress shift and destressing (§6.6.1.1). I then describe and formalize some preliminaries on reduction, i.e., predicates which pick out the relevant context and domain for reduction (§6.6.1.2). I formalize reduction in a piecemeal fashion because it sensitive to multiple conditions on the input. This requires the use of helper predicates to streamline the formalization. I formalize how to faithfully output segments and nodes which are outside of the context of reduction (§6.6.1.3). I formalize the resyllabification factors in §6.6.1.4. I then formalize how to reduce vowels to schwas (§6.6.1.5) and how to delete a vowel and trigger resyllabification (§6.6.1.6). I emphasize that the phonological process is computationally local, and it is definable with QF logic. This section is essentially an extended exercise in using helper predicates for a complex process (see Chapter 4: §4.3.4, §4.3.5).
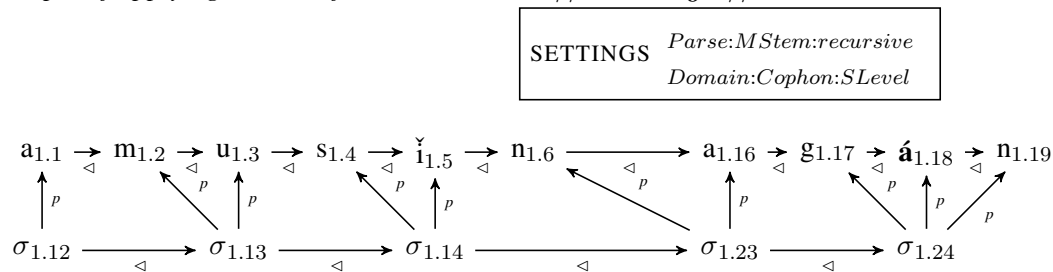
6.6.1.1   Locality of stress shift and destressing

Given an input form like $//$amusín-agan$//$, the first step in the stem-level cophonology is to apply stress shift: $//$amusìn-agán$//$. The logical definition for stress assignment in complex words is the same as in simplex words. However, only in complex words do we find *destressed* vowels like $ì$ where the diacritic marks the loss of stress. I show the input and output below.

(506)   a. *Input for applying stress shift in the derivative* $//$amusín-agan$//$

$$\boxed{\begin{array}{ll} \text{SETTINGS} & \textit{Parse:MStem:recursive} \\ & \textit{Domain:Cophon:SLevel} \end{array}}$$

$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to í_{0.5} \to n_{0.6} \longrightarrow a_{0.16} \to g_{0.17} \to a_{0.18} \to n_{0.19}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.23} \longrightarrow \sigma_{0.24}$

b. *Output of applying stress shift in the derivative //amusĭn-agán//*

SETTINGS	*Parse:MStem:recursive*
	Domain:Cophon:SLevel

$$a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \to ĭ_{1.5} \to n_{1.6} \longrightarrow a_{1.16} \to g_{1.17} \to á_{1.18} \to n_{1.19}$$

$$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.14} \longrightarrow \sigma_{1.23} \longrightarrow \sigma_{1.24}$$

Stress assignment is a transduction with a copy set of size 1. As said, I formalized stress assignment in Chapter 5: §5.5. The main output functions are repeated below. These output functions will shift stress from the input stressed vowel $i_{0.5}$ (and its syllable $\sigma_{0.14}$) to the rightmost full vowel $a_{1.18}$ (and its syllable $\sigma_{1.24}$).

(507)   *QF output functions for assigning stress in Copy 1*

- $\phi\texttt{stressed:syll}(x^1) \overset{\text{def}}{=} \textbf{StressDomain}(\textsc{settings}) \wedge \textsf{syll}(x) \wedge \textbf{rightmost\_full\_syll}(x)$
- $\phi\texttt{stressed:vowel}(x^1) \overset{\text{def}}{=} \textbf{StressDomain}(\textsc{settings}) \wedge \textsf{vowel}(x) \wedge \textbf{rightmost\_full\_vowel}(x)$

Importantly, the vowel $ĭ_{1.5}$ is now a destressed vowel along with its destressed syllable $\sigma_{1.14}$.

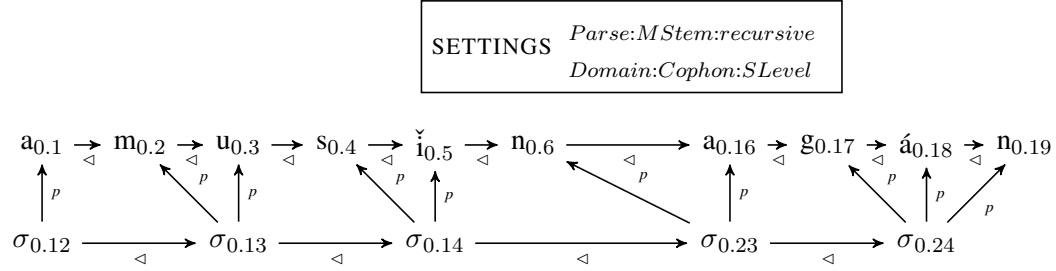(508)   *QF output functions for destressing items in Copy 1*

- $\phi\texttt{destressed:syll}(x^1) \overset{\text{def}}{=} \textsf{stressed:syll}(x) \wedge \neg\phi\texttt{stressed:syll}(x^1)$
- $\phi\texttt{destressed:vowel}(x^1) \overset{\text{def}}{=} \textsf{stressed:vowel}(x) \wedge \neg\phi\texttt{stressed:vowel}(x^1)$

I do not explain again how these functions will generate the right output. What's important is that stress shift is equivalent to stress assignment. It is a local process *once* we factor out its morphological triggers in the form of SETTINGS labels. Interested readers are encouraged to reread Chapter 5: §5.5 and work out for themselves why the functions will work.
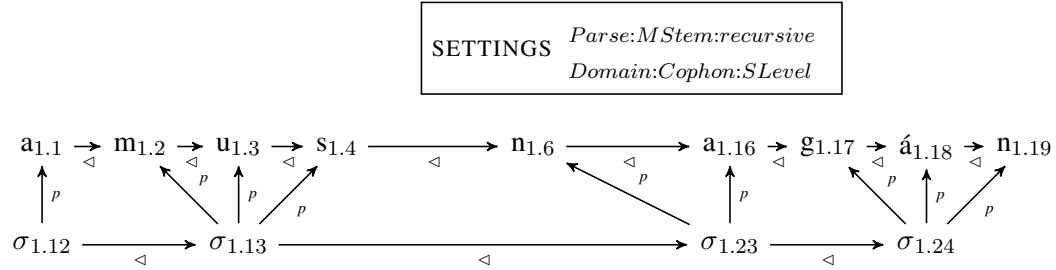
6.6.1.2   Preliminaries

The output of stress shift feeds destressed high vowel reduction or DHR. Reduction is a separate transduction which uses a copy set of size 1. I show the input and output below, without any morphological or PStem nodes.

(509) a. *Input of vowel reduction in* //amusĭn-agán//

SETTINGS $Parse:MStem:recursive$
$Domain:Cophon:SLevel$

$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to ĭ_{0.5} \to n_{0.6} \longrightarrow a_{0.16} \to g_{0.17} \to á_{0.18} \to n_{0.19}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.23} \longrightarrow \sigma_{0.24}$

b. *Output of vowel reduction in* //amusĭn-agán// → amusn-agán

SETTINGS $Parse:MStem:recursive$
$Domain:Cophon:SLevel$

$a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \longrightarrow n_{1.6} \longrightarrow a_{1.16} \to g_{1.17} \to á_{1.18} \to n_{1.19}$

$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.23} \longrightarrow \sigma_{1.24}$

First, I formalize whether reduction *should* apply by checking that that we are in the right stratum, either the stem-level cophonology or the Eastern Armenian PStem cophonology. The helper predicate below checks if the SETTINGS has the domain label of the the right stratum.[32]

(510) *QF user-defined predicate for checking that we are in the cophonology domain for vowel reduction*

- **ReductionDomain**(SETTINGS) $\stackrel{\text{def}}{=}$ Domain:Cophon:SLevel(SETTINGS)$\vee$
  [Domain:Cophon:PStem(SETTINGS) $\wedge$ Eastern(SETTINGS)]

As for the nodes affected by vowel reduction, the user-defined predicates below select them: high vowels, a destressed high vowel $ĭ_{0.5}$, its syllable *sĭ* at $\sigma_{0.14}$, its onset $s_{0.4}$, the segment after the destressed high vowel $n_{0.6}$, the previous syllable *mu* at $\sigma_{0.13}$, and following syllable *na* at $\sigma_{0.23}$. These predicates are locally-computed.

(511) a. *FO user-defined predicates for finding nodes involved in vowel reduction*

- **high**$(x) \stackrel{\text{def}}{=}$ u$(x) \vee$ i(x)
- **destressed_high_vowel**$(x) \stackrel{\text{def}}{=}$ destressed:vowel$(x) \wedge$ **high**(x)
- **destressed_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x)\wedge\exists y[$**destressed_high_vowel**$(y)\wedge$PDom:syll_nuc$(x,y)]$
- **destressed_ons**$(x) \stackrel{\text{def}}{=}$ consonant$(x)\wedge\exists y[$**destressed_high_vowel**$(y)\wedge$succ:seg$(x,y)]$
- **following_seg**$(x) \stackrel{\text{def}}{=}$ seg$(x) \wedge \exists y[$**destressed_high_vowel**$(y) \wedge$ succ:seg$(y,x)]$
- **previous_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x) \wedge \exists y[$**destressed_syll**$(y) \wedge$ succ:syll(x,y)
- **following_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x) \wedge \exists y[$**destressed_syll**$(y) \wedge$ succ:syll(y,x)

b. *QF user-defined predicates for finding nodes involved in vowel reduction*

- **high**$(x) \stackrel{\text{def}}{=}$ u$(x) \vee$ i(x)

---

[32]Dialect labels are explained in §6.6.3.1 in the context of inflection.

- **destressed_high_vowel**$(x) \stackrel{\text{def}}{=}$ destressed:vowel$(x) \wedge$ **high**(x)
- **destressed_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x) \wedge$ **destressed_high_vowel**$(\mathsf{F_M}$:PDom:syll_nuc$(x))$
- **destressed_ons**$(x) \stackrel{\text{def}}{=}$ consonant$(x) \wedge$ **destressed_high_vowel**$(\mathsf{F_L}$:succ:seg$(x))$
- **following_seg**$(x) \stackrel{\text{def}}{=}$ seg$(x) \wedge$ **destressed_high_vowel**$(\mathsf{F_R}$:succ:seg$(x))$
- **previous_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x) \wedge$ **destressed_syll**$(\mathsf{F_L}$:succ:syll$(x))$
- **following_syll**$(x) \stackrel{\text{def}}{=}$ syll$(x) \wedge$ **destressed_syll**$(\mathsf{F_R}$:succ:syll$(x))$

All these nodes are part of the context of vowel reduction. The predicate below picks out all of these nodes. All these predicates are computationally local.

(512) *QF user-defined predicate for selecting nodes in the context of vowel reduction*

- **ReductionContext**$(x) \stackrel{\text{def}}{=}$ **destressed_high_vowel**$(x) \vee$ **destressed_syll**$(x) \vee$
  destressed_ons$(x) \vee$ **previous_syll**$(x) \vee$
  following_syll$(x) \vee$ **following_seg**(x)

Given the domain and relevant nodes for reduction, there are different possible scenarios for any given node during vowel reduction. I list below different base-derivative pairs. The high vowel is deleted in some pairs, or reduced to a schwa in others.

(513) *Different derivatives and outcomes in vowel reduction*

	Base		Input to reduction	Output derivative	
	Delete				
a.	*amusín*	'husband'	*amusin̆-agán*	*amusn-agán*	'marital'
b.	*barsíg*	'Persian'	*barsi̇g-astán*	*barsg-astán*	'Persia'
	Reduce to schwa				
c.	*kír*	'writing'	*ki̇r-óɣ*	*kər-óɣ*	'writer'
d.	*aɣmúg*	'noise'	*aɣmŭg-él*	*aɣməg-él*	'to make noise'
e.	*barísp*	'fortress'	*barĭsp-él*	*barəsp-él*	'to fortify'

Either a segment is part of the reduction context or it is not (it is unaffected). Among nodes which are affected and part of the reduction context, the high vowel is generally deleted *amusn-agán* unless deletion would cause an unsyllabifiable consonant cluster: //ki̇r-óɣ//→kər-óɣ, *kr-óɣ.

For the running example //amusi̇n-agán//, the underlined segments are far from the destressed high vowel. They are unaffected segments which are faithfully outputted . For //amusi̇n-agán//, the underlined segments undergo the deletion scenario whereby the high vowel must delete and trigger resyllabification: *amus.n-agán*. For a different input like //ki̇r-óɣ//, the high vowel should reduce to a schwa: *kər-óɣ*.

I summarize these scenarios below.

(514) *Scenarios involved in vowel reduction*

a. **Unaffected Scenario**: vowel reduction does not apply to this segment because it is not part of the context for vowel reduction (not a destressed high vowel or adjacent to one).

b. **Schwa Scenario**: vowel reduction applies does apply to this segment because it is a destressed high vowel or part of its context. The destressed high vowel is turned into a schwa.

c. **Deletion Scenario**: vowel reduction does apply to this segment because it is a destressed high vowel or part of its context. The destressed high vowel is deleted.

The division of labor across these scenarios is shown below for different base-derivative pairs. I underline the relevant segments. I don't show the relevant syllables.

(515)   *Division of labor across the scenarios in reduction*

Input	Scenarios			Output
	Unaffected	**Schwa**	**Deletion**	
amusĭn-agán	*a̠mu̠sin-a̠gán*	*amusi̠g-agán*		*amusn-agán*
barsĭg-astán	*ba̠rsin-a̠stán*	*barsi̠g-astán*		*barsg-astán*
kĭr-óɣ	*kĭr-óɣ*		*kĭr-óɣ*	*kər-óɣ*
aɣmŭg-él	*a̠ɣmŭg-é̠l*		*aɣmŭg-él*	*aɣməg-él*
barĭsp-él	*ba̠ri̠sp-él*		*bari̠sp-él*	*barəsp-él*

In the rest of this section, I formalize helper predicates which handle these different scenarios.

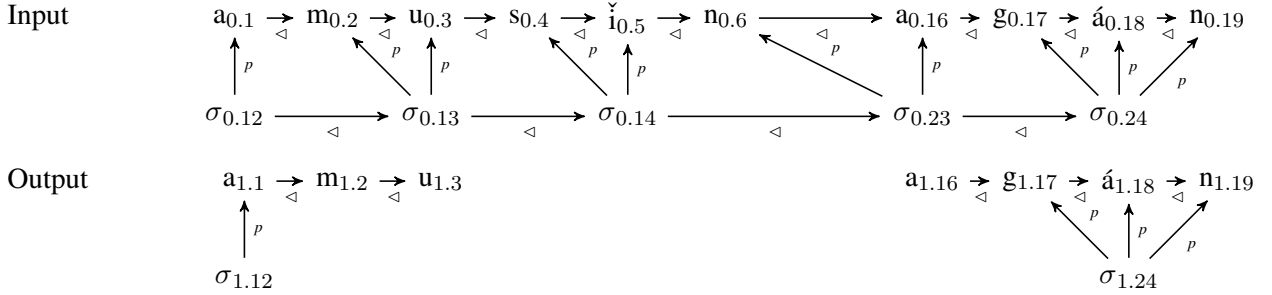### 6.6.1.3 Unaffected scenario: Faithfully outputting irrelevant nodes

For nodes in the the **Unaffected Scenario**, vowel reduction does not affect them. We *should* output all nodes and relations which don't involve the nodes in the context of reduction. This is formalized by the helper predicates below.

(516)   *QF helper predicates for faithfully outputting nodes which aren't part of the context of reduction*

- **should__lab:unaffected**$(x) \overset{\text{def}}{=}$ **ReductionDomain**(settings) $\land$ lab$(x) \land$
  $$[\neg\textbf{ReductionContext}(x)]$$
- **should__rel:unaffected**$(x, y) \overset{\text{def}}{=}$ **ReductionDomain**(settings) $\land$ rel$(x, y) \land$
  $$[\neg\textbf{ReductionContext}(x) \land \neg\textbf{ReductionContext}(y)]$$

In the case of //amusĭn-agán//, these helper predicates output all the morphological nodes, PStems, the underlined segments //a̠mu̠sĭn-a̠gán//, and the underlined syllable nodes //a̠.mu.sĭ.n-a.gán//. I show the eventual output of these predicates below for *amusin-agan*. The other segments will be generated by helper predicate which I will later introduce. Note that the output is not generated yet until the above helper predicates are used in an output function.

(517)   *Applying vowel reduction in* //amusĭn-agán//→amusn-agán *– eventually outputting unaffected nodes*

Input

$$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to \breve{\imath}_{0.5} \to n_{0.6} \longrightarrow a_{0.16} \to g_{0.17} \to á_{0.18} \to n_{0.19}$$

$$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.23} \longrightarrow \sigma_{0.24}$$

Output

$$a_{1.1} \to m_{1.2} \to u_{1.3} \qquad\qquad a_{1.16} \to g_{1.17} \to á_{1.18} \to n_{1.19}$$

$$\sigma_{1.12} \qquad\qquad \sigma_{1.24}$$

6.6.1.4   Formalizing resyllabification contexts for reduction

The other two scenarios of **Schwa Scenario** and **Deletion Scenario** involve more work. Given a destressed high vowel, we check if its deletion would create a resyllabifiable consonant cluster. In this section, I define relevant predicates for locally checking if resyllabification (and thus deletion) should apply.

First, we check that the high vowel's onset *could* resyllabify with the previous syllable: the onset *s* in //a.mu.sĭ.n-a.gán// can resyllabify as a coda in *a.mus.n-a.gán*. This information is formalized by the predicate **can_left_resyllabify**(x). The onset x is resyllabifiable if it is a consonant which either i) follows a vowel, or ii) follows a vowel-consonant sequence such that the onset and the preceding consonant could form a falling sonority complex coda. This predicate is locally-computed.

(518)   a.   *FO user-defined predicate for checking if an onset can resyllabify into its left context as a coda*
- **can_left_resyllabify**$(x) =$ consonant$(x)\wedge$
  $[\exists v[\text{vowel}(v) \wedge \text{succ:seg}(v,x)]\vee$
  $[\exists u,v[\text{vowel}(u)\wedge\text{consonant}(v)\wedge\text{succ:seg}(u,v)\wedge\textbf{good\_CC}(v,x)]]$

b.   *QF user-defined predicate for checking if an onset can resyllabify into its left context as a coda*
- **can_left_resyllabify**$(x) =$ consonant$(x)\wedge$
  $[\text{vowel}(\text{F}_\text{R}\text{:succ:seg}(x))\vee$
  $[\text{consonant}(\text{F}_\text{R}\text{:succ:seg}(x)) \wedge \text{vowel}(\text{F}_\text{R}\text{:succ:seg}^2(x))\wedge$
  $\textbf{good\_CC}(\text{consonant}(\text{F}_\text{R}\text{:succ:seg}(x)))]]$

For example, the underlined onset in //amusĭn-agán// and //barsĭg-astán// are resyllabifiable in their output form: the following input-output pairs are resyllabifiable: *amusn-agán, barsg-astán*. The onsets here satisfy the predicate **can_left_resyllabify**(x) by either following a vowel or a VC with falling sonority. In contrast, the onsets in //kĭr-óɣ// and //aɣmŭg-él// are not resyllabifiable: *kər-óɣ, *kr-óɣ* and *aɣməg-el, *aɣmg-él*. These onsets do not satisfy the predicate **can_left_resyllabify**(x) because they are word-initial or follow a VC sequence where the C has lower sonority.

A second condition on deletion is that the destressed high vowel must not have a complex coda in the base *barísp*, thus a coda in the intermediate representation //baris.p-él// (discussed by Ġaragyowlyan 1979). The output is reduction to schwa instead of deletion: *barəsp-él, *barsp-él*. The predicate below checks if some syllable x is an open syllable or not, by checking if it has an inner coda y. The predicate is locally computible.

(519)   a. *FO user-defined predicate to check if a syllable is an open syllable*
  - $\textbf{open\_syll}(x) \overset{\text{def}}{=} \textsf{syll}(x) \wedge \neg\exists y[\textsf{PDom:syll\_coda1}(x,y)]$

   b. *QF user-defined predicate to check if a syllable is an open syllable*
  - $\textbf{open\_syll}(x) \overset{\text{def}}{=} \textsf{syll}(x) \wedge \textsf{F}_\textsf{M}\textsf{:PDom:syll\_coda1}(x)$

Thus a destressed high vowel is deletable based on whether its onset is resyllabifiable, and whether it is an open syllable. These two conditions are summarized into the user-predicate below. For some segment vowel x, this predicate is TRUE if i) x is a destressed high vowel, ii) its syllable z has a resyllabifiable onset y, and iii) its syllable z is an open syllable. This predicate is locally-computed.

(520)   a. *FO user-defined predicate for checking if a destressed high vowel can be deleted*
  - $\textbf{deletable\_destressed\_vowel}(x) \overset{\text{def}}{=} \textbf{destressed\_high\_vowel}(x) \wedge$
    $\exists y, z[\textbf{destressed\_ons}(y) \wedge \textbf{destressed\_syll}(z) \wedge$
    $\textbf{can\_left\_resyllabify}(y) \wedge \textbf{open\_syll}(z)$

   b. *QF user-defined predicate for checking if a destressed high vowel can be deleted*
  - $\textbf{deletable\_destressed\_vowel}(x) \overset{\text{def}}{=} \textbf{destressed\_high\_vowel}(x) \wedge$
    $\textbf{destressed\_ons}(\textsf{F}_\textsf{R}\textsf{:succ:seg}(x)) \wedge$
    $\textbf{can\_left\_resyllabify}(\textsf{F}_\textsf{R}\textsf{:succ:seg}(x)) \wedge$
    $\textbf{open\_syll}(\textsf{F}_\textsf{D}\textsf{:PDom:syll\_nuc}(x))$

In the case of the input //amusǐn-agán//, this predicate is TRUE of the vowel $i_{0.5}$ (x) because the vowel is a destressed high vowel, ii) its syllable $\sigma_{0.14}$ (z) has the consonant $s_{0.4}$ (y) as its onset and this consonant s can be resyllabified to the preceding segments, and iii) the destressed high vowel's syllable $\sigma_{0.14}$ (z) is an open syllable.

It is useful to know if some node x is in the reduction context *and* if the reduction context contains a deletable destressed vowel y. The user-defined predicate **DeletionContext** does this. The underlined segments in //amusǐn-agán// satisfy this predicate because we have a deletable destressed vowel i; in contrast, the underlined segments //kǐr-óɣ// do not satisfy this predicate.

(521)   *FO user-defined predicate for selecting nodes in the reduction context when have a deletable destressed vowel*
  - $\textbf{DeletionContext}(x) \overset{\text{def}}{=} \textbf{ReductionContext}(x) \wedge \exists y[\textbf{deletable\_destressed\_vowel}(y)]$

This predicate is locally computible. The existential quantifier y finds a deletable destressed vowel. However, because x is part of the reduction context, then y is within the local context of x. We need to find y by checking where x is in the reduction context. For example, if x is a destressed syllable, then y must be its nucleus.

(522)   *QF user-defined predicate for selecting nodes in the reduction context when have a deletable destressed vowel*
  - $\textbf{DeletionContext}(x) \overset{\text{def}}{=} \textbf{ReductionContext}(x) \wedge$
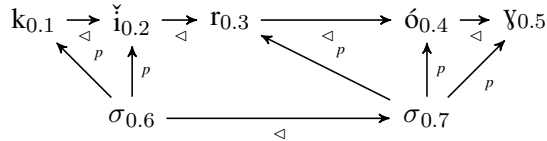 $[\,[\textbf{destressed\_high\_vowl}(x) \wedge$

$$\begin{aligned}
&\textbf{deletable\_destressed\_vowel}(x)]\lor\\
&[\textbf{destressed\_syll}(x)\land\\
&\qquad\textbf{deletable\_destressed\_vowel}(\mathsf{F_M{:}PDom{:}syll\_nuc}(x))]\lor\\
&[\textbf{destressed\_ons}(x)\land\\
&\qquad\textbf{deletable\_destressed\_vowel}(\mathsf{F_L{:}succ{:}seg}(x))]\lor\\
&[\textbf{following\_seg}(x)\land\\
&\qquad\textbf{deletable\_destressed\_vowel}(\mathsf{F_R{:}succ{:}seg}(x))]\lor\\
&[\textbf{previous\_syll}(x)\land\\
&\qquad\textbf{deletable\_destressed\_vowel}(\mathsf{F_M{:}PDom{:}syll\_nuc}(\mathsf{F_L{:}succ{:}syll}(x)))]\lor\\
&[\textbf{following\_syll}(x)\land\\
&\qquad\textbf{deletable\_destressed\_vowel}(\mathsf{F_M{:}PDom{:}syll\_nuc}(\mathsf{F_R{:}succ{:}syll}(x)))]\,]\,]
\end{aligned}$$

The above local predicates do the brunt of work in predicting when we should trigger reduction to schwa vs. deletion and the ensuing resyllabification. The next two sections use these predicates to generate the right outputs.

6.6.1.5 Schwa scenario: Formalizing reduction to schwa

I first formalize the **Schwa Scenario** for a possible input-output pair like //kĭr-óɣ//→*kər-óɣ*. Here, the destressed high vowel is reduced to a schwa instead of deleting because deletion would form an unsyllabifiable consonant cluster.

(523)   a.   *Input of vowel reduction in //kĭr-óɣ//*

   b.   *Input of vowel reduction in //kĭr-óɣ//→kər-óɣ with reduction to schwa*
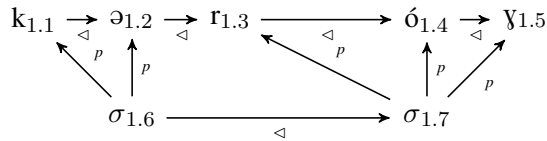
The high vowel $i_{0.2}$ (x) does not satisfy the predicate **deletable_destressed_vowel**(x) because its onset $k_{0.1}$ (y) is not resyllabifiable. The word-initial onset does not satisfy the predicate **can_left_resyllabify**(x) because it does not follow a vowel or a VC sequence that it could syllabify with.

For the **Schwa Scenario**, different segments are processed differently. The destressed high vowel $\breve{i}_{0.2}$ should reduce to a schwa, while all other nodes in the reduction context $(k_{0.1}, r_{0.3}, \sigma_{0.6}, \sigma_{0.7})$ should be faithfully outputted. The following helper predicates will distinguish the high vowel from the rest of the reduction context. The high vowel in *kĭr-óɣ* *should* output as a schwa iff i) we are in the context of vowel reduction, ii) the high vowel is a destressed high vowel, and iii) this destressed vowel is not deletable.

(524)   *QF helper predicate for outputting a destressed high vowel as a schwa if it can't be deleted*

- **should__schwa__schwa:vowel**(x^1) $\overset{\text{def}}{=}$ **ReductionDomain**(SETTINGS)$\wedge$
  destressed_high_vowel$(x)\wedge\neg$**deletable_destressed_vowel**(x)

All other underlying labels for this destressed high vowel should output faithfully, e.g., that it is a segment or a vowel. This is formalized with the helper predicate below.

(525)   *QF helper predicate for outputting other labels of a destressed high vowel in the **Schwa Scenario***

- For every lab $\in L - \{$schwa$\}$:
  should__lab__schwa:vowel(x^1) $\overset{\text{def}}{=}$ **ReductionDomain**(SETTINGS) $\wedge$ lab$(x)\wedge$
  destressed_high_vowel$(x)\wedge\neg$**deletable_destressed_vowel**(x)

For all other nodes in the reduction context $(k_{0.1}, r_{0.3}, \sigma_{0.6}, \sigma_{0.7})$, their labels and relations *should* be faithfully outputted via the predicate below. We need to specify that in this scenario that i) we have the right cophonology domain for reduction, ii) the relevant node is part of the context of vowel reduction, iii) it is not the destressed high vowel, and iv) we are not in the deletion context, i.e., there is no deletable destressed high vowel.

(526)   *QF helper predicate for outputting labels of other nodes in the context of reduction in the **Schwa Scenario***

- **should__lab__schwa:other**(x^1) $\overset{\text{def}}{=}$ **ReductionDomain**(SETTINGS) $\wedge$ lab$(x)\wedge$
  ReductionContext$(x) \wedge \neg$**destressed_high_vowel**$(x)\wedge$
  $\neg$**DeletionContext**(x)

In the **Schwa Scenario**, all these potential changes in labels are summarized in the helper predicates below: one for the schwa label, and one for any other label.

(527)   *QF helper predicate for summarizing what labels should be outputted in the **Schwa Scenario***

- For the label schwa
  should__schwa__schwa(x) $\overset{\text{def}}{=}$ **should__schwa__schwa:vowel**$(x)\vee$
  should__schwa__schwa:other(x)
- For other labels **lab** $\in L - \{$schwa$\}$
  should__lab__schwa(x) $\overset{\text{def}}{=}$ **should__lab__schwa:vowel**$(x)\vee$
  should__lab__schwa:other(x)

In the **Schwa Scenario**, all relations involving the reduction context *should* be faithfully outputted. The helper predicate below again checks that i) we have the right cophonology domain, and ii) that the relevant relation involves a node which is in the reduction context, and iii) we are not in the deletion context, i.e., there is no deletable destressed vowel.

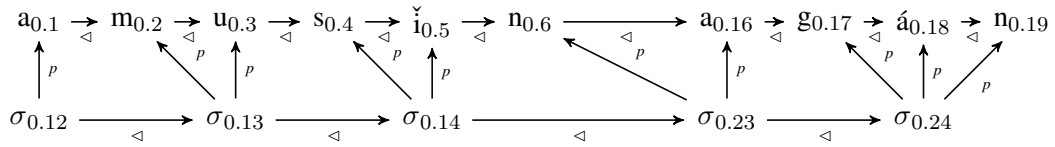(528)   *QF helper predicate for faithfully outputting relations in the **Schwa Scenario***

- **should__rel__schwa**(x^1, y^1) $\overset{\text{def}}{=}$ **ReductionDomain**(SETTINGS) $\wedge$ rel$(x,y)\wedge$
  $[$**ReductionContext**$(x) \vee$ **ReductionContext**$(y)]\wedge$
  $\neg$**DeletionContext**$(x) \wedge \neg$**DeletionContext**$(x)\wedge$

No quantifiers were used in definition the helper predicates, thus they are computationally local.
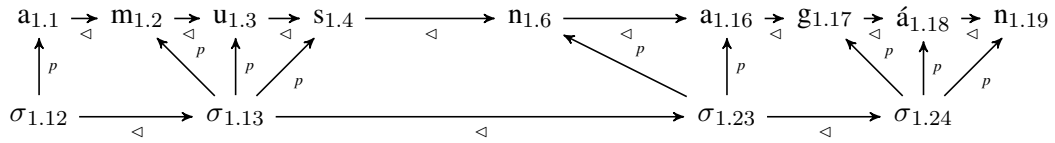
6.6.1.6 Deletion scenario: Formalizing deletion

Vowel deletion in the **Deletion Scenario** is more complicated because it involves resyllabification. I repeat the input-output of vowel reduction for *amusn-agan*. I omit the SETTINGS, morphological, and PStem nodes.

(529)   a.   *Input of vowel reduction in //amusĭn-agán//*

$$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to \breve{\imath}_{0.5} \to n_{0.6} \longrightarrow a_{0.16} \to g_{0.17} \to á_{0.18} \to n_{0.19}$$

$$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.23} \longrightarrow \sigma_{0.24}$$

   b.   *Output of vowel reduction in //amusĭn-agán//→amusn-agán*

$$a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \longrightarrow n_{1.6} \longrightarrow a_{1.16} \to g_{1.17} \to á_{1.18} \to n_{1.19}$$

$$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.23} \longrightarrow \sigma_{1.24}$$

In the case of the input //amu<u>sĭn</u>-agán//, the underlined segments within the reduction context satisfy the deletion context, i.e., the high vowel *i* is deletable. Within the deletion context, all nodes *should* be faithfully outputted *except* for the destressed high vowel $\breve{\imath}_{0.5}$ and its destressed syllable $\sigma_{0.14}$. This is formalized with the helper predicate below. As before, we check that the right cophonology domain for reduction is active via the SETTINGS.

(530)   *QF helper predicate for outputting the labels of all nodes in the reduction context except for the deleting high vowel and its syllable in the **Deletion Scenario***

- **should__lab__deleted**$(x) \stackrel{\text{def}}{=}$ **ReductionDomain**(SETTINGS) $\wedge$ lab$(x)\wedge$
  DeletionContext$(x)\wedge$
  $\neg$**destressed_high_vowel**$(x) \wedge \neg$**destressed_syll**(x)

To handle resyllabification from //a.mu.<u>sĭn</u>-a.gán// to *a.mus.n-a.gán*, we need to change certain relations in the input:

1. **Inner Coda Creation**: The onset $s_{0.4}$ must resyllabify as an inner coda to the preceding syllable $\sigma_{0.13}$ or *.mus.*, if the syllable is an open syllable
2. **Syllable Adjacency**: The syllables before ($\sigma_{0.13}$) and after ($\sigma_{0.23}$) the destressed syllable *.mus.<i>.na* must be made adjacent via immediate precedence
3. **Segment Adjacency**: The segments before ($s_{0.4}$) and after ($n_{0.5}$) the destressed high vowel *s<i>n* must be made adjacent via immediate precedence

These 3 *potential* changes are formalized with the helper predicates below. They all check that we are in the right cophonology domain, that the relevant nodes are part of the deletion context, and that they single out the nodes described in the above list.

(531)   *QF helper predicates used in the **Deletion scenario** in order to formalize...*

- *Inner Coda Creation*
  should__PDom:coda1__deleted:new$(x, y) \stackrel{\text{def}}{=}$
  $\qquad$ **ReductionDomain**(SETTINGS) $\wedge$ **DeletionContext**$(x)\wedge$
  $\qquad$ **destressed_ons**$(y) \wedge$ **previous_syll**$(x) \wedge$ **open_syll**(x)

- *Syllable Adjacency*
  should__succ:syll__deleted:new$(x, y) \stackrel{\text{def}}{=}$
  $\qquad$ **ReductionDomain**(SETTINGS) $\wedge$ **DeletionContext**$(x)\wedge$
  $\qquad$ **previous_syll**$(x) \wedge$ **following_syll**(x)

- *Segment Adjacency*
  should__succ:seg__deleted:new$(x, y) \stackrel{\text{def}}{=}$
  $\qquad$ **ReductionDomain**(SETTINGS) $\wedge$ **DeletionContext**$(x)\wedge$
  $\qquad$ **destressed_ons**$(x) \wedge$ **following_seg**(x)

If the destressed high vowel was deletable but its onset followed a VC sequence, i.e., a closed syllable, then the onset should resyllabify as an outer coda: //bar.sĭ.g-a.gán//→*bars.g-agán*. This is formalized by the helper predicate below. Note that we need to check that the previous syllable is not an open syllable.

(532)   *QF helper predicates used in the **Deletion scenario** in order to formalize Outer Coda Creation*

- **should__PDom:coda2__deleted:new**$(x, y) \stackrel{\text{def}}{=}$
  $\qquad$ **ReductionDomain**(SETTINGS) $\wedge$ **DeletionContext**$(x)\wedge$
  $\qquad$ **destressed_ons**$(y) \wedge$ **previous_syll**$(x) \wedge \neg$**open_syll**(x)

The above 4 helper predicates for resyllabification should create *new* binary relations in the output, i.e., new prosodic dominances for codas and new immediate precedences for syllables and segments. Within the deletion context, some of the *old* underlying prosodic dominances or immediate precedences should be faithfully outputted, e.g., the syllable preceding the high vowel in //amusĭn-agán// should still follow the syllable *a*: *amusn-agán* . These 'old' relations should be faithfully outputted via the helper predicates below.

(533)   *QF helper predicate for faithfully outputting underlying prosodic dominances in the deletion context in the **Deletion Scenario***

- For every relation rel $\in$ {PDom:coda1, prec:syll, prec:seg, PDom:coda2}
  should__rel__deleted:old$(x, y) \stackrel{\text{def}}{=}$ **ReductionDomain**(SETTINGS) $\wedge$ rel$(x, y)\wedge$
  $\qquad$ [**DeletionContext**$(x) \vee$ **DeletionContext**$(y)]\wedge$
  $\qquad \neg$[**destressed_high_vowel**$(x)\vee$**destressed_high_vowel**$(y)\vee$
  $\qquad$ **destressed_syll**$(x) \vee$ **destressed_syll**$(y)]$

The helper predicate checks that we are in the right reduction domain, that one of the nodes is in the deletion context, and that none of the nodes are the destressed high vowel or the destressed syllable.

All other relations within the deletion context should be outputted faithfully as well. This is formalized via the helper predicate below. Again, we output the relations involving any nodes *except* for the destressed high vowel or its syllables.

(534) *QF helper predicate for faithfully outputting other relations in the deletion context in the **Deletion Scenario***

- For every relation rel $\in R - \{\mathsf{PDom{:}coda1}, \mathsf{prec{:}syll}, \mathsf{prec{:}seg}, \mathsf{PDom{:}coda2}\}$
  should__rel__deleted:other$(x, y) \overset{\text{def}}{=}$

  $\quad\quad$ **ReductionDomain**(SETTINGS) $\wedge$ rel$(x, y)\wedge$
  $\quad\quad$ **[DeletionContext**$(x) \vee$ **DeletionContext**$(y)]\wedge$
  $\quad\quad$ $\neg$**[destressed_high_vowel**$(x)\vee$**destressed_high_vowel**$(y)\vee$
  $\quad\quad$ **destressed_syll**$(x) \vee$ **destressed_syll**$(y)]$

Together, the above relations for the reduction context in the **Deletion Scenario** are summarized in the helper predicates below.

(535) *QF helper predicates for outputting prosodic relations in the **Deletion Scenario***
- For every relation rel $\in \{\mathsf{PDom{:}coda1}, \mathsf{prec{:}syll}, \mathsf{prec{:}seg}, \mathsf{PDom{:}coda2}$
  should__rel__deleted$(x, y) \overset{\text{def}}{=}$ **should__rel__deleted:old**$(x, y)\wedge$**should__rel__deleted:new**(x, y)
- For every relation rel $\in R - \{\mathsf{PDom{:}coda1}, \mathsf{prec{:}syll}, \mathsf{prec{:}seg}, \mathsf{PDom{:}coda2}$
  should__rel__deleted$(x, y) \overset{\text{def}}{=}$ **should_rel__deleted:other**(x, y)

All of the above helper predicates across the three scenarios (**Unaffected, Schwa, Deletion**) are used in the *two* output functions below. The above helper predicates kept track of various changes which should apply: deletion, resyllabification, etc. The output functions below *implement* these changes.

(536) *QF output functions for vowel reduction*
- For every lab $\in L$:
  $\phi\mathtt{lab}(x^1) \overset{\text{def}}{=}$ **should__lab__unaffected**$(x)\vee$**should__lab__schwa**$(x)\vee$**should__lab__deleted**(x)
- For every relation rel $\in R$:
  $\phi\mathtt{rel}(x^1, y^1) \overset{\text{def}}{=}$ **should__rel__unaffected**$(x, y)\vee$
  $\quad\quad$ **should__rel__schwa**$(x, y) \vee$ **should__rel__deleted**(x, y)

This completes the formalization of vowel reduction. All predicates were locally defined. However, to ensure consistency and readability of these predicates, we made extensive use of helper predicates.

6.6.2 Prosodic cophonologies: Non-locality of cophonology selection

In the previous section, the main examples were derivatives like *amusn-agan* 'marital' from *amusín* 'husband'. Because these words contained a derivational suffix, they underwent the stem-level cophonology, a cophonology or rule domain which is triggered by *morphological* structure. Armenian likewise has cophonologies which are triggered by *prosodic* structure. As explained in the Introduction chapter (§1.1.3), when PStems are misaligned via overparsing, they trigger the PStem-level cophonology. I formalize how this cophonology is triggered.

Unlike derivational suffixes, C-initial inflectional suffixes trigger the word-level cophonology of just stress shift without reduction: *amusin-nér, *amusən-nér* 'husband-PL'. But when the inflectional suffix is V-initial,
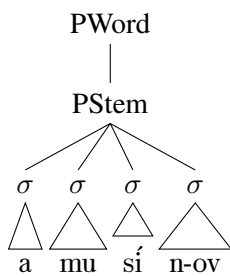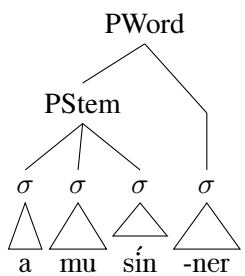
this suffix *can* trigger vowel reduction in some dialects like Eastern Armenian *amusn-óv*, but not Western Armenian *amusin-óv*. The word-level phonology then vacuously applies.

(537)   *Reduction across types of inflection and dialect*

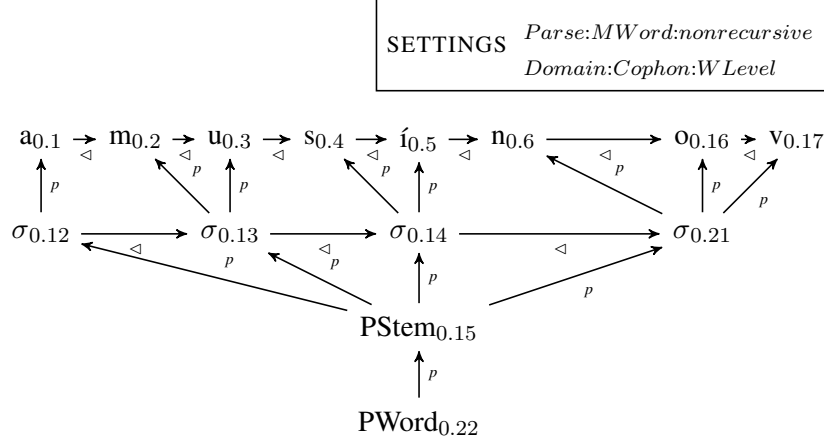|	V-initial		C-initial
Dialect	Eastern	Western	Both
Base	*amusín*	*amusín*	*amusín*
Inflected	*amusn-óv*	*amusin-óv*	*amusin-nér*

As explained in Chapter 1: §1.1.3, I analyze this fact as the existence of the *prosodic* cophonology which is triggered when a V-initial suffix is added to a stem and causes the prosodic misalignment of the prosodic stem (PStem). The prosodic analysis was formalized in §6.5.2.1. Once the prosody is in place by misaligning the PStem, the PStem-level cophonology applies. In Eastern Armenian, the PStem-level cophonology triggers stress shift and reduction; but in Western Armenian, the PStem-cophonology only triggers stress shift.

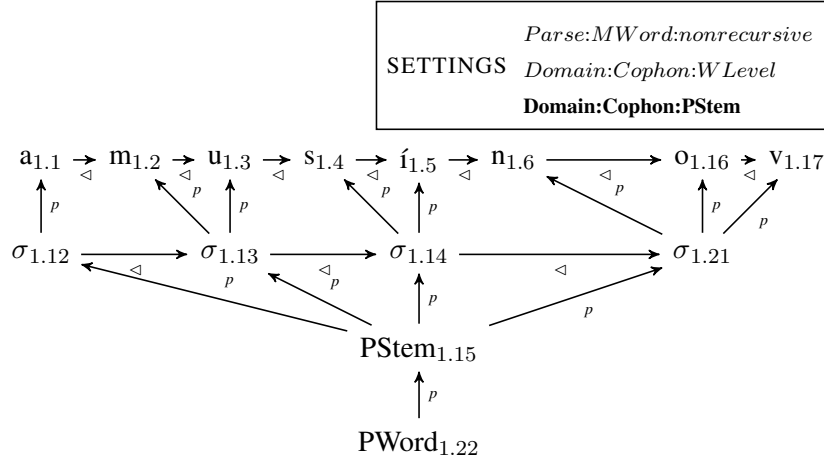(538)   *PStem cophonologies across dialects and inflection-shape*

Morphology	V-initial inflection in...		C-initial inflection in...
Dialect	Eastern	Western	Both
Prosody			
Does PStem apply?	✓	✓	✗
PStem-level processes	stress reduction	stress	
Output	amusn-óv	amusin-óv	amusin-nér

I formalize the selection of the PStem-cophonology based on PStem overparsing. To check if the PStem has overparsed, we check that the PStem contains an inflectional suffix. I am agnostic if this step happens after all the prosody is over, i.e., after a PWord is generated, or if it occurs right after the PStem has misaligned. Regardless, this step precedes the application of phonological rules. I show the input and output for when inflection is V-initial. The PStem dominates the syllables of the V-initial inflectional suffix *-ov*. I omit morphological nodes. The relevant change to the SETTINGS is adding the label Domain:Cophon:PStem in bold.

(539) a. *V-initial: Input to setting the PStem-level cophonology in* $//(\text{amus}\acute{\text{i}}\text{n-ov})_s//$

SETTINGS	$Parse{:}MWord{:}nonrecursive$
	$Domain{:}Cophon{:}WLevel$

$a_{0.1} \rightarrow m_{0.2} \rightarrow u_{0.3} \rightarrow s_{0.4} \rightarrow \acute{\text{i}}_{0.5} \rightarrow n_{0.6} \longrightarrow o_{0.16} \rightarrow v_{0.17}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.21}$

$\text{PStem}_{0.15}$

$\text{PWord}_{0.22}$

b. *V-initial: Output of setting the PStem-level cophonology in* $//(\text{amus}\acute{\text{i}}\text{n-ov})_s//$

SETTINGS	$Parse{:}MWord{:}nonrecursive$
	$Domain{:}Cophon{:}WLevel$
	Domain:Cophon:PStem

$a_{1.1} \rightarrow m_{1.2} \rightarrow u_{1.3} \rightarrow s_{1.4} \rightarrow \acute{\text{i}}_{1.5} \rightarrow n_{1.6} \longrightarrow o_{1.16} \rightarrow v_{1.17}$

$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.14} \longrightarrow \sigma_{1.21}$

$\text{PStem}_{1.15}$

$\text{PWord}_{1.22}$

This process must update the SETTINGS with the right PStem-level cophonology label. This is a transduction with a copy set of size 1. I assume it happens after the prosody, but before phonological rule application. In Copy 1, all labels and relations are faithfully outputted except for domain labels for prosodic cophonologies.

(540) *QF output functions for vacuous changes in updating the* SETTINGS
- For every label lab $\in L - \{\text{Domain:Cophon:PStem}\}$:
  $\phi\texttt{lab}(x^1) \overset{\text{def}}{=} \text{lab}(x)$
- For every relation rel $\in R$:
  $\phi\texttt{rel}(x^1, y^1) \overset{\text{def}}{=} \text{rel}(x, y)$

On the surface, we know if a PStem is misaligned or overparsed *if* the PStem contains any segments from an inflectional suffix. This information is broken down into the following predicates.

(541)   a. *FO user-defined predicate for checking if a segment is part of an inflectional morpheme in an MWord*
- **seg_in_MWord**$(x) \overset{\text{def}}{=} \text{seg}(x) \wedge \exists u, v, w[\text{MWord}(u) \wedge \text{morpheme}(v) \wedge \text{morph}(w) \wedge$
$\text{MDom}(u,v) \wedge \text{MDom}(v,w) \wedge \text{MDom}(w,x)]$

   b. *FO user-defined predicate for checking if the PStem has some segment*
- **seg_in_PStem**$(x, y) \overset{\text{def}}{=} \text{seg}(x) \wedge \text{PStem}(y) \exists z[\text{syll}(z) \wedge \text{PDom:PStem\_syll}(y, z) \wedge$
$[\text{PDom:syll\_ons}(z, x) \vee \text{PDom:syll\_nuc}(z, x) \vee$
$\text{PDom:syll\_coda1}(z, x) \vee \text{PDom:syll\_coda2}(z, x)]$

   c. *FO user-defined predicate for checking if a PStem has inflectional segments*
- **infl_in_PStem**$(x) \overset{\text{def}}{=} \text{PStem}(x) \wedge \exists y[\text{seg}(y) \wedge \textbf{seg\_in\_PStem}(y, x) \wedge \textbf{seg\_in\_MWord}(y)]$

The predicate **seg_in_MWord**(x) checks if some segment x is part of an MWord, i.e., it it is part of an inflectional suffix. This predicate was introduced in §6.5.2.1. The predicate **seg_in_PStem**(x, y) checks if a segment x is dominated by the PStem y by checking if there is some syllable z which is between the PStem y and the segment x (whether x acts as z's onset, nucleus, or coda). In *(amusin)$_s$-ner*, this predicate is satisfied by the PStem and the segments *a,m,u,s,i,n*. But in *(amusin-ov)$_s$*, the predicate picks the suffix segments *-ov* as well. The predicate **infl_in_PStem**(x) checks if the PStem x contains an inflectional segment y. The predicate is true for the PStem in *(amusin-ov)$_s$* but not in *(amusin)$_s$-ner*.

Of these predicates, the first two are locally-computed and QF definable. However, the third predicate **infl_in_PStem**(x) is not locally-computible. This predicate involves giving a property to the PStem x based on examining the properties of its granddaughter y. But, a PStem can have an unbounded number of daughter syllables (and thus granddaughter segments) because of n-ary branching (cf. Chapter 4: §4.5.3). Thus given a PStem x, an existential quantifier is needed to search through the space of segments y and to check if any of these segments are part of the PStem.

(542)   a. *QF user-defined predicate for checking if a segment is part of an inflectional morpheme in an MWord*
- **seg_in_MWord**$(x) \overset{\text{def}}{=} \text{seg}(x) \wedge \text{morph}(\text{F}_\text{D}\text{:MDom}(x)) \wedge \text{morpheme}(\text{F}_\text{D}\text{:MDom}^2(x)) \wedge$
$\text{MWord}(\text{F}_\text{D}\text{:MDom}^3(x))$

   b. *QF user-defined predicate for checking if the PStem has some segment*
- **seg_in_PStem**$(x, y) \overset{\text{def}}{=} \text{seg}(x) \wedge \text{PStem}(y) \wedge$
$\text{F}_\text{D}\text{:PDom:PStem\_syll}(\text{F}_\text{D}\text{:PDom:syll\_ons}(x)) = y \vee$
$\text{F}_\text{D}\text{:PDom:PStem\_syll}(\text{F}_\text{D}\text{:PDom:syll\_nuc}(x)) = y \vee$
$\text{F}_\text{D}\text{:PDom:PStem\_syll}(\text{F}_\text{D}\text{:PDom:syll\_coda1}(x)) = y \vee$
$\text{F}_\text{D}\text{:PDom:PStem\_syll}(\text{F}_\text{D}\text{:PDom:syll\_coda2}(x)) = y$

With these predicates, we know some PStem x overparses its MStem if it satisfies the predicate **infl_in_PStem**(x). This information is encoded into the SETTINGS via the label and output function below. I redundantly check that the misaligned PStem has the right cophonology label.

(543)   a. *Unary label for the domain of the PStem cophonology*
- Domain:Cophon:PStem(SETTINGS): the SETTINGS has the domain of the PStem cophonology because the PStem expanded

   b. *FO output function for updating the* SETTINGS *with the PStem cophonology label*

- ϕ`Domain:Cophon:PStem`$(\text{SETTINGS}^1) \stackrel{\text{def}}{=}$ `Parse:MWord:nonrecursive`$(\text{SETTINGS}) \land$
  $\exists x[\text{PStem}(x) \land \text{Cophon:PStem}(x) \land \mathbf{infl\_in\_PStem}(x)]$

For the input *(amusín-ov)$_s$* with V-initial inflection, the SETTINGS now has labels for *two* separate cophonologies to apply: the word-level cophonology and PStem-level cophonology. The competition and ordering of these two domains is handled in §6.6.3.3. Contrast this with the word *(amusin)$_s$-ner* which has C-initial inflection. The input and output are shown below and are identical. Here, the PStem does not dominate any inflectional segment. Thus, the output will not contain a label for the PStem-level cophonology.

(544)   a. *C-initial: Input to* trying to set the PStem-level cophonology in //(amusín)$_s$-ner//
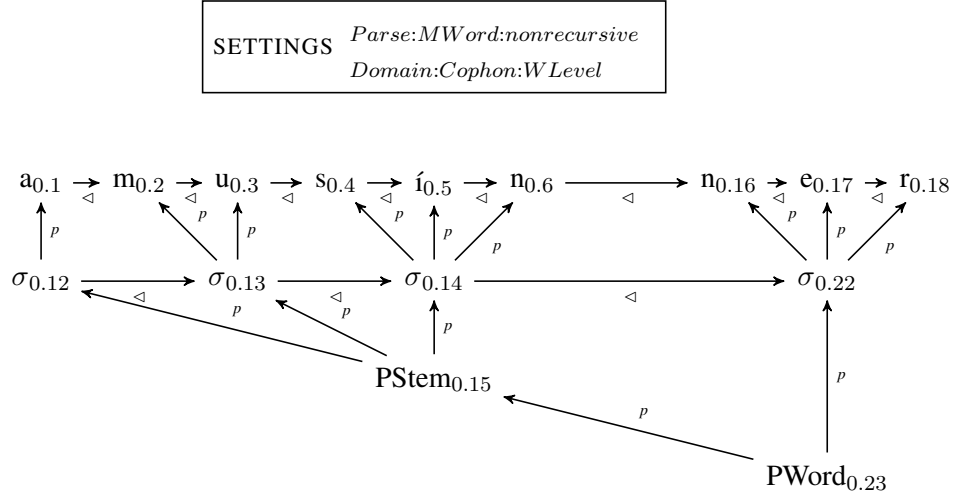
    b. *C-initial: Output of* trying to set the PStem-level cophonology in //(amusín)$_s$-ner//
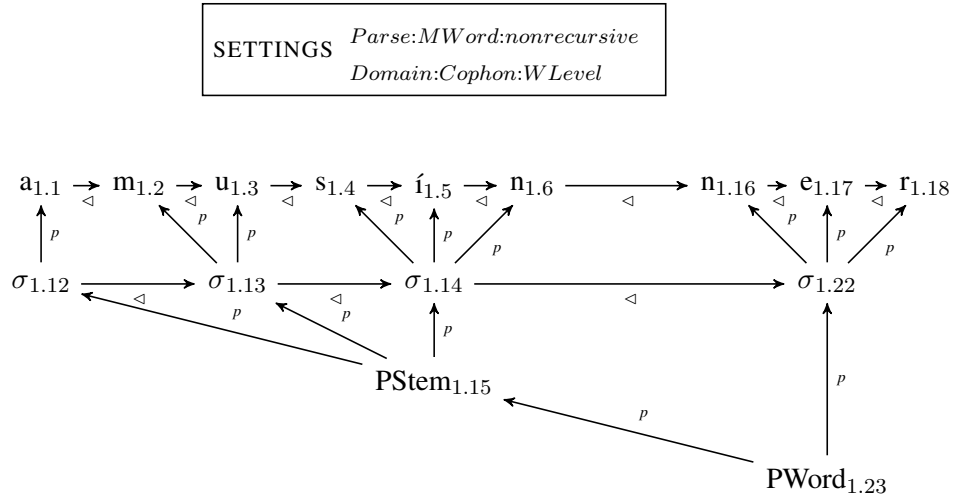
6.6.3   Word-level cophonology: Varied application of the right cophonology

Having formalized the main processes (reduction and stress shift), the stem-level cophonology, and the PStem-level cophonology, I now turn to the word-level cophonology. Depending on dialect, on the shape of the suffix, and on the size of PStem, different cophonologies and rules may apply before inflectional morphology. This is summarized in the table below.

(545)   *Illustrating the application of different cophonologies in inflection*

	V-initial inflection in...		C-initial inflection in...
	Eastern	Western	Both
Generating PWord			
Applying cophonology of...			
EArm PStem-level Stress Reduction	amusĭn-óv amusn-óv		
WArm PStem-level Stress		amusĭn-óv	
Word-level Stress	amusn-óv	amusin-óv	amusin-nér
Output	amusn-óv	amuson-óv	amusin-nér

Before V-initial inflection, the PStem-level cophonology applies. In Eastern Armenian, this cophonology includes stress shift and vowel reduction. In Western Armenian, this cophonology includes only stress shift. The PStem-level cophonology does not apply before C-initial inflection. At the end, all types of inflected items undergo the word-level cophonology which only includes stress shift.

The illustration above has the PStem-level cophonology apply *before* the word-level cophonology. However, in the formalization, I assume that the PStem-level and word-level cophonologies *simultaneously* apply. In C-initial inflection *amusin-nér*, the word-level process of stress shift applies. In V-initial inflection, the PStem-level and word-level processes of stress shift apply. In Western Armenian, reduction does not apply because reduction is not part of the word-level or WArm PStem-level: *amusin-óv*. In Eastern Armenian, reduction applies because it is part of the EArm PStem-level: *amusn-óv*.

I first go over the basic notation and predicates for formalizing dialects and rule domains (§6.6.3.1). I show how the word-level phonology applies in C-initial inflection (§6.6.3.2). I then finally show how both the PStem- and word-level cophonologies apply in V-initial inflection (§6.6.3.3).

6.6.3.1 Computing dialects and domains

Throughout the previous two chapters, the SETTINGS constant encoded only two types of information: Domain labels for cophonologies, and Parse labels for prosody. The Domain and Prosody labels were determined by examining the properties of the morphologically topmost tree (its label, what it dominates), and by checking if there existed a misaligned PStem. In this section, I further enrich the SETTINGS by letting it encode the relevant *dialect* of the derivation..

(546)   *Unary labels for dialect* SETTINGS

- Western(SETTINGS) or Eastern(SETTINGS)

The individual rules for stress and reduction were formalized in previous chapters or sections: Chapter 5: §5.5 for stress, this chapter at §6.6.1 for reduction. Each rule was specified to occur in some domain based on the SETTINGS of the derivation. For stress shift, it applies whenever we are in the stem-level, PStem-level, or word-level cophonologies. This was encoded in the user-predicate (547a), repeated from Chapter 5: §5.5. For reduction, it applies in the stem-level cophonology and the EArm PStem-level cophonology. This was encoded in the predicate (547b), repeated from §6.6.1. This predicates are all we need to understand how cophonologies compete and apply in inflection.

(547)   a. *QF user-defined predicate for morphophonological domain of stress assignment*

- **StressDomain**(SETTINGS) $\stackrel{\text{def}}{=}$ Domain:Cophon:SLevel(SETTINGS) $\vee$
  Domain:Cophon:WLevel(SETTINGS) $\vee$
  Domain:Cophon:PStem(SETTINGS)

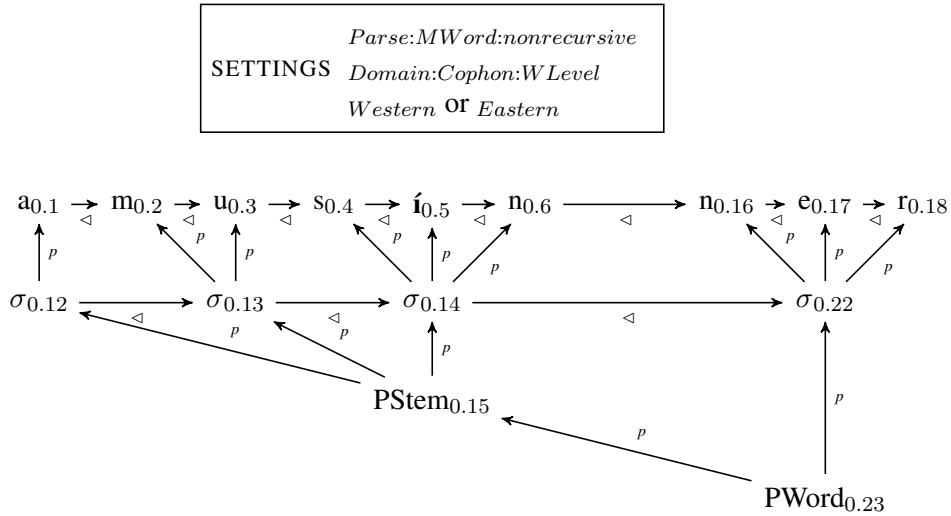b. *QF user-defined predicate for checking that we are in the cophonology domain for vowel reduction*

- **ReductionDomain**(SETTINGS) $\stackrel{\text{def}}{=}$
  Domain:Cophon:SLevel(SETTINGS)$\vee$
  [Domain:Cophon:PStem(SETTINGS) $\wedge$ Eastern(SETTINGS)]
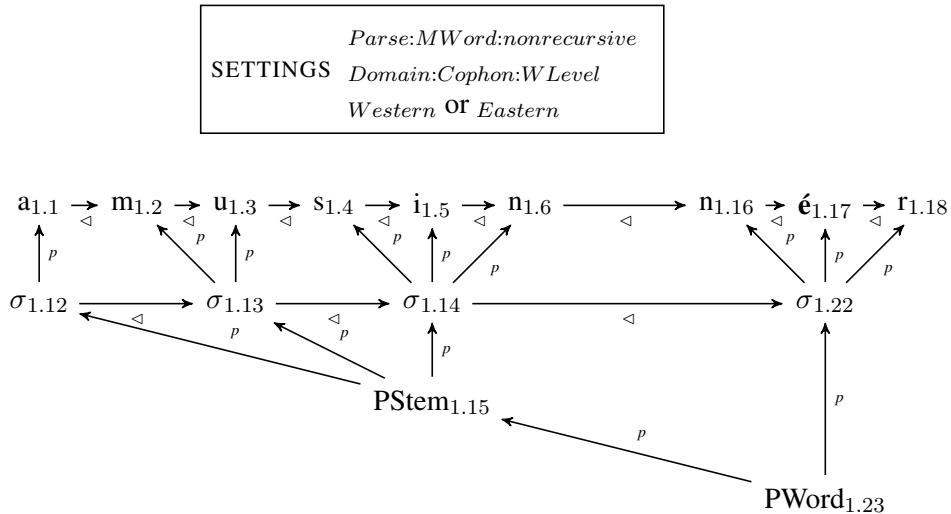
6.6.3.2 Activation of the word-level phonology

I formalize how the word-level phonology applies. In C-initial inflection, only stress shift applies: *amusin-nér*. The SETTINGS has the domain label of the word-level cophonology, as explained in §6.3.1. I show the input below. I omit morphological nodes. The stressed vowel is in bold. The SETTINGS can have the dialectal label of either Western or Eastern.

(548) *Input to cophonology application – C-initial inflection in* amusin-ner

$$\begin{array}{l}\text{SETTINGS}\quad\begin{array}{l}Parse{:}MWord{:}nonrecursive\\Domain{:}Cophon{:}WLevel\\Western\text{ or }Eastern\end{array}\end{array}$$

$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to \acute{\imath}_{0.5} \to n_{0.6} \longrightarrow n_{0.16} \to e_{0.17} \to r_{0.18}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.22}$

$\text{PStem}_{0.15}$

$\text{PWord}_{0.23}$

The predicate **StressDomain**(SETTINGS) is TRUE because the SETTINGS contains the label for the word-level cophonology Domain:Cophon:WLevel. The rule of stress shift will apply using the *same* formula as in Chapter 5: §5.5. I do not repeat the formalization here, but interested readers are encouraged to work this out for themselves. Reduction does not apply because the input does not satisfy the **ReductionDomain**(SETTINGS). I show the output below.
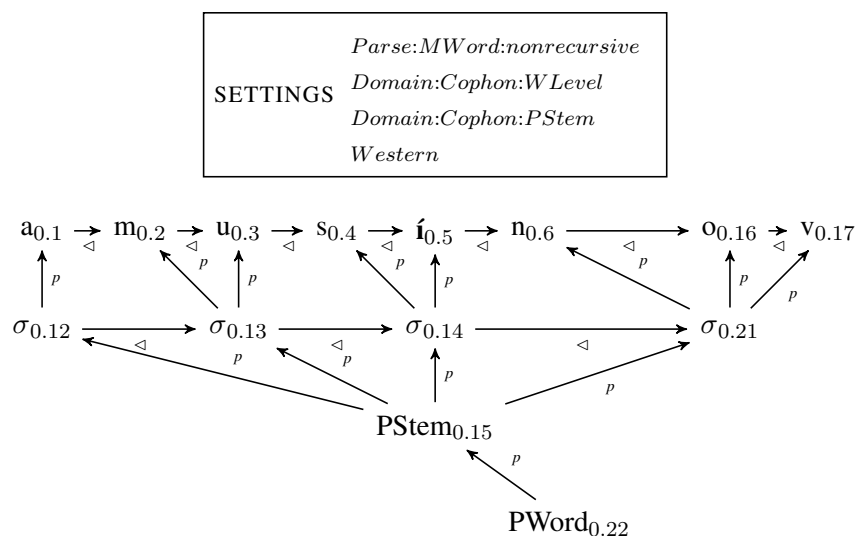
(549) *Output of cophonology application – C-initial inflection in* amusin-ner

$$\begin{array}{l}\text{SETTINGS}\quad\begin{array}{l}Parse{:}MWord{:}nonrecursive\\Domain{:}Cophon{:}WLevel\\Western\text{ or }Eastern\end{array}\end{array}$$

$a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \to i_{1.5} \to n_{1.6} \longrightarrow n_{1.16} \to \acute{e}_{1.17} \to r_{1.18}$

$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.14} \longrightarrow \sigma_{1.22}$
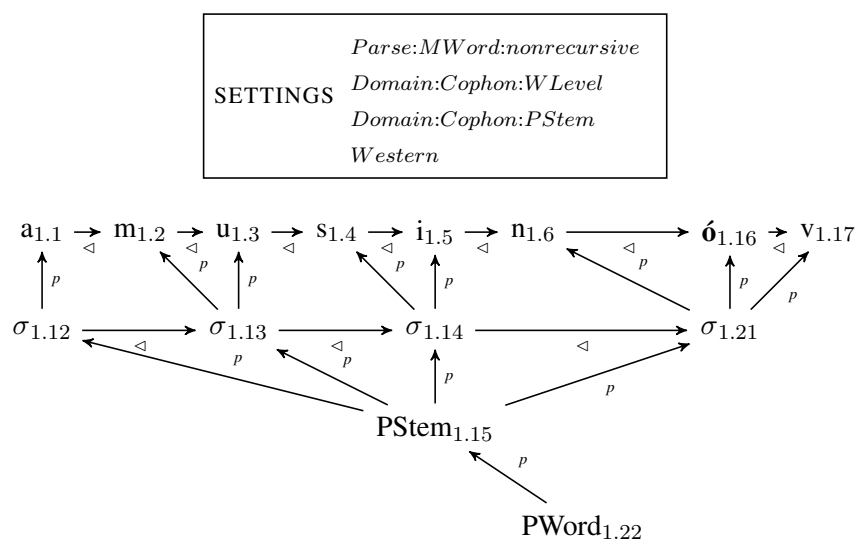
$\text{PStem}_{1.15}$

$\text{PWord}_{1.23}$

6.6.3.3 Simultaneous activation of the PStem-level and word-level phonology

Applying rules before V-initial inflection is slightly more complicated. I first explain with Western Armenian *amuson-óv*. Here, we have stress shift but no reduction. The input and output are shown below.

(550)   a. *Input to cophonology application – V-initial inflection in Western Armenian* $//\text{amusìn-ov}//$

$$
\boxed{
\begin{array}{ll}
\text{SETTINGS} & \begin{array}{l} \textit{Parse:MWord:nonrecursive} \\ \textit{Domain:Cophon:WLevel} \\ \textit{Domain:Cophon:PStem} \\ \textit{Western} \end{array}
\end{array}
}
$$

$a_{0.1} \rightarrow m_{0.2} \rightarrow u_{0.3} \rightarrow s_{0.4} \rightarrow í_{0.5} \rightarrow n_{0.6} \longrightarrow o_{0.16} \rightarrow v_{0.17}$

$\sigma_{0.12} \longleftarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.21}$

$\text{PStem}_{0.15}$

$\text{PWord}_{0.22}$

b. *Output of cophonology application – V-initial inflection in Western Armenian* amusin-óv

$$
\boxed{
\begin{array}{ll}
\text{SETTINGS} & \begin{array}{l} \textit{Parse:MWord:nonrecursive} \\ \textit{Domain:Cophon:WLevel} \\ \textit{Domain:Cophon:PStem} \\ \textit{Western} \end{array}
\end{array}
}
$$

$a_{1.1} \rightarrow m_{1.2} \rightarrow u_{1.3} \rightarrow s_{1.4} \rightarrow i_{1.5} \rightarrow n_{1.6} \longrightarrow ó_{1.16} \rightarrow v_{1.17}$

$\sigma_{1.12} \longleftarrow \sigma_{1.13} \longrightarrow \sigma_{1.14} \longrightarrow \sigma_{1.21}$

$\text{PStem}_{1.15}$

$\text{PWord}_{1.22}$

The SETTINGS has <u>two</u> domain labels for the word-level and PStem-level cophonologies. These are Domain:Cophon:WLevel and Domain:Cophon:PStem. Intuitively, the two cophonologies compete over which will apply: stress shift in the PStem-level: *(amusin-óv)$_s$* or stress shift in the word-level: *amusin-óv*. But for the formalization, this competition doesn't exist. The input satisfies the domain for stress shift **StressDomain**(SETTINGS) because the SETTINGS has at least *one* of the right domain labels: either Domain:Cophon:PStem or Domain:Cophon:Word.

Reduction does not apply because the input does not satisfy the predicate **ReductionDomain**(SETTINGS). The SETTINGS has the label for the PStem cophonology, but the input is from Western Armenian, not Eastern Armenian.

Moving on to V-initial inflection in Eastern Armenian, the input is shown below. The SETTINGS has the
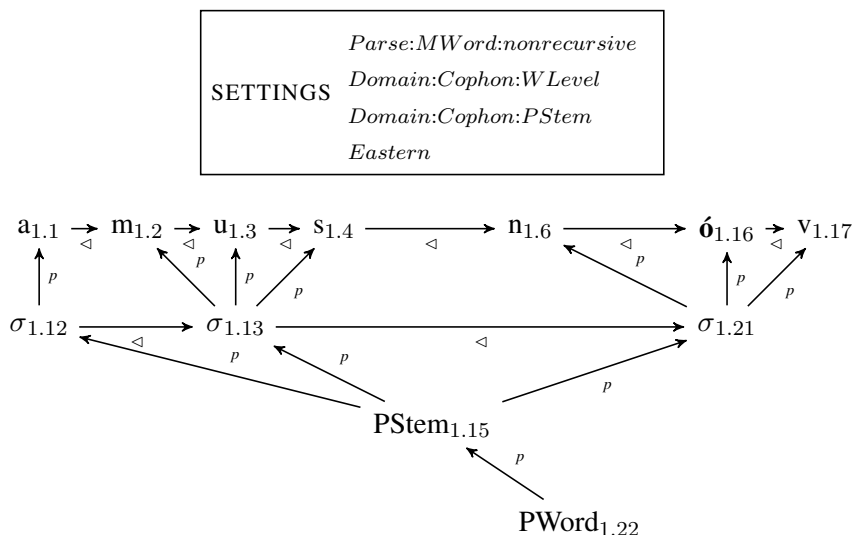
domain labels of the PStem-level and word-level cophonologies. It also encodes the right dialect as Eastern Armenian.

(551) *Input to cophonology application – V-initial inflection in Eastern Armenian* amusn-ov

$$
\begin{array}{|ll|}
\hline
 & Parse{:}MWord{:}nonrecursive \\
\text{SETTINGS} & Domain{:}Cophon{:}WLevel \\
 & Domain{:}Cophon{:}PStem \\
 & Eastern \\
\hline
\end{array}
$$

$a_{0.1} \to m_{0.2} \to u_{0.3} \to s_{0.4} \to í_{0.5} \to n_{0.6} \longrightarrow o_{0.16} \to v_{0.17}$

$\sigma_{0.12} \longrightarrow \sigma_{0.13} \longrightarrow \sigma_{0.14} \longrightarrow \sigma_{0.21}$

$\text{PStem}_{0.15}$

$\text{PWord}_{0.22}$
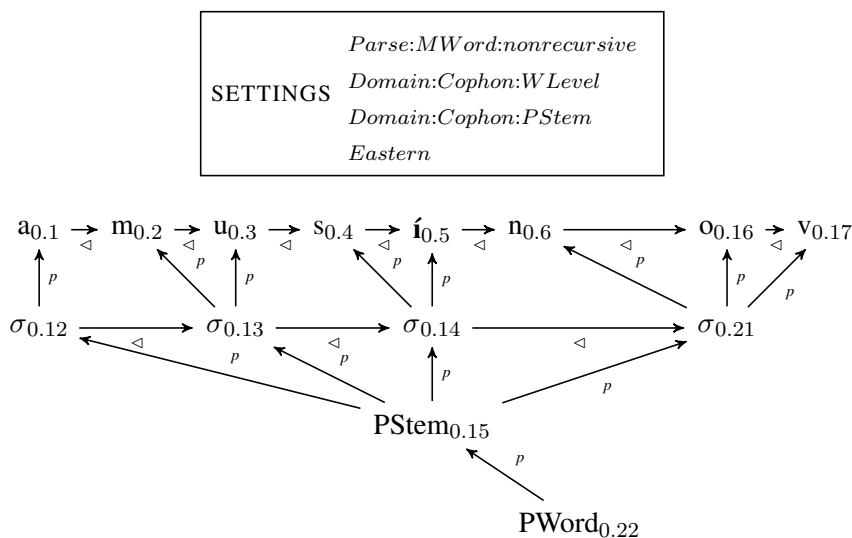
Stress-shift applies because the predicate **StressDomain**(SETTINGS) is satisfied, just as in Western Armenian above. But unlike the previous examples, the predicate **ReductionDomin**(SETTINGS) is also satisfied. The SETTINGS has the label of the PStem cophonology *and* the label for Eastern Armenian. Thus reduction applies. The output is shown below. I do not repeat the relevant output functions which cause vowel reduction. Readers are encouraged to look back at the formula for reduction in §6.6.1 and see how reduction applies.

(552) *Output of cophonology application – V-initial inflection in Eastern Armenian* amusn-ov

$$
\begin{array}{|ll|}
\hline
 & Parse{:}MWord{:}nonrecursive \\
\text{SETTINGS} & Domain{:}Cophon{:}WLevel \\
 & Domain{:}Cophon{:}PStem \\
 & Eastern \\
\hline
\end{array}
$$

$a_{1.1} \to m_{1.2} \to u_{1.3} \to s_{1.4} \longrightarrow n_{1.6} \longrightarrow ó_{1.16} \to v_{1.17}$

$\sigma_{1.12} \longrightarrow \sigma_{1.13} \longrightarrow \sigma_{1.21}$

$\text{PStem}_{1.15}$

$\text{PWord}_{1.22}$

The current formalization does not capture the intuition that the word-level cophonology is post-cyclic (Booij and Rubach 1987). A cophonology is post-cyclic if it applies after *all* morphological operations have applied. Formalizing this intuition is not straightforward. I sketch out a possible formalization in Chapter 9 which would need to make use of a novel mechanism to determine that there are no more morphological operations to apply.

6.7 Conclusion and ubiquity of locality

This section formalized a substantial chunk of the morphology-phonology interface of Armenian, with a focus on general aspects of the interactionist or cyclic model. The take-away is that most morphophonological processes are computationally local.

The formalization used a small set of 6 words as case studies: one simplex from Chapter 5, and 5 complex from the current chapter. The latter 5 case studies were spread across morphological derivation, inflection, and compounding. For **Morphology**, the generation of morphological structure was shown to be computationally local, including overt affixation and compounding. For the prosody and phonological rule domains, all the described processes were computationally local *once* we factorized the SETTINGS of the derivation. In the SETTINGS stage, we used non-local information to find the properties of the morphologically-topmost node and existing prosodic structure. This information was then encapsulated into a constant called the SETTINGS. With this constant, the rest of the interface was local.

In terms of the **Prosody**, once we factor out the SETTINGS, we only need local computation to generate, restructure, fuse, or misalign prosodic nodes. These nodes include prosodic stems, prosodic words, their recursive forms, and their alternations in compounds. Likewise for the **Phonology**, we only need local information to apply morphologically-triggered rules like stress and reduction once we know that we are supposed to trigger those rules (based on examining the SETTINGS).

Part III

Other computational aspects of the morphology-phonology interface

Chapter 7

Computational aspects of affixation

7.1 Constraints on morphological structure: Overview

The previous chapters formalized the derivation of simplex and complex words. Their derivation included a set of *morphological* transductions to generate a covert or overt affix: ∅, *-agan*, *-ov*, *-a-*. In the previous chapters, the added morpheme was a suffix and non-alternating (it had no allomorphs). This chapter examines the computation of morphological processes. In morphological theory, there is a dichotomy between Item-and-Arrangement and Item-and-Process models. In §7.2, I discuss how these models do not generally have significant computational differences (cf. Roark and Sproat 2007:ch3). I then move on to two aspects of morphology: affix order (linearization) and allomorphy. Computationally, I show that affix order tends to be order-preserving, and that allomorphy tends to be computationally local. Below, I state when these computational properties hold. Both order-preservation and computational locality act as formal computational constraints for morphology. The results in this chapter support Chandlee (2017) who finds that various affixation processes are Input-Strictly Local.

For illustration, I discuss these results over relatively simple representations (§7.3). The first set of results concern the linearization of prefixes vs. suffixes. I introduce the formal concept of order-preservation as a constraint on possible input-output correspondences (§7.4). I illustrate with reduplication. I then apply this concept to the affix order (§7.5). I show how suffixation, prefixation, and mobile affixation are order-preserving, while reduplication is not. Briefly, a suffix (prefix) must be defined as the output correspondent of the final (initial) segment in order to make affixation be order-preserving.

I then move on to allomorphy and show how it tends to be computationally local. In phonologically-conditioned allomorphy (§7.6) , the trigger of the allomorph is a phonological property of the input. To make it computationally local, the allomorphy must reference a property which is within a bounded distance from the affix's edge (i.e., the final segment for a suffix). In Armenian, this property can be whether the final segment is a vowel or consonant (§7.6.1) or whether the final segment is part of a monosyllabic word (§7.6.2). Non-locality does occur though in mobile affixation (§7.5.3).

Similarly for morphologically-conditioned allomorphy (§7.7), computationally locality requires that the morphological trigger is within a finite bound from the affix's edge (i.e., the final segment for a suffix). This trigger can either be on the morphological node which dominates the edge (§7.7.1) or is within a finite bound

from the edge (§7.7.2). I illustrate this locality from Armenian declension and conjugation classes. However, not all types of morphologically-conditioned allomorphy are computationally local. Non-locality is derived if the trigger and target are on opposite sides of the input, i.e., prefix-suffix dependencies (§7.7.3.1), or if the allomorphy references the topmost morphological node via feature percolation (§7.7.3.2).

7.2   Dichotomies in morphological theory

Before analyzing affixation, I briefly discuss how computational morphology has dealt with controversies in theoretical morphology. In theoretical morphology, there is debate over how morphological structure should be formalized. Two rough extremes are *item-and-arrangement* (IA) and *item-and-process* (IP) (Hockett 1942). These two approaches have been formalized and contrasted with at least four similar mechanisms (553).

(553)   *Summary of some different morphological formalizations*

	Item-and-Process		Item-and-Arrangement	
System:	Roark and Sproat (2007)	This thesis	Beesley and Karttunen (2003)	Ellison (1993)
Function?	✓	✓	✓	✗
Input	*cat*⋈	*cat*	$\sqrt{\text{CAT}}$ PL	*cat*#Σ^*
Mechanism:	Plural FST	Plural logical transduction	Spell-out FST	FSA intersection
	$\emptyset \rightarrow$-s/_⋈	cf. Armenian *-agan* (§6.2.1.1)	PL$\rightarrow$-s	union with Σ^*#-s
Output	*cat-s*	*cat-s*	*cats*	*cats*

In IP, morphological operations are *functions*. Consider English pluralization. Given an input *cat*, the plural is function which generates an output *cats*. To formalize these functions in this dissertation, I used affix-specific logical transductions. Another mechanism is finite-state transducers and their composition (Roark and Sproat 2007). Given an input ⋊*cat*⋈, an FST for plural formation adds *-s* at the end ⋈ of the input.

In contrast, in IA approaches, the input is not a base or root like *cat*. The input is instead a concatenated sequence of morphological items (morphemes, feature bundles, etc.): $\sqrt{cat}$+PL. These morphological items are spelled out or given phonological form: *cat-s*. An IA model is used in many finite-state applications of morphology, e.g., Beesley and Karttunen (2003)'s XFST system. Like IA, these finite-state systems can model non-concatenative morphology, but with some unwieldiness (Sproat 1992b; Roark and Sproat 2007).

IA models have also been formalized with finite-state intersection. In the general framework of One-Level Declarative Phonology (Scobbie et al. 1996), Ellison (1993) formalizes morphological operations as the intersection of different FSA formalizations.[1] An English root like *cat* is modeled as the regular expression *cat*#Σ^* where # is a special boundary symbol which can precede any number of segments in Σ^*. The English plural is a regular expression Σ^*#*s*. The intersection of these two regular expressions is the word *cats*. Such an approach formalizes the *language* of morphologically-well formed words, but it does not formalize the *transformation* that generates them.

---

[1]Ellison (1993)'s FSA-based formalization for allomorphy is conceptually similar to many theoretical accounts of allomorphy (cf. Hudson 1986; Scheer 2016; Papillon 2020).

In this dissertation, I formalized morphological operations with logical transductions that were more similar to IP models than to IA models. This was not because I argue that IP models are in any way more 'correct' than IA models. In fact, it was because it was *subjectively* easier to define morphology as a set of logical transductions or functions (item-and-process) instead of as some set of logical statements (item-and-arrangement). Outside of subjective utility, the difference between IA vs. IP models is computationally unclear at worst and nonexistent at best. There are different theoretical incarnations for both IA (Lieber 1980; Selkirk 1982; Halle and Marantz 1993) and IP models (Aronoff 1976; Anderson 1992; Stump 2001). The differences between these two models are largely theoretical (Aronoff 1976; Halle and Marantz 1993; Embick 2013; Trommer 2012). There is arguably little to no *computational* difference between them. Roark and Sproat (2007:ch3) detail how one can translate a finite-state system from one model to the other, making both models computationally inter-translatable. In fact, my formalization blurs the line between IA and IP because I treat my representations as individual items (trees), but I generate them via functions. In Chapter 9, I sketch an approach to modeling outwards-sensitive allomorphy which further blurs the line between IA and IP.

7.3 Simpler representation

The previous chapters used an enriched system of representation in order to highlight a word's morphology and prosody. For example, consider the free-standing stem *dun* 'house'. In the more explicit representation in (554c), various labels and relations are used to encode prosodic nodes, morphological nodes, prosodic relations, and morphological relations.

(554)   a. dun   'house'   c. *Explicit representation of a word* dun

b. *Tree structures for* dun
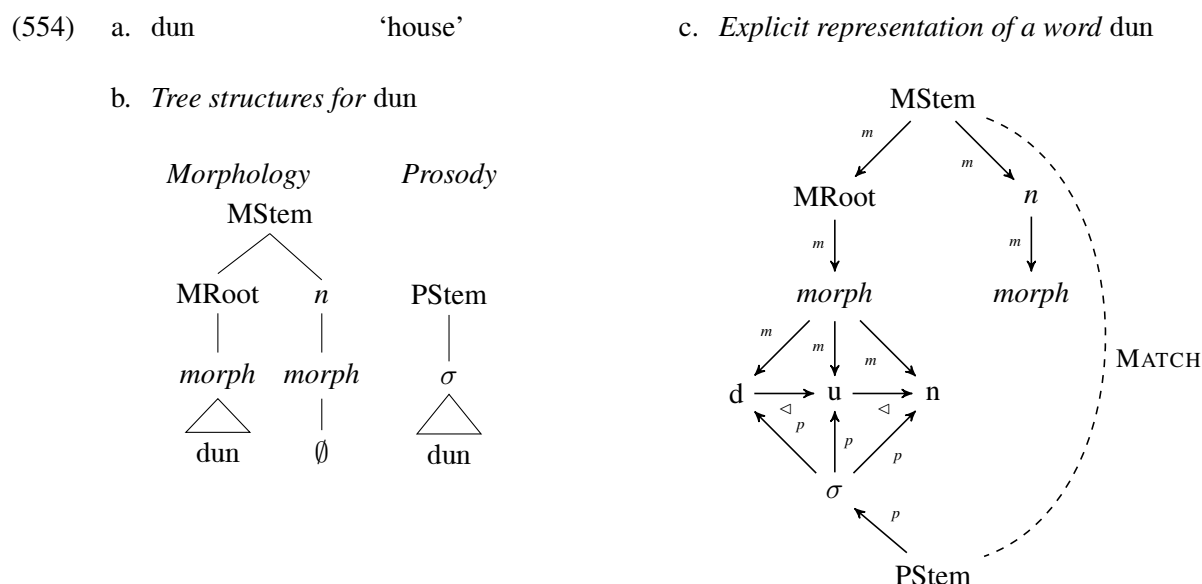
For parts of this chapter, I use a much simpler representation. I show only the segments, without any notation for the morphology or prosody. This is only for *illustrative* purposes. In §7.6.2, some prosodic structure is needed to formalize syllable-counting allomorphy.

(555)   *Explicit representation of a word* dun *in terms of only its segments*

$$d \xrightarrow{\triangleleft} u \xrightarrow{\triangleleft} n$$

Throughout this chapter, I regularly refer to the initial and final segments of the input via the following predicates. Checking that some segment is initial or final is locally-computible and QF-definable.

(556)   a. *User-defined predicates for finding the initial and final segment*
   - $\mathbf{initial{:}seg}(x) \overset{\text{def}}{=} \mathsf{seg}(x) \land \neg \exists y[\mathsf{succ{:}seg}(y, x)]$
   - $\mathbf{final{:}seg}(x) \overset{\text{def}}{=} \mathsf{seg}(x) \land \neg \exists y[\mathsf{succ{:}seg}(x, y)]$

   b. *QF user-defined predicates for finding the initial and final segment*
   - $\mathbf{initial{:}seg}(x) \overset{\text{def}}{=} \mathsf{seg}(x) \land \mathsf{F_R{:}succ{:}seg}(x) = \text{NULL}$
   - $\mathbf{final{:}seg}(x) \overset{\text{def}}{=} \mathsf{seg}(x) \land \mathsf{F_L{:}succ{:}seg}(x) = \text{NULL}$

7.4   Formal concept of order-preservation

I previously discussed how morphological processes have computational properties which are not affected by differences in morphological theories. For example, in previous chapters, I illustrated how morphology has the tendency of being computationally local. In this section, I introduce an additional computational tendency: *order-preservation*. Order-preservation is a constraint on possible input-output correspondences in terms of how output correspondents are ordered across different copies. Order-preserving transductions are easier to compute than non-order-preserving ones; they can be converted to 1-way FSTs. Most but not all types of morphological processes are order-preserving. I first informally introduce this concept with a case study from reduplication (§7.4.1). I formalize it (§7.4.2), and discuss its significance (§7.4.3).

7.4.1   Iconic vs. non-iconic representations in reduplication

Consider a hypothetical partial reduplication process that copies a word-initial CV substring: *pati*∼*papati*. The relationship between the two strings can be represented in (at least) two ways: iconically (557a) vs. non-iconically (557b). These representations 1) encode different segment alignments and 2) assign certain copies to certain segments. The term order-preserving is explained later.

(557)   a. *Iconic and Non-order-preserving*   b. *Non-iconic and Order-preserving*

Input: $p_{0.1} \xrightarrow{<} a_{0.2} \xrightarrow{<} t_{0.3} \xrightarrow{<} i_{0.4}$ Input: $p_{0.1} \xrightarrow{<} a_{0.2} \xrightarrow{<} t_{0.3} \xrightarrow{<} i_{0.4}$

Output: $p_{1.1} \xrightarrow{<} a_{1.2}$ Output: $p_{1.1}$

$\qquad\qquad\qquad \swarrow_{<}$ $\qquad\qquad\quad \downarrow_{<}$

$p_{2.1} \xrightarrow{<} a_{2.2} \longrightarrow\!\!\!\!\times t_{2.3} \longrightarrow\!\!\!\!\times i_{2.4}$ $a_{2.1}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \downarrow_{<}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad p_{3.1} \xrightarrow{<} a_{3.2} \xrightarrow{<} t_{3.3} \xrightarrow{<} i_{3.4}$

The iconic representation uses two copies of the input, while the non-iconic one uses three. For the iconic representation, the two copies of *pa* (the reduplicant and base) each correspond to the underlying segments *pa*. They are spread across the Copy 1 and 2. This correspondence relation is linguistically intuitive and matches the correspondence relations posited in the theoretical literature (McCarthy and Prince 1995). In contrast, in the non-iconic representation, all of the reduplicant's segments correspond to the *first* segment *p*. Copies 1-2 are reserved for the reduplicant while Copy 3 is reserved for the base.

The iconic and non-iconic outputs are generated by two intensionally different processes or transductions. For an input *pati* below, assume a simple phoneme inventory {p,t,a,i}. The predicates in (558a) reference the first and second in the input. Assume that all inputs start with a CVC substring. These predicates are locally-definable

(558)   a. *FO user-defined predicates for reduplication*
   - $\mathbf{first}(x) \overset{\text{def}}{=} \neg\exists w[\mathsf{succ{:}seg}(w, x)]$
   - $\mathbf{second}(x) \overset{\text{def}}{=} \exists w[\mathsf{succ{:}seg}(w, x) \wedge \mathbf{first}(w)]$

   b. *QF user-defined predicates for reduplication*
   - $\mathbf{first}(x) \overset{\text{def}}{=} \mathsf{F_R{:}succ{:}seg}(x) = \text{NULL}$
   - $\mathbf{second}(x) \overset{\text{def}}{=} \mathsf{F_R{:}succ{:}seg}^2(x) = \text{NULL}$

To generate the iconic output, we need a logical transduction that uses a copy set of size 2 and the following output functions (559). In Copy 2, the underlying labels and relations are faithfully outputted for the base (559a). In Copy 1, the reduplicant is created by outputting and ordering the first and second segments (559b). To link the two copies, the second segment of Copy 1 immediately precedes the first segment of Copy 2 (559c). This process is computationally local.

(559)   *QF logical transduction for iconic reduplication*

   a. *QF output functions for unary labels and binary relations on Copy 2 for the base*
   - For every label $\mathsf{lab} \in L$:
     $\phi\mathtt{lab}(x_2) \overset{\text{def}}{=} \mathsf{lab}(x)$
   - For every relation $\mathsf{rel} \in R$:
     $\phi\mathtt{rel}(x_2, y_2) \overset{\text{def}}{=} \mathsf{rel}(x, y)$

   b. *QF output functions for unary labels and binary relations over Copy 1 for the reduplicant*
   - For every label $\mathsf{lab} \in L$:
     $\phi\mathtt{lab}(x_1) \overset{\text{def}}{=} \mathsf{lab}(x) \wedge [\mathbf{first}(x) \vee \mathbf{second}(x)]$
   - $\phi\mathtt{succ{:}seg}(x_1, y_1) \overset{\text{def}}{=} \mathsf{succ{:}seg}(x, y) \wedge \mathbf{first}(x) \wedge \mathbf{second}(y)$

   c. *QF output functions for linking the two copies*
   - $\phi\mathtt{succ{:}seg}(x_1, y_2) \overset{\text{def}}{=} \mathbf{second}(x) \wedge \mathbf{first}(y)$

As for the non-iconic process, we use a transduction with a copy set of size 3, and we use the output functions in (560). In Copy 3, the base is faithfully outputted with all of its labels and relations (560a). In Copy 1, the first segment is outputted faithfully as the reduplicant consonant $p_{1.1}$. Labeling the reduplicant's vowel is superficially more complicated (560b). In Copy 2, the first segment's correspondent $a_{2.1}$ gets the *same* labels as the underlying second segment $a_{0.2}$ in the input. Ordering these reduplicated segments is straightforward (560c). These functions are likewise all computationally local.

(560)   *QF logical transduction for non-iconic reduplication*

    a.   *QF output functions for unary labels on Copy 3 for the base*
- For every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x_3) \stackrel{\text{def}}{=} \mathsf{lab}(x)$
- For every relation $\mathsf{rel} \in R$:
  $\phi\mathtt{rel}(x_3, y_3) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

    b.   *QF output functions for unary labels in Copies 1-2 for the reduplicant*
- For all every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x_1) \stackrel{\text{def}}{=} \mathbf{first}(x) \wedge \mathsf{lab}(x)$
- For all every label $\mathsf{lab} \in L$:
  $\phi\mathtt{lab}(x_2) \stackrel{\text{def}}{=} \mathbf{first}(x) \wedge \mathsf{lab}(\mathsf{F_R}\text{:}\mathsf{succ}\text{:}\mathsf{seg}(x))$

    c.   *QF output functions for binary relations between copies 1,2, and 3*
- $\phi\mathtt{succ\text{:}seg}(x_1, y_2) \stackrel{\text{def}}{=} \mathbf{first}(x) \wedge \mathbf{first}(y)$
- $\phi\mathtt{succ\text{:}seg}(x_2, y_3) \stackrel{\text{def}}{=} \mathbf{first}(x) \wedge \mathbf{first}(y)$

These completes the formalization of the two partial reduplicative processes as two separate transductions. Both formalizations are computationally local. The next section looks at their computational differences.

7.4.2   Criteria for order-preserving functions

The two transductions above generate the same output *pa∼pati* for the input *pati*. They are thus extensionally equivalent in what they output. However, the two transductions are not intensionally equivalent because they generate difference correspondence structures for the output. I repeat the relevant structures below. This correspondence or alignment difference relates to order-preservation (Filiot and Reynier 2016).

(561)   a.   *Iconic and non-order preserving*   b.   *Non-iconic and order-preserving*

Given an input *pati*, a transduction which generates the iconic representation is **not** order-preserving, while a transduction which generates the non-iconic representation **is** order-preserving. Order-preservation is based on two criteria: **Inter-Copy Precedence** and **Intra-Copy Precedence**. Each criterion has a formal and informal definition. Both criteria must be satisfied so that a transduction or function is order-preserving.

The first criterion for order-preservation is **Inter-Copy Precedence**. If a segment x generally precedes y in the input, then all of x's output correspondents must generally precede y's output correspondents.

Visually, a transduction must not create any *right-to-left* arcs. Formally, a transduction is order-preserving only if there are no distinct input segments x, y such that x generally precedes y **but** the output correspondent y_d in some Copy d generally precedes the output correspondent x_c in some Copy d. c, d can be the same Copy.

For the iconic output (557a), x, y, x_c, y_d are $p_{0.1}, a_{0.2}, p_{2.1}, a_{1.2}$. The transduction is not order-preserving because: 1) in the input, $/p_{0.1}/$ precedes $/a_{0.2}/$; 2) but in the output, $[a_{1.2}]$ precedes $[p_{2.1}]$. Thus, we have the ordering '1.**2**' < '2.**1**'. The second slot in the index gets smaller. In contrast, the non-iconic output (557b) satisfies this criterion: there are no pairs of underlying segments x, y such that x generally precedes y while the output correspondent y_c generally precedes x_d. The underlying $/p_{0.1}/$ has three output correspondents: $[p_{1.1}]$, $[a_{2.1}]$, $[p_{3.1}]$. However, these precede each other in a sequence and they precede the output vowel $[a_{3.2}]$: '1.**1**' < '2.**1**' < '3.**1**' < '3.**2**'. The second slot in the indexes does not get smaller.

The second criterion for order-preservation is **Intra-Copy Precedence**. For a given input segment x, its multiple output correspondents are ordered from the lowest Copy to the highest. Visually, there are no vertical upwards-going edges. Formally, let x be an input segment which has two output correspondents x_c and x_d in Copies c, d. If c is less than d, then x_c must generally precede y_d. Both the iconic and non-iconic representations satisfy this criterion. In the iconic representation, the consonant $p_{0.1}$ has two output correspondents $p_{1.1}$ and $p_{2.1}$. It is **not** the case that $p_{2.1}$ generally precedes $p_{1.1}$: '**1**.1' < '**2**.1' Similarly for the non-iconic representation, the underlying consonant $p_{0.1}$ has three output correspondents $p_{1.1}, a_{2.1}, p_{3.1}$; these are consecutively ordered as '**1**.1' < '**2**.1' < '**3**.1'. The first slot in the indexes gets bigger while the second slot stays the same. We don't have the order '**2**.1' < '**1**.1' or '**3**.1' < '**2**.1' < '**1**.1'.

7.4.3   Significance of order-preservation and finite-state technology

To summarize, partial reduplication can be modeled either with an order-preserving or non-order preserving transduction. The question now is why this property of 'order-preserving' matters. The motivation is that any string-to-string MSO transduction can be converted to 1-way finite-state transducers (FST) as long as the transduction is order-preserving (Filiot and Reynier 2016). On a larger scale, the role of order-preservation is about asking what are the desired intensional descriptions of our grammars, i.e., their strong generative capacity.

On the one hand, the iconic representation captures a linguistic insight on the nature of reduplication as actively copying segments; while the non-iconic representation does not. But on the other hand, order-preserving transductions are easier to compute and can be converted to 1-way FSTs. In fact, all partial reduplication processes can be modeled as order-preserving transductions and thus computed by a 1-way FST (Roark and Sproat 2007; Chandlee and Heinz 2012; Chandlee 2017). A non-iconic yet order-preserving transduction is possible because the size of the reduplicant is fixed. By being fixed, we know beforehand the required size of the copy set. In the case of initial-CV copying, we only need a copy set of size 3: Copy 1-2 to output the reduplicant, Copy 3 for the base. To model a larger partial reduplicative process like initial-CVC copying, we would need an additional Copy 4: Copy 1-3 for the reduplicant, Copy 4 for the base.

In contrast, total reduplication cannot be modeled by an order-preserving transduction at all because there is no fixed bound on the size of the reduplicant. Thus, there is no fixed size on the copy set for an order-preserving total reduplication transduction. Because total reduplication can't be computed by a order-preserving MSO transduction, it also cannot be computed by a 1-way FST (Culy 1985). It needs more expressive classes of finite-state calculus, e.g., 2-way FSTs which can go back and forth on the input

(Dolatian and Heinz 2018b, forthcoming).

The logical concept of order-preservation is related to the finite-state concept of origin semantics (Bojańczyk 2014). Consider some string-to-string function $f(x) = y$ and an FST T which computes f, the origin semantics of f is the origin information of each of its output symbols y_j. For an output symbol y_j, its origin information is the input symbol x_i that was used by T to generate y_j. For example, consider a simple function f_{ab} which maps the string ab to itself. This function can be implemented with two different 1-way FSTs. These FSTs differ in when they output the symbols a,b. I show these FSTs below. In the bottom row, I visualize the origin information that these FSTs create for the mapping $ab \mapsto ab$. I use a type of correspondence graph that's called an origin graph (Bojanczyk et al. 2017).

(562)   *FSTs with different origin semantics for $ab \mapsto ab$*

1-way FST	
Origin information	

Given the same input, the two FSTs generate the same output. They are thus equivalent in their extensional description or in their weak generative capacity. But, the two FSTs create different origin information. This makes them differ in their origin semantics. Thus, they differ in their intensional description and in their strong generative capacity (cf. the role of strong generative capacity in syntax: Miller 1999).

Typing back to order-preservation, partial reduplication can be formalized with either order-preserving or non-order-preserving transductions. When converted to finite-state implementations, these transductions differ in the origin information that they create. The order-preserving non-iconic transduction can be implemented with a 1-way FST, while the iconic but non-ordering-preserving transduction cannot. The latter needs the power of 2-way FSTs in order to generate the same origin information (Dolatian and Heinz 2018b, forthcoming), This connection between order-preserving and 1-way FSTs comes up in the logical formalization of affix linearization and allomorphy, which I turn to next.

7.5 Order-preservation in affix order

In previous chapters, all morphological functions generated suffixes. I defined the suffix segments as output correspondents of the final input segment. In this section, I explain why I did this. Briefly, this choice made the morphological transductions be *order-preserving*. The desideratum of order-preservation goes beyond extensional equivalence, but it constrains the set of possible hidden structures. Not all morphological processes are order-preserving, but it is a striking fact that most are or can be.

The previous section showed that reduplication is not order-preserving. In this section, I show that suffixation, prefixation, and mobile affixation are order-preserving. I first discuss two possible formalizations

for affixation based on input-to-affix correspondences (§7.5.1). If a suffix segment is defined as the output correspondent of a final segment, then this formalization is **Edge-Matching** and order-preserving. Otherwise, if a suffix segment is defined as the output correspondent of an initial segment, then the formalization is **Edge-Mismatching** and non-order-preserving. I show a similar dichotomy for prefixation (§7.5.2). Mobile affixation is a special case where a transduction can't be both local and order-preserving (§7.5.3). Generating morphological nodes is also order-preserving (§7.5.4).

7.5.1 Order-preservation in suffix linearization

In the previous chapters, morphological operations such as suffixation were formalized using unique transductions for every existing suffix. For example, the ablative is marked by the suffix -*e*. The suffix displays no productive allomorphy. The representation of the ablative word *dun-e* is shown below.

(563)   a. dun   'house' $d \xrightarrow{\triangleleft} u \xrightarrow{\triangleleft} n$

      b. dun-e   'house-ABL' $d \xrightarrow{\triangleleft} u \xrightarrow{\triangleleft} n \xrightarrow{\triangleleft} e$

Ignoring the underlying morphological and prosodic structure, generating the suffix -*e* is a straightforward transduction with a copy set of size 2. I first show a formalization of suffixation in the **Edge-Matching Formalization**. The input and output are shown below.

(564)   *Input and output of an edge-matching suffix* – dun-e

        Input $d_{0.1} \xrightarrow{\triangleleft} u_{0.2} \xrightarrow{\triangleleft} n_{0.3}$

        Output   Copy 1 $d_{1.1} \xrightarrow{\triangleleft} u_{1.2} \xrightarrow{\triangleleft} n_{1.3}$

                                           $\downarrow \triangleleft$

           Copy 2 $e_{2.3}$
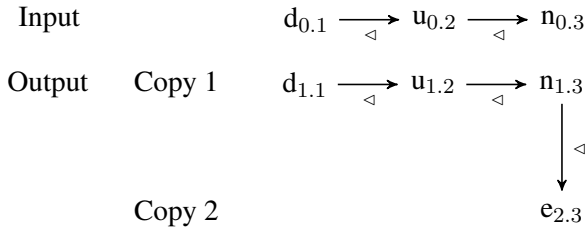
In the edge-matching formalization, Copy 1 is dedicated to outputting the base (565a). In Copy 2, the suffix segment $e_{2.3}$ is generated as an output correspondent of the base-final segment $n_{0.3}$ (565a). The suffix $e_{2.3}$ is externally linearized with the base-final segment $n_{1.3}$ via immediate successor (565c).

(565)   *Generating an edge-matching suffix*
      a. *QF output functions for faithfully outputting the base*
          • For every label lab $\in L$:
              $\phi\texttt{lab}(x_1) \stackrel{\text{def}}{=} \textsf{lab}(x)$
          • For every relation rel $\in R$:
              $\phi\texttt{rel}(x_1, y_1) \stackrel{\text{def}}{=} \textsf{rel}(x, y)$
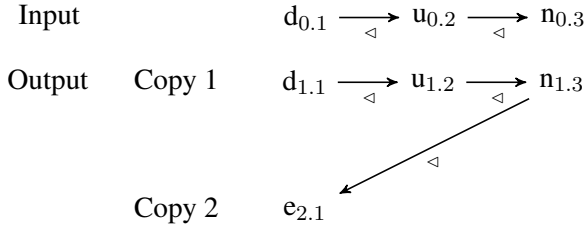   b. *QF output functions for outputting the suffix segment* -e
          • $\phi\texttt{e}(x_2) \stackrel{\text{def}}{=} \textbf{final:seg}(x)$

    c. *QF output function for externally linearizing the suffix with the base*
- $\phi\texttt{succ:seg}(x_1, y_2) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y)$

Next consider the **Edge-Mismatching Formalization** of suffixation. Here, the suffix's segment $e_{2.1}$ is defined as an output correspondent of the *initial* segment $n_{0.3}$. I show the input-output below.

(566)   *Generating an edge-mismatching prefix – the suffix in Copy 2 is based on the 'initial' segment*

The suffix $e_{2.1}$ is generated (567a) and linearized with the base (567b).

(567)   *Generating an edge-mismatching suffix*
    a. *QF output functions for generating the suffix segment*
- $\phi\texttt{e}(x_2) \overset{\text{def}}{=} \textbf{initial:seg}(x)$

    b. *QF output function for externally linearizing the suffix with the base*
- $\phi\texttt{succ:seg}(x_1, y_2) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{initial:seg}(y)$

Both the **Edge-Matching** and **Edge-Mismatching** formalizations work and produce the same output string of segments. Both are likewise computationally local. I repeat their input-output structures below. However, the two formalizations are not intensionally equivalent: they use different hidden structures in terms of input-output correspondence. They differ in order-preservation.

(568)   *Generating a suffix with Edge-Matching vs. Edge-Mismatching*

First off, both formalizations respect **Intra-Copy Precedence**. The suffix in Copy 2 is in a later copy than the base in Copy 1. In the edge-matching formalization, the suffix $e_{2.3}$ succeeds the base-final segment $n_{1.3}$:

'**1**.3' < '**2**.3'. We do not have a sequence like '**2**.3' < '**1**.3'. Visually, there were no vertical upwards-going arcs. The same holds in the edge-mismatching formalization. The input segment $d_{0.1}$ has two output correspondents $d_{1.1}$ and $e_{2.1}$. The base segment $d_{1.1}$ generally precedes the suffix segment $e_{2.1}$: '**1**.1' < 1**2**.' Visually, there were no vertical upwards-going edges like from $e_{2.1}$ to $d_{1.1}$.

However, the two transductions differ with respect to **Inter-Copy Precedence**. The edge-matching formalization satisfies it while the edge-mismatching formalization violates it. Consider first the edge-matching formalization. The base vowel $u_{0.2}$ underlyingly precedes the base-final segment $n_{0.3}$. None of the output correspondents of $u_{0.2}$ ($=u_{1.2}$) succeed the output correspondents of $n_{0.3}$ ($=n_{1.3}, e_{2.3}$). That is, we didn't have a sequence like '**2**.3' < '**1**.2'. Visually, there were no right-to-left curves. In contrast, the edge-mismatching formalization violates Inter-Copy Precedence. In the input, the initial segment $d_{0.1}$ generally precedes the final segment $n_{0.3}$: '**0**.1' < '**0**.3'. It has two output correspondents: $d_{1.1}$ in the base, $e_{2.1}$ in the suffix. $d_{1.1}$ generally precedes the base-final segment, but $n_{1.3}$ precedes $e_{2.1}$: '**1**.1' < '**1**.3' < '**2**.1'. Visually, we have a 'right-to-left' curve from $n_{1.3}$ to $e_{2.1}$. The transduction is *not* order-preserving.

To summarize, the edge-mismatching formalization is not order-preserving. Thus, it cannot be implemented by 1-way finite-state transducer, a commonly used computational implementation in computational phonology. There is thus no visible benefit in treating simple suffixation with a non order-preserving transduction. In fact, order-preservation is found in other affixation processes as well.

7.5.2 Order-preservation in prefix linearization

Like suffixation, prefixation can defined with an order-preserving (edge-matching) or a non-order-preserving (edge-mismatching) transduction. For example, the indicative in Armenian is the prefix /ɡ-/. A later rule of schwa epenthesis applies in word-initial consonant clusters.

(569) a. /ɡ-ude/   [ɡ-ude]   'he eats'
      b. /ɡ-kale/   [ɡə-kale]   'he walks'

Prefixation is a transduction with copy set of size 2. To be order-preserving, the prefix segments are defined in terms of the *initial* segment of the base; otherwise, the prefix is defined in terms of the final segment. I show the input and output for both formalizations below.

(570) *Generating a prefix with Edge-Matching vs. Edge-Mismatching*

	Edge-Matching *and order-preserving*	*Edge-Mismatching* *and not order-preserving*
Input	$u_{0.1} \longrightarrow d_{0.2} \longrightarrow e_{0.3}$	$u_{0.1} \longrightarrow d_{0.2} \longrightarrow e_{0.3}$
Generating Prefix		
Copy 1	$g_{1.1}$ $\downarrow$	$g_{1.3}$
Copy 2	$u_{2.1} \longrightarrow d_{2.2} \longrightarrow e_{2.3}$	$u_{2.1} \longrightarrow d_{2.2} \longrightarrow e_{2.3}$

I focus on the edge-matching transduction first. In Copy 2, the base is faithfully outputted (571a). The prefix $g_{1.1}$ is generated in Copy 1 as an output correspondent of the initial segment $a_{0.1}$ (571b). The prefix and base are linearized (571c).

(571)   *Generating an edge-matching prefix*

    a. *QF output functions for faithfully outputting the base*
- For every label lab $\in L$:
  $\phi\mathtt{lab}(x_3) \stackrel{\text{def}}{=} \mathsf{lab}(x)$
- For every relation rel $\in R$:
  $\phi\mathtt{rel}(x_3, y_3) \stackrel{\text{def}}{=} \mathsf{rel}(x, y)$

    b. *QF output functions for generating the prefix segment*
- $\phi\mathtt{g}(x_1) \stackrel{\text{def}}{=} \mathbf{initial\!:\!seg}(x)$

    c. *QF output functions for externalizing linearizing the prefix with the base*
- $\phi\mathtt{succ\!:\!seg}(x_1, y_2) \stackrel{\text{def}}{=} \mathbf{initial\!:\!seg}(x) \wedge \mathbf{initial\!:\!seg}(y)$

This transduction is order-preserving because it respects Intra-Copy Precedence and Inter-Copy Precedence. The prefix segment $g_{1.1}$ precedes the base-initial segment $u_{2.1}$: '**1**.1' < '**2**.1'. None of the output correspondents of other segments like $d_{0.2}$ (= $d_{2.2}$) precede the output correspondents of $g_{0.1}$ (= $g_{1.1}$, $u_{2.1}$). We don't have a sequence like '2.**2**' < '1.**1**'.

The edge-mismatching version is similarly defined. The difference is that the prefix is generated as an output segment $g_{1.3}$ of the input-final segment $e_{0.3}$ (572a). The base and prefix are then linearized (572b).

(572)   *Generating an edge-mismatching prefix*

    a. *QF output functions for generating the prefix segment*
- $\phi\mathtt{g}(x_1) \stackrel{\text{def}}{=} \mathbf{final\!:\!seg}(x)$

    b. *QF output functions for externalizing linearizing the prefix with the base*
- $\phi\mathtt{succ\!:\!seg}(x_1, y_2) \stackrel{\text{def}}{=} \mathbf{final\!:\!seg}(x) \wedge \mathbf{initial\!:\!seg}(y)$

The edge-mismatching transduction is not order-preserving because it violates Inter-Copy Precedence. The initial input segment $u_{0.1}$ precedes the final segment $e_{0.3}$. But the output correspondent of $u_{0.1}$ is $u_{2.1}$, and it succeeds the output correspondent $g_{1.3}$ of $e_{0.3}$: '1.**3**' < '2.**1**'. There's a right-to-left curve.
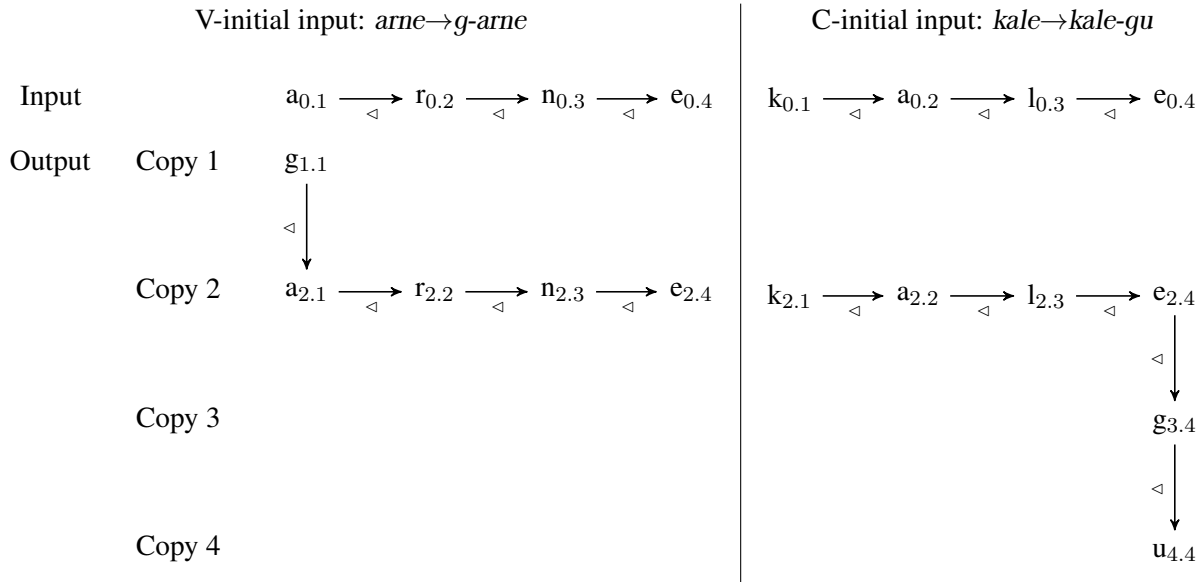
7.5.3   Mobile affixes and conflicts between locality, order-preservation, and representations

In general, affix order is stable: an affix's allomorphs are placed in the same location. However, there are rare cases of mobile affixes (Noyer 1994; Fulmer 1991, 1997; Kim 2010, 2015) which some argue are theoretically impossible (Paster 2006, 2009). For example, in the Hamshen dialect of Armenian, the indicative is the prefix *g-* for V-initial words and the *suffix -gu* for C-initial words.

(573)   a. g-arne   'he takes'
        b. kale-gu   'he walks'

Mobile affixation is computationally more complex than stable affixation. It can be formalized in many different ways, each with its own computational consequence. For example, the edge-matching formalization is order-preserving but not computationally local, while the edge-mismatching formalization is computationally local but not order-preserving. Both transductions use a copy set of size 4. I illustrate the edge-matching formalization first. The prefix *g-* is generated in Copy 1 as the output correspondent of the initial segment, while the suffix *-gu* is generated in Copes 3-4 as output correspondents of the final segment. The input-output are shown below.

(574)   *Input and output for edge-matching mobile affix*

| | | V-initial input: *arne→g-arne* | C-initial input: *kale→kale-gu* |

V-initial input: *arne→g-arne*   C-initial input: *kale→kale-gu*

Input $a_{0.1} \longrightarrow_{\lhd} r_{0.2} \longrightarrow_{\lhd} n_{0.3} \longrightarrow_{\lhd} e_{0.4}$ $k_{0.1} \longrightarrow_{\lhd} a_{0.2} \longrightarrow_{\lhd} l_{0.3} \longrightarrow_{\lhd} e_{0.4}$

Output   Copy 1 $g_{1.1}$

$\lhd \downarrow$

Copy 2 $a_{2.1} \longrightarrow_{\lhd} r_{2.2} \longrightarrow_{\lhd} n_{2.3} \longrightarrow_{\lhd} e_{2.4}$ $k_{2.1} \longrightarrow_{\lhd} a_{2.2} \longrightarrow_{\lhd} l_{2.3} \longrightarrow_{\lhd} e_{2.4}$

$\lhd \downarrow$

Copy 3 $g_{3.4}$

$\lhd \downarrow$

Copy 4 $u_{4.4}$

In Copy 2, the base is faithfully outputted (not formalized). The choice of affix depends on if the first segment is a vowel (575a). The prefix and suffix are respectively generated as output correspondents of the initial and final segment (575b). When generating the suffix *-gu*, an existential quantifier must be used to check if the initial segment is a consonant or not: $\exists z[\mathbf{initial\_C}{:}\mathbf{seg}(z)]$. When linearizing the allomorphs, we check that we have the right C/V-initial contexts for each allomorph. The allomorph *-gu* is internally linearized (575c). External linearization is the crucial step. The prefix *g* is defined as the output correspondent of the initial segment $a_{0.1}$, and it is linearized before the initial segment $a_{2.1}$ (575d). For the suffix *-gu*, it is defined as the output correspondent of the final segment $e_{0.4}$; and it is linearized after the final segment $e_{1.4}$ (575e).

(575)   *Generating an edge-matching mobile affix*
   a. *QF user-defined predicates for checking if the initial segment is a vowel or consonant*
      - $\mathbf{initial\_V}{:}\mathbf{seg}(x) \overset{\text{def}}{=} \mathbf{initial}{:}\mathbf{seg}(x) \land \mathsf{vowel}(x)$
      - $\mathbf{initial\_C}{:}\mathbf{seg}(x) \overset{\text{def}}{=} \mathbf{initial}{:}\mathbf{seg}(x) \land \mathsf{consonant}(x)$
   b. *FO output functions for generating the mobile allomorphs*
      i. *Outputting the prefix* g- *in Copy 1*
         - $\phi\mathsf{g}(x_1) \overset{\text{def}}{=} \mathbf{initial\_V}{:}\mathbf{seg}(x)$

   ii. *Outputting the suffix* -gu *in Copies 3-4*
- $\phi\mathtt{g}(x_3) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \exists z[\textbf{initial\_C:seg}(z)]$
- $\phi\mathtt{u}(x_4) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \exists z[\textbf{initial\_C:seg}(z)]$

c. *FO output functions for internally linearizing the* -gu *allomorph*
- $\phi\mathtt{succ{:}seg}(x_3, y_4) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y) \wedge \exists z[\textbf{initial\_C:seg}(z)]$

d. *QF output functions for externalizing linearizing the prefix allomorph* g- *with the base*
- $\phi\mathtt{succ{:}seg}(x_1, y_2) \stackrel{\text{def}}{=} \textbf{initial\_V:seg}(x) \wedge \textbf{initial\_V:seg}(y)$

e. *FO output functions for externally linearizing the* -gu *allomorph*
- $\phi\mathtt{succ{:}seg}(x_2, y_3) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y) \wedge \exists z[\textbf{initial\_C:seg}(z)]$

In the edge-matching formalization, generating and linearizing the suffix is order-preserving and can be implemented by a 1-way FST. Because the suffix's segments are defined as output correspondents of the final segment $e_{0.4}$, they can follow the base in Copy 1 without violating order-preservation. But in return, the transduction is not local. An existential quantifier is needed to check that the initial segment is a vowel.[2]

Unlike all the previous morphological processes, mobile affixation is our first case of computational non-locality in morphology. It is not my goal to make all morphological processes be computationally local, but to determine 1) what factors cause non-locality, and 2) what alternative analyses can remove non-locality. For the latter point, if we want to make mobile affixation be local, then we must either use a non-order-preserving formalization or use an alternative representation. I discuss both.

An alternative formalization is an edge-*mismatching* formalization. Here, both the prefix and suffix are defined as output correspondents of the initial segment. I illustrate below.

(576) *Input and output for generating an edge-mismatching mobile affix*
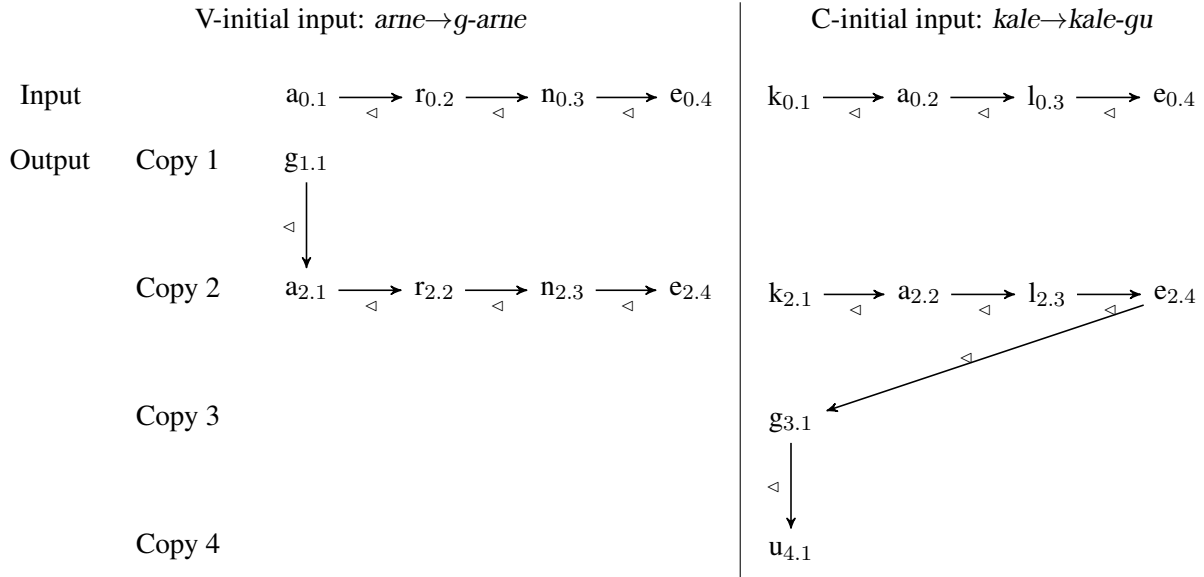
Both the prefix and suffix are generated as output correspondents of the initial segment (577a). External and internal linearization keeps track of these different allomorphy contexts. For the suffix *-gu*, it is defined as the output correspondent of the initial segment $k_{0.1}$; and it is linearized after the final segment $e_{1.4}$ (577c).

(577)   *Generating an edge-mismatching mobile affix*

      a. *QF output functions for generating the affix segments*

         i. *Outputting the prefix* g- *in Copy 1*

            • $\phi g(x_1) \stackrel{\text{def}}{=} \textbf{initial\_V:seg}(x)$

         ii. *Outputting the suffix* -gu *in Copies 3-4*

            • $\phi g(x_3) \stackrel{\text{def}}{=} \textbf{initial\_C:seg}(x)$

            • $\phi u(x_4) \stackrel{\text{def}}{=} \textbf{initial\_C:seg}(x)$

      b. *QF output functions for internally linearizing the suffix allomorph*

         • $\phi \texttt{succ:seg}(x_3, y_4) \stackrel{\text{def}}{=} \textbf{initial\_C:seg}(x) \wedge \textbf{initial\_C:seg}(y)$

      c. *QF output functions for externalizing linearizing the suffix allomorph* -gu *with the base*

         • $\phi \texttt{succ:seg}(x_2, y_3) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{initial\_C:seg}(y)$

This transduction is local because no quantifiers were used to find the initial or final segment. However, the transduction is not order-preserving when it generates the suffix *-gu*. The underlying initial segment $k_{0.1}$ has two output correspondents: $k_{1.1}, g_{3.1}$. $k_{2.1}$ precedes the base-final vowel $e_{2.4}$, but $e_{2.4}$ precedes $g_{3.1}$: '2.**1**' < '2.**4**' < '3.**1**'. The second slot in the index gets smaller. This creates a right-to-left edge from $e_{2.4}$ to $g_{3.1}$. With this exact intensional description, this transduction cannot be modeled by a 1-way FST. While there are 1-way FSTs which are extensionally equivalent to this transduction, none of them have the same origin semantics because the origins semantics of 1-way FSTs is always order-preserving

The above discussion shows that, with our simple input representations, we can't treat mobile affixation as both computationally local and computationally order-preserving. Another alternative formalization is to change our input representation. Hypothetically, we can define the prefix and suffix as output correspondents of the initial and final segments respectively. To capture the role of the input-initial segment in determining the allomorphy, we could use a constant INITIAL that refers or points back to the input-initial segment. With this constant, we can compute the allomorphy without using any quantifiers, and by still being order-preserving.

(578)   *Generating mobile allomorphs with a constant* INITIAL

      a. *Outputting the prefix* g- *in Copy 1*

         • $\phi g(x_1) \stackrel{\text{def}}{=} \textbf{initial:seg}(x) \wedge \textbf{initial\_V:seg}(\text{INITIAL})$

      b. *Outputting the suffix* -gu *in Copies 3-4*

         • $\phi g(x_3) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{initial\_C:seg}(\text{INITIAL})$

         • $\phi u(x_4) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{initial\_C:seg}(\text{INITIAL})$

All three of the above formalizations work and are extensionally equivalent. However, they each sacrifice one computational desideratum, whether its locality, order-preservation, or simple representations. This makes them intensionally different, with different strong generative capacity. For this dissertation, I will use the first formalization, and I will treat mobile affixation as order-preserving but not computationally local.

### 7.5.4   Order-preservation and morphological dominance

Up until now, order-preservation was discussed in terms of the binary relation of *immediate successor* among segments. A transduction is order-preserving as long as suffix (prefix) segments are defined in terms of the final (initial) segment. The result and condition also extend to the binary relation of *morphological dominance* among morphological nodes. When formalizing all the morphological processes in this thesis, morphological nodes were generated as output correspondents of the morphologically topmost node. Because of this, generating and linearizing new morphological nodes is *order-preserving* in terms of morphological dominance. To illustrate, consider simple suffixation of *-e* to form the word *dun-e* 'house-ABL'. The input and output are shown below with the full morphological structure.

(579)   *Input and output for generating a suffix* -e *including the morphology*
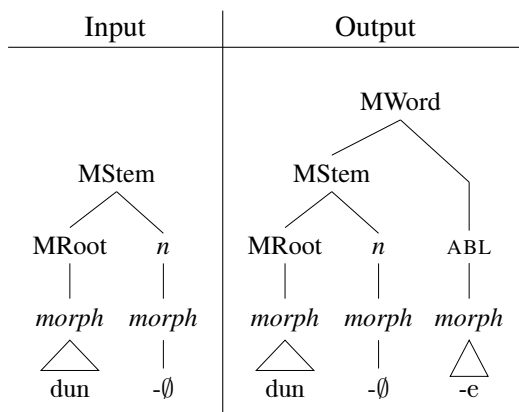
The above structure is made explicit below. I omit any prosodic nodes.

(580)   *Explicit input and output for generating a suffix* -e
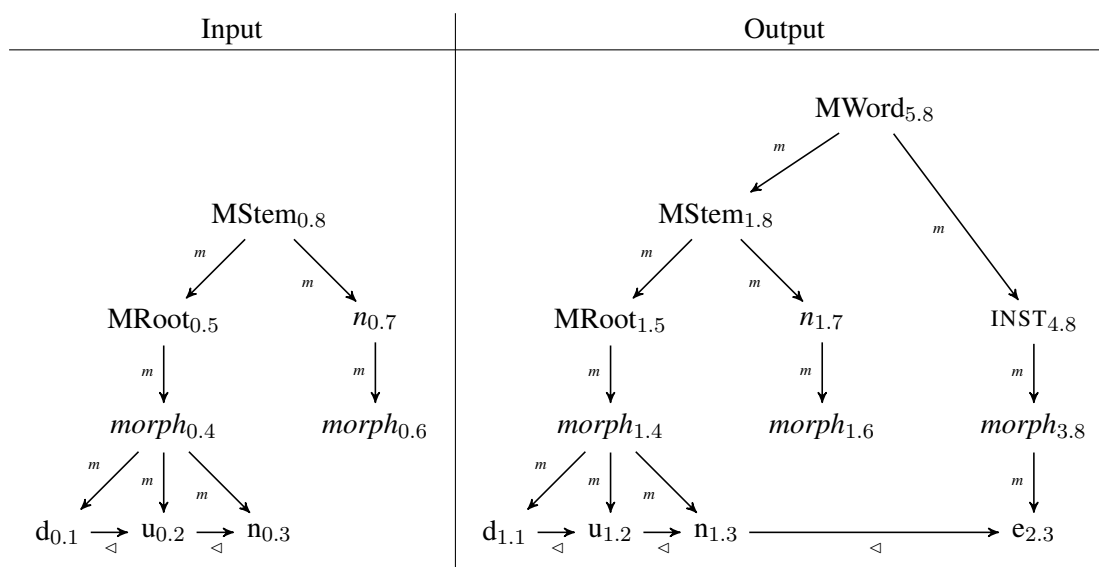
The output is generated by a morphological transduction with a copy set of size 5. In Copy 1, the base is faithfully outputted. In Copy 2, the suffix segment *e* is generated as an output correspondent of the final segment *n* and linearized. I omit the functions for Copy 1 and Copy 2. Crucially in Copies 3-4, the new morphological nodes are generated as output correspondents of the input's morphologically topmost node, the MRoot (581a-ii). The new nodes are internally linearized via morphological dominance (581b-i). For external linearization, the affix's MStem dominates the base's MRoot (581b-ii).

(581) *Predicates and output functions for generating an inflectional suffix* -e 'ABL'

    a. **Outputting labels**

        i. Relevant user-defined predicates:

            • $\mathbf{MTopmost}(x) \overset{\text{def}}{=} \mathsf{MNode}(x) \wedge \neg\exists y[\mathsf{MDom}(y, x)]$

        ii. Output the new morphological nodes

            • $\phi\texttt{morph}(x_3) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$

            • $\phi\texttt{abl}(x_4) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$

            • $\phi\texttt{MWord}(x_5) \overset{\text{def}}{=} \mathbf{MTopmost}(x)$

    b. **Linearization – Outputting relations**

        i. Internal linearization of the suffix

            • $\phi\texttt{MDom}(x_3, y_2) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{final{:}seg}(y)$

            • $\phi\texttt{MDom}(x_4, y_3) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

            • $\phi\texttt{MDom}(x_5, y_4) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

        ii. External linearization of morphology

            • $\phi\texttt{MDom}(x_6, y_1) \overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$

Readers can verify for themselves that generating the suffix segment $e_{2.3}$ is order-preserving. I focus on the morphological nodes. The suffix's MNodes are defined as output correspondents of the input's morphologically topmost node *MStem*$_{0.8}$ (581a-ii). The suffix's MWord$_{5.8}$ then dominates the base's MStem$_{1.8}$ (581b-ii). The transduction is order-preserving because the input's topmost node is dominated by its own output correspondent: '1.**8**' < '5.**8**'. It is not the case that the output MStem$_{1.8}$ is now dominated by the output correspondent of a lower item like the MRoot$_{0.5}$: *'1.**8**' < '5.**5**'.

To summarize, order-preservation requires that we carefully define a new affix's segments and morphological nodes. The morphological nodes must always be defined in terms of morphologically topmost node.

7.6   Locality of phonologically-conditioned allomorphy

In the previous sections, the morphological processes were *non-alternating* and involved a *single* allomorph. In contrast, allomorphy is when the affix has different shapes in different contexts. There are roughly two types of allomorphy: phonologically-conditioned allomorphy and morphologically-conditioned allomorphy. I discuss the first type of allomorphy in this section, and the second type in the next.

Over the next two sections, I show that affix allomorphy is not necessarily more complex than non-alternating affixation. In fact, the typology of phonologically-conditioned allomorphy has a strong tendency to be

local (Paster 2005, 2006, 2009; Nevins 2011). I formalize a simple case of phonologically-conditioned allomorphy and show that it is computationally local (§7.6.1). I present a special case of syllable-counting allomorphy, and show that it is also local (§7.6.2). Crucially however, not all cases of phonologically-conditioned allomorphy are local. As previously shown, mobile affixation is not computationally local (§7.5.3).

7.6.1 Allomorphy based on the final segment

An example of phonologically-conditioned allomorphy is found in Armenian. The definite suffix is *-n* after V-final bases, *-ə* after C-final bases.

(582) a. agra   'tooth'   (583) a. park   'glory'
           agra-n   'tooth-DEF'   park-ə   'glory-DEF'

Generating the definite suffix is a simple transduction with a copy set of size 2. I show the input-output below for both a V-final and C-final base.

(584)   *Generating suffix allomorphy in the definite*

V-final input: *agra→agra-n*   C-final input: *dun→dun-ə*
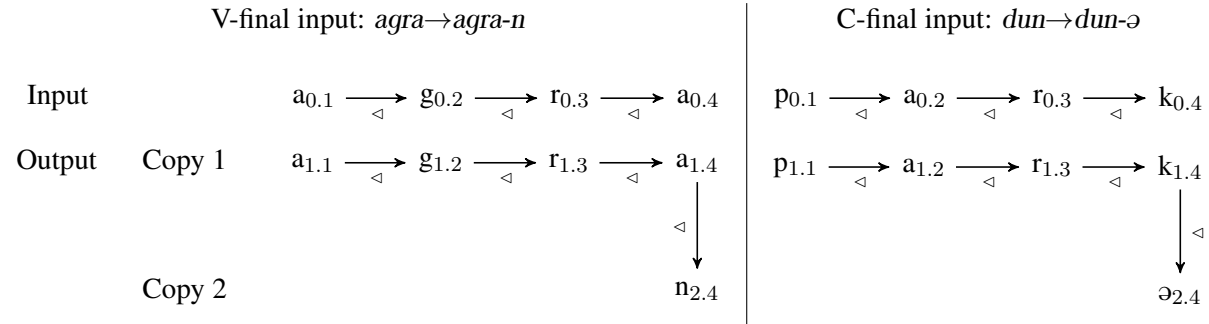
In Copy 1, the input is faithfully outputted as the base. The choice of allomorph depends on if the final segment is a vowel or consonant (585a). In Copy 2, the suffix is generated as $n_{2.4}$ if it is an output correspondent of a final vowel $a_{0.4}$, and as $-ə_{2.4}$ if it is an output correspondent of a final consonant $k_{0.4}$ (585b). The suffix's segment is externally linearized with the base via immediate successor. It is doesn't matter which allomorph is picked.

(585)   *Generating phonologically-conditioned allomorphs for the definite suffix*
         a. *QF user-defined predicate for checking if the final segment is a vowel or consonant*
             • **final_V:seg**$(x) \stackrel{\text{def}}{=}$ **seg**$(x) \wedge$ **final:seg**$(x) \wedge$ vowel(x)
             • **final_C:seg**$(x) \stackrel{\text{def}}{=}$ **seg**$(x) \wedge$ **final:seg**$(x) \wedge$ consonant(x)
         b. *QF output functions for generating the segments of the right suffix allomorph*
             • $\phi n(x_2) \stackrel{\text{def}}{=}$ **final_V:seg**(x)
             • $\phi ə(x_2) \stackrel{\text{def}}{=}$ **final_C:seg**(x)
         c. *QF output function for externally linearizing the suffix allomorphs with the base's final segment*
             • $\phi \texttt{succ:seg}(x_1, y_2) \stackrel{\text{def}}{=}$ **final:seg**$(x) \wedge$ **final:seg**(y)

Despite showing allomorphy, generating the definite suffix is still computationally local because no existential quantifiers are needed to access any long-distant information. I now turn to a more complex example.

7.6.2 Allomorphy based on syllable-counting

As shown above, allomorphy is not itself more complex than non-alternating affixation. What is important is *how* we formalize the allomorph's segments as output correspondents of the input. As another case study, I formalize the Armenian plural suffix. The plural displays allomorphy conditioned by syllable count: *-er* after monosyllabic bases, and *-ner* after polysyllabic bases.

(586)   a. dar   'character'   b. andar   'forest'
           dar-er   'character-PL'   andar-ner   'forest-PL'

In compounds, the plural is involved in a bracketing paradox. If the second stem is monosyllabic, exocentric compounds are transparently pluralized as polysyllabic bases with *-ner* (587a), but endocentric compounds are paradoxically pluralized as <u>monosyllabic</u> bases with *-er* (587b).
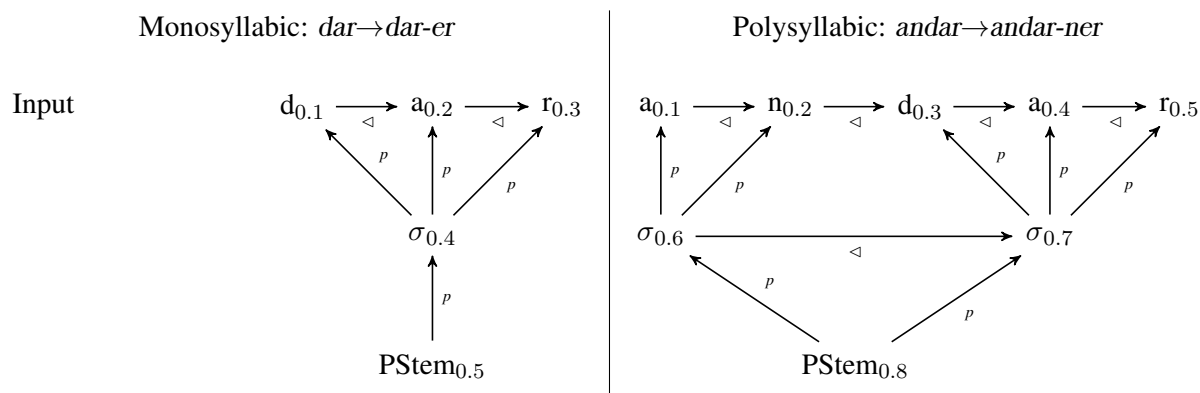
(587)   a. t͡ʃúr   'water'   b. t͡ʃúr   'water'
           pós   'hole'   kújn   'color'
           t͡ʃər-a-pós   'water-hole'   t͡ʃər-a-kújn   'water-colored'
           t͡ʃər-a-<u>pos</u>-ér 'water-holes'   *t͡ʃər-a-kújn-ér   'water-colored (objects)'
           *t͡ʃər-a-<u>pos</u>-nér   t͡ʃər-a-kujn-nér

The plural paradox is triggered by compound prosodic structure. As formalized in Chapter 6: 6.5.3, endocentric compounds are parsed into two separate PStems, while exocentric compounds are a single PStem. In contrast, simple roots form a single PStem. The plural counts the number of syllables in the rightmost PStem.
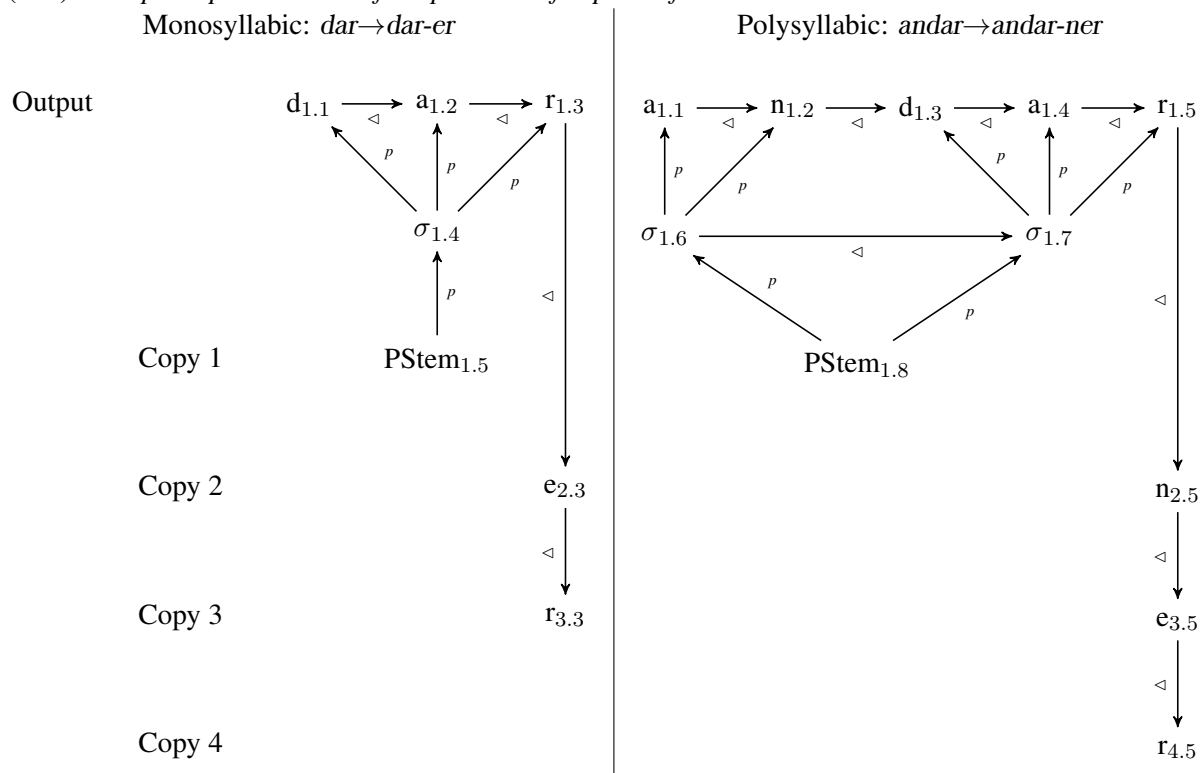
(588)   *Different internal PStem structures for compounds and roots*

Compound		Root	
Endocentric	*Exocentric*	*Monosyllabic*	*Polysyllabic*

Generating the plural allomorphy thus requires more enriched prosodic structure than a simple string of segments. To that end, I explicitly show prosodic nodes in this section, but not morphological nodes. In the case of the singular simple roots, their representation is below.

(589)   *Input representation of simplex roots for plural formation*

Monosyllabic: *dar→dar-er*   Polysyllabic: *andar→andar-ner*

Input

$d_{0.1} \rightarrow a_{0.2} \rightarrow r_{0.3}$ $\quad$ $a_{0.1} \rightarrow n_{0.2} \rightarrow d_{0.3} \rightarrow a_{0.4} \rightarrow r_{0.5}$

$\sigma_{0.4}$ $\qquad\qquad$ $\sigma_{0.6} \longrightarrow \sigma_{0.7}$

$PStem_{0.5}$ $\qquad\qquad$ $PStem_{0.8}$

I show the final output below. For the monosyllabic input *dar*, the suffix segments $e_{2.3}$, $r_{3.3}$ are defined as output correspondents of the final segment $r_{0.3}$. Similarly for the polysyllabic input *andar*, the final segment is $r_{0.5}$. The suffix segments are $n_{2.5}$, $e_{3.5}$, and $r_{4.5}$.

(590)   *Output representation of simplex roots for plural formation*

Monosyllabic: *dar→dar-er*   Polysyllabic: *andar→andar-ner*

Output

$d_{1.1} \rightarrow a_{1.2} \rightarrow r_{1.3}$ $\quad$ $a_{1.1} \rightarrow n_{1.2} \rightarrow d_{1.3} \rightarrow a_{1.4} \rightarrow r_{1.5}$

$\sigma_{1.4}$ $\qquad\qquad$ $\sigma_{1.6} \longrightarrow \sigma_{1.7}$

Copy 1 $\qquad$ $PStem_{1.5}$ $\qquad\qquad$ $PStem_{1.8}$

Copy 2 $\qquad\qquad$ $e_{2.3}$ $\qquad\qquad\qquad$ $n_{2.5}$

Copy 3 $\qquad\qquad$ $r_{3.3}$ $\qquad\qquad\qquad$ $e_{3.5}$

Copy 4 $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $r_{4.5}$

Generating the plural for simplex roots is a transduction with a copy set of size 4. In Copy 1, the input is faithfully outputted as the base. I don't show the relevant functions. The suffix allomorphs are chosen based on the number of syllables in the input. Suffixes are defined in terms of the final segment of the input in order to ensure order-preservation. We thus need to check for the monosyllabicity requirement in terms of the final segment. Specifically, we know that we should use the monosyllabic-selecting suffix *-er* if:

1. The final segment x is part of a syllable y
2. The syllable y is part of a PStem z
3. There is no syllable w which precedes the syllable y **and** is in the same PStem z

The user-defined predicate **allomorph_er**(x) checks that the final segment x fits this context for choosing the *-er* allomorph. It is locally-computed and QF-definable (591b).

(591) a. *FO User-defined predicates for checking if the input ends in a monosyllabic PStem*
- **allomorph_er**$(x) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land \exists y[\text{syll}(y) \land$ **PDom:syll_seg**$(y, x) \land$
$\exists z[\text{PStem}(z) \land \text{PDom:PStem\_syll}(z, y) \land$
$\neg \exists w[\text{syll}(w) \land \text{succ:syll}(w, y) \land \text{PDom:PStem\_syll}(z, w)]]]$

b. *QF user-defined predicates for checking if the input ends in a monosyllabic PStem*
- **allomorph_er**$(x) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$
$F_D\text{:PDom:PStem\_syll}(F_D\text{:PDom:syll\_seg}(x)) \neq \text{NULL} \land$
$F_D\text{:PDom:PStem\_syll}(F_R\text{:succ:syll}(F_D\text{:PDom:syll\_seg}(x))) \neq$
$F_D\text{:PDom:PStem\_syll}(F_D\text{:PDom:syll\_seg}(x))$

If the predicate **allomorph_er**(x) is true, then we generate the suffix *-er* in Copies 2-3 (592a); otherwise, we generate the suffix *-ner* in Copies 2-4 (592b).

(592) *QF output functions for generating the plural allomorphs*
a. *Generating the allomorph* -er
- $\phi\text{e}(x_2) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$ **allomorph_er**(x)
- $\phi\text{r}(x_3) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$ **allomorph_er**(x)

b. *Generating the allomorph* -ner
- $\phi\text{n}(x_2) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land \neg$**allomorph_er**(x)
- $\phi\text{e}(x_3) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land \neg$**allomorph_er**(x)
- $\phi\text{r}(x_4) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land \neg$**allomorph_er**(x)

The allomorphs are internally (593a) and externally linearized (593b) via the functions below. Note that we do not need to specify which allomorph is being linearized between Copies 1,2,3. Thus, we do not check for the allomorphy condition for them. But we do need to check the allomorphy condition in order to linearize Copy 3 with Copy 4: if *-er* is used, then there is nothing in Copy 4 to linearize.
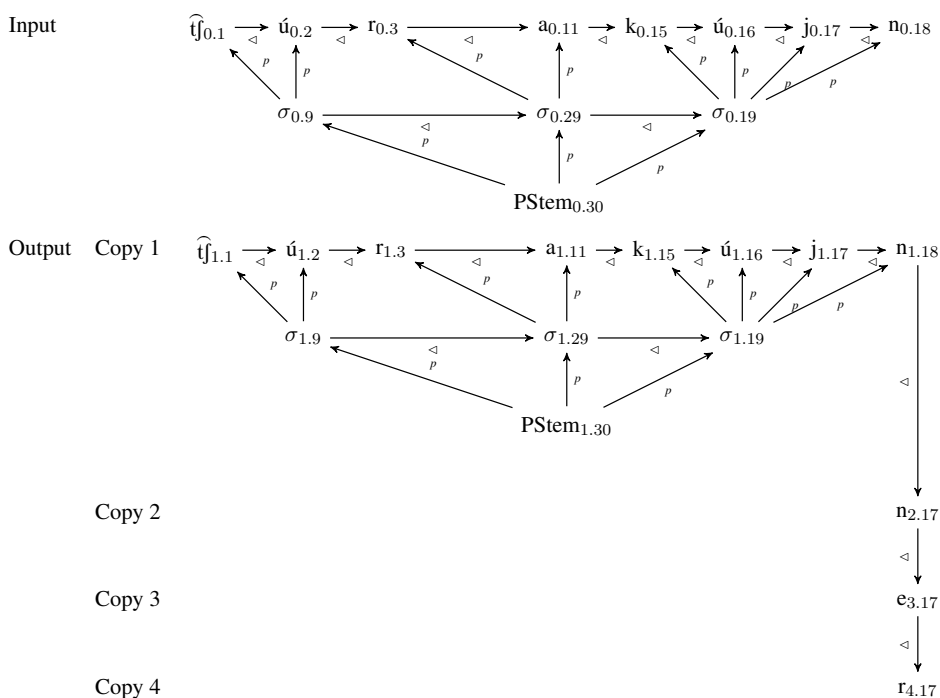
(593) a. *QF output functions for internally linearizing the allomorphs*
- $\phi\text{succ:seg}(x_2, y_3) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$ **final:seg**(y)
- $\phi\text{succ:seg}(x_3, y_4) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$ **final:seg**$(y) \land \neg$**allomorph_er**(x)

b. *QF output function for externally linearizing the plural with the base*
- $\phi\text{succ:seg}(x_1, y_2) \stackrel{\text{def}}{=}$ **final:seg**$(x) \land$ **final:seg**(y)

These functions will generate the right plural allomorph for compounds. Readers can work out the derivation for themselves to verify. I show the input and output for endocentric and exocentric compounds below. The indexing keeps track of the morphological nodes from Chapter 6: §6.3.2.3.

(594)   a.   *Input and output to pluralizing an endocentric compound* : $\widehat{tʃ}$ər-a-pós→$\widehat{tʃ}$ər-a-<u>pós</u>-er 'water-holes'

Input $\widehat{tʃ}_{0.1} \to$ ə$_{0.2} \to$ r$_{0.3} \longrightarrow$ a$_{0.11} \to$ p$_{0.15} \to$ ó$_{0.16} \to$ s$_{0.17}$

$\sigma_{0.9} \longrightarrow \sigma_{0.28} \longrightarrow \sigma_{0.18}$

PStem$_{0.10} \longrightarrow$ PStem$_{0.19}$

Output   Copy 1 $\widehat{tʃ}_{1.1} \to$ ə$_{1.2} \to$ r$_{1.3} \longrightarrow$ a$_{1.11} \to$ p$_{1.15} \to$ ó$_{1.16} \to$ s$_{1.17}$

$\sigma_{1.9} \longrightarrow \sigma_{1.28} \longrightarrow \sigma_{1.18}$

PStem$_{1.10} \longrightarrow$ PStem$_{1.19}$

Copy 2   e$_{2.17}$

Copy 3   r$_{3.17}$

Copy 4
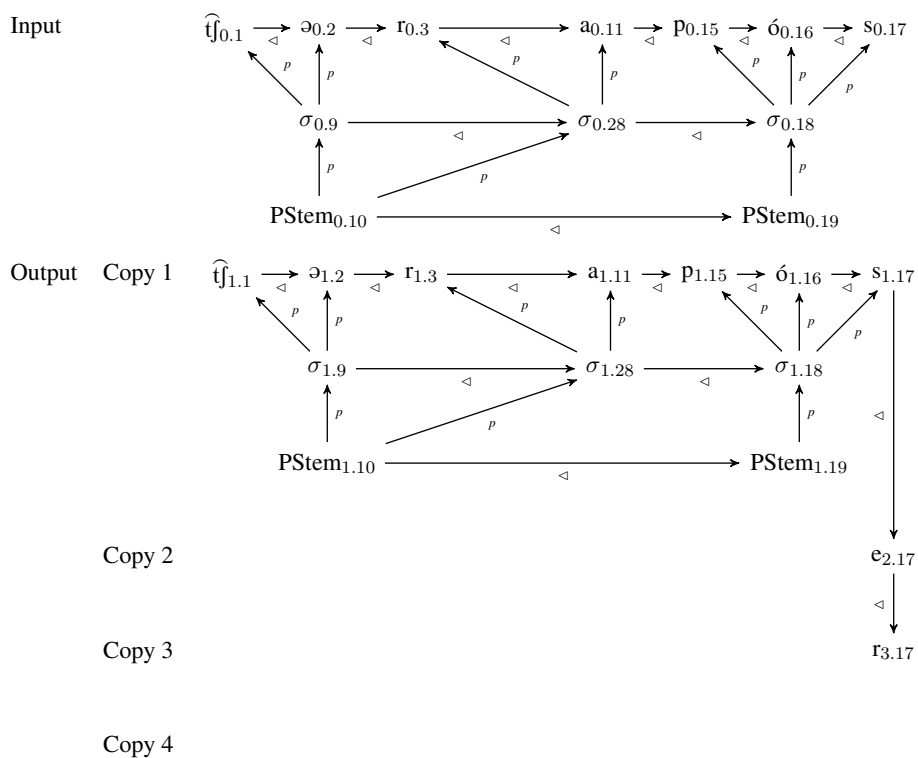
b.   *Input and output to pluralizing an exocentric compound* : $\widehat{tʃ}$ər-a-kújn→$\underline{\widehat{tʃ}ər\text{-}a\text{-}kújn}$-ner 'water-colored (objects)'

Input $\widehat{tʃ}_{0.1} \to$ ú$_{0.2} \to$ r$_{0.3} \longrightarrow$ a$_{0.11} \to$ k$_{0.15} \to$ ú$_{0.16} \to$ j$_{0.17} \to$ n$_{0.18}$

$\sigma_{0.9} \longrightarrow \sigma_{0.29} \longrightarrow \sigma_{0.19}$

PStem$_{0.30}$

Output   Copy 1 $\widehat{tʃ}_{1.1} \to$ ú$_{1.2} \to$ r$_{1.3} \longrightarrow$ a$_{1.11} \to$ k$_{1.15} \to$ ú$_{1.16} \to$ j$_{1.17} \to$ n$_{1.18}$

$\sigma_{1.9} \longrightarrow \sigma_{1.29} \longrightarrow \sigma_{1.19}$

PStem$_{1.30}$

Copy 2   n$_{2.17}$

Copy 3   e$_{3.17}$

Copy 4   r$_{4.17}$

7.7   Locality of morphologically-conditioned allomorphy

The previous allomorphy processes were all *phonologically-conditioned*. Allomorphy can likewise be *morphologically-conditioned*, meaning that the choice depends on non-phonological, morphological, and arbitrary properties of the input. As with phonologically-conditioned allomorphy, morphologically-conditioned allomorphy tends to be a local process (Embick 2010, 2013; Bobaljik 2012; Gribanova 2010, 2015; Merchant 2015; Moskal 2015b,a; Gribanova and Shih 2017; Gribanova and Harizanov 2017).

In this section, I illustrate how such allomorphy is computationally local in Armenian inflection, specifically declension classes and case-marking (§7.7.1). Although computational locality can involve strict adjacency between triggers and targets, I show a process can be local even if it violates strict adjacency, e.g., in Armenian conjugation classes and theme vowels (§7.7.2). I show what factors can create non-local allomorphy (§7.7.3), whether from prefix-suffix dependencies (7.7.3.1) or from feature percolation (§7.7.3.2).

7.7.1   Locality in Armenian inflection

As an example of morphologically-conditioned allomorphy, consider the Western Armenian dative. It is is the suffix *-i* for regular nouns (595a) but the suffix *-u* for plural nouns after the plural suffix (595b), for some arbitrary irregular nouns (595c), and for nominalized infinitivals (595d). There are also a few other irregular contexts for *-u*, and a few other unproductive irregular allomorphs of the dative.

(595)   a. kam   'nail'   c. ʒam   'time'
            kam-i   'nail-DAT   ʒam-u   'time-DAT

        b. kam-er   'nail-PL'   d. kerel   'to scratch'
            kam-er-u   'nail-PL-DAT   kerel-u   'scratching-DAT'

In the table below, I show the morphological structure of these different Armenian words. I underline the **trigger morpheme**: the node which contains the relevant morphological feature which triggers the allomorphy. It is either a root (MRoot$_{\text{IRREG}}$) or an affix (PL, INF).
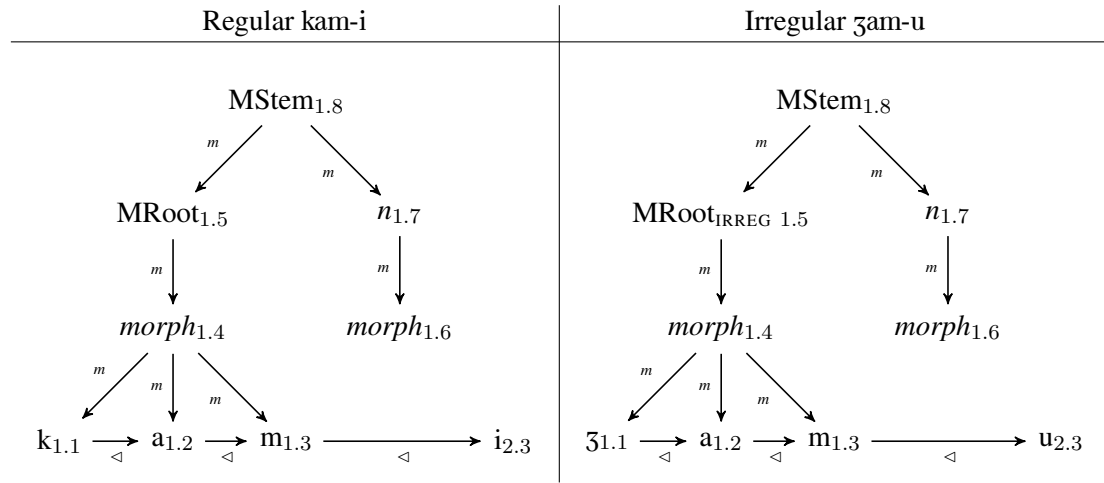
In itself, morphologically-conditioned allomorphy is not more complex than phonologically-conditioned allomorphy. What matters is where the trigger morpheme lies in the input. If the allomorphs are all suffixes (prefixes), then the formalization is computationally local if the trigger morpheme is a bounded distance away from the final (initial) segment. In the cases above, the allomorphy is computationally local because the trigger morpheme dominates a morph which dominates the final segment: *kam-e***r**, *ʒa***m**, *ker-e-***l**.

(596) *Input and output for the Armenian dative*

	Regular suffix	Irregular suffix		
	Regular root	Plural word	Irregular root	Infinitival
Base	MStem → MRoot, *n*; MRoot → *morph* → kam; *n* → *morph* → ∅	MWord → MStem, PL; MStem → MRoot, *n*; MRoot → *morph* → kam; *n* → *morph* → ∅; PL → *morph* → -er	MStem → MRoot$_{IRREG}$, *n*; MRoot$_{IRREG}$ → *morph* → ʒam; *n* → *morph* → ∅	MStem → MStem, INF; MStem → MRoot, *v*; MRoot → *morph* → ker; *v* → *morph* → -e; INF → *morph* → -l
Dative	MWord → MStem, DAT; MStem → MRoot, *n*; MRoot → *morph* → kam; *n* → *morph* → ∅; DAT → *morph* → -i	MWord → MWord, DAT; MWord → MStem, PL; MStem → MRoot, *n*; MRoot → *morph* → kam; *n* → *morph* → ∅; PL → *morph* → -er; DAT → *morph* → -u	MWord → MStem, DAT; MStem → MRoot$_{IRREG}$, *n*; MRoot$_{IRREG}$ → *morph* → ʒam; *n* → *morph* → ∅; DAT → *morph* → -u	MWord → MStem, DAT; MStem → MStem, INF; MStem → MRoot, *v*; MRoot → *morph* → ker; *v* → *morph* → -e; INF → *morph* → -l; DAT → *morph* → -u

To illustrate, consider the regular and irregular roots *kam, ʒam* below. I show the input without prosodic nodes. I show a partial output which doesn't show the suffix's MNodes.

(597)   a.   *Input to regular and irregular dative formation* – /kam, ʒam/

| Regular /kam/ | Irregular /ʒam/ |

MStem$_{0.8}$ → MRoot$_{0.5}$, $n_{0.7}$; MRoot$_{0.5}$ → *morph*$_{0.4}$; $n_{0.7}$ → *morph*$_{0.6}$; *morph*$_{0.4}$ → k$_{0.1}$ ◁ a$_{0.2}$ ◁ m$_{0.3}$

MStem$_{0.8}$ → MRoot$_{IRREG\ 0.5}$, $n_{0.7}$; MRoot$_{IRREG\ 0.5}$ → *morph*$_{0.4}$; $n_{0.7}$ → *morph*$_{0.6}$; *morph*$_{0.4}$ → ʒ$_{0.1}$ ◁ a$_{0.2}$ ◁ m$_{0.3}$
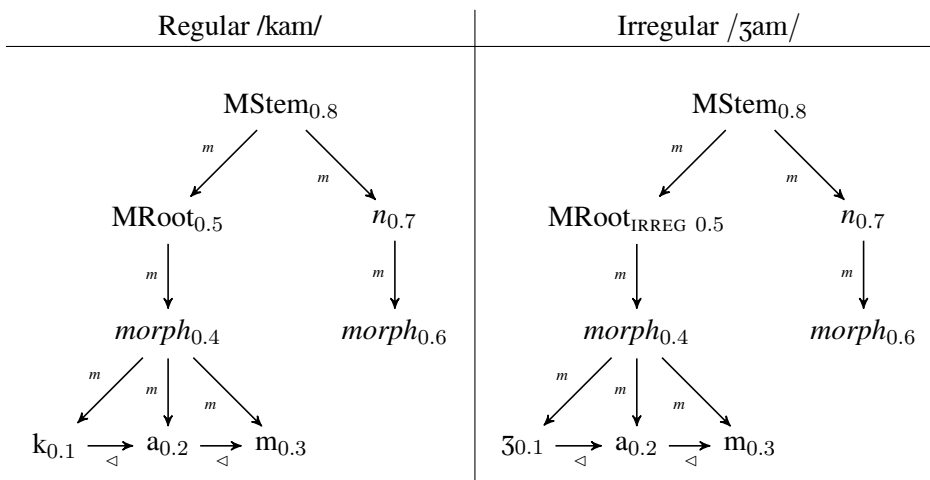
b. *Output of regular and irregular dative formation* – kam-i, ʒam-u

Regular kam-i	Irregular ʒam-u

$$\text{MStem}_{1.8}$$

Regular kam-i:

$\text{MStem}_{1.8} \xrightarrow{m} \text{MRoot}_{1.5}$, $\text{MStem}_{1.8} \xrightarrow{m} n_{1.7}$

$\text{MRoot}_{1.5} \xrightarrow{m} morph_{1.4}$

$n_{1.7} \xrightarrow{m} morph_{1.6}$

$morph_{1.4} \xrightarrow{m} \text{k}_{1.1}$, $morph_{1.4} \xrightarrow{m} \text{a}_{1.2}$, $morph_{1.4} \xrightarrow{m} \text{m}_{1.3}$

$\text{k}_{1.1} \xrightarrow{\triangleleft} \text{a}_{1.2} \xrightarrow{\triangleleft} \text{m}_{1.3} \xrightarrow{\triangleleft} \text{i}_{2.3}$

Irregular ʒam-u:

$\text{MStem}_{1.8} \xrightarrow{m} \text{MRoot}_{\text{IRREG } 1.5}$, $\text{MStem}_{1.8} \xrightarrow{m} n_{1.7}$

$\text{MRoot}_{\text{IRREG } 1.5} \xrightarrow{m} morph_{1.4}$

$n_{1.7} \xrightarrow{m} morph_{1.6}$

$morph_{1.4} \xrightarrow{m} \text{ʒ}_{1.1}$, $morph_{1.4} \xrightarrow{m} \text{a}_{1.2}$, $morph_{1.4} \xrightarrow{m} \text{m}_{1.3}$

$\text{ʒ}_{1.1} \xrightarrow{\triangleleft} \text{a}_{1.2} \xrightarrow{\triangleleft} \text{m}_{1.3} \xrightarrow{\triangleleft} \text{u}_{2.3}$

The user-defined predicate below finds a segment's morpheme. This predicate is satisfied for the final segment $m_{0.3}$ and its morpheme the $\text{MRoot}_{0.5}$. For convenience, I define a unary function which returns a segment's morpheme, with some conditions on when this function is defined.

(598) *Finding a segment's morpheme via...*

    a. *FO user-defined predicate*

        • **morpheme_of_seg**$(x,y) \overset{\text{def}}{=} \mathsf{morpheme}(x) \wedge \mathsf{seg}(y) \wedge$
$$\exists z[\mathsf{morph}(z) \wedge \mathsf{MDom}(x,z) \wedge \mathsf{MDom}(z,y)]$$

    b. *QF user-defined unary function*

        • $\mathbf{F_R}$:**morpheme_of_seg**$(y) \overset{\text{def}}{=} \mathsf{F_D}\text{:}\mathsf{MDom}^2(y)$

           The function is defined provided that x satisfies....
$$\mathsf{seg}(y) \wedge \mathsf{morph}(\mathsf{F_D}\text{:}\mathsf{MDom}(y)) \wedge \mathsf{morpheme}(\mathsf{F_D}\text{:}\mathsf{MDom}^2(y))$$
           The function is otherwise undefined

The user-defined predicate below checks if some morpheme is the trigger morpheme for the Armenian irregular dative *-u*, i.e., if the morpheme carries a the label of a plural, irregular root, or infinitival morpheme. These labels are primitive unary labels in the input. This predicate picks out the irregular MRoot in *ʒam* but not the regular MRoot in *kam*.

(599) *QF user-defined predicate to check if a morpheme is the trigger morpheme for the irregular dative*

        • **trigger_morpheme**$(x) \overset{\text{def}}{=} \mathsf{morpheme}(x) \wedge [\mathsf{plural}(x) \vee \mathsf{irregular}(x) \vee \mathsf{infinitival}(x)]$

The output functions below will generate the right suffixes. In Copy 2, we generate the irregular suffix *-u*$_{2.3}$ as an output correspondent of the final segment $m_{0.3}$ (x) if i) the final segment x is part of the morpheme $\text{MRoot}_{0.5}$ (y) and ii) the morpheme y is the trigger morpheme which has the right labels (plural, irregular, or infinitival). Otherwise, the regular suffix *-i* is generated because the final segment is not part of a trigger morpheme. These functions are locally-computed and QF-definable (600b). I don't show the functions which involve the base, i.e., how the base is faithfully outputted in Copy 1 and linearized with the suffix.

(600) a. *FO output functions for generating the irregular dative* -u *and regular dative* -i

- $\phi\text{u}(x_2) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \exists y[\textbf{morpheme\_of\_seg}(y, x) \wedge \textbf{trigger\_morpheme}(y)]$
- $\phi\text{i}(x_2) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \exists y[\textbf{morpheme\_of\_seg}(y, x) \wedge \neg\textbf{trigger\_morpheme}(y)]$

b. *QF output functions for generating the irregular dative* -u *and regular dative* -i

- $\phi\text{u}(x_2) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{trigger\_morpheme}(\textbf{F}_\textbf{R}\text{:}\textbf{morpheme\_of\_seg}(x))$
- $\phi\text{i}(x_2) \stackrel{\text{def}}{=} \textbf{final:seg}(x) \wedge \neg\textbf{trigger\_morpheme}(\textbf{F}_\textbf{R}\text{:}\textbf{morpheme\_of\_seg}(x))$

Thus, generating these allomorphs is computationally local. The computation simply references the morphological nodes which are 'close' to the final segment.

7.7.2 Apparent non-locality in conjugation classes

A common thought in linguistic theory is that a process is local if and only if the trigger and target of the process are adjacent (Odden 1994). Strict adjacency is also argued for in morphological allomorphy (Allen 1979; Embick 2010), i.e., that the trigger morpheme is the closest morpheme to the allomorph. However, computational locality is a *looser* form of linear locality or adjacency. A process is computationally local as long as the trigger and target are within a fixed bound. It does not matter whether this distance is zero, one (Božič 2019), or larger.

To illustrate, consider Armenian verb conjugation. Verbs can surface with one of three theme vowels *e,i,a*. These theme vowels are semantically content-less (cf. Aronoff 1994). The choice tends to correlate with transitivity but not always (Donabédian 1997). Roots which take these vowels form separate conjugation classes: Class E, Class I, and Class A (Kogian 1949; Boyacioglu 2010). Infinitivals surface with the suffix -*l* after the theme vowel. The 1SG, 2SG, and 3SG present surface with the suffix -*m, -s,* and zero -∅ after the theme vowel.

(601) *Paradigm of infinitivals and present singular verbs*

	Root	Infinitival		Present		
				1SG	2SG	3SG
Features		$\sqrt{}$-TH-INF		$\sqrt{}$-TH-1SG	$\sqrt{}$-TH-2SG	$\sqrt{}$-TH-3SG
Fixed morphs		$\sqrt{}$-TH-l		$\sqrt{}$-TH-m	$\sqrt{}$-TH-s	$\sqrt{}$-TH-∅
a. Class E	$\sqrt{}$ker-	ker-e-l	'to scratch'	ker-e-m	ker-e-s	ker-e-
b. Class I	$\sqrt{}$xos-	xos-i-l	'to speak'	xos-i-m	xos-i-s	xos-i-
c. Class A	$\sqrt{}$gart-	gart-a-l	'to read'	gart-a-m	gart-a-s	gart-a-

I focus on the theme vowels. I assume that roots have a diacritic which triggers the right theme vowel: CLASS:E, CLASS:I, CLASS:A which I represent below as a subscript. I assume that the theme vowel is generated as a suffix that is adjoined to a covert verbalizer *v* suffix (Oltra-Massuet 1999a,b). The trigger morpheme for the right theme vowel is the underlined root. I omit the morph nodes.

(602)   *Morphological structure of simple infinitivals*

Class E	Class I	Class A

MWord — MStem — v — MRoot$_E$, v, TH, INF
ker-   -∅   -e   -l

MWord — MStem — v — INF — MRoot$_I$, v, TH, -l
xos-   -∅   -i

MWord — MStem — v — MRoot$_A$, v, TH, INF
gart-   -∅   -a   -l
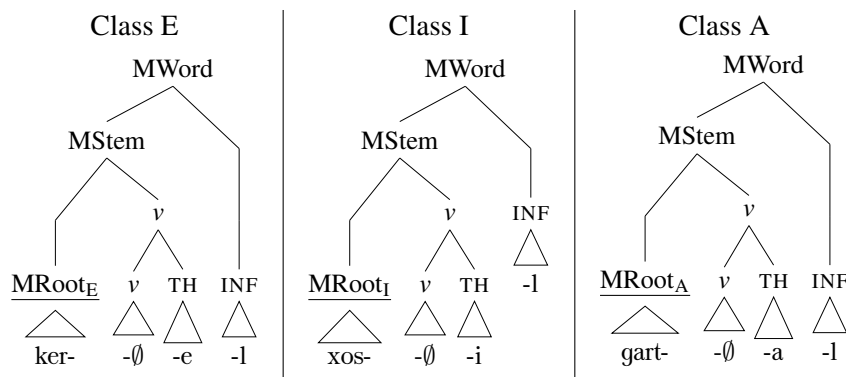
Given a Class-E root *ker-* or Class-I root *xos-* as the input, the trigger morpheme for the theme vowel is the MRoot. The root is local to the input-final segment. I show an illustrative input and output below. The v and TH are added simultaneously in the same cycle, while the infinitival is added in a later cycle.

(603)   *Input and output to adding theme vowels*

Class E		Class I	
Input	Output	Input	Output

MStem — v — MRoot$_E$, v, TH
morph   *morph*   *morph*   *morph*
ker-   ker-   -∅   -e

MStem — v — MRoot$_I$, v, TH
morph   *morph*   *morph*   *morph*
xos-   xos-   -∅   -i

MRoot$_E$ | *morph* | ker-

MRoot$_I$ | *morph* | xos-
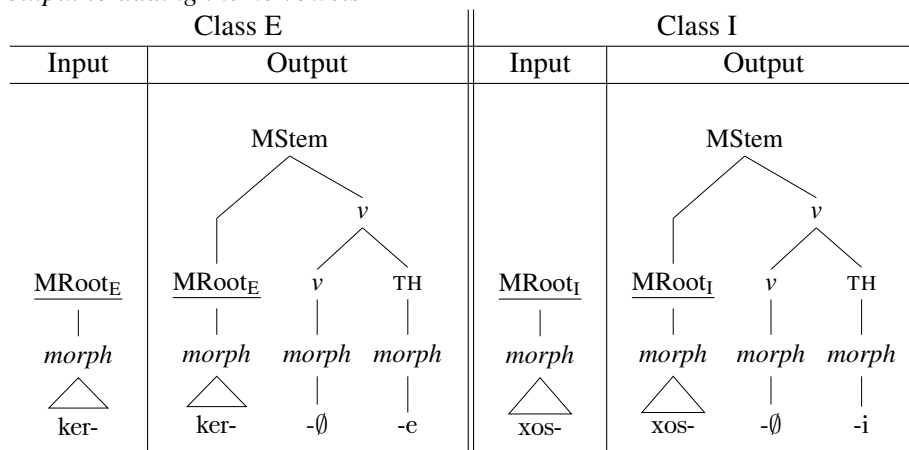
I show the more explicit input and output below. I only show the segments, and omit the morphological nodes of the suffix. The covert v suffix has no segment intervening between the base and theme vowel.

(604)   a. *Explicit input to generating theme vowels*

Class E *ker-*	Class I *xos-*

MRoot$_{E\ 0.5}$
m ↓
morph$_{0.4}$
m ↙ m ↓ m ↘
k$_{0.1}$ →$_◁$ e$_{0.2}$ →$_◁$ r$_{0.3}$

MRoot$_{I\ 0.5}$
m ↓
morph$_{0.4}$
m ↙ m ↓ m ↘
x$_{0.1}$ →$_◁$ o$_{0.2}$ →$_◁$ s$_{0.3}$
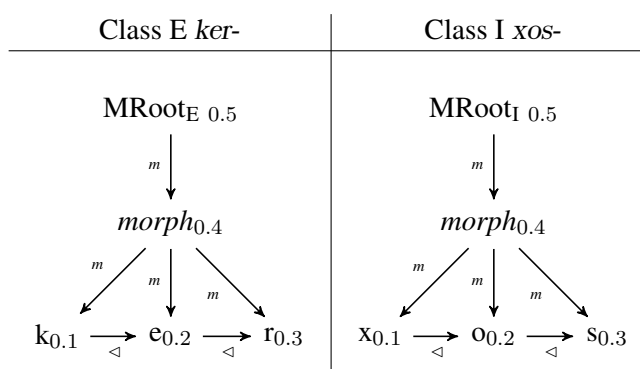
b. *Intermediate output of generating theme vowels*

Class E *ker-*	Class I *xos-*

$\mathrm{MRoot_{E}}_{\ 1.5}$

$\downarrow m$

$morph_{1.4}$

$k_{1.1} \rightarrow e_{1.2} \rightarrow r_{1.3} \longrightarrow e_{2.3}$

$\mathrm{MRoot_{I}}_{\ 1.5}$

$\downarrow m$

$morph_{1.4}$

$x_{1.1} \rightarrow o_{1.2} \rightarrow s_{1.3} \longrightarrow i_{2.3}$
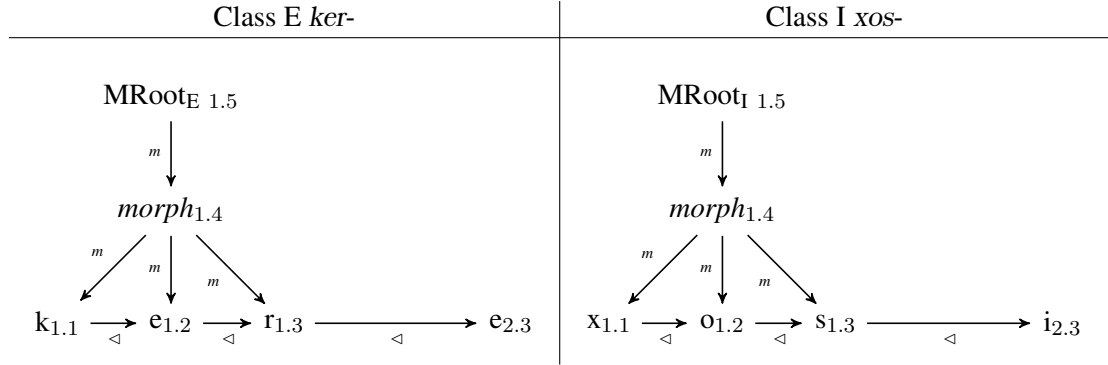
The output is partially generated via the output functions below. I assume the input has unary labels for the different classes of verbs. The function $\phi\mathsf{e}(x_2)$ will generate the $-e_{2.3}$ theme vowel in Copy 2 if i) it is the output correspondent of the final segment $r_{0.3}(x)$, ii) the final segment x belongs to a morpheme $\mathrm{MRoot}_{0.5}(y)$, iii) and y has the right CLASS:E feature. The other output functions $\phi\mathsf{i}(x_2), \mathsf{a}(x_2)$ are analogously defined. They are all locally-computible (605b). I do not show the output functions needed to generate the base or the morphological nodes of the suffix.

(605)　　a. *FO output functions to generate the right theme vowel based on the root*
- $\phi\mathsf{e}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \exists y[\mathbf{morpheme\_of\_seg}(y,x) \wedge \mathsf{Class{:}E}(y)]$
- $\phi\mathsf{i}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \exists y[\mathbf{morpheme\_of\_seg}(y,x) \wedge \mathsf{Class{:}I}(y)]$
- $\phi\mathsf{a}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \exists y[\mathbf{morpheme\_of\_seg}(y,x) \wedge \mathsf{Class{:}A}(y)]$

　　b. *QF output functions to generate the right theme vowel based on the root*
- $\phi\mathsf{e}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \mathsf{Class{:}E}(\mathbf{F_R}{:}\mathbf{morpheme\_of\_seg}(x))$
- $\phi\mathsf{i}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \mathsf{Class{:}I}(\mathbf{F_R}{:}\mathbf{morpheme\_of\_seg}(x))$
- $\phi\mathsf{a}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \mathsf{Class{:}A}(\mathbf{F_R}{:}\mathbf{morpheme\_of\_seg}(x))$

The computation so far is local. However, Armenian conjugation does show signs of non-locality in the past perfect. In the past perfect, we replace the infinitival suffix *-l* with two suffixes: the aorist or perfective suffix *-t͡s-* and an agreement suffix: *ker-e-l* ∼ *ker-e-t͡s-i* 'I scratched'. The theme vowel *-i-* of Class-I verbs is replaced by the theme vowel *-e-* in bold: *xos-**i**-l* ∼ *xos-**e**-t͡s-a* 'I spoke'. The change in theme vowels is a case of outwards-sensitive allomorphy (cf. Bobaljik 2000). It is difficult to formalize in a purely cyclic model. I discuss possible solutions in Chapter 9. But the crucial point is the agreement suffixes which differ across classes. The agreement suffixes are *-i, -ir, -∅* for Class E and A, but *-a, -ar, -av* for Class I. These are in bold.

(606)   *Paradigm of infinitivals and past perfect singular verbs*

	Root	Infinitival		Past Perfect		
				1SG	2SG	3SG
Features		$\sqrt{}$-TH-INF		$\sqrt{}$-TH-PRF-1SG	$\sqrt{}$-TH-PRF-2SG	$\sqrt{}$-TH-PRF-3SG
Fixed morphs		$\sqrt{}$-TH-l		$\sqrt{}$-TH-t͡s-i/a	$\sqrt{}$-TH-t͡s-ir/ar	$\sqrt{}$-TH--t͡s-∅/av
a. Class E	$\sqrt{}$ker-	ker-e-l	'to scratch'	ker-e-t͡s-i	ker-e-t͡s-ir	ker-e-t͡s
b. Class I	$\sqrt{}$xos-	xos-i-l	'to speak'	xos-**e**-t͡s-**a**	xos-**e**-t͡s-**ar**	xos-**e**-t͡s-**av**
c. Class A	$\sqrt{}$gart-	gart-a-l	'to read'	gart-a-t͡s-i	gart-a-t͡s-ir	gart-a-t͡s

The agreement suffixes are chosen based on the class features of the root. However, if the input to generating the person suffixes includes both the theme vowel and the perfective suffix *ker-e-t͡s-*, then the input-final segment is not part of the root. The trees below illustrate. I underline the trigger morpheme, and I put in bold the morph nodes and segments which separate the MRoot from the final segment.

(607)   a.   *Input to adding the agreement suffixes in the past perfect*
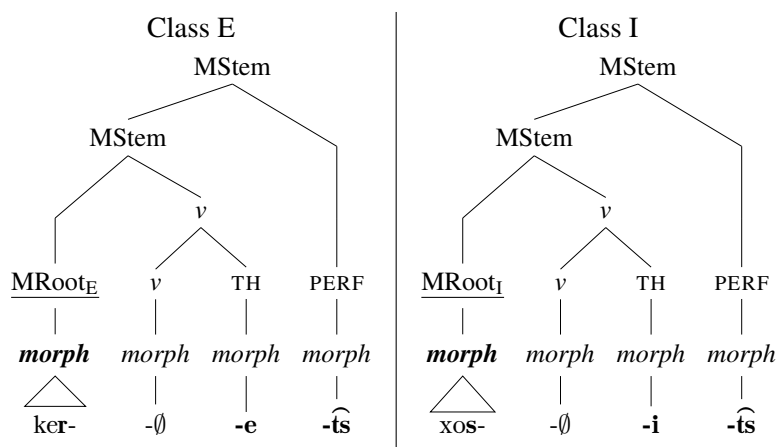
b.   *Output of adding the agreement suffixes in the past perfect*
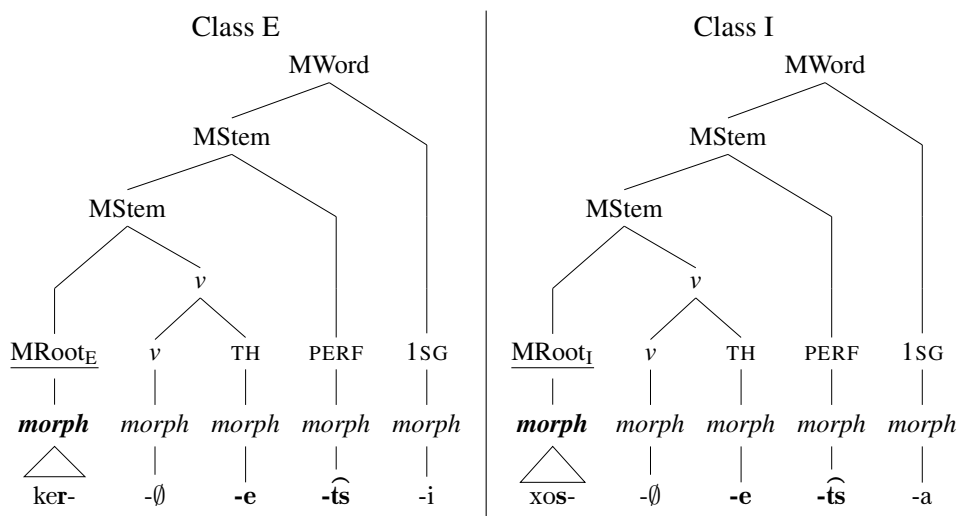
On the surface, the allomorphy in agreement appears long-distance. However, this is incorrect because Armenian morphology does not recursively add morphemes to verbs. The MRoot is always at a fixed maximal distance from the perfective suffix -$\widehat{ts}$. Specifically, in simple regular verbs, the perfective -$\widehat{ts}$ is always at most 2 segments away from the root's final segment: *ke<u>r</u>-e-ts-*, *xo<u>s</u>-i-ts-*. Thus, generating the right agreement suffix is still computationally local. I show below the explicit input and partial output for a Class-E and Class-I past perfect verb. I omit the MStem nodes.

(608) a. *Explicit input to generating agreement suffixes in the past perfect*

Class E *ker-* | Class I *xos-*
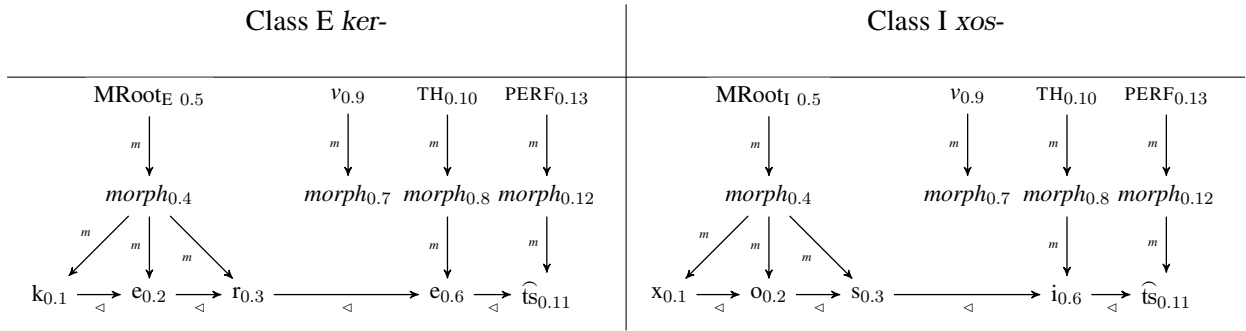
  b. *Partial output of generating agreement suffixes in the past perfect*

Class E *ker-* | Class I *xos-*
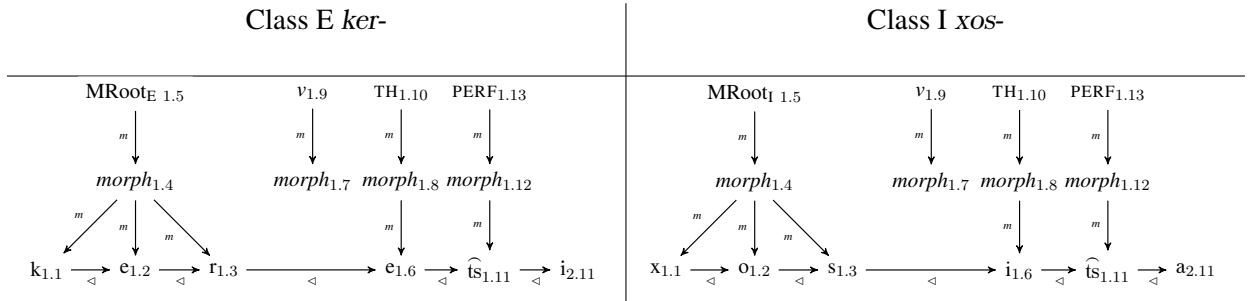
I focus on generating the agreement suffix's segment. I assume the other nodes are handled by other output functions. Given an MRoot x and segment y, the predicate **MRoot_pre_perf_seg**(x, y) below checks if some MRoot$_{0.5}$ (x) precedes a segment $\widehat{ts}$ (y) such that i) y is part of the perfective suffix, ii) y follows two segments v ($r_{0.3}$ or $s_{0.3}$) and w ($e_{0.6}$ or $i_{0.6}$), and iii) v belongs to the MRoot. The predicate can be replaced by a user-defined unary function (609b) which returns this MRoot x if given the perfective segment y. It locally-computed and QF-definable.

(609) a. *FO user-defined predicate for finding the MRoot given the perfective segment*
- **MRoot_pre_perf_seg**$(x, y) \stackrel{\text{def}}{=} \mathsf{MRoot}(x) \wedge \mathsf{seg}(y) \wedge$
  $\exists z[\mathsf{morpheme}(z) \wedge \mathsf{perfect}(z) \wedge \mathsf{MDom}(z, y)] \wedge$
  $\exists v, w[\mathsf{seg}(v) \wedge \mathsf{seg}(w) \wedge \mathsf{succ:seg}(v, w) \wedge \mathsf{succ:seg}(w, y) \wedge$
  $\mathsf{MDom}(x, v)]$

  b. *QF user-defined unary function for finding the MRoot given the perfective segment*
- **F$_\mathbf{R}$:MRoot_pre_perf_seg**$(y) \stackrel{\text{def}}{=} \mathsf{F_D{:}MDom}(\mathsf{F_R{:}succ{:}seg}^2(y))$
  The function is defined provided that x satisfies....
  $\mathsf{seg}(y) \wedge \mathsf{perfective}(\mathbf{F_R}\textbf{:morpheme\_of\_seg}(y))$

The function is otherwise undefined

With these formulas, the output function below then generates the right 1SG suffixes by checking if the MRoot has the right class labels. They are locally-computible (610b).

(610)   a. *FO output function for generating the* 1SG *past perfect agreement suffixes*
   - $\phi\mathtt{i}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \exists y[\mathbf{MRoot\_pre\_perf\_seg}(y,x) \wedge$
$$[\mathsf{Class{:}E}(y) \vee \mathsf{Class{:}A}(y)]]$$
   - $\phi\mathtt{a}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \exists y[\mathbf{MRoot\_pre\_perf\_seg}(y,x) \wedge \mathsf{Class{:}I}(y)]$

   b. *QF output function for generating the* 1SG *past perfect agreement suffixes*
   - $\phi\mathtt{i}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge [\mathsf{Class{:}E}(\mathbf{F_R}{:}\mathbf{MRoot\_pre\_perf\_seg}(x)) \vee$
$$\mathsf{Class{:}A}(\mathbf{F_R}{:}\mathbf{MRoot\_pre\_perf\_seg}(x))]]$$
   - $\phi\mathtt{a}(x_2) \stackrel{\text{def}}{=} \mathbf{final{:}seg}(x) \wedge \mathsf{Class{:}I}(\mathbf{F_R}{:}\mathbf{MRoot\_pre\_perf\_seg}(x))]$

Thus, generating the agreement suffix is still computationally local even though the trigger morpheme is not adjacent to the input-final segment. As long as the trigger morpheme is still a predictable finitely bounded distance away from the final segment, the computation is local.

7.7.3   When allomorphy is non-local

The above results may imply that morphologically-conditioned allomorphy is always local. In this section, I show two cases where this is not so. First, suffix (prefix) allomorphy is computationally non-local if the trigger morpheme is an unbounded distance away from the final (initial) segment (§7.7.3.1). Second, allomorphy is non-local if it references feature percolation to the topmost morphological node (§7.7.3.2).

7.7.3.1   Non-locality from opposite-sided triggers

Computational locality requires that the trigger and target are within a finite bound. For allomorphy, computational locality is violated when the affix is a suffix (prefix) but the morphological trigger is on the other side of the input. This problem is analogous to phonologically-conditioned mobile affixation (§7.5.3). In fact, prefix-suffix dependencies are a common issue in claiming morphology as computationally local or even finite-state (Langendoen 1981; Carden 1983; Hammond 1992, 1993; Bjorkman and Dunbar 2016; Aksënova and De Santo 2018).

Armenian does not have morphologically-conditioned allomorphy which references prefix-suffix dependencies. I instead illustrate a hypothetical example with a constructed language that has two plural allomorphs: regular *-s* and irregular *-ren*. The irregular suffix *-ren* is used when the input starts with an irregular prefix *non-*, such that this prefix takes scope over the rest of the input. For example, the root *dog* takes a regular plural *dog-s*.[3] But, when the irregular prefix *non-* is added, then the plural form is *non-dog-ren*. I illustrate their tree structures below.[4] The trigger morpheme is underlined; the intervening segments are in bold.

---

[3]I omit using a covert category suffix to turn the root *dog* to a free-standing noun MStem.

[4]A similar case study is found in Russian verbs where we have prefix-suffix dependencies based on the telicity of affixes (Aksënova and De Santo 2018).

(611)   *Non-locality from prefix-suffix dependencies when computed over segments*

	Regular plural			Prefixed irregular plural	
Singular	Plural		Singular	Plural	

Regular plural — Singular:
MRoot — *morph* — △ *dog*

Regular plural — Plural:
MWord → MRoot, PL
MRoot — *morph* — △ *dog*
PL — *morph* — △ *-s*

Prefixed irregular plural — Singular:
MStem → n_{IRREG}, MRoot
n_{IRREG} — **morph** — △ *non-*
MRoot — *morph* — △ **dog**

Prefixed irregular plural — Plural:
MWord → MStem, PL
MStem → n_{IRREG}, MRoot
n_{IRREG} — **morph** — △ *non-*
MRoot — *morph* — △ **dog**
PL — *morph* — △ *-ren*

Because of order-preservation, the suffix is generated as an output correspondent of the final segment. The computation of the allomorph is non-local because the final segment *g* can be an unbounded distance away from the prefix *non-*. In terms of segments, there can be an unbounded number of segments between the prefix and final segment because there is no bound on the size of roots.

Interestingly, over morphological nodes, the computation is still non-local and not QF. The generalization is that the input must *start* with the irregular prefix. There could be an unbounded number of intervening morphological nodes between the final segment and the irregular prefix. To illustrate, assume that the noun *dog* can take the regular prefix *pre-* to form a regular noun *pre-dog*. This word is regularly pluralized as *pre-dog-s*. But if this derivative takes the irregular prefix *non-*, then the doubly-prefixed derivative takes the irregular plural *non-pre-dog-ren*. I show the intervening morphological nodes in bold.

(612)   *Non-locality in prefix-suffix dependencies when computed over morphological nodes*

	Regular & Prefixed			Irregular & Doubly-prefixed	
Singular	Plural		Singular	Plural	

Regular & Prefixed — Singular:
MStem → n, MRoot
n — *morph* — △ *pre-*
MRoot — *morph* — △ *dog*

Regular & Prefixed — Plural:
MWord → MStem, PL
MStem → n, MRoot
n — *morph* — △ *pre-*
MRoot — *morph* — △ *dog*
PL — *morph* — △ *-s*

Irregular & Doubly-prefixed — Singular:
MStem → n_{IRREG}, **MStem**
MStem → n, **MRoot**
n_{IRREG} — *morph* — △ *non-*
n — *morph* — △ *pre-*
MRoot — *morph* — △ **dog**

Irregular & Doubly-prefixed — Plural:
MWord → **MStem**, PL
MStem → n_{IRREG}, **MStem**
MStem → n, **MRoot**
n_{IRREG} — *morph* — △ *non-*
n — *morph* — △ *pre-*
MRoot — **morph** — △ *dog*
PL — *morph* — △ *-ren*

Thus, given our simple tree-based representations for morphology, prefix-suffix dependencies are not computationally local. To make these processes local, we need to enrich our input's representation, e.g., by formally encoding the derivational history of the input (cf. Aksënova and De Santo 2018). I discuss one possible formalization in Chapter 9 in the context of outwards-looking allomorphy.

7.7.3.2 Non-locality because of feature percolation

Another way that morphologically-conditioned allmorphy is non-local is if it references a trigger morpheme which is deeply embedded in the input. In these cases, most analyses assume an operation of feature percolation (Lieber 1980, 1989) such that the triggering morphological feature is part of the topmost morphological node (cf. Williams 1981).

To illustrate, consider the English word *tooth*. Its irregular plural is *teeth*. Assume the irregular plural allomorph is a floating feature [+F] for [+FRONT]. The trigger morpheme for the irregular plural is an irregular label on the root.

(613) *Irregular plurals in English* tooth

Singular	Plural

MWord

MStem   MStem

MRoot$_{\text{IRREG}}$ *n*   MRoot$_{\text{IRREG}}$ *n*   PL

morph   *morph*   *morph*   *morph*   *morph*

tooth   -∅   *teeth*   -∅   [+F]
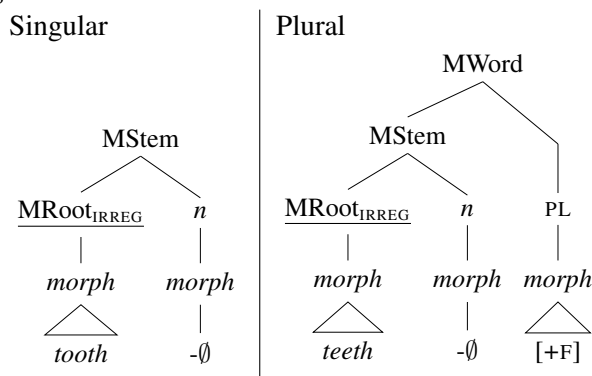
Problems arise in compounding. When the word forms an endocentric compound, the compound inherits the irregular plural: *milk-teeth*. But if the word forms an exocentric compound, then the compound is regularly pluralized: *saber-tooth-s*. A traditional analysis is that somehow *tooth* in *milk-teeth* forms the head of the compound and percolates its irregular features up the tree to the topmost MNode. In contrast in the head-less *saber-tooth-s*, the irregular feature does not percolate all the way up to the topmost node.[5]

I illustrate below using the compound structures from §6.2.2. The stems are concatenated to form a morphological concatenation *MConc*. For easier illustration, I omit the MStem$_C$ which dominates the entire compound and which takes its own zero-suffix. The irregular label $_{\text{IRREG}}$ percolates up the endocentric compound in bold, but not *all* the way up to the exocentric compound.

---

[5]Irregular inflection is likewise inflected in Armenian endocentric compounds, see Chapter 3.

(614)   *Irregular plurals in English compounds*

   a. *Input*

Endocentric Compound	Exocentric Compound

MConc$_{\textbf{IRREG}}$

MStem   **MStem**$_{R\ \textbf{IRREG}}$

MRoot   *n*   MRoot$_{\text{IRREG}}$ *n*

morph   *morph*   *morph*   *morph*

milk   -∅   *tooth*   -∅

MConc

MStem   **MStem**

MRoot   *n*   MRoot$_{\text{IRREG}}$ *n*

morph   *morph*   *morph*   *morph*

saber   -∅   *tooth*   -∅

   b. *Output*

Endocentric Compound	Exocentric Compound

MWord

MConc$_{\textbf{IRREG}}$

MStem   **MStem**$_{R\ \textbf{IRREG}}$

MRoot   *n*   MRoot$_{\text{IRREG}}$ *n*   PL

morph   *morph*   *morph*   *morph*   *morph*

milk   -∅   *teeth*   -∅   [+F]

MWord

MConc

MStem   **MStem**

MRoot   *n*   MRoot$_{\text{IRREG}}$ *n*   PL

morph   *morph*   *morph*   *morph*   *morph*

saber   -∅   *tooth*   -∅   -s

To generate the right plural in a compound, we need to check that 1) the final segment *th* belongs to an irregular root *tooth*, and that 2) the irregular features of the root percolated all the way up the tree. The problem is that the plural allomorphs are suffixes (or suffixal features) and must be defined as output correspondents of the final segment. If there is no bound on the distance between the final segment and the topmost node, then determining the second factor on feature percolation is not computationally local.

Thus, contra linguistic intuitions, feature percolation does not always make long-distant allomorphy into local allomorphy. Superficially, feature percolation seems to require quantification. Furthermore, depending on the specific theory, feature percolation can itself be computationally difficult to implement (Ritchie et al. 1992:ch4).

7.8   Conclusion

This chapter examined computational aspects of affixation and allomorphy. After first discussing the computational insignificance of certain morphological controversies, I introduced and formalized the concept

of order-preservation. As a constraint on possible input-output correspondences, order-preservation was found to be prevalent in affix order and linearization. Although reduplication is not generally order-preserving, simple affixation (suffixation, prefixation, mobile affixes) are order-preserving transductions. As order-preserving transductions, these morphological processes can be implemented with 1-way FSTs which share the same intensional description or strong generative capacity of our logical formalization. As a caveat though, this conclusion depends on what representations we use. If we used more enriched representations, e.g., using constants to refer to the initial segment, then we can describe more processes as being computationally and order-preserving. Whether this richer representation is empirically warranted is open question which I did not pursue.

Moving on to allomorphy, I showed how certain types of allomorphy can or can't be computationally local. There is a tendency for phonologically- and morphologically-conditioned to be local. This tendency was found in Armenian. Computational locality is found regardless whether the allomorphy is computed over phonological segments or over morphological nodes. Non-locality is limited to certain problem areas, including mobile affixation, prefix-suffix dependencies, and feature percolation.

Chapter 8

Locality with and without cyclicity

8.1 Global information and locality in a cyclic interface

The formalization in this thesis is cyclic and it utilized SETTINGS encapsulation as a way to encode long-distance morphological information. In this chapter, I relax these two assumptions and I show how much of the interface is still computationally local. I first clarify these two assumptions.

First, my formalization is cyclic because of how morphology and phonology are interleaved. Given some input, we first applied a **morphological** transduction. We **examined** the morphology and encapsulated some global information about the input into the constant SETTINGS. With this information, **prosodic** and **phonological** processes then applied. The sum of the morphological, encapsulation, prosodic, and cophonological processes constitutes a single cycle. The output of this cycle is then submitted as the *input* for another morphological transduction. This triggered another round of phonological transductions.

Second, I encapsulated different types of information into the SETTINGS. The SETTINGS has general information such as what dialect the derivation is in. I also encoded different types of long-distance information which was derived by examining the morphological and prosodic structure of the input. However, I only encapsulated information about the properties of the morphologically topmost node: what cophonology label it had, what types of morphological nodes it dominated, and what prosodic constituents were associated with the topmost node's daughters. Doing so lets us separate the (often local) segmental aspects of a phonological rule from its (possibly global) morphological triggers.

With this architecture, the bulk of the interface was shown to be computationally local. In this chapter, I illustrate how non-locality was minimized by using both cyclicity and this constant SETTINGS. I reformulate some prosodic and phonological processes but without the constant SETTINGS. Thus, we can no longer encapsulate any potentially global information into one accessible constant. This makes computing the right cophonology (§8.2) and prosody (§8.3) potentially non-local.

These are the results. In the case of cophonologies, constituent-based definitions for cophonologies were earlier shown to be non-local (Chapter 5:§5.6). In contrast, an alternative morpheme-based definition can sometimes be computationally local (§8.2). For the prosody, a cyclic prosody parse is computationally local with or without the SETTINGS (§8.3). Interestingly, post-cyclic or non-cyclic prosody is computationally local for non-compounds (§8.3.2.1), but it is non-local for compounds (§8.3.2.2).

8.2   Locality and non-locality in cophonology domains

Informally, a cophonology is a set of rules which apply in some domain. The phonological environment triggering the change can be local but its cophonological trigger can be non-local. However, the non-locality depends on how we *analyze* the location of this cophonological trigger. Throughout this dissertation, I used a constituent-based definition of cophonologies where the trigger is the topmost MNode. As explained earlier in Chapter 5: §5.6, this requires non-local computation. For this reason, I encapsulated this non-local trigger into the SETTINGS constant. In this section, I decompose the above argument and analysis (§8.2.1). I show that an alternative *morpheme*-based definition for cophonologies can be computationally local (§8.2.2).[1]

8.2.1   Components of rule domains

To illustrate a cophonology, the rule of final stress applies in *amusín* because we're in the stem-level cophonology.

(615)   *Application of final stress*

$$/\text{amusin}/ \qquad\qquad [\text{amusín}] \qquad \text{'husband'}$$

/amusin/	[amusín] 'husband'
MStem$_{SLevel}$	MStem$_{SLevel}$
MRoot n	MRoot n
morph *morph*	*morph* *morph*
/amusin/ $\emptyset$	amusín $\emptyset$
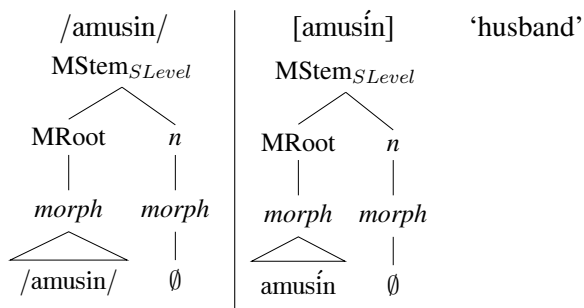
In this dissertation, I factorized a rule domain into its:

1. *morphological trigger*, e.g., stress applies because the topmost node is an MStem.
2. *cophonology label*, e.g., the MStem$_{SLevel}$ has the label for the *stem-level* cophonology.
3. *phonological target*, e.g., a full vowel *i*.
4. *phonological context*, e.g., the full vowel is the rightmost one *in#*.

The phonological target and context can be formalized into a single SPE-rule, set of constraints, or an FST. The phonological target and context are often within a bounded distance from each other, i.e., rules have a local phonological context (Odden 1994; Chandlee and Heinz 2018).

This dissertation focused on formalizing and evaluating the morphological triggers and their cophonology labels. Throughout chapters §5-6, I defined the trigger in terms of morphological constituents, specifically the topmost MNode. The cophonology label was *on* these constituents. However, as explained in Chapter 5: §5.6, this constituent-based definition of triggers is computationally non-local. A priori, there can be an unbounded distance between the trigger and target, i.e. for stress, the topmost MStem node could be an unbounded distance away from the rightmost full vowel. The unboundedness can come from recursively adding covert affixes or prefixes.

---

[1]Throughout this section on cophonologies, I assume the derivation is cyclic.

The issue of non-locality is clearer when we look at vowel reduction. Reduction applies in the stem-level but not word-level The destressed vowel reduces in *amusn-agán* (616b) but not *amusin-nér* (616c). The former has a topmost MStem node, while the latter has a topmost MWord node. Because of zero affixation, there could be no bound on the number of intervening morphological nodes between the target vowel and the topmost trigger.

(616)   *Application of reduction with constituent-based cophonologies*

a. Root	b. Derivative	c. Inflected form
amusín 'husband'	*amusn-agan* 'marital'	*amusin-ov* 'husband-PL'

I encapsulated the potentially non-local trigger into a constant SETTINGS. This made the derivation simpler to understand by separating the non-local trigger from the rest of the local morphophonology.

8.2.2   Morpheme-based rule domains

Interestingly, some of the non-locality is resolved in morpheme-based definitions for cophonologies. As said, this dissertation assumed a constituent-based definition for cophonologies: the trigger was the topmost MNode and that it carried the label. In contrast, in a morpheme-based definition, the cophonology is triggered by morphemes which carry these cophonology labels. That is, we apply final stress when the target vowel is part of an MRoot (617a) or suffix (617b,c). And, we apply reduction if the target destressed vowel is before a final suffix which is derivational (*-agan* 617b), not inflectional (*-ov* 617c). I assume that roots carry the stem-level label. I use the smaller subscripts $_{SL,WL}$ for space.

(617)   *Application of reduction with morpheme-based cophonologies*

a. Root	b. Derivative	c. Inflected form
amusín 'husband'	*amusn-agan* 'marital'	*amusin-ov* 'husband-PL'

To illustrate with reduction, I show an explicit input and output for /amusĭn-agán/ below. I omit any morphological nodes above morphemes. I omit the PStem.

(618)   a. *Input for morpheme-based application of stem-level reduction* //amusĭn-agán//
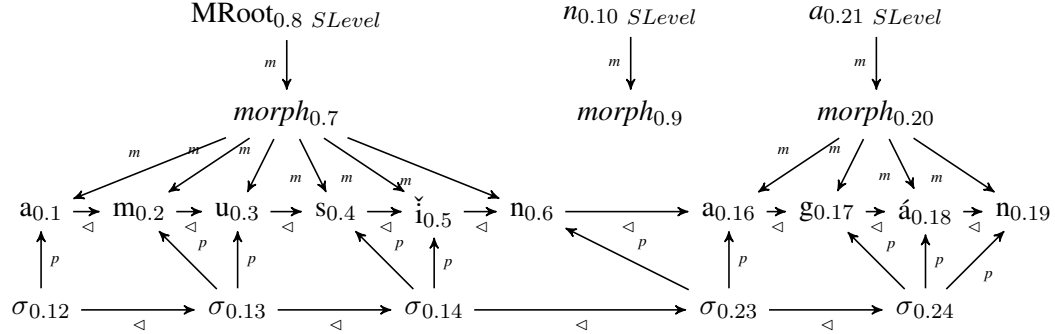
b. *Output of morpheme-based application of stem-level reduction* amusn-agán
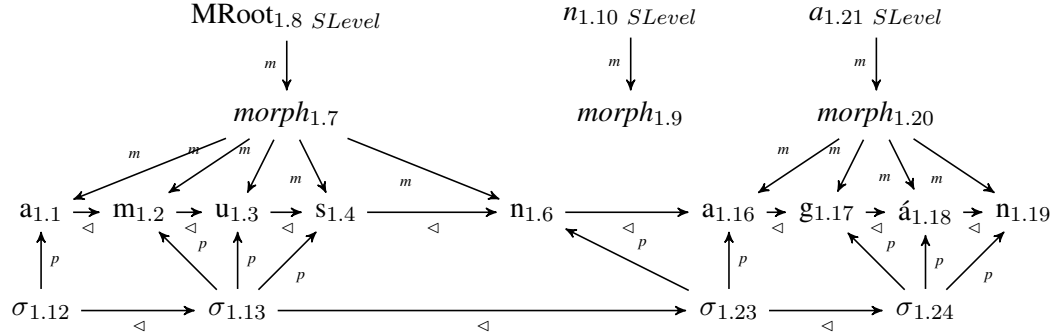
For this input-output pair, the target of reduction is the destressed high vowel $i_{0.5}$, while the trigger is the derivational suffix *-agan* or $a_{0.20}$. Specifically, the trigger is the morpheme, not the MStem. This morpheme carries the stem-level cophonology label $_{SLevel}$ or Cophon:SLevel(x)

To compute this morpheme-based process of reduction, we need to find the trigger given the target. This is facilitated by using several predicates. Given two vowels x, y, the predicate **next_vowel**(x, y) checks if y comes after x, i.e., they are adjacent on the tier of vowels. We determine this by checking if the syllable u of x precedes the syllable v of y. In the case of //a.mu.sĭ.n-a.gán//, the predicate is satisfied by the pair $i_{0.5}$, $a_{0.9}$ because the syllable $\sigma_{0.14}$ precedes $\sigma_{0.23}$. This predicate is locally computed; we can define a QF unary function to find the subsequent vowel y of some vowel x.

(619)   *Finding the next vowel via a...*

    a. *FO user-defined predicate*

- **next_vowel**$(x, y) \stackrel{\text{def}}{=}$ vowel$(x) \wedge$ vowel$(y) \wedge \exists u, v[$syll$(u) \wedge$ syll$(v) \wedge$
  PDom:syll_nuc$(u) \wedge$ PDom:syll_nuc$(v) \wedge$ succ:syll$(u, v)]$

    b. *QF user-defined unary function*

- $\mathbf{F_L}$:**next_vowel**$(x) \stackrel{\text{def}}{=} \mathsf{F_M}$:PDom:syll_nuc$(\mathsf{F_L}$:succ:syll$(\mathsf{F_D}$:PDom:syll_nuc$(x)))$
  The function is defined provided that x satisfies....
      vowel$(x) \wedge$ vowel$(\mathsf{F_M}$:PDom:syll_nuc$(\mathsf{F_L}$:succ:syll$(\mathsf{F_D}$:PDom:syll_nuc$(x))))$

The function is otherwise undefined

The predicate **morpheme_of_seg**(x, y) checks if a segment y is part of a morpheme x. For example, the morpheme *-agan* or $a_{0.21}$ is the morpheme of the suffix vowel $a_{0.16}$. The predicate can be converted to a QF unary function to find the morpheme x of a segment y.

(620)   *Finding a segment's morpheme via...*
   a.   *FO user-defined predicate*
      - **morpheme_of_seg**$(x, y) \stackrel{\text{def}}{=}$ morpheme$(x) \land$ seg$(y)\land$
         $$\exists z[\text{morph}(z) \land \text{MDom}(x, z) \land \text{MDom}(z, y)]$$
   b.   *QF user-defined unary function*
      - $\mathbf{F_R}$**:morpheme_of_seg**$(y) \stackrel{\text{def}}{=} \mathsf{F_D}$:MDom$^2(x)$
         The function is defined provided that x satisfies....
         $$\text{seg}(y) \land \text{morph}(\mathsf{F_D}\text{:MDom}(y)) \land \text{morpheme}(\mathsf{F_D}\text{:MDom}^2(y))$$
         The function is otherwise undefined

With all these predicates in place, we can now determine if some vowel should undergo destressed reduction or not. The predicates **target:destressed_high_vowel**(x) and **trigger:der_suffix**(x) pick the target and trigger of vowel reduction: a destressed high vowel and a derivational suffix. These predicates are straightforward. The derivational suffix must have the right cophonology label.

(621)   *QF user-defined predicates for checking if a node is...*
   a.   *The target – a destressed high vowel*
      - **target:destressed_high_vowel**$(x) \stackrel{\text{def}}{=} [\text{i}(x) \lor \text{u}(x)] \land \text{destressed:vowel}(x)$
   b.   *The trigger – a derivational suffix*
      - **trigger:der_suffix**$(x) \stackrel{\text{def}}{=}$ morpheme$(x) \land$ der$(x) \land$ Cophon:SLevel(x)

We apply reduction if *both* the target and trigger are found *together* in the input. The predicate **reducible**(x) is satisfied if i) x is a a destressed high vowel (the target), ii) the subsequent or next vowel y exists, iii) this vowel is part of a morpheme z, and iv) this morpheme z is a derivational suffix (the trigger). For //amusĭn-agán//, the target x is $i_{0.5}$, the next vowel y is $a_{0.16}$, and the trigger z is the suffix morpheme *-agan*, i.e., $a_{0.21}$. This predicate is locally computible using our user-defined QF unary functions.

(622)   a.   *FO user-defined predicates for checking if a destressed high vowel is reducible*
      - **reducible**$(x) \stackrel{\text{def}}{=}$ **target:destressed_high_vowel**$\land$
         $$\exists y, z[\text{vowel}(y) \land \text{morpheme}(z) \land \textbf{next\_vowel}(x, y)\land$$
         $$\textbf{morpheme\_of\_seg}(z, y) \land \textbf{trigger:der\_suffix}(z)]$$
   b.   *QF user-defined predicates for checking if a destressed high vowel is reducible*
      - **reducible**$(x) \stackrel{\text{def}}{=}$ **target:destressed_high_vowel**$\land$
         $$\textbf{trigger:der\_suffix}(\mathbf{F_R}\textbf{:morpheme\_of\_seg}(\mathbf{F_L}\textbf{:next\_vowel}(x)))$$

With this predicate in place, the various output functions in Chapter 6: §6.6.1 can be reformulated to reference this predicate and then apply reduction. The predicate **reducible**(x) is not satisfied when the

destressed high vowel precedes an inflectional suffix as in *amusin-ov*. Here, the high vowel precedes a morpheme $PL_{0.20}$ ($= w$). This morpheme is not stem-level because it is inflectional.

(623)   *Input to blocking vowel reduction in the inflected word* amušin-nér
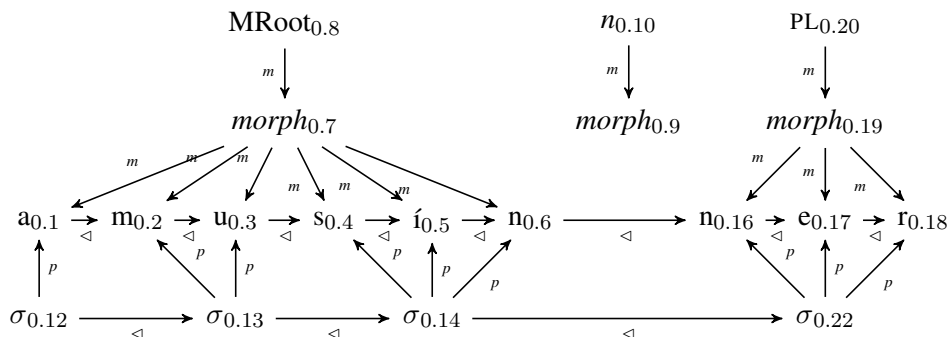
This entire formalization is local because it only references the morphemes and syllables which are adjacent to the destressed high vowel via $\mathsf{MDom}(x, y), \mathsf{PDom:syll\_nuc}(x, y), \mathsf{succ:syll}(x, y)$. This strategy reduces constituency-effects in cophonologies to morpheme-specific rules (Pater 2007, 2009; Finley 2010). In this analysis, we no longer need the SETTINGS encapsulation to factorize the long-distance trigger from the target. However, without an adequate cross-linguistic typology of stratal or cyclic phonological rules, it is unclear if this strategy would always work or always be local (cf. Inkelas and Zoll 2007; Inkelas 2008).

8.3   Locality and non-locality in prosodic parsing

The above discussion showed that non-local information can sometimes be avoided in defining the morphological trigger for phonological rule domains. Similar problems and solutions arise in prosody. I focus on prosodic mappings, not syllabification.[2]

In Chapters §5-6, I showed that a host of prosodic processes are computationally local. I list below all the different prosodic processes, the properties of the topmost morphological node (MNode) (= the trigger), the prosodic change (= the target), and where I formalized the process. All of these processes are computationally local *given* the analytical assumptions that I made. These assumptions were that i) the derivation was cyclic, and ii) we had a constant SETTINGS which encapsulated the properties of the topmost MNode.

---

[2]Syllabification is local regardless of cyclicity or SETTINGS encapsulation. In the formalizations of syllabification (Chapter 5: §5.4.1) and resyllabification (Chapter 6: §6.4), no reference was made to morphological constituents like MStems or MWords. Syllabification around prefixation is sensitive to the left-boundary of roots. I did not discuss those. However, they only need to reference morpheme boundaries.

(624)   *Prosodic processes which are local with cyclicity and the* SETTINGS

Prosodic processes in...	Trigger: Topmost MNode is...	Target or Prosodic Change	Section
Simplex Stem Prosodic Generation	Unparsed MStem which dominates an MRoot	MStem matched to new PStem	Chapter 5: §5.4.3
Derivation Prosodic Restructuring	Unparsed MStem which dominates a parsed MStem	MStem matched with old PStem Old MStem wrapped into new PStem	Chapter 6: §6.5.1.1
Prosodic Recursion	Unparsed MStem which dominates a parsed MStem	MStem matched with new PStem	Chapter 6: §6.5.1.2.1
Prosodic Flattening	Parsed MStem which dominates a parsed MStem	PStems merge	Chapter 6: §6.5.1.2.2
Inflection Prosodic Misalignment	Unparsed MWord which dominates a parsed MStem which is a contracted PStem	PStem is expanded	Chapter 6: §6.5.2.1
Prosodic Layering	Unparsed MWord which dominates a parsed MStem	MWord matched with new MStem PWord dominates PStem	Chapter 6: §6.5.2.2
Compounding PStem linearization	Unparsed compound MStem$_C$ which dominates two parsed MStems	Two PStems are linearized	Chapter 6: §6.5.3.3
Prosodic Subsumption	Unparsed endocentric compound MStem$_C$ which dominates two parsed MStems	MStem$_C$ subsumed into rightmost PStem	Chapter 6: §6.5.3.4
Prosodic Fusion	Unparsed exocentric compound MStem$_C$ which dominates two parsed MStems	MStem$_C$ matched with new PStem Old PStems fused into new PStem	Chapter 6: §6.5.3.5

In this section I show that these prosodic processes are *still* computationally local if we use cyclicity *without* SETTINGS encapsulation (§8.3.1). Furthermore, if we used neither cyclicity nor encapsulation, then most of these processes are still local (§8.3.2). Specifically, the post-cyclic prosodic parse of non-compounds is computationally local (§8.3.2.1), while the post-cyclic parse of compounds is not local (§8.3.2.2).

8.3.1   Locality of cyclic prosody without the SETTINGS

If the derivation is still cyclic but we don't use SETTINGS encapsulation, then prosodic mapping is still local. This is because of the following. The SETTINGS encapsulated information about the topmost MNode. However, the prosodic change involves either creating a prosodic node (PNode) as the output correspondent of the topmost MNode, or modifying a PNode which is matched with an immediate dominee of the topmost MNode. Thus, the properties of the topmost Node are still accessible to the relevant PNode.

I illustrate the above alternative analysis by reformulating Prosodic Restructuring as in a derivative //(amusi)$_s$-agan// (Chapter 6: §6.5.1.1). In this process, the PStem expands because the morphology added an overt derivational suffix: //(amusi-agan)$_s$//. In the simple input-output representations below, the height of the PStem shows which MStem it is matched with.

(625)   *Input and output for prosodic restructuring in* $//(\text{amusi})\text{n-agan}//$
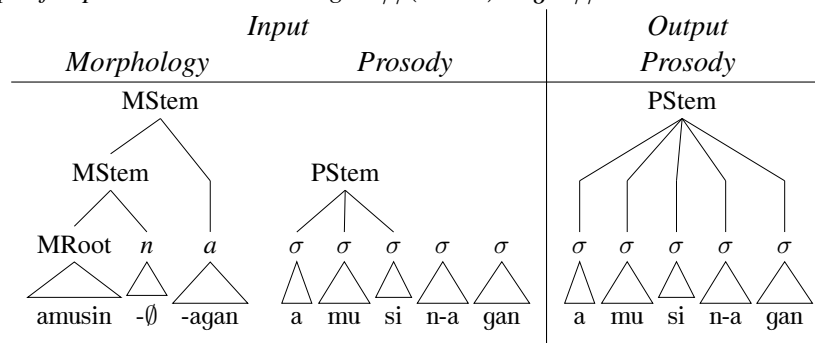
In terms of its input, the morphology contains an unparsed MStem which dominates a parsed MStem. Because they are both MStems, the higher MStem recursively dominates the lower MStem. Prosodically, the lower MStem is matched with a PStem. As for the output, the lower MStem's PStem will expand and incorporate the higher MStems' segments. The higher MStem is matched with this PStem, while the lower MStem is wrapped into it.

I show the explicit input. There is no SETTINGS constant. The high unparsed MStem is $\text{MStem}_{0.22}$; it dominates the lower $\text{MStem}_{0.11}$ which is matched with the $\text{PStem}_{0.15}$.

(626)   *Input to prosodic restructuring without the settings*: $//(\text{amusi})_s\text{n-agan}//$
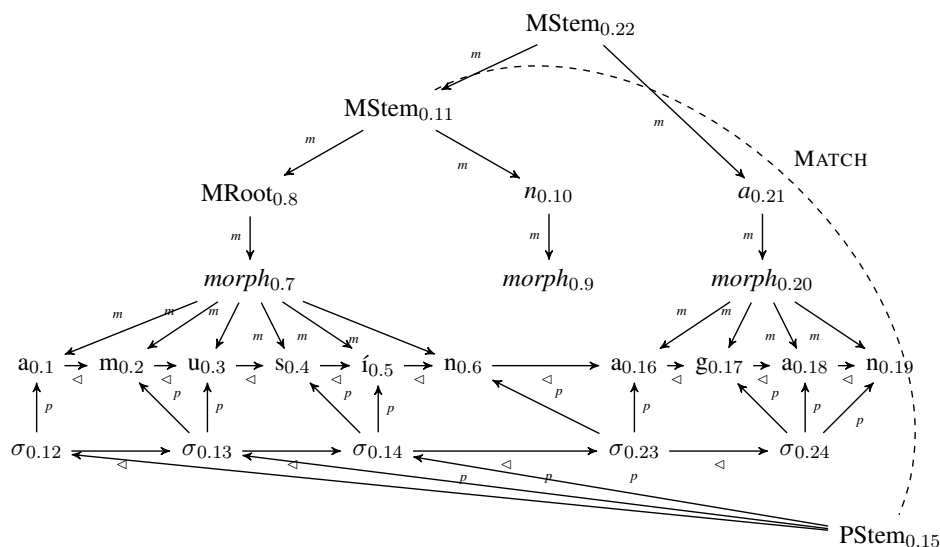
In the output, the only changes are that i) $\text{PStem}_{1.15}$ is now matched with the high $\text{MStem}_{1.22}$, ii) $\text{PStem}_{1.15}$ has the low $\text{MStem}_{1.11}$ wrapped into it, and iii) $\text{PStem}_{1.15}$ dominates the high $\text{MStem}_{1.22}$'s syllables

(627)   *Output of prosodic restructuring without the settings:* $//(\text{amus\'in-agan})_s//$
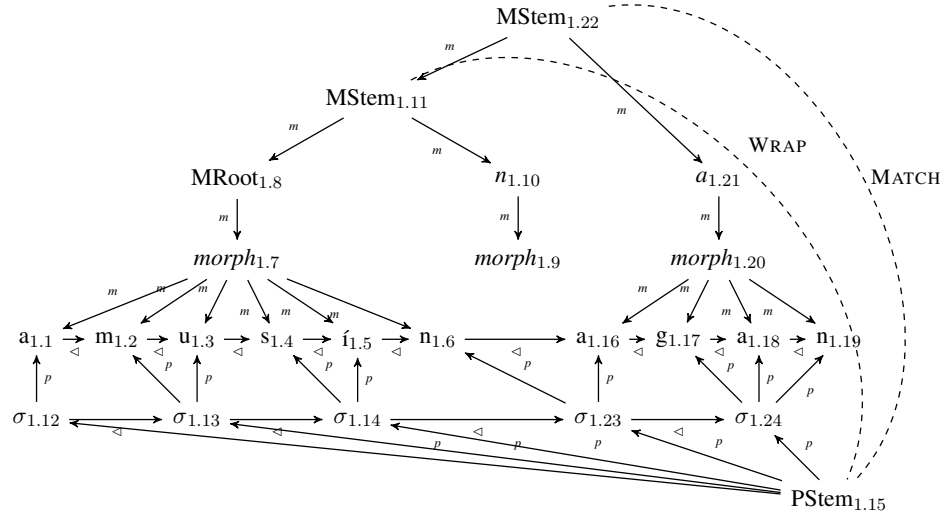
As explained in Chapter 6: §6.5.1.1, the following predicates pick out the unparsed higher $\text{MStem}_{0.22}$ and the expanding $\text{PStem}_{0.15}$. I add the predicate **MStem:parsed** to pick the lower parsed $\text{MStem}_{0.11}$. Their interpretation is based on their local context, i.e., what they dominate and/or match with. In fact, the 3 predicates all reference the same 3 nodes: $\text{MStem}_{0.11}$, $\text{MStem}_{0.22}$, and $\text{PStem}_{0.15}$. All these predicates are locally-computible.

(628)   a.   *FO user-defined predicates for finding the higher MStem, expanding PStem, and lower MStem*
- **MStem:unparsed**$(x) \stackrel{\text{def}}{=} \text{MStem}(x) \wedge \textbf{MTopmost}(x) \wedge \neg \exists y [\text{PStem}(y) \wedge \text{Match:stem}(x,y)]$
$$\exists u, v[\text{MStem}(u) \wedge \text{MDom}(x,u) \wedge \text{PStem}(v) \wedge \text{Match:stem}(u,v)]$$
- **PStem:expanding**$(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge$
$$\exists y, z[\text{Match:stem}(y,x) \wedge \textbf{MStem:unparsed}(z) \wedge$$
$$\text{MDom}(z,y)] \wedge$$
$$\neg \exists y[\text{PStem}(y) \wedge \text{succ:PStem}(x,y)]$$
- **MStem:parsed**$(x) \stackrel{\text{def}}{=} \text{MStem}(x) \wedge \exists y[\text{PStem}(y) \wedge \text{Match:stem}(x,y)] \wedge$
$$\exists z[\text{MStem}(z) \wedge \text{MDom}(z,x) \wedge \textbf{MStem:unparsed}]$$

   b.   *QF user-defined predicates for finding the higher MStem, expanding PStem, and lower MStem*
- **MStem:unparsed**$(x) \stackrel{\text{def}}{=} \text{MStem}(x) \wedge \textbf{MTopmost}(x) \wedge \text{F}_\text{L}\text{:Match:stem}(x) = \textsc{null}$
$$\text{MStem}(\text{F}_\text{M}\text{:MDom}(x)) \wedge$$
$$\text{F}_\text{L}\text{:Match:stem}(\text{F}_\text{M}\text{:MDom}(x)) \neq \textsc{null}$$
- **PStem:expanding**$(x) \stackrel{\text{def}}{=} \text{PStem}(x) \wedge$
$$\textbf{MStem:unparsed}(\text{F}_\text{D}\text{:MDom}(\text{F}_\text{R}\text{:Match:stem}(x)))$$
$$\text{F}_\text{L}\text{:succ:PStem}(x) = \textsc{null}$$
- **MStem:parsed**$(x) \stackrel{\text{def}}{=} \text{MStem}(x) \wedge \text{F}_\text{L}\text{:Match:stem}(x) \neq \textsc{null} \wedge$
$$\textbf{MStem:unparsed}(\text{F}_\text{D}\text{:MDom}(x))$$

In the original formulation from Chapter 6: §6.5.1.1, these steps reference the SETTINGS label Parse:MStem:recursive. For example, in Copy 1, we faithfully output all nodes and all relations except for those involving PStems.

The original formulation referenced Parse:MStem:recursive. In the reformulation, this statement is removed.

(629)   *Faithfully outputting labels and relations except for those involving PStems*

   a.   For every label lab $\in L$:
   - Original
     $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS) $\wedge$ lab(x)
   - Reformulated
     $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=}$ lab(x)

   b.   For every relation rel $\in R - \{$Match:stem, Wrap:stem, PDom:PStem_syll,
                                              PDom:PStem_PStem, PDom:PWord_PStem$\}$:
   - Original
     $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS) $\wedge$ rel(x, y)
   - Reformulated
     $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=}$ rel(x, y)

In Copy 1, the expanding PStem y^1 (y is '0.15') should be matched with the higher unparsed MStem x^1 (x is '0.22'). This is encoded in the original output function. In the reformulated version, we simply remove the statement Parse:MStem:recursive(...). The predicate **MStem:unparsed**(x) already picks out the MStem$_{0.22}$ as a topmost unparsed MStem which dominates a parsed MStem, while **PStem:expanding**(y) picks out that PStem.

(630)   *QF output function for matching the expanding PStem with the higher unparsed MStem*
   - Original
     $\phi\texttt{Match:stem}(x^1, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS) $\wedge$ MStem$(x) \wedge$ PStem$(y) \wedge$
                                 MStem:unparsed$(x) \wedge$ **PStem:expanding**(y)
   - Reformulated
     $\phi\texttt{Match:stem}(x^1, y^1) \stackrel{\text{def}}{=}$ MStem$(x) \wedge$ PStem$(y) \wedge$
                                 MStem:unparsed$(x) \wedge$ **PStem:expanding**(y)

The lower MStem x^1 (x is '0.11') is associated with the expanding PStem y^1 (y is '0.15') via a wrapping relationship. The original function references the SETTINGS. The reformulated version does not. The predicate **PStem:expanding**(y) picks a PStem$_{0.15}$ which was underlyingly matched with a dominated MStem$_{0.11}$ (x).

(631)   *QF output function for wrapping the lower MStem into the expanding PStem*
   - Original
     $\phi\texttt{Wrap:stem}(x^1, y^1) \stackrel{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS) $\wedge$ MStem$(x) \wedge$ PStem$(y) \wedge$
                                 PStem:expanding$(y) \wedge$ Match:stem(x, y)
   - Reformulated
     $\phi\texttt{Wrap:stem}(x^1, y^1) \stackrel{\text{def}}{=}$ MStem$(x) \wedge$ PStem$(y) \wedge$ **PStem:expanding**$(y) \wedge$ Match:stem(x, y)

The expanding PStem must incorporate its old input syllables and newly incorporate the suffix's syllables. The predicate **syll_of_MStem** finds the local syllables of an MStem, i.e., the suffix syllables for the higher MStem$_{0.22}$. This predicate is locally computed.

(632)   a. *FO user-defined predicates for finding an MStem's closest syllables*

- **syll_of_MStem**$(x, y) \overset{\text{def}}{=}$ syll$(x) \wedge$ MStem$(y) \wedge$
  $\exists u, v, w[$morpheme$(u) \wedge$ morph$(v) \wedge$ seg$(w) \wedge$
  MDom$(y, u) \wedge$MDom$(u, v) \wedge$MDom$(v, w) \wedge$PDom:syll_nuc$(x, w)]$

     b. *QF user-defined predicates for finding an MStem's closest syllables*

- **syll_of_MStem**$(x, y) \overset{\text{def}}{=}$ syll$(x) \wedge$ MStem$(y) \wedge$
  seg$(\mathsf{F_M}$:PDom:syll_nuc$(x)) \wedge$
  morph$(\mathsf{F_D}$:MDom$(\mathsf{F_M}$:PDom:syll_nuc$(x))) \wedge$
  morpheme$(\mathsf{F_D}$:MDom$^2(\mathsf{F_M}$:PDom:syll_nuc$(x))) \wedge$
  $y = \mathsf{F}$:$_\mathsf{D}$:MDom$^3(\mathsf{F_M}$:PDom:syll_nuc$(x))$

Syllable incorporation was formalized with two helper predicates. The first predicate allows the expanding PStem to faithfully dominate the syllables which it had in the input: $\sigma_{0.12}, \sigma_{0.13}, \sigma_{0.14}$. The reformulated version removes the reference to the SETTINGS; instead we check that the PStem x is the expanding PStem.

(633)   *QF helper predicates for letting a PStem dominates its underlying syllables*

- Original
  should__PDom:PStem_syll__old$(x, y) \overset{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS)$\wedge$
              PStem$(x) \wedge$ syll$(y) \wedge$ PDom:PStem_syll(x, y)
- Reformulated
  should__PDom:PStem_syll__old$(x, y) \overset{\text{def}}{=}$ PStem$(x) \wedge$ syll$(y) \wedge$
              PStem:expanding$(x) \wedge$ PDom:PStem_syll(x, y)

The second helper predicate will let the expanding PStem incorporate the suffix's syllables $\sigma_{0.23}, \sigma_{0.24}$. The reformulated version simply removes reference to the SETTINGS. The formula already picks out an expanding PStem x. This predicate is QF-definable.

(634)   a. *FO helper predicates for letting the expanding PStem dominate the syllables of the new higher MStem's suffixes*

- Original
  should__PDom:PStem_syll__new$(x, y) \overset{\text{def}}{=}$ Parse:MStem:recursive(SETTINGS)$\wedge$
          PStem$(x) \wedge$ syll$(y) \wedge$ **PStem:expanding**$(x) \wedge$
          $\exists z[$**MStem:unparsed**$(z) \wedge$ **syll_of_MStem**(y, z)
- Reformulated
  should__PDom:PStem_syll__new$(x, y) \overset{\text{def}}{=}$
          PStem$(x) \wedge$ syll$(y) \wedge$ **PStem:expanding**$(x) \wedge$
          $\exists z[$**MStem:unparsed**$(z) \wedge$ **syll_of_MStem**(y, z)

     b. *QF helper predicate for letting the expanding PStem dominate the syllables of the new higher MStem's suffixes*

- **should__PDom:PStem_syll__new**$(x, y) \overset{\text{def}}{=}$
          PStem$(x) \wedge$ syll$(y) \wedge$ **PStem:expanding**$(x) \wedge$
          MStem:unparsed$(\mathsf{F_D}$:MDom$(\mathsf{F_R}$:Match:stem$(x))) \wedge$
          syll_of_MStem$(y, \mathsf{F_D}$:MDom$(\mathsf{F_R}$:Match:stem$(x)))$

These helper predicates are used for the output function below which outputs all correct prosodic dominances from PStems to syllables.

(635) *QF output function to make higher PStem dominate the syllables of its two MStems*
   - ϕPDom:PStem_syll$(x^1, y^1) \overset{\text{def}}{=}$ **should__PDom:PStem_syll__old**$(x, y) \lor$
     should__PDom:PStem_syll__new(x, y)

This completes the *local* generation of a larger PStem from a recursive MStem via prosodic restructuring. I emphasize that reformulated version of prosodic restructuring was *still* local. This is all because the prosodic changes referenced the same MStems and PStem which were in a local relationship with each other. I do not formalize the other prosodic processes. They are likewise local without the SETTINGS because of the local relationships between the morphological triggers and the prosodic targets.

Thus, prosodic parsing is local when cyclic. The SETTINGS does not help improve locality. It does however make it easier to organize formulas in prosodic parsing.

8.3.2 Locality and non-locality is post-cyclic prosodic parses

Because of how cyclicity generates only one new MStem/MWord at a time, the prosodic parse is local. But if the derivation were non-cyclic, then the prosodic parse *could* be non-local. A non-cyclic derivation means that the entire morphological structure is given as an input without any prosodic constituents. All PNodes are generated at once. However, non-locality depends on whether the output could contain more than one PStem or PWord. For simplicity, I assume that the input is already syllabified. I show that the post-cyclic parse of non-compounds is local (§8.3.2.1), while it is non-local for compounds (§8.3.2.2).

8.3.2.1 Post-cyclic parsing is local in non-compounds

If the input is a non-compound word, parsing this item without cyclicity is surprisingly still local. It is computationally local because the prosody will create only a single PStem and PWord. I illustrate with the word *kord͡z-avor-ner-ov* which contains a root, overt derivational suffix, and two overt inflectional suffixes.

(636) a. kord͡z   'work'
       b. kord͡z-avor   'worker'
       c. kord͡z-avor-ner   'workers'
       d. kord͡z-avor-ner-ov 'with workers'

Morphologically, the word contains two layers of MStems. These MStems are under two layers of MWords. The higher MStem is parsed to (matched with) a PStem, while the higher MWord is parsed to (matched with) a PWord. The lower MStem and lower MWord are wrapped into the corresponding PStem and PWord. The trees below illustrate this. I visualize these types of prosodic relations below with subscripts.

(637) *Post-cyclic prosodic mapping of a non-compound word* $((\text{kordz-avor})_s\text{-ner-óv})_w$

Input Morphology | Output Prosody

MWord$_{matched}$

MWord$_{wrapped}$

MStem$_{matched}$

MStem$_{wrapped}$

MRoot   *n*   *n*   PL   INST

morph   *morph*   *morph*   *morph*   *morph*

kordz   ∅   -avor   -ner   -óv

PWord

PStem

σ   σ   σ   σ   σ

kor   dz-a   vor   -ne   r-ov
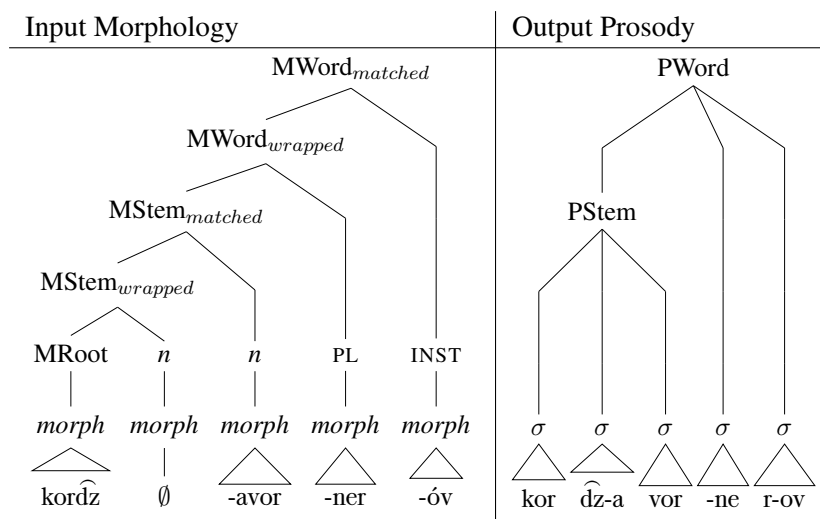
In a post-cyclic prosodic parse, we do not need global information to parse a non-compound. In the output, we always have exactly one PStem and one PWord. For the morphology, exactly one MStem is recursively undominated; meaning that it isn't dominated by a MStem. This MStem is matched with the output's PStem. Every other MStem in the input is recursively dominated by an MStem; in the output, it is wrapped into the output's PStem. The same match-wrap distinctions occur for the MWord layers.

The transduction for post-cyclic parsing requires a copy set of size 2. I show that this transduction is computationally local. I do not show the explicit input and output. Because of their size, they're not easily readable. The compact representations below are sufficient. I assume that the input is already syllabified. In the output, the morphology, segments, and syllables are generated in Copy 1, while the prosodic constituents are generated in Copy 2. I show indexes for the relevant nodes; the subscripts mark prosodic associations.

(638)   a.   *Partially-explicit input for post-cyclic prosody parse of a non-compound word* //kordz-avor-ner-ov//

Input | Morphology | Prosody

MWord$_{0.27}$

MWord$_{0.22}$

MStem$_{0.16}$

MStem$_{0.9}$

MRoot   *n*   *n*   PL   INST

morph   *morph*   *morph*   *morph*   *morph*

kordz   ∅   -avor   -ner   -óv

$\sigma_{0.28}$ $\sigma_{0.29}$ $\sigma_{0.30}$ $\sigma_{0.31}$ $\sigma_{0.32}$
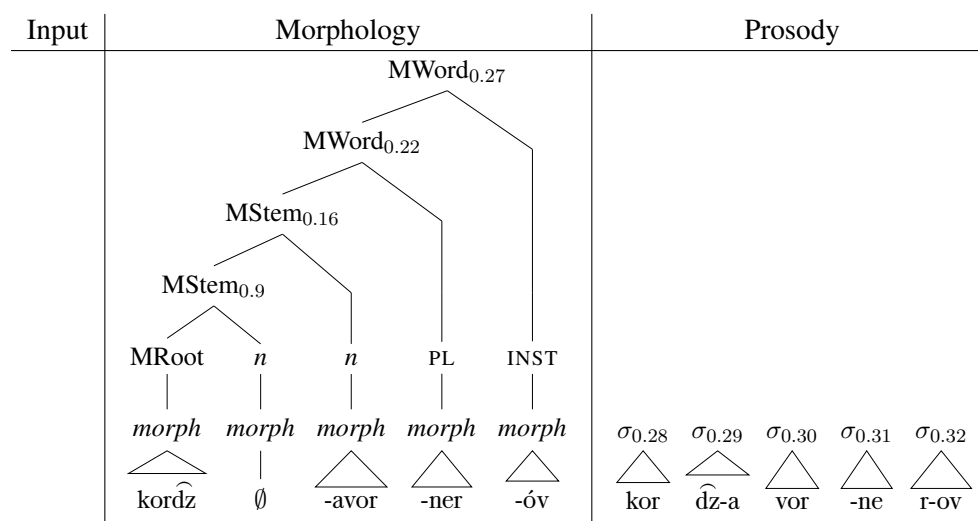
kor   dz-a   vor   -ne   r-ov

b. *Partially-explicit output for post-cyclic prosody parse of a non-compound word* $((\text{kordz-avor})_s\text{-ner-ov})_w$
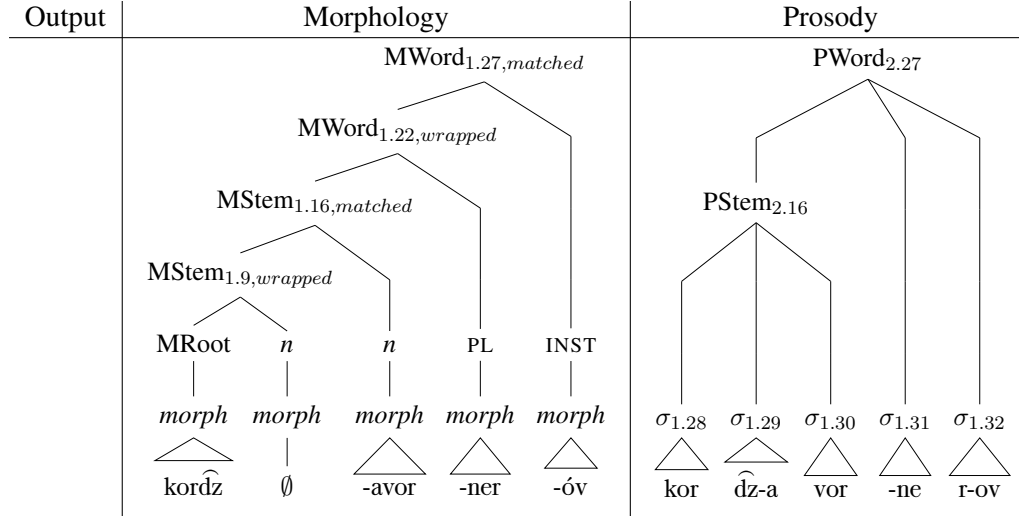
In Copy 1, all segments, syllables, and MNodes are faithfully outputted. There are no underlying PStems or PWords.

(639) *QF output functions for faithfully outputting labels and relations in Copy 1*

- For every label $\text{lab} \in L$:
  $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \text{lab}(x)$
- For every relation $\text{rel} \in R$:
  $\phi\texttt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \text{rel}(x, y)$

The predicates below check if some MStem or MWord is recursively dominated or not. They are locally-computed and QF-definable. An MStem (MWord) is recursively dominated if it is morphologically dominated by another MStem (MWord). It is recursively undominated otherwise. For *kord̂z-avor-ner-ov*, $\text{MStem}_{0.9}$ and $\text{MWord}_{0.22}$ are recursively dominated. These introduce the zero *n*-suffix and the inflectional suffix *-ner*. $\text{MStem}_{0.16}$ and $\text{MWord}_{0.27}$ are recursively undominated; they introduce the derivational suffix *-avor* and inflectional suffix *-ner*. These two are the rightmost derivational and inflectional suffix respectively.

(640) a. *FO user-defined predicates for finding a recursively dominated MStem/MWord*

- **MStem:rec_dominated**$(x) \stackrel{\text{def}}{=} \text{MStem} \wedge \exists y[\text{MStem}(y) \wedge \text{MDom}(y, x)]$
- **MWord:rec_dominated**$(x) \stackrel{\text{def}}{=} \text{MWord} \wedge \exists y[\text{MWord}(y) \wedge \text{MDom}(y, x)]$

b. *QF user-defined predicates for finding a recursively dominated MStem/MWord*

- **MStem:rec_dominated**$(x) \stackrel{\text{def}}{=} \text{MStem} \wedge \text{MStem}(\text{F}_\text{D}:\text{MDom}(x))$
- **MWord:rec_dominated**$(x) \stackrel{\text{def}}{=} \text{MWord} \wedge \text{MWord}(\text{F}_\text{D}:\text{MDom}(x))$

c. *QF user-defined predicates for finding a recursively undominated MStem/MWord*

- **MStem:rec_undominated**$(x) \stackrel{\text{def}}{=} \text{MStem} \wedge \neg\textbf{MStem:rec\_dominated}(x)$
- **MWord:rec_undominated**$(x) \stackrel{\text{def}}{=} \text{MWord} \wedge \neg\textbf{MWord:rec\_dominated}(x)$

In Copy 2, we generate $\text{PStem}_{2.16}$ and $\text{PWord}_{2.27}$ as output correspondents of the recursively undominated $\text{MStem}_{0.16}$ and $\text{MWord}_{0.27}$. If the input is a non-compound, there will be exactly one recursively undominated MStem or MWord.

(641) *QF output function to post-cyclically generate a PStem (PWord) for a recursively undominated MStem (MWord)*
- $\phi\texttt{PStem}(x^2) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathbf{MStem\text{:}rec\_undominated}(x)$
- $\phi\texttt{PWord}(x^2) \stackrel{\text{def}}{=} \mathsf{MWord}(x) \wedge \mathbf{MWord\text{:}rec\_undominated}(x)$

The undominated $\text{MStem}_{1.16}$ and $\text{MWord}_{1.27}$ will be matched with their own $\text{PStem}_{2.16}$ and $\text{PWord}_{2.27}$. The nodes x (for the undominated MNode) and y (its generated PNode) must both be underlyingly the same MNode ($x = y$).

(642) *QF output functions to match undominated MStem (MWord) with the output PStem (PWord)*
- $\phi\texttt{Match:stem}(x^1, y^2) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathbf{MStem\text{:}rec\_undominated}(x) \wedge x = y$
- $\phi\texttt{Match:word}(x^1, y^2) \stackrel{\text{def}}{=} \mathsf{MWord}(x) \wedge \mathbf{MWord\text{:}rec\_undominated}(x) \wedge x = y$

In the output, there is only a single $\text{PStem}_{2.16}$ and $\text{PWord}_{2.27}$. The dominated $\text{MStem}_{1.9}$ and $\text{MWord}_{1.24}$ (x) will be wrapped into this PStem and PWord. In the input, all we need to check is if the surface PStem (PWord) was generated from the recursively undominated MStem (PWord) y.

(643) *QF output functions to wrap dominated MStems (MWord) in the PStem (PWord) of the recursively undominated MStem (MWord)*
- $\phi\texttt{Wrap:stem}(x^1, y^2) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathbf{MStem\text{:}rec\_dominated}(x)\wedge$
  $\mathsf{MStem}(y) \wedge \mathbf{MStem\text{:}rec\_undominated}(y)$
- $\phi\texttt{Wrap:word}(x^1, y^2) \stackrel{\text{def}}{=} \mathsf{MWord}(x) \wedge \mathbf{MWord\text{:}rec\_dominated}(x)\wedge$
  $\mathsf{MWord}(y) \wedge \mathbf{MWord\text{:}rec\_undominated}(y)$

The generated PStem (PWord) must dominate the syllables of all MStems (MWords). These syllables are found by the predicates below. They check if a syllable x is close to some MStem or MWord node t, i.e., syllables whose nuclei belong to morphemes which are immediately dominated by the MStem/MWord. These predicates are locally-computible.

(644) a. *FO user-defined predicates for finding syllables which are part of an MStem or MWord*
- $\mathbf{syll\_of\_any\_MStem}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge$
  $\exists t, u, v, w[\mathsf{MStem}(t) \wedge \mathsf{morpheme}(u) \wedge \mathsf{morph}(v) \wedge \mathsf{seg}(w) \wedge$
  $\mathsf{MDom}(t,u) \wedge \mathsf{MDom}(u,v) \wedge \mathsf{MDom}(v,w) \wedge \mathsf{PDom\text{:}syll\_nuc}(x,w)]$
- $\mathbf{syll\_of\_any\_MWord}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge$
  $\exists t, u, v, w[\mathsf{MWord}(t) \wedge \mathsf{morpheme}(u) \wedge \mathsf{morph}(v) \wedge \mathsf{seg}(w) \wedge$
  $\mathsf{MDom}(t,u) \wedge \mathsf{MDom}(u,v) \wedge \mathsf{MDom}(v,w) \wedge \mathsf{PDom\text{:}syll\_nuc}(x,w)]$

b. *QF user-defined predicates for finding syllables which are part of an MStem or MWord*
- $\mathbf{syll\_of\_any\_MStem}(x) \stackrel{\text{def}}{=} \mathsf{syll}(x)\wedge$
  $\mathsf{seg}(\mathsf{F_M\text{:}PDom\text{:}syll\_nuc}(x))\wedge$

$$\text{morph}(\mathsf{F_D}{:}\mathsf{MDom}(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x))) \wedge$$
$$\text{morpheme}(\mathsf{F_D}{:}\mathsf{MDom}^2(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x))) \wedge$$
$$\mathsf{MStem}(\mathsf{F}{:}_\mathsf{D}{:}\mathsf{MDom}^3(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x)))$$

- **syll_of_any_MWord**$(x) \overset{\text{def}}{=} \mathsf{syll}(x) \wedge$
$$\mathsf{seg}(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x)) \wedge$$
$$\text{morph}(\mathsf{F_D}{:}\mathsf{MDom}(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x))) \wedge$$
$$\text{morpheme}(\mathsf{F_D}{:}\mathsf{MDom}^2(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x))) \wedge$$
$$\mathsf{MWord}(\mathsf{F}{:}_\mathsf{D}{:}\mathsf{MDom}^3(\mathsf{F_M}{:}\mathsf{PDom}{:}\mathsf{syll\_nuc}(x)))$$

For example, the syllables of the lower MStem$_{0.9}$ are $\sigma_{0.28}$ *.kor.*. The syllables of the higher MStem$_{0.16}$ are $\sigma_{0.29}$ *.d̂za.*, $\sigma_{0.30}$ *.vor.*. The syllables of the lower MWord$_{0.22}$ are $\sigma_{0.31}$ *.ne.*. And the syllables of the higher MWord$_{0.27}$ are $\sigma_{0.32}$ *.rov.*

The output contains only one PStem and PWord. The PStem (PWord) will dominate all the syllables which belong to any MStem (MWord).[3] The computation is local; the syllables are within a finite bound from their MStems.

(645)   *QF output function to let the PStem (PWord) dominate the syllables of MStems (MWords)*

- $\phi\mathtt{PDom{:}PStem\_syll}(x^2, y^1) \overset{\text{def}}{=} \phi\mathtt{PStem}(x^2) \wedge \mathsf{syll}(y) \wedge \textbf{syll\_of\_any\_MStem}(y)$
  equivalent to
  $\phi\mathtt{PDom{:}PStem\_syll}(x^2, y^1) \overset{\text{def}}{=} \textbf{MStem{:}rec\_undominated}(x) \wedge \mathsf{syll}(y) \wedge$
  $$\textbf{syll\_of\_any\_MStem}(y)$$
- $\phi\mathtt{PDom{:}PWord\_syll}(x^2, y^1) \overset{\text{def}}{=} \phi\mathtt{PWord}(x^2) \wedge \mathsf{syll}(y) \wedge \textbf{syll\_of\_any\_MWord}(y)$
  equivalent to
  $\phi\mathtt{PDom{:}PWord\_syll}(x^2, y^1) \overset{\text{def}}{=} \textbf{NWord{:}rec\_undominated}(x) \wedge \mathsf{syll}(y) \wedge$
  $$\textbf{syll\_of\_any\_MWord}(y)$$

Lastly, the PWord must dominate the PStem. We do not need to check that the PWord's MWord generally dominates the PStem's MStem. That requires long-distance computation and is unneeded. The output only has one PStem and PWord which were generated from the recursively undominated MStem and MWord.

(646)   *QF output function to let the PWord dominate the PStem*

- $\phi\mathtt{PDom{:}PWord\_PStem}(x^2, y^2) \overset{\text{def}}{=} \phi\mathtt{PWord}(x^2) \wedge \phi\mathtt{PStem}(y^2)$
  equivalent to
  $\phi\mathtt{PDom{:}PWord\_PStem}(x^2, y^2) \overset{\text{def}}{=} \textbf{MWord{:}rec\_undominated}(x) \wedge$
  $$\textbf{MStem{:}rec\_undominated}(y)$$

I emphasize that this entire computation was local. This is because the input contained only one recursively undominated MStem and only one recursively undominated MWord. Thus, the output would only have one PStem and one PWord. This uniqueness lets the post-cyclic parsing be local.

---

[3]These functions must be modified in order to trigger PStem expansion before V-initial inflection. One possibility is to define a separate PStem expansion transduction which follows post-cyclic parsing.

8.3.2.2 Post-cyclic parsing is non-local in compounds

In contrast to non-compound words, if the output *does* contain more than one PWord or PWord, then post-cyclic parsing is not local. This is because in compounds, we need to reference long-distance information like general morphological dominance in order to know if some MStem and its syllables are generally dominated by the left vs. right stem in the compound.

Consider the nonce word below. It is an endocentric compound. I set aside the linking vowel. The topmost node is an *MStem$_C$*. It is made up of a *morphological concatenation* node or MConc. This MConc consists of two internally complex or recursive MStems: *MStem$_L$* and *MStem$_R$*. These two higher MStems map to two PNodes. The MStems dominate a long chain of MStems. Specifically, *MStem$_L$* (*MStem$_R$*) generally dominates a chain of MStems from *MStem$_A$* (*MStem$_D$*) to *MStem$_L$* (*MStem$_R$*), including *MStem$_B$* (*MStem$_E$*). These lower MStems are wrapped into the compound's two PNodes. I show the prosodic associations via subscripts.

(647) *Post-cyclic parsing of a large endocentric compound into two PWords*
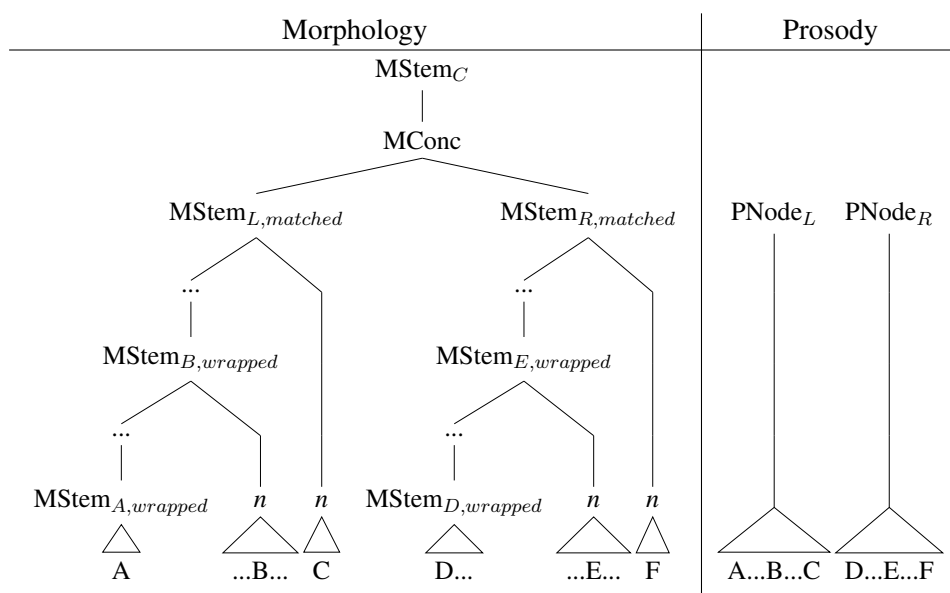
Such complex compounds are rare in Armenian, but they can be found in English: *origin-al-ity sens-itiv-ity* (= being sensitive to originality). English compounds can freely combine any two nouns, even if the two nouns are internally complex and have multiple derivational suffixes. For English, the compound *origin-al-ity sens-itiv-ity* would be parsed into two separate PWords *(origin-al-ity)$_w$ (sens-itiv-ity)$_w$*.

(648)   a. *Partially explicit input and output for post-cyclically parsing a compound* origin-al-ity sens-itiv-ity

	Morphology	Prosody
Input		
Output		

The post-cyclic parsing of a compound is a transduction with a copy set of size 2. I illustrate this with the English example.

In Copy 2, *MStem_L* and *MStem_R* must be parsed into PWord$_L$ and PWord$_R$. These two MStems are marked as recursively undominated. They are not *immediately* dominated by an MStem$_C$. They must be matched with their own PWord. All the lower MStems are recursively dominated. The output functions below reference the predicate **MStem:rec_undominated**(x) from the previous section.

(649)   a. *QF output function to generate PWords from recursively undominated MStems*
  - $\phi\mathtt{PWord}(x^2) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \textbf{MSem:rec\_undominated}$

   b. *QF output function to match undominated MStems with PWords*
  - $\phi\mathtt{Match:stem}(x^1, y^2) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \textbf{MSem:rec\_undominated} \wedge (x = y)$

The problem is syllable incorporation. The predicate **syll_of_MStem**(x, y) selects the syllables x whose nuclei are dominated by an MStem y's morpheme. For example, the syllables of MStem$_L$ are the syllables *.li.ty*, while the syllables of MStem$_R$ are *.vi.ty*. These syllables are local to the undominated MStems; *.li.ty* and *.vi.ty* are dominated by the suffixes *-ity* and *-ity* which are immediately dominated by MStem$_L$ and MStem$_R$. This predicate is locally-computed and QF-definable.

(650)   a. *FO user-defined predicate for finding syllables of an MStem*

- $\mathbf{syll\_of\_MStem}(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{MStem}(y) \wedge$
$\exists u, v, w[\mathsf{morpheme}(u) \wedge \mathsf{morph}(v) \wedge \mathsf{seg}(w) \wedge$
$\mathsf{MDom}(y, u) \wedge \mathsf{MDom}(u, v) \wedge \mathsf{MDom}(v, w) \wedge \mathsf{PDom:syll\_nuc}(x, w)]$

   b. *QF user-defined predicate for finding syllables of an MStem*

- $\mathbf{syll\_of\_MStem}(x, y) \stackrel{\text{def}}{=} \mathsf{syll}(x) \wedge \mathsf{MStem}(y) \wedge$
$\mathsf{seg}(\mathsf{F_M{:}PDom{:}syll\_nuc}(x)) \wedge$
$\mathsf{morph}(\mathsf{F_D{:}MDom}(\mathsf{F_M{:}PDom{:}syll\_nuc}(x))) \wedge$
$\mathsf{morpheme}(\mathsf{F_D{:}MDom}^2(\mathsf{F_M{:}PDom{:}syll\_nuc}(x))) \wedge$
$y = \mathsf{F{:}_D{:}MDom}^3(\mathsf{F_M{:}PDom{:}syll\_nuc}(x))$

Using this predicate, the output function $\phi\mathtt{PDom{:}PWord\_syll}(x, y)$ will cause PWord_L and PWord_R to dominate the syllables which are local to MStem_L and MStem_R, i.e. syllables *.li.ty.* and *.vi.ty.*

(651)   *QF output function to let PWords$_L$ and PWord$_R$ dominate the syllables of MStem$_L$ and MStem$_R$*

- $\phi\mathtt{PDom{:}PWord\_syll}(x^2, y^1) \stackrel{\text{def}}{=} \phi\mathtt{PWord}(x^2) \wedge \phi\mathtt{syll}(y^1) \wedge$
$\mathbf{MStem{:}rec\_undominated}(x) \wedge \mathbf{syll\_of\_MStem}(y, x)$

However, syllable incorporation is incomplete. The syllables of MStem_A are *.o.ri.gi.* These syllables must incorporate into PWord_L because MStem_A is under MStem_L. But this information is non-local to MStem_A. We cannot use immediate morphological dominance to know if the syllables of MStem_A should incorporate to PWord_L vs. PWord_R. We need long-distance or general morphological dominance: we need to check if MStem_A is *generally* or non-immediately dominated by MStem_L.

This non-local information is formalized below in the predicate $\mathbf{MStem{:}dominator\_of}(x, y)$. Given two MStems x, y, this predicate checks if x is the closest recursively undominated MStem which dominates y. That is, the predicate is satisfied by MStem_L with MStem_A, but not MStem_R or MStem_C with MStem_A. The predicate references general or long-distance morphological dominance $\mathbf{gen\_MDom}(x, y)$ which was defined in Chapter 4: §4.5.1 as an MSO predicate. The predicate below essentially creates a tier of MStems based on non-local information.

(652)   *MSO user-defined predicate to find the topmost dominating MStem of a recursively dominated MStem*

- $\mathbf{MStem{:}dominator\_of}(x, y) \stackrel{\text{def}}{=} \mathsf{MStem}(x) \wedge \mathsf{MStem}(y) \wedge$
$\mathbf{MStem{:}rec\_undominated}(x) \wedge \mathbf{MStem{:}rec\_dominated}(y) \wedge$
$\mathbf{gen\_MDom}(x, y) \wedge$
$\neg\exists z[\mathbf{MStem{:}rec\_undominated}(z) \wedge$
$\mathbf{gen\_MDom}(x, z) \wedge \mathbf{gen\_MDom}(z, y)]$

For the English compound, the predicate $\mathbf{MStem{:}dominator\_of}(x, y)$ is satisfied by MStem_L as x, and MStem_A as y. This is because 1) MStem_L is recursively undominated while MStem_A is recursively dominated, 2) MStem_L generally dominates MStem_A, and 3) there is no other recursively undominated MStem z which comes between MStem_L and MStem_A. This last condition ensures that MStem_C is not considered the dominator of MStem_A. This condition essentially creates multiple tiers of MStems based on

morphological dominance similar to how tiers have been constructed for syntax in subregular syntax (Vu et al. 2019; Graf and Shafiei 2019b).

Using this non-local information, the function $\phi\texttt{PDom:PWord\_syll}(x, y)$ must be redefined in order to let PWord$_L$ dominate the syllables *.o.ri.gi.na.*, and let PWord$_R$ dominate syllables *.sen.si.ti.*.

(653) *MSO output function to let PWord$_L$ and PWord$_R$ dominate the syllables of its undominated and dominated MStems*

- $\phi\texttt{PDom:PWord\_syll}(x^2, y^1) \overset{\text{def}}{=} \phi\texttt{PWord}(x^2) \wedge \phi\texttt{syll}(y^1) \wedge$
  MStem:rec_undominated$(x) \wedge$
  [**syll_of_MStem**$(y, x) \vee$
  $\exists z[$**MStem:rec_dominated**$(z) \wedge$**syll_of_MStem**$(y, z) \wedge$
  MStem:dominator_of$(x, z)]]$

The disjunct **syll_of_MStem**(y, x) makes sure that the syllables *.li.ty.* (y) are still dominated by PWord$_L$ (x). The new disjunct $\exists z[...$**MStem:dominator_of**$(x, z)]$ ensures that the syllables *.o.ri.gi.* (y) are also dominated by PWord$_L$ (x). This function will cause PWord$_L$ (x^2) to dominate the syllables *.o.ri.gi.* (y) because PWord$_L$ is defined as an output correspondent for an underlying recursively undominated MStem$_L$ as x, the syllables *.o.ri.gi.* belongs to a recursively dominated MStem$_A$ as z, and MStem$_L$ is the closest dominator of MStem$_A$.

For completeness, I provide below the output functions which will cause the dominated MStems to get wrapped into PWord$_L$ and PWord$_R$. A recursively dominated MStem$_A$ (x^1) is wrapped into PWord$_L$ (y^2) because PWord$_L$ is defined as an output correspondent of a recursively undominated MStem$_L$ (y). MStem$_L$ y is the closest dominator of MStem$_A$. This function references long-distance information, analogous to the above formalization of syllable incorporation.

(654) *MSO output function to let dominated MStems get wrapped into PWords*

- $\phi\texttt{Wrap:stem}(x^1, y^2) \overset{\text{def}}{=} \phi\texttt{MStem}(x^1) \wedge \phi\texttt{PWord}(y^2) \wedge$
  MStem:rec_dominated$(x) \wedge$**MStem:rec_undominated**$(y) \wedge$
  $\wedge$**MStem:dominator_of**(y, x)

To summarize, the post-cyclic parsing of compounds is non-local. We need to reference long-distance information like general morphological dominance in order to know if some MStem and its syllables are generally dominated by the left vs. right stem in the compound. The same problem of non-locality pops up if we want to compute higher levels of prosodic constituents and the syntax-phonology interface, e.g., mapping flat prosodic phrases from recursive syntactic phrases.[4]

---

[4] The non-locality problem cannot be solved by 1) parsing the entire compound into recursive PWords for every MStem and then 2) flattening the recursive PWords into a single PWord. With this strategy, we still need non-local or long-distance information to see if any two PWords should be flattened into the same new PWord or not. Intuitively, generating recursive PWords just involves replacing the node label of MStems for PWords. The problem of just outputting the two correct PWords still remains.

8.4 Conclusion

The main goal of this chapter was to examine the ubiquity of computational locality in the morphology-phonology interface. The result is that, even with fewer restrictions on how the interface is organized, the bulk of the interface is still computationally local.

Chapters 5-6 used two architectural assumptions and formalizations: cyclicity and the SETTINGS constant. With these two concepts, I factorized cophonologies and prosody into their morphophonological trigger and their morphophonological target. Although the trigger could be far from the target, the rest of the process was shown to be computationally local. In this chapter, I showed that morpheme-based definitions of cophonologies *can* be computationally local. Much more interesting was the prosody. Even without the SETTINGS constant, prosodic mapping is computationally local as long as the derivation is cyclic. If the derivation is non-cyclic, parsing a non-compound is local, but parsing a compound is non-local. This is surprising, and it is due to how many unique PNodes we have to generate.

(1) *Post-cyclic parsing of a large endocentric compound into two PWords by first creating recursive PWords*
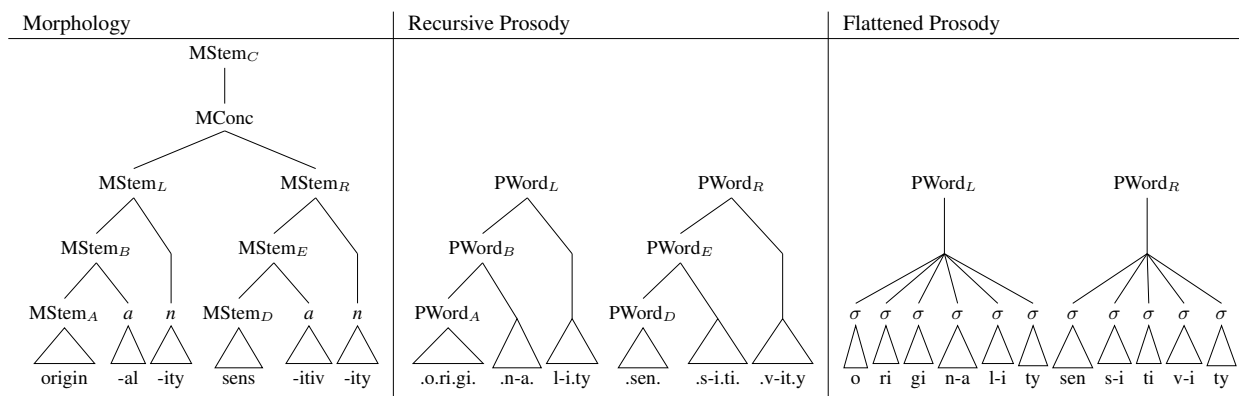
Chapter 9

Computation of cyclicity

9.1 Formal aspects of cyclicity: Overview

This dissertation utilized a cyclic architecture of the morphology-phonology interface. Thanks to cyclicity, a significant chunk of the interface was shown to be computationally local. In this chapter, I show problems inherent to cyclicity. I sketch a plausible solution for them.

In §9.2, I go over problems in cyclicity, both computational and empirical. Computationally, cyclicity is a difficult concept to formalize because it involves potentially unbounded number of interactions or cycles between phonology and morphology. Even if these individual processes are computationally local or simple, applying these processes an unbounded number of times *can* be computationally more complex or take exponential time (Johnson 1972; Ristad 1990; Coleman 1995a). Empirically, the main problem behind cyclicity are cases where a linguistic process acts counter-cyclically by accessing information from future cycles. Two common examples of this problem are post-cyclic phonology and outwards-sensitive allomorphy.

But despite the computational and empirical conundrums, cyclicity is an empirically real phenomenon (Bermúdez-Otero 2011), even though its analytical details can be controversial (Cole 1995a). Cyclic phenomena in *real* natural language patterns do not show computational blowup, implying there may be other constraints which restrict the problem in ways we need to discover. This is similar to the problem of how two-level FST formalizations of morphology, *two-level morphology* (Koskenniemi 1983b,a, 1984), can be computationally complex and NP-hard in theory (Barton 1986; Barton et al. 1987; Wareham 1999) but not in practice (Koskenniemi and Church 1988; Ritchie et al. 1992; Kornai 2009).[1]

Thus in practice, cyclicity seem manageable. I argue that phonological cyclicity is manageable because many factors control how often morphological processes can apply, which limits the impact of 'unboundedness'. Specifically, a given derivation utilizes a predictable number of cycles based on how many morphological processes must apply. These morphological processes are themselves predictable from what operations we choose to do. Informally, the input to the entire derivation should encode the full sequence of these desired operations in order to make cyclicity become computationally feasible.

---

[1] The potential NP-hardness of two-level morphology is independent of the generative capacity of two-level morphology which is still computationally simple and less expressive than rational relations (Ritchie 1989, 1992).

I flesh out the above idea by using an **Operation List** (§9.3). An Operation List is a formalization of the derivational history of a given lexical item. This list provides a way to encode derivational look-ahead and planning. Informally, the Operation List is a linear representation for the morphotactics of a word (cf. Beesley and Karttunen 2003). It acts as a linear approximation for morphosyntactic features (Stump 2001) and for tree-based bottom-up spell-out (Halle and Marantz 1993). I formalize and illustrate the use of the Operation List in planning morphological processes and in accessing derivational look-ahead. I discuss the significance of the Operation List in allowing us to compose the number of cycles (§9.4). I conclude in §9.5.

9.2 Problems in cyclicity

Cyclicity is a cornerstone of phonological theory (Chomsky et al. 1956; Chomsky and Halle 1968; Kiparsky 1982b; Bermúdez-Otero 2011). However, cyclicity has both computational (§9.2.1) and empirical (§9.2.2) shortcomings. The computational problem is about how cyclicity affects the generative capacity or computational complexity of linguistic processes. The empirical problem is about how certain linguistic processes appear counter-cyclic or access information from future morphological operations.

9.2.1 Computational problem of cyclicity

Although cyclicity is a common theoretical concept, there are little to no computational results or implementations of cyclicity (Sproat 1992b:108). Cyclicity is computationally problematic because it requires that there is no a priori bound on how often a rule will apply. However, the lack of a bound means that even though the individual components of a cyclic analysis (the rules) may be computationally simple, the sum total of the composition of these components can be computationally more complex.

A groundbreaking result in computational phonology is that SPE rewrite rules can be converted to 1-way finite-state transducers (Kaplan and Kay 1994). These transducers compute rational functions and have wide utility in Natural Language Processing (Roche and Schabes 1997). Because of this result, the composition of multiple finite number of rules is finite-state definable. However, for an SPE rule to be finite-state definable, Johnson (1972) showed that the rule cannot take its own output as its input. Intuitively, this means that the same rules cannot apply for an unbounded number of cycles. Otherwise, the rules would no longer be finite-state definable, but would get closer to needing a Turing Machine (Coleman 1995a, 1998:77ff) or being computationally undecidable (Ristad 1990). Kaplan and Kay (1994:365) put it nicely as:

> In the worst case, in fact, we know that the computations of an arbitrary Turing machine can be simulated by a rewriting grammar with unrestricted rule reapplication

To illustrate, consider a rule of epenthesis: $\emptyset \rightarrow ab/a\_b$ (cf. Kaplan and Kay 1994). This rule will add the string *ab* between an *a* and a *b*. This rule is definable with FSTs and with FO logic. But given an input string *ab*, if this rule applied an unbounded number of times on its own output, it would generate the context-free language $a^n b^n$. The same problem can be illustrated with unbounded circumfixation (cf. Aksënova et al. 2016). Given an input *X*, adding a circumsuffix *a–b* once is FST-definable and FO-definable. But adding the circumfix an unbounded number of times will generate the context-free language $a^n X b^n$.

In response to this problem, there are two intuitive solutions. One is to abandon the use of the cycle altogether. The other is to place a bound on the number of cycles (Peters and Ritchie 1973; Kaplan and Kay 1994). The first alternative was used in earlier work in One-Level Declarative Phonology (Cole 1990, 1995b; Coleman 1995a; Cole and Coleman 1992). Cyclic rule domains can be replaced with inviolable monostratal constraints in a context-free grammar. These constraints do not generate an output form an input, but they state what is a well-formed word. They have the benefit of being computationally definable and implementable, but they do not faithfully formalize the transformation or *function* behind cycles and strata.

To my knowledge, the second alternative has not be seriously developed because no a priori bound is clear. I instead take a practical intermediate approach. I assume that there is no bound k on the number of cycles for all words w. Instead, I assume that, for a given word w, there is a bound k in *run-time* or in *practice*. In section §9.3, I enrich the input representation in order to encode all future morphological operations. By doing this encoding for derivational look-ahead, we impose a bound on how many cycles we will ultimately use in run-time.

9.2.2 Empirical problems in cyclicity

Alongside its computational problems, cyclicity likewise has empirical problems. I discuss two of them: post-cyclic phonology (§9.2.2.1) and outwards-sensitive allomorphy (§9.2.2.2).

9.2.2.1 Post-cyclic phonology

In this thesis, the phonological processes that I described were *cyclic*, not *postcyclic*. A process is post-cyclic if it must apply after all morphological processes have applied. For example, devoicing is a common postcyclic process. In Dutch, syllable-final codas are devoiced (Booij and Rubach 1987). Devoicing is a postcyclic rule that applies after all word-level morphology is completed.

(655) a. /hɛld/   'hero'   b. /hɛld-in/   'heroine'
           *[.hɛld.]   [hɛl.d-in]   *[hɛl.t-in]
           [.hɛlt.]

To apply final devoicing, we must know that the output will *not* undergo any more morphological operations. This requires a form of derivational look-ahead in order to see if there are any more morphological operations.

9.2.2.2 Outwards-sensitive allomorphy

Besides postcyclic phonology, another empirical problem for cyclicity is outwards-sensitive allomorphy. This is when the choice of allomorph is dependent on properties of *later* morphological operations, not on the current input.[2] This outwards-sensitive allomorphy can be phonologically- or morphologically-conditioned.

---

[2]Some processes can be both inwards- and outwards-sensitive, e.g., the Armenian plural possessive (Arregi et al. 2013; Wolf 2013).

Phonologically-conditioned outwards-sensitive allomorphy is empirically rare (Paster 2006), but attested (Wolf 2008, 2013; Deal and Wolf 2017). For example, the Western Armenian definite suffix displays allomorphy which is conditioned by mostly *just* the input: *-n* if the base ends in a vowel: *agra-n*, *-ə* if the base ends in a consonant: *kar-ə*. However, the vowel-selecting *-n* is also used *before* V-initial clitics: *kar-n=e*. These clitics are structurally higher than the suffix and are added later.

(656)	a.	i.	agra	'tooth'	b.	i.	kar	'rock'
		ii.	agra-n	'tooth-DEF'		ii.	kar-ə	'rock-DEF'
		iii.	agra-n=e	'tooth-DEF-is'		iii.	kar-n=e	'rock-DEF-is'
				= *is the tooth*				= *is the rock*
			agra-n=al	'tooth-DEF-also'			kar-n=al	'rock-DEF-also'
				= *also the tooth*				= *also the rock*

The above allomorphy is thus outwards-sensitive and phonologically conditioned.[3] Outwards-sensitive allomorphy can likewise be morphologically-conditioned, which is empirically much more common (Bobaljik 2000). As previewed in Chapter 7: §7.7.2, Armenian regular verbs form three conjugation classes based on three theme vowels: *e,i,a*. The *-i-* theme vowel shows outwards-sensitive allomorphy in certain morphological contexts. When an infinitival with *-i-* is nominalized with the definite suffix, the theme vowel stays the same: *xos-i-l-ə*. But if the nominalized infinitival takes a case marker, then the *-i-* theme vowel is 'replaced' by the *e* theme vowel: *xos-e-l-ov*. The changed theme vowel is underlined and in bold.

(657)   *Simple paradigm of infinitivals and their nominalizations*

	Root	Infinitival	Nominalization			
			definite	genitive	ablative	instrumental
Features		√-TH-INF	√-TH-INF-DEF	√-TH-INF-GEN	√-TH-INF-ABL	√-TH-INF-INST
Fixed morphs		√-TH-l	√-TH-l-ə	√-TH-l-u	√-TH-l-e	√-TH-l-ov
a. Class E	√ker-	ker-e-l	ker-e-l-ə	ker-e-l-u	ker-e-l-e	ker-e-l-ov
b. Class I	√xos-	xos-i-l	xos-i-l-ə	xos-**e**-l-u	xos-**e**-l-e	xos-**e**-l-ov
a. Class A	√gart-	gart-a-l	gart-a-l-ə	gart-a-l-u	gart-a-l-e	gart-a-l-ov

The Armenian cases of allomorphy are intuitively local because there is a finite bound between the alternating suffix (the definite, the theme vowel) and the trigger morpheme (the case marker, the clitic). This bound is 1 and 2 respectively. However, cases of non-local or long-distant outwards-sensitive allomorphy are argued to exist (cf. Bobaljik 2000).

Formalizing outwards-sensitive allomorphy is difficult in a purely cyclic model like the one I used in this thesis. In the next section, I sketch a solution to this problem.

---

[3]The allomorphy is more extreme in Eastern Armenian where it is subject to phrase-level prosodic phonology. The definite allomorph *-n* is used if the following word is V-initial and not separated by a prosodic break (Dum-Tragut 2009:108). There is little extensive work on what pragmatic and intonational factors affect this allomorphy (Gulakian 1965; Ġaragyowlyan 1974:156; Xačatryan 1988:58; Mkrtčyan 2015).

9.3 Derivational history and look-ahead
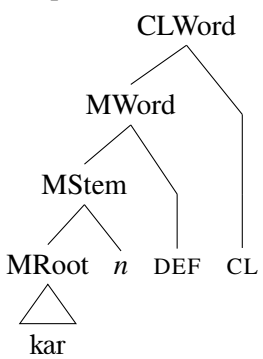
In this thesis, the derivation was cyclic and the input was a piece of morphological structure (root, stem, word) which was incrementally enlarged. On the one hand, the computational problem of cyclicity is the lack of a bound on the number of cycles. On the other hand, the empirical problem is that morphophonological processes can show derivational look-ahead. I handle both problems by enriching the input to encode the word's derivational history and future. I formalize this with Operation Lists (§9.3.1) which are cyclically adjusted during the derivation (§9.3.2). With this list, I trigger the right morphological processes (§9.3.3) and provide derivational look-ahead for outwards-sensitive allomorphy (§9.3.4). In short, this list is a globally accessible resource that can be used by local computations.

As I explain below, an Operation List is similar to a flattened version of a morphological tree (cf. Bjorkman and Dunbar 2016). The concept of an operation list is inspired from finite-state treatments of morphotactics (Koskenniemi 1983b, 1984; Beesley and Karttunen 2003) and from Aksënova and De Santo (2018) analysis of derivational locality. I use a string-like list instead of trees or sets because strings are computationally simpler.

9.3.1 Operation Nodes and Operation Lists

The empirical challenges against cyclicity require a type of derivational look-ahead. To illustrate, consider the Armenian definite suffix again. The suffix shows look-ahead in that it surfaces as *-n* if the a V-initial clitic is later added. Generating this entire output requires 3 cycles: one for the covert nominalizer, one for the definite suffix, and one for the clitic. I assume that clitics form a novel type of morphological nodes called *cliticized words* (CLWord).

(658)   *Deriving outwards-sensitive allomorphy with vs. without lookahead*

	a. Input without look-ahead	b. Input with look-ahead
Input	/kar/	/kar - *n* - DEF - CL /
Cycle 1	kar-∅	kar-∅
Cycle 2	kar-∅-ə	kar-∅-n
Cycle 3	*kar-∅-ə e	kar-∅-n e

In the current formalization (658a), this allomorphy cannot be computed because the input only contains the abstract root. An alternative is to let the input include the root and information on what are the

future morphological operations (658)b. This alternative representation is commonly used in contemporary morphological theories, whether in the form of a tree of morphological features with bottom-up spell-out (Halle and Marantz 1993) or an unordered set of morphosyntactic features (Stump 2001).

Trees are conceptually simple, but they are a richer data structure than strings. In contrast, unordered sets in themselves do not encode enough explicit information on what sequence of rules to apply. I develop a practical, intermediate approach between trees and sets: *operation list*. An Operation List is a sequence of nodes called *operation nodes*. I visualize this concept below for the input root *kar*.

(659)   *Input to updating the operation list* – kar
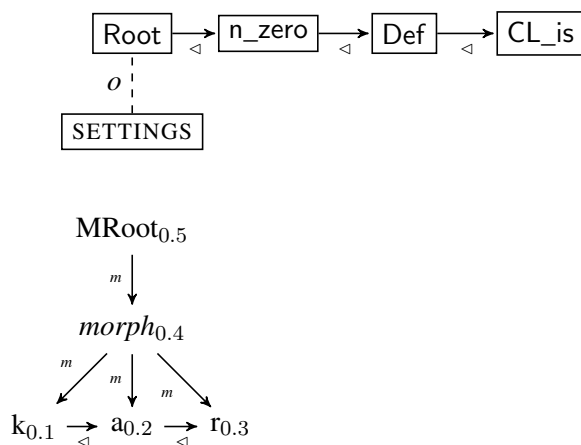
The operation nodes which are visualized as rectangles. The interpretation of each node is that it encodes past, current, and future morphological operations. Their label encodes what morphological process they are associated with: an input root, a covert nominalizer, definite formation, and cliticizing a copula. The current operation is linked to the SETTINGS constant via a dashed line labeled o.

Formally, this representation uses the following set of input labels and relations. Operation nodes have the label $\mathsf{Oper}(x)$. Each of these operation nodes is ordered via a type of successor relation $\mathsf{succ:Oper}(x,y)$. Each has some label which encodes its morphological process: $\mathsf{Oper:Root}(x)$ $\mathsf{Oper:n\_zero}(x)$, $\mathsf{Oper:Def}(x)$, $\mathsf{Oper:CL\_is}(x)$. In the graphs, I show these labels without the substring *Oper:*. The list of operations must start with a node with the label $\mathsf{Oper:Root}(x)$. The SETTINGS constant is associated with exactly one of the operation nodes via the relation $\mathsf{operate\_at}(x,y)$; this relation means that we have just applied that operation node's morphological process.

(660)   a.   *Unary labels for operations*
- $\mathsf{Oper}(x)$: the node x is an operation
- $\mathsf{Oper:Root}(x)$: we must generate the input
- $\mathsf{Oper:n\_zero}(x)$: we must generate the covert nominalizer
- $\mathsf{Oper:Def}(x)$: we must generate the definite suffix -*ə*,-*n*
- $\mathsf{Oper:CL\_is}(x)$: we must generate the clitic =*e* 'is'

b.   *Binary labels for operations*
- $\mathsf{succ:Oper}(x,y)$: the operation x immediately precedes the operation y

- operate_at(x, y): the operation y is the current morphological operation that we must apply. It is linked with the SETTINGS constant as x.

The relation operate_at(x, y), specifically operate_at$(\text{SETTINGS}, y)$, can be replaced by two unary functions. Likewise, the relation succ:Oper(x, y) can be replaced by two unary functions.

(661) a. *Unary functions from the relation* operate_at(x, y):
- $\mathsf{F_L}$:operate_at(x): return the operation node y which is connected with the SETTINGS constant x
- $\mathsf{F_R}$:operate_at(y): return the SETTINGS constant x which is connected with the operation node y

b. *Unary functions from the relation* succ:Oper(x, y):
- $\mathsf{F_L}$:succ:Oper(x): return the operation node y which follows the operation node x
- $\mathsf{F_R}$:succ:Oper(y): return the operation node x which precedes the operation node y

The operation list tells us what morphological process to apply. This is referenced during the **Morphology** stage of a cycle, which I turn to next.

9.3.2 Progressing in the Operation List

Throughout this dissertation, I decomposed a cycle into four components: Morphology, Settings, Prosody, and Phonology. In this chapter, I further refine this architecture by introducing an additional **Operation** stage before the **Morphology** (662). In this stage, we examine the Operation List and proceed onto the next operation. The Operation stage occurs at the beginning of every cycle.

(662) *Sketch of an interaction model with an Operation stage*
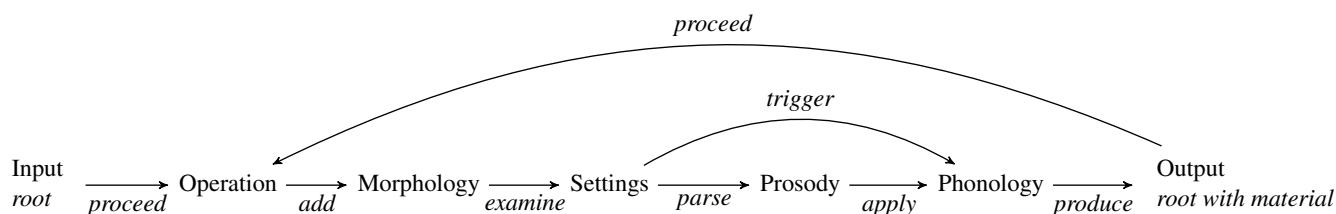
Every derivation starts with the SETTINGS connected to the first operation node: the *Root* node. We progress the operation list by making the SETTINGS re-associate with the subsequent operation node. I show the input and output below.

(663)   *Input and output of proceeding on the operation list for the root* kar
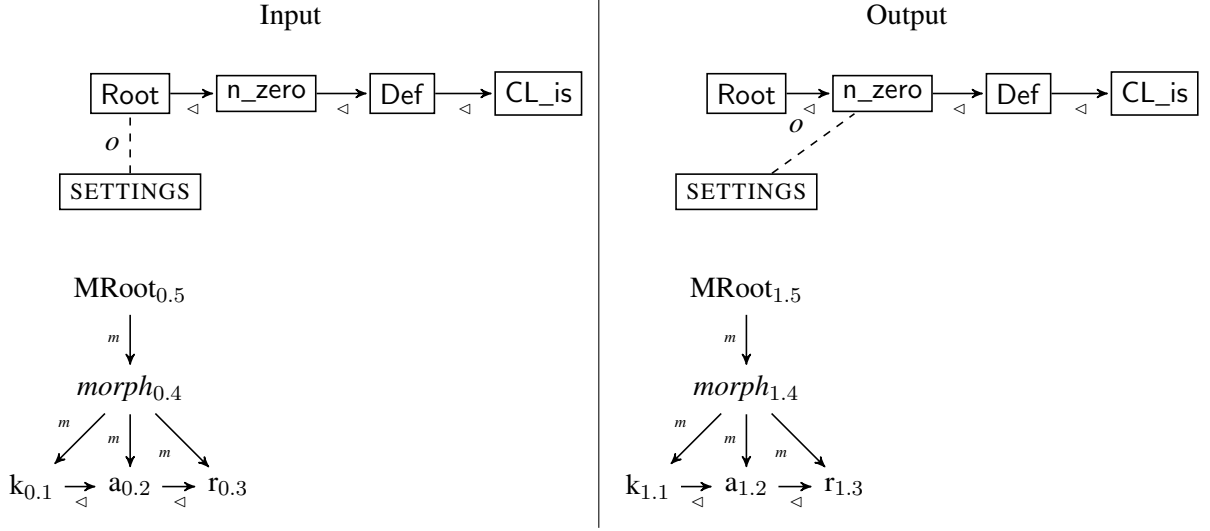
Input	Output

Proceeding on the Operation List is a transduction with a copy set of size 1. The input nodes are faithfully outputted (664). All relations are faithfully outputted except for the relation operate_at(x, y). What is changed is the association between the SETTINGS and the operation nodes.

(664)   *QF output functions to faithfully output the base when updating the operation list, except for* operate_at(x, y)
- For every label lab $\in L$:
  $\phi\texttt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x)$
- For every relation rel $\in R - \{$operate_at$\}$:
  $\phi\texttt{re}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{re}(x, y)$

We find the current operation node *Oper:Root* and the subsequent operation node *Oper:n_zero*. We do so by checking what operation node is associated with the SETTINGS constant via operate_at(x, y). This choice is formalized by the user-predicate **Oper_current**(x). The subsequent operation node is what node follows the current operation node. It is found by the predicate **Oper:subsequent**(x).

(665)   a. *QF user-defined predicate to find the current operation node*
- **Oper:current**$(x) \stackrel{\text{def}}{=} \mathsf{Oper}(x) \wedge \mathsf{operate\_at}(\textsc{settings}, x)$

     b. *FO user-defined predicate to find the subsequent operation node*
- **Oper:subsequent**$(x) \stackrel{\text{def}}{=} \mathsf{Oper}(x) \wedge \exists y[\mathsf{Oper}(y) \wedge \textbf{Oper:current}(y) \wedge \mathsf{succ:Oper}(y, x)]$

In the output, the SETTINGS is no longer associated with the current node *Oper:Root*. The SETTINGS is instead associated with the subsequent operation node *Oper:n_zero*. This is done by the following output function. It is locally computible by only using QF unary functions.

(666)   a. *FO output function to shift the* SETTINGS *to the subsequent operation node*
- $\phi\texttt{operate\_at}(\textsc{settings}^1, y^1) \stackrel{\text{def}}{=} \mathsf{Oper}(y) \wedge \textbf{Oper:subsequent}(y)$

    b. *QF output function to shift the* SETTINGS *to the subsequent operation node*
- $\phi \texttt{operate\_at}(\text{SETTINGS}^1, y^1) \stackrel{\text{def}}{=} \text{Oper}(y) \wedge \text{F}_\text{L}:\text{succ}:\text{Oper}(\text{F}_\text{L}:\texttt{operate\_at}(\text{SETTINGS}))$

Progressing on the Operation List is a computationally local process. I next turn to the Morphology.

9.3.3 Morpheme-specific morphology

A language consists of a finite number of possible morphological processes which can be applied in some order. In previous chapters, I defined **morphological** processes in terms of logical transductions which just applied. Given some input, I did not specify what controlled the choice of morphological processes and their corresponding logical transduction. In this section, I redefine morphological transductions and make them reference the operation list. Specifically, we examine the label of the current operation node. We apply the *type* of morphological process which this node is labeled for.

To illustrate, I use operation lists to trigger covert nominalization for the root *kar*. I show the input and output below. The transduction uses a copy set of size 4.

(667)   *Input and output for adding a covert nominalizer* kar-∅
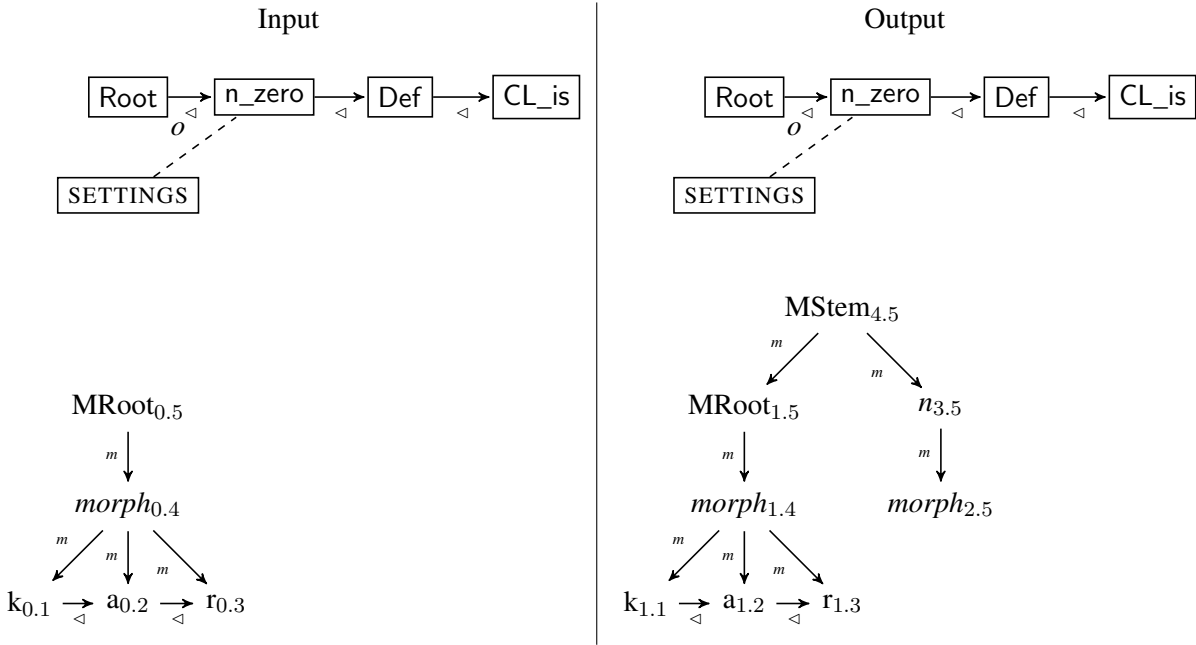
Intuitively, we add the zero suffix $-\emptyset$ if the current operation node tells us to. This is formulated by the logical statement below which doesn't take any input variables (668a). The statement is locally-computed and QF-definable because the current operation node can be locally found from the SETTINGS constant.

(668)   *Checking that the current Operation Node triggers covert nonimalizer via...*
    a. *FO logical statement*
       $\mathbf{Current\,Oper}{:}\mathbf{n\_zero} \stackrel{\text{def}}{=} \exists y[\text{Oper}(y) \wedge \mathbf{Oper}{:}\mathbf{current}(y) \wedge \text{Oper}:\text{CL\_is}(y)]$

b. *QF logical statement*
   $$\mathbf{CurrentOper:n\_zero} \overset{\text{def}}{=} \mathsf{Oper:n\_zero}(\mathsf{F_L:operate\_at}(\textsc{settings}))$$

In the original formulation in Chapter 5: §5.2, the zero suffix $-\emptyset$ was generated without an Operation List. I reformulate its generation. We simply add the condition $\mathbf{CurrentOper:n\_zero}$ to all of the output functions for this process. For example, in Copy 1, all segments and nodes are outputted faithfully. In the reformulated version, we add the condition $\mathbf{CurrentOper:n\_zero}$.

(669) *QF output functions for outputting the base...*

	i. without Operation List	*ii. with Operation List*
For every label $\mathsf{lab} \in L$:		
$\phi\mathtt{lab}(x^1)$	$\overset{\text{def}}{=} \mathsf{lab}(x)$	$\overset{\text{def}}{=} \mathsf{lab}(x) \wedge \mathbf{CurrentOper:n\_zero}$
For every relation $\mathsf{rel} \in R$:		
$\phi\mathtt{rel}(x^1, y^1)$	$\overset{\text{def}}{=} \mathsf{rel}(x,y)$	$\overset{\text{def}}{=} \mathsf{rel}(x,y) \wedge \mathbf{CurrentOper:n\_zero}$

As for the zero suffix itself, the original functions below generate the suffix's morphological nodes (670a), internally linearizes them (670b), and externally linearizes them with the base (670c). Their reformulation simply adds the condition $\mathbf{CurrentOper:n\_zero}$.

(670) *QF output functions for generating the covert suffix*
   a. *Outputting the morphological nodes*

	i. without Operation List	*ii. with Operation List*
$\phi\mathtt{morph}(x^2)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{CurrentOper:n\_zero}$
$\phi\mathtt{noun}(x^3)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{CurrentOper:n\_zero}$
$\phi\mathtt{MStem}(x^4)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{CurrentOper:n\_zero}$

   b. *Internally linearizing the suffix*

	i. without Operation List	*ii. with Operation List*
$\phi\mathtt{MDom}(x^4, y^3)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \ldots \wedge \mathbf{CurrentOper:n\_zero}$
$\phi\mathtt{MDom}(x^3, y^2)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \ldots \wedge \mathbf{CurrentOper:n\_zero}$

   c. *Externally linearizing the suffix with the base*

	i. without Operation List	*ii. with Operation List*
$\phi\mathtt{MDom}(x^8, y^1)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \mathbf{MTopmost}(y)$	$\overset{\text{def}}{=} \mathbf{MTopmost}(x) \wedge \ldots \wedge \mathbf{CurrentOper:n\_zero}$

The output of the morphological transduction is then fed to the transductions for updating the $\textsc{settings}$, generating a prosodic parse, and applying the stem-level cophonological rules. For illustrative purposes, I put aside prosody and phonological rule application.

9.3.4 Derivational look-ahead for outwards-sensitive allomorphy

The previous section established how to use the Operation List to select morphological processes and their corresponding transductions. I now use the list to trigger outwards-sensitive allomorphy. After generating

the covert suffix, we move to the next cycle. We update the operation list by shifting the SETTINGS to the next operation node Oper:Def. The input and output are shown below. I omit prosodic nodes.

(671) *Input and output of proceedings on the operation list for the noun* kar-∅

Input	Output

Root → n_zero → Def → CL_is

o ⋯ SETTINGS

$MStem_{0.8}$

$MRoot_{0.5}$ $n_{0.7}$

$morph_{0.4}$ $morph_{0.6}$

$k_{0.1} \rightarrow a_{0.2} \rightarrow r_{0.3}$

Root → n_zero → Def → CL_is

o ⋯ SETTINGS

$MStem_{1.8}$

$MRoot_{1.5}$ $n_{1.7}$

$morph_{1.4}$ $morph_{1.6}$

$k_{1.1} \rightarrow a_{1.2} \rightarrow r_{1.3}$
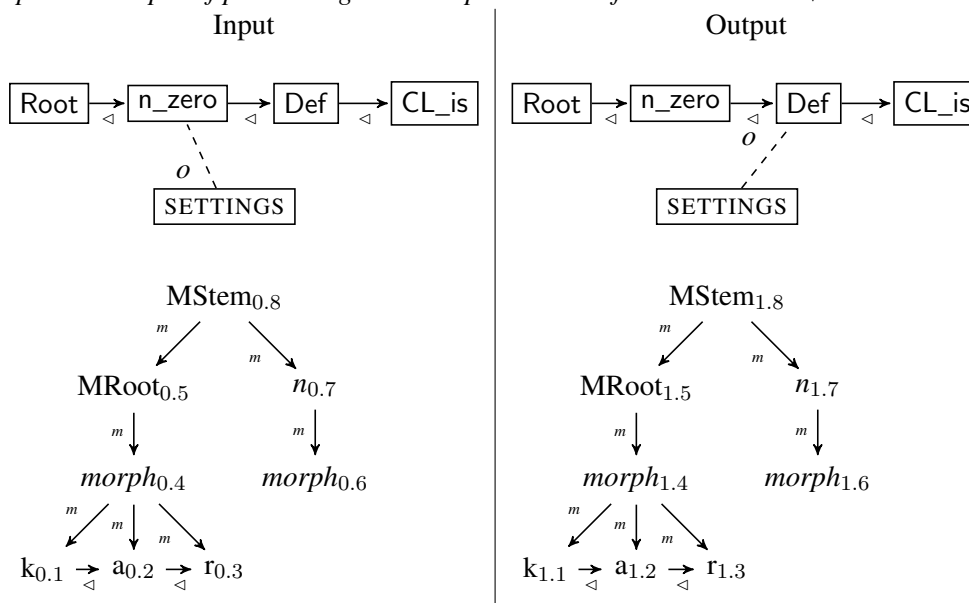
The output of the above Operations stage is then fed to the Morphology. The morphology will add the definite suffix allomorph. Although the base is C-final, the subsequent (future) morphological operation is cliticization. This outwards-sensitive allomorphy triggers the allomorph -*n*.

(672) *Input and output for adding the definite suffix* kar-n

Input	Output

Root → n_zero → Def → CL_is

o ⋯ SETTINGS

$MStem_{0.8}$

$MRoot_{0.5}$ $n_{0.7}$

$morph_{0.4}$ $morph_{0.6}$

$k_{0.1} \rightarrow a_{0.2} \rightarrow r_{0.3}$

Root → n_zero → Def → CL_is

o ⋯ SETTINGS

$MWord_{5.8}$

$MStem_{1.8}$

$MRoot_{1.5}$ n $DEF_{4.8}$

$morph_{1.4}$ $morph$ $morph_{3.8}$

$k_{1.1} \rightarrow a_{1.2} \rightarrow r_{1.3} \longrightarrow n_{2.3}$
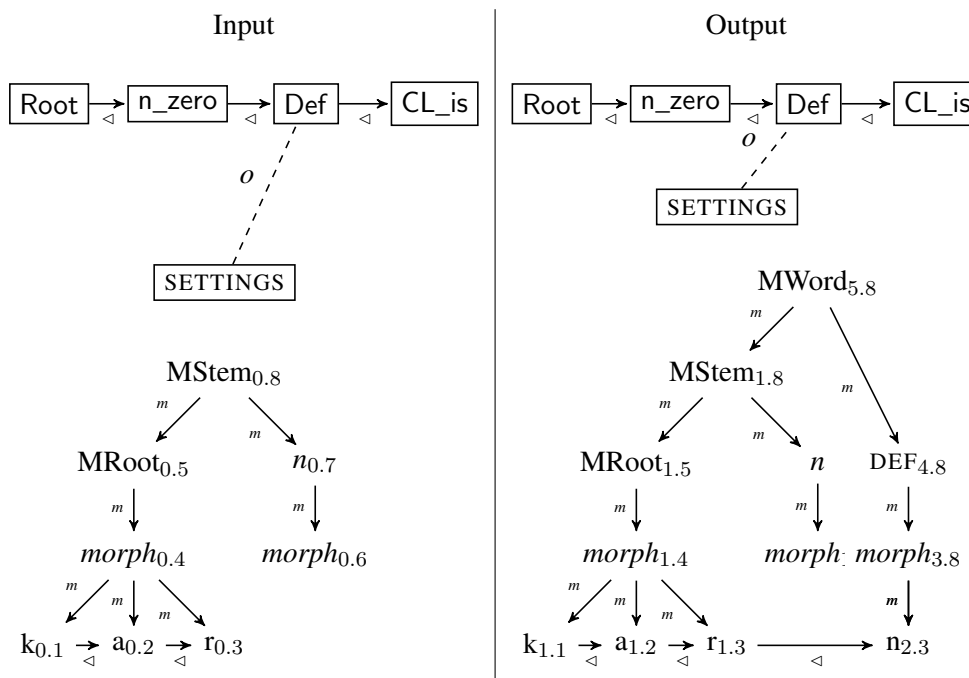
The current operation node is an instruction to generate the definite suffix. This information is encapsulated into the logical statement below (673a). It is locally-computed from the SETTINGS constant (673b).

(673)   *Checking that the current Operation Node triggers the definite suffix via...*
    a. *FO logical statement*
       $\mathbf{CurrentOper:Def} \stackrel{\text{def}}{=} \exists y[\mathsf{Oper}(y) \wedge \mathbf{Oper:current}(y) \wedge \mathsf{Oper:Def}(y)]$
    b. *QF logical statement*
       $\mathbf{CurrentOper:Def} \stackrel{\text{def}}{=} \mathsf{Oper:Def}(\mathsf{F_L:operate\_at}(\text{SETTINGS}))$

Using this statement, we apply the morphological transduction that generates the definite suffix. It uses a copy set of size 5. In Copy 1, the base is faithfully outputted.

(674)   *QF output functions for vacuous identity in Copy 1*
    - For every label $\mathsf{lab} \in L$:
      $\phi\mathtt{lab}(x^1) \stackrel{\text{def}}{=} \mathsf{lab}(x) \wedge \mathbf{CurrentOper:Def}$
    - For every relation $\mathsf{rel} \in R$:
      $\phi\mathtt{rel}(x^1, y^1) \stackrel{\text{def}}{=} \mathsf{rel}(x, y) \wedge \mathbf{CurrentOper:Def}$

Outside of cliticized words, the definite suffix shows inwards-sensitive allomorphy. The suffix *-n* after a V-final base, *-ə* elsewhere after a C-final base. These contexts are formalized by the local predicates below.

(675)   *QF user-defined predicates to check for inwards-sensitive contexts for definite allomorphy*
    - Check if base-final segment is a vowel:
      $\mathbf{final\_V:seg}(x) \stackrel{\text{def}}{=} \mathsf{vowel}(x) \wedge \mathbf{final:seg}(x)$
    - Check if base-final segment is a consonant:
      $\mathbf{final\_C:seg}(x) \stackrel{\text{def}}{=} \mathsf{consonant}(x) \wedge \mathbf{final:seg}(x)$

But in cliticized words, the suffix shows outwards-sensitive allomorphy. It surfaces as *-n* after a V-final base or *before* a V-initial clitic. It is *-ə* elsewhere (after a consonant, not before a V-initial clitic). With the operation list, we can predict if a vowel-initial clitic will be added by examining the labels of the subsequent operation node. That is, the suffix allomorph *-n* is used if the base ends in a vowel *or* if the subsequent operation node has the label $\mathsf{Oper:CL\_is}$. This information is locally-computed from the SETTINGS constant.

(676)   *Check if subsequent operation is adding a V-initial clitic via...*
    - *FO logical statement*
      $\mathbf{SubseqOper:CL\_is} \stackrel{\text{def}}{=} \mathsf{Oper}(x) \wedge \mathbf{Oper:subsequent}(x) \wedge \mathsf{Oper:CL\_is}(x)$
    - *QF logical statement*
      $\mathbf{SubseqOper:CL\_is} \stackrel{\text{def}}{=} \mathsf{Oper:CL\_is}(\mathsf{F_L:succ:Oper}(\mathsf{F_L:operate\_at}(\text{SETTINGS})))$

In Copy 2, we generate the affix allomorphs as output correspondents for the base final segment $r_{0.3}$. We add the suffix allomorph *-n* if the base ends in a vowel or if the subsequent operation is cliticization (677a). The allomorph *-ə* is generated if the base ends in a consonant and if the subsequent operation is not the clitic 'is' (677b). To generate either allomorph, we make sure that the current operation is definite formation.

(677) *QF output functions for generating the suffix segments*
   a. *Generating the allomorph* -n
      - $\phi\text{n}(x^2) \overset{\text{def}}{=} \textbf{final:seg} \wedge [\textbf{final\_V:seg}(x) \vee \textbf{SubseqOper:CL\_is}] \wedge \textbf{CurrentOper:Def}$
   b. *Generating the allomorph* -ə
      - $\phi\text{n}(x^2) \overset{\text{def}}{=} \textbf{final:seg} \wedge [\textbf{final\_C:seg}(x) \wedge \neg\textbf{SubseqOper:CL\_is}] \wedge \textbf{CurrentOper:Def}$

Both of these functions are locally-computed because the Operation List is local to every segment in the input via the SETTINGS constant. For the input *kar*, the suffix is generated as an *-n*. Even though the base is C-final, we know that a vowel-initial clitic *e* will be later added. The rest of the computation is straightforward and uses the output functions to generate the morphological nodes and the linearization.[4] The derivation will continue onto the next cycle to generate the clitic: *kar-n-e*.

(678) *QF output functions to generate and linearize the definite suffix*
   1. *QF output functions to generate the new morphological nodes*
      - $\phi\text{morph}(x^3) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{CurrentOper:Def}$
      - $\phi\text{def}(x^4) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{CurrentOper:Def}$
      - $\phi\text{MWord}(x^5) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{CurrentOper:Def}$
   2. *QF output functions to internally linearize the suffix*
      - $\phi\text{MDom}(x^3, y^2) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{final:seg}(y) \wedge \textbf{CurrentOper:Def}$
      - $\phi\text{MDom}(x^4, y^3) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y) \wedge \textbf{CurrentOper:Def}$
      - $\phi\text{MDom}(x^5, y^4) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y) \wedge \textbf{CurrentOper:Def}$
   3. *QF output function to externally linearize the suffix's segments and morphology*
      - $\phi\text{MDom}(x^5, y^1) \overset{\text{def}}{=} \textbf{MTopmost}(x) \wedge \textbf{MTopmost}(y) \wedge \textbf{CurrentOper:Def}$
      - $\phi\text{succ:seg}(x^1, y^2) \overset{\text{def}}{=} \textbf{final:seg}(x) \wedge \textbf{final:seg}(y) \wedge \textbf{CurrentOper:Def}$

The above strategy with operation lists works, but it is more intuitive for cases where the outwards-sensitive allomorphy is *morphologically*-conditioned rather than phonologically-conditioned. It can model outwards-sensitive *phonologically*-conditioned allomorphy like with the definite suffix, but *only if* we can predict the future form of the clitic based on morphological context. We know a future clitic will be vowel-initial if the subsequent operation node tells us which clitic will be added. In this case, the subsequent node says the 'is' clitic will be added. If we know that 'is' can only be the vowel-initial *=e*, then we can generate the correct definite suffix *-n*. But this approach reduces phonologically-conditioned allomorphy to some type of morphologically-conditioned allomorphy: *use* -n *if the definite precedes the* 'is' *clitic or other* **specific** *clitics which happen to all be V-initial*.

9.4 Function composition of 'unbounded' cyclicity

As explained in §9.2.1, the main computational problem behind cyclicity is its unboundedness. Even though a single function be computationally simple and only need QF or FO logic, the composition of an

---

[4]A small problem in this formalization is preventing prosodic processes from referencing the outwards-sensitive allomorph *-n*. Given an input *kar* and output *kar-n*, we incorrectly predict that the output will be syllabified as *kar-ən with schwa epenthesis. We would need to either block schwa epenthesis from applying here, or have a rule that deletes this epenthetic schwa once the V-initial clitic is added to form *kar-n-e* instead of *kar-ən-e.

infinite number of these functions is not necessarily QF or FO. This is a common and early criticism against the role of cycles in generative linguistics and transformational grammars (Peters and Ritchie 1973; Levelt 1974), which has been subsequently applied to cyclic phonology (Cole and Coleman 1992; Cole 1995b).

However, in practice, speakers make a conscious choice to apply some morphological process to generate a new derivative, inflected item, or compound. Thus, upon forming a new word, speakers plan the number of cycles they need. I formalized this intuitive planning of morphological processes with Operation Lists. This means that we can compose entire cyclic derivations into a single set of functions. I illustrate below.

In the beginning of the derivation, the input is an entire list of planned operations. Once an individual operation is selected, the morphological and phonological transductions apply in a single cycle. Each cycle consists of a finite sequence of 5 components: Operation, Morphology, Settings, Prosody, and Phonology. Each component is formalized as a QF or FO logical transduction, which itself can consist of smaller transductions. The entire sequence of components forms a chain of logical transductions that are in feed-forward relationship (679aa). This finite sequence can composed into a single FO transduction (679ab). Thus, every cycle is at most FO.

(679)   a. *Interactionist model with 5 components and recursive cyclicity*
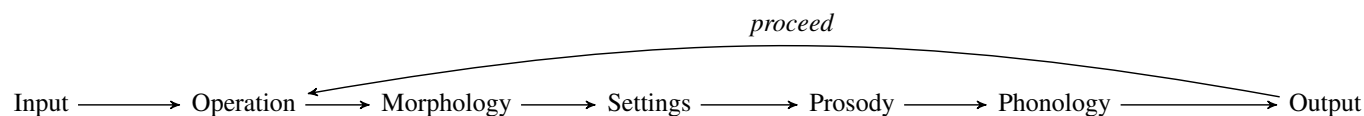
b. *Composing the 5 components in a single cycle*
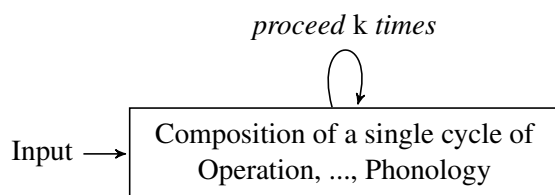
But what about for the entire sequence of cycles? Recall that the Operation List is finite, e.g., of size k. Thus, a given derivation will involve only a predictable finite number of cycles (680aa). As long as these cycles are logically definable with FO or MSO logic (which they are in thesis), then they can be composed into a single FO or MSO function or transduction (680ab).

(680)   a. *Cyclic computation with exactly k cycles*
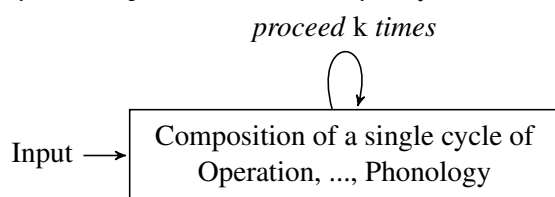
This is equivalent to...
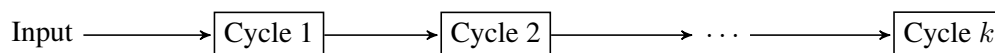
b. *Composition of the interactionist with k-bounded cyclicity*

Input $\longrightarrow$ | Composition of k cycles of (Operation, ..., Phonology)^k |
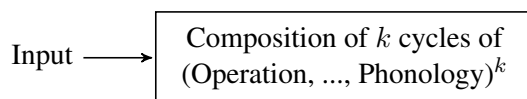
The composition of the entire sequence of cycles can only apply in run-time.[5] It cannot apply in compilation. Because of unbounded cyclicity, there is no number k such that all words w will use at most k cycles or morphological operations. This bound is impossible because of recursive morphological processes like recursive affixation, recursive compounding, etc. Schematically, I am not arguing for $\exists k, \forall w$ [*apply at most* k *cycles*]. However, for every word w, there exists a number k of cycles which is *specific* to that word w, i.e., I conceptualize this as $\forall w, \exists k$ [*apply exactly* k *cycles*].

9.5 Conclusion

This section focused on problems in cyclic architectures. Computationally, cyclicity is difficult to formalize or restrain because there are no a priori bounds on the number of cycles. Empirically, cyclic architectures can get unwieldy in the case of outwards-sensitive allomorphy. I tackle both of these problems by developing a formalization for derivational look-ahead.

The empirical problem was tackled with Operation Lists. An operation list is a finite list of instructions over what morphological processes must apply in the entire derivation. This list encodes the set of morphosyntactic features of the lexical item (cf. Stump 2001) which are linearly organized. The linear organization resembles traditional Item-and-Arrangement models for morphotactics (Hockett 1942; Beesley and Karttunen 2003), and are a computationally simpler representation than trees (cf. Halle and Marantz 1993; Embick and Noyer 2001; Embick 2010, 2015). This list is **accessible** throughout the derivation by being connected with the SETTINGS constant. With this list, we can locally determine what will be the subsequent morphological operations. This solves the empirical problem of outwards-sensitive allomorphy. A similar solution can be sketched for post-cyclic phonology.[6]

As for the computational problem, the Operation List intuitively asserts a finite bound on the number of cycles. This bound is not universal, i.e., there is no bound k such that all words will undergo at most k cycles. However, for every individual word, there is such a bound. Thus, we can compose a single word's derivation in run-time in order to generate a single set of FO logical transductions for the morphology-phonology interface.

---

[5]This is similar to Walther (2000)'s strategy for modeling total reduplication. He develops a system whereby complex FSTs are generated for every word w in run-time. Each FST computes total reduplication for a single word w. Given a finite lexicon, the union of all these finite-number of FSTs generates a finite language. However, he suggests that these FSTs are not stored or compiled. They are generated at run-time so that we could (in practice) have an infinite number of FSTs which can reduplicate an infinite number of words.

[6]Using an Operation List, we can apply the post-cyclic cophonology if the current Operation Node is the final Operation Node.

Chapter 10

Conclusion

In this dissertation, we asked the following fundamental question:

Q1) *What principles control the alternation in the pronunciation of morphemes?*

Using Armenian as a case study, we found that three core principles are the three separate modules of:

1. **Morphology**:   What is the internal morphological composition of words
2. **Prosody**:   What is the internal prosodic composition of words
3. **Phonological Rule Domains**: In what types of domains do phonological rules ($\sim$ cophonologies) apply

With this empirical background, question Q1 was followed up by two questions:

Q2) *How are these principles combined or organized?*
Q3) *How are these organizations computed?*

In this chapter, I review all these questions. For questions Q1 and Q2, the results required a model of the interface which was interactionist and cyclic whereby the three modules feed each other in a potentially unbounded number of ways, much like classical lexical phonology (Kiparsky 1982b).

Question Q3 then asks how such a model can be formally defined and computed. I answered this question using graph-to-graph logical transductions with Monadic Second Order (MSO) logic. Besides making the interface be computationally definable, the logical formalism likewise showed that the bulk of the interface is computationally simple or local because most of the formulas can be expressed in Quantifier-Free logic. This reinforced the often-assumed hypothesis that morphophonological processes show locality restrictions. The formalism likewise identified those aspects of the interface that are computationally non-local.

The computational model that I developed was inspired from early work in level-ordered phonology (Siegel 1974; Allen 1979) and classical lexical phonology (Kiparsky 1982b). I did not consider the various theoretical devices that were added to the simple interactionist model, e.g., the Strict Cyclicity Condition (Kean 1974), Morphological Level-Ordering (Siegel 1974), Structure Preservation (Kiparsky 1985; Myers 1991), or Bracket Erasure (Pesetsky 1979). Many of these additional principles are problematic (Cole 1995a; Bermúdez-Otero 2008), e.g., the SCC was disproven (Kiparsky 1993), Level-Ordering is not adequate (Fabb 1988), and Structure Preservation is not a strict principle (Harris 1987; Kaisse and Hargus 1994). Bracket

Erasure is one of the few classical principles which is mostly robust (Orgun 2002); the main modification is that stem/root boundaries tend to evade bracket erasure (Hargus 1985; Shaw 2009; Inkelas 2014) and that affixation can reference the most recently added morpheme, i.e., potentiation (Hammond 1992).

The computational formalism is agnostic about any potential limits on the number of possible strata or cophonologies in the language (cf. Inkelas and Orgun 1995; Lemus 1996; Vogel 2016:27). It is likewise agnostic over the possibility that only some but not all morphological constructions can arbitrarily trigger cyclic phonology (cf. Halle and Vergnaud 1987a; Szpyra 1989). But regardless, the computational results apply to *any* interactionist model. The simple interactionist model has had many incarnations over the decades (cf. Scheer 2011). But regardless, our computational results apply to many of these incarnations, including:

- Lexical Phonology (Booij 1980, 1987, 1988a,b, 1994, 2000; Kiparsky 1982b,a, 1983, 1985; Booij and Rubach 1984, 1987; Rubach 1985, 2008; Hargus 1985; Kaisse and Shaw 1985; Mohanan 1986; Hargus and Kaisse 1993; Wiese 1994; Giegerich 1999; Kaisse and McMahon 2011)
- Prosodic Lexical Phonology (Booij 1985; Inkelas 1989, 1993; Fitzpatrick-Cole 1994; Han 1995; Mansfield 2017)
- Stratal OT (Booij 1997; Rubach 1997, 2003; Bermúdez-Otero 1999, 2011, 2012, 2016, 2018, prep; Kiparsky 2000, 2015; Trommer 2013)
- Transderivational OT (Kenstowicz 1996; Benua 1997; Burzio 1998; Raffelsiefen 1999, 2005; Steriade 2000, 2008b; Downing et al. 2005)
- Cophonology Theory (Orgun 1994, 1996, 1998; Inkelas and Orgun 1995, 2003; Inkelas et al. 1996, 1997; Anttila 2002; Inkelas and Zoll 2007; Inkelas 1998, 2008, 2014; Sande 2017; Sande et al. 2020)
- Phase-based Phonology (Marvin 2002; Newell 2008; Samuels 2011; Embick 2010, 2014; Scheer 2011, 2012; Newell and Piggott 2014; d'Alessandro and Scheer 2015; McPherson and Hayes 2016; Newell et al. 2017; Guekguezian 2017a; Sande et al. 2020)

There are many more interactionist models which don't fall in the above categories (cf. Scheer 2011; Elordieta 2014), but which still abide by our computational results (Wolf 2008; Hanson and Inkelas 2009; Inkelas 2014; Shwayder 2015). The results likewise apply to non-interactionist but cyclic approaches to the morphology-phonology interface (Chomsky et al. 1956; Chomsky and Halle 1968; Brame 1974; Mascaró 1976; Halle and Vergnaud 1987a,b; Halle and Kenstowicz 1991). They likewise apply to theoretical frameworks which combine prosodic structure with phonological rule domains (cophonologies, phase boundaries) without necessarily assuming cyclicity or interleaving (Selkirk 1980, 1986, 1996, 2011; Nespor and Vogel 1986; Vogel 1989, 2008, 2016; Rice 1992, 1993; Peperkamp 1997; Kager et al. 1999; Hall and Kleinhenz 1999; Grijzenhout and Kabak 2009; Miller 2018, 2020).

10.1   Q1 & Q2: Cyclic phonology of Armenian

In terms of the empirical contributions of the dissertation, we showed that Armenian phonological processes are sensitive to various morphological, prosodic, cophonological, and organizational structures. Chapter 2 looked at these processes in detail and I argued for the existence of three rule domains: the stem-level, word-level, and PStem-level rule-domains or cophonologies. Each cophonology is triggered by its own morphological or prosodic constituent, and each is associated with separate set of phonological processes in the two Armenian dialects. I recapitulate each cophonology below.

(681)   *Distribution of processes and domains across cophonologies*

Cophonology	Stem-level	PStem-level		Word-level
Morphological domain	Derivation	V-initial Inflection		Inflection
Relevant constituent	MStems	Misaligned PStems		MWords
Process ⟍ Dialect	Both	EArm	WArm	Both
Destressed Diphthong *uj* Reduction	✓	✗	✗	✗
Destressed High Vowel Reduction	✓	✓	✗	✗
Stress Shift	✓	✓	✓	✓

In Western Armenian, stress shift applies before both derivational and inflectional morphology. In contrast, other processes such as destreseed high vowel reduction (DHR) are sensitive to the type of morphemes which are added in the course of the derivation. DHR applies before derivational morphology (682a.ii), but not inflectional morphology (682a.iii,iv). Other phonological processes showed the same restriction, including destressed diphthong reduction (DDR, 682b) and numerous vowel hiatus repair rules such as vowel deletion (682c). It doesn't matter if the inflectional suffix is vowel-initial (V-Infl) or consonant-initial (C-Infl).

(682)   *Different processes in Western Armenian and their domains*

	a. DHR		b. DDR		c. Vowel Hiatus	
i. Root	amusín	'husband'	zərújts͡	'conversation'	təʃnamí	'enemy'
ii. Der	amusn-utjún	'marriage'	zəruts͡-él	'to converse'	təʃnam-agán	'hostile'
iii. V-Infl	amusin-óv	'husband-INST'	zərújts͡-óv	'conversation-INST'	təʃnamij-óv	'enemy-INST'
iv. C-Infl	amusin-nér	'husband-PL'	zərújts͡-nér	'conversation-PL'	təʃnami-nér	'enemy-PL'

These processes are organized into different strata or rule domains: derivational morphology triggers the stem-level cophonology which includes DHR, while inflectional morphology triggers the word-level cophonology which excludes DHR.

Cross-dialectal data from Eastern Armenian likewise showed the need for prosodic structure. Eastern Armenian minimally differs from Western Armenian in that DHR applies in derivation (683b) and in vowel-initial inflection (683c), but not consonant-initial inflection (683e).

(683)   a. Base   amusín   'husband'
         b. Der   amusn-utjún   'marriage'
         c. V-Infl   amusn-óv   'husband-INST'   Eastern
         d.   amusin-óv   'husband-INST'   Western
         e. C-Infl   amusin-nér   'husband-PL'   Eastern & Western

However, it is not the case that all stem-level rules overapply in V-initial inflection; DDR is still limited to derivational morphology (684b), not inflectional morphology (684c,684d).

(684)   a. Western   Eastern
             zərújts͡   zərújts͡ʰ   'conversation'

   b. zəruts͡-él   zəruts͡ʰ-él   'to converse'

   c. zərujt͡s-óv   zərujt͡sʰ-óv   'conversation-INST'

   d. zərujt͡s-nér   zərujt͡sʰ-nér   'conversation-PL'

To capture the overapplication of DHR in Eastern Armenian, I argued that there is a prosodic constituent which straddles the boundary between the morphological stem and vowel-initial inflection. This constituent is the Prosodic Stem (PStem, Downing 1999a). It is isomorphic to the morphological stem (MStem) in derivation (685b) and C-initial inflection (685d). It expands before V-initial inflection due to resyllabification (685c) and becomes non-isomorphic from the MStem. Once the PStem expands, it triggers its own set of phonological rules, i.e., its own cophonology. In Eastern Armenian, the PStem arbitrarily triggers the application of DHR but not other processes. In Western Armenian, the PStem cophonology is identical to the word-level cophonology. The emergence of this PStem cophonology was explained in terms of the diachronic history of Armenian vowel reduction and inflection.

(685)   *Different PStem structures in the two Armenian dialects*

   a. Base   b. Derivative   c. V-Infl   d. C-Infl

amusín   amusn-utjún   amus(i)n-óv   amusin -nér
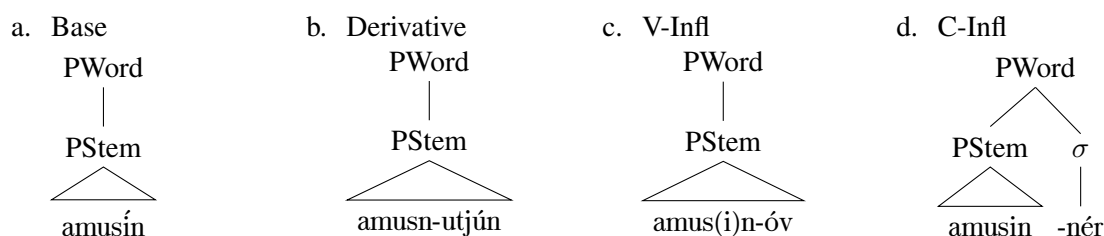
The sum total of the analysis from Chapter 2 was the existence of these multiple cophonologies, their activation within a cyclic and interactionist model, and the existence of an intermediate prosodic constituent such as the PStem. Later in Chapter 3, I showed that these three aspects of Armenian were crucial to understanding a bracketing paradox in compounds. In simplex words, the plural suffix has two allomorphs: *-er* after monosyllabic bases (686a-i) and *-ner* after polysyllabic bases (686a-ii). In compounds, we find a paradox when the second stem is monosyllabic. While exocentric compounds are transparently pluralized as polysyllabic bases with *-ner* (686b-ii), endocentric compounds which are paradoxically pluralized as monosyllabic bases with *-er* (686b-i).

(686) a. i. pág   'yard, lot'   ii. panág   'army'

          pag-ér   'yards, lots'   panag-nér   'armies'

    b. i. ant͡srév + t͡ʃúr   'rain + water'   ii. t͡ʃár + sírd   'evil + heart'

          ant͡srev-a-t͡ʃúr   'rain-water'   t͡ʃar-a-sírd   'evil-hearted'

          ant͡srev-a-t͡ʃur-ér   'rain-waters'   t͡ʃar-a-sird-nér 'evil-hearted people'

In the paradoxical case, the plural was found to count the number of syllables in the second stem, i.e., the semantic head of the compound: *ant͡srev-a-t͡ʃur-er* 'rain-water-s'. This paradoxical behavior was analyzed in terms of a cyclic head operation whereby the plural targets the head of the compound. I showed that counter-cyclic alternatives (e.g., Marantz 1988; Sproat 1985; Newell 2005) are problematic because they could not explain the internal phonology of compounds. Similar to derivational morphology, both exocentric
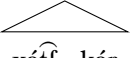
(687a-ii) and endocentric (687b-ii) compounds are phonologically alike because they undergo the same cyclic stem-level rules, e.g., high vowel reduction. On the one hand, the two stems in an endocentric compound are phonologically coherent towards each other and form one unit for the stem-level cophonology: *t͡ʃər-a-pos* 'water-hole'. But on the other hand, they take paradoxical plurals whereby the plural ignores the first stem: *t͡ʃər-a-pos-ér* 'water-holes'.

(687) a. i. aznív   'sincere'   ii. aznív + sírd   'sincere + heart'
               aznəv-utjún   'sincerity'   aznəv-a-sírd   'sincere-hearted'
               azniv-ov   'sincere-INST'   <u>aznəv-a-sird</u>-nér   'sincere-hearted (people)'

       b. i. t͡ʃúr   'water'   ii. t͡ʃúr + pós   'water + hole'
               t͡ʃər-ajín   'aquatic'   t͡ʃər-a-pós   'water-hole'
               t͡ʃur-ov   'water-INST'   t͡ʃər-a-<u>pos</u>-ér   'water-holes'

As a last piece to the piece, I also showed that internal prosodic constituents such as the PStem are likewise active in compound phonology. Exocentric bisyllabic compounds are always pluralized transparently with *-ner* (688a). But endocentric bisyllabic compounds can pluralize either transparently with *-ner* or paradoxically with *-er* (688b).

(688) a. kár + daS-el   'stone + to carve'   b. xát͡ʃ + kár   'cross + stone'
           kar-dáʃ   'stone carver, mason'   xat͡ʃ-kár   'cross-stone'
           *kar-<u>daʃ</u>-ér   'stone carvers, masons'   xat͡ʃ-<u>kar</u>-ér   'cross-stones'
           <u>kar-daʃ</u>-nér   xat͡ʃ-kar-nér

The variation was analyzed in terms of prosodic mapping. The semantic head *h* is mapped to a prosodic head *p*. This *p* can optionally expand in bisyllabic words. The plural suffix then counts the number of syllables in *p*. I showed that *p* cannot be a foot or PWord but must be a PStem as a last resort.

(689) a. xat͡ʃ-kár   'cross-stone'   b. xat͡ʃ-kár   'cross-stone'
           xat͡ʃ-<u>kar</u>-ér 'cross-stones'   xat͡ʃ-kar-nér 'cross-stones'

In sum, the above fragment of Armenian encompasses a large chunk of the morphology and phonology of the language. The fragment shows that Armenian is clearly sensitive to separate strata or cophonologies which act as different phonological rule domains. These domains are triggered by abstract morphological and prosodic structures. These different factors interact cyclically within a simple interactionist model, and their interaction can create bracketing paradoxes.

10.2   Q3: Computational locality of cyclic phonology

With the data and interactionist model set up, the next step was to determine how this model can be computed. Using MSO graph-to-graph transductions in Chapter 5, we defined each individual component of the morphology-phonology interface and defined how one component can be processed into the next.

(690)   *Sketch of an interactionist model*
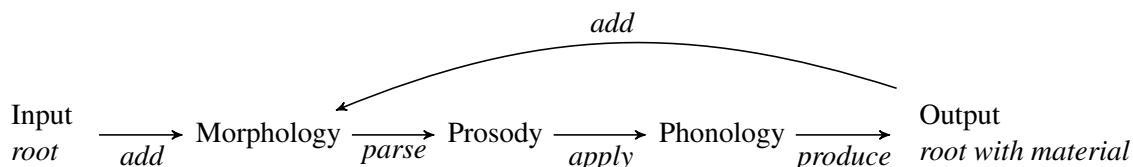
The three core benefits or results are listed below. I go through each of them. All of these benefits arise from the fact that our computational system is couched within Model Theory. By using logic, we develop a unified framework where we compare different types of intensional descriptions (representations) in order to understand their computational properties.

1. **Faithfulness**: The bulk of the interface was computationally and faithfully defined
2. **Explicitness**: The interface was made computationally explicit
3. **Locality**: The bulk of the interface is computationally local

The first benefit is **faithfulness**. As a computational tool, I used logic in order to faithfully replicate the multiple hierarchical structures which are present in linguistic theory (and in the empirical data). Alternative common formalisms such as finite-state calculus would sacrifice this hierarchical structure. In a sense, the present formalism generalizes the work of one-level, monostratal Declarative Phonology (Bird 1995; Coleman 1998) into a Two-Level perspective.

The second benefit is that the formalism **explicitly** shows how an interactionist model works. The formalism allowed us to uncover implicit factors in the derivation. The most important implicit factor is the SETTINGS of the derivation. This is the stage where the linguist analyzes the morphological structure and implicitly knows what to prosodically parse and what rules to apply. Using the SETTINGS as a descriptive tool, we factorized the morphological, prosodic, and phonological processes into their targets and triggers.

(691)   *Sketch of an interactionist model with an explicit stage for the* SETTINGS
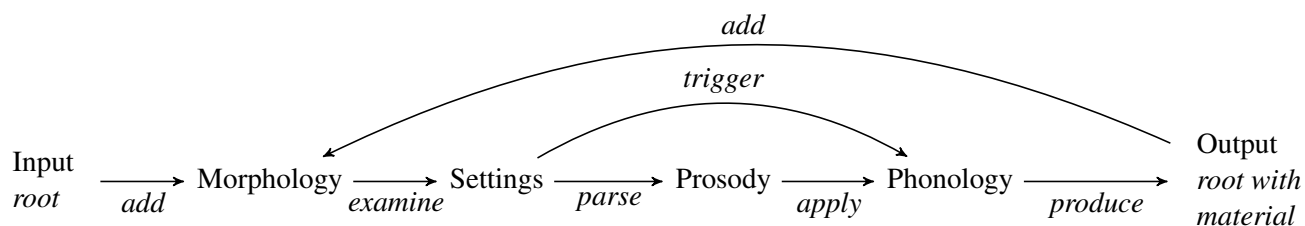
The third major benefit of the formalism is that it allows us to understand the **generative capacity** or computational tendencies of the morphology-phonology interface. Our logical formalization was written

in MSO logic, a rich and flexible type of logical formalism. However, we did not need the full power of MSO logic. In Chapters 5-7, using the above factorization, I showed that the brunt of the interface consists of computationally simple or **local** processes. The locality of this information can be defined with Quantifier-Free (QF) logic. In fact, by assuming interactionism and the SETTINGS factorization, nearly the entire interface is QF. Non-locality is restricted to the generation of tiers, different types of allomorphy, the need for settings examination for cophonology selection, and post-cyclic prosody.

(692)   *Aspects of the morphology-phonology interface which are defined*

Morphology		Local?	**Prosody**		Local?
Affixation			Syllabification		
Covert affix	§5.2	✓	Generating syllables	§5.4.1	✓
Overt affix	§6.2.1	✓	Ordering syllables	§5.4.2	✓
Compound Morphology	§6.2.2	✓	Resyllabification	§6.4	✓
Tiers over dominance	§8.3.2.2	✗	Tier over syllables	§5.4.2.1	✓
Affix order and linearization			Prosodic mapping		
Reduplication	§7.4.1	✓	Generating prosodic stem	§5.4.3	✓
Suffixation	§7.5.1	✓	Restructuring prosodic stem	§6.5.1.1	✓
Prefixation	§7.5.2	✓	Recursive prosody	§6.5.1.2.1	✓
Mobile affixation	§7.5.3	✗	Flattening recursive prosody	§6.5.1.2.2	✓
Morphological layering	§7.5.4	✓	Prosodic misalignment	§6.5.2.1	✓
Allomorphy			Prosodic layering	§6.5.2.2	✓
Phonologically-conditioned	§7.6	✓/✗	Compound prosody	§6.5.3	✓
Morphologically-conditioned	§7.7	✓/✗	Post-cyclic prosody	§8.3.2	✓/✗
Outwards-sensitive	§9.2.2.2	✓/✗			
Examination of the SETTINGS			**Phonology**		
Cophonologies triggered by			Processes in Armenian		
Morphological constituents	§5.3.2	✗	Stress	§5.5	✓
	§6.3.1	✗		§6.6.1.1	✓
Prosodic constituents	§6.6.2	✗	Reduction	§6.6.1	✓
Morphemes	§8.2.2	✓/✗	Cross-linguistic processes		
Parsing instructions	§5.3.1	✗	Aspiration, metathesis, etc	§4.3	✓
	§6.3.2	✗			

The goal of this dissertation was not to make everything in morphology-phonology become computationally local. Rather the goal is to understand which representations and analytical choices can create locality or non-locality. For example, by peeling away some of our assumptions of the interface, we still find that a significant chunk of the interface is local in Chapter 8. If we assume interactionism but no SETTINGS factorization, almost everything is still local. The only new area of non-locality is defining phonological rule domains. These domain rules affect the target segment but they can be triggered by abstract morphological nodes which can be at any distance from the target segment.

Going further, we likewise saw that interactionism itself affects the locality of certain processes. If we assume a non-interactionist model for prosody, i.e., postcyclic prosodic parsing, the bulk of word-level prosody is locally computed. The only traces of non-locality are a) parsing compounds into multiple prosodic nodes, and b) flattening potentially unbounded number of recursive prosodic nodes into one.

Lastly in Chapter 9, we showed that an additional implicit factor in an interactionist or cyclic architecture

is the knowledge of what future morphological operation to apply. When this knowledge is made explicit, we can handle phenomena such as outwards-sensitive allomorphy. This knowledge is made explicit in the form of an Operation List which acts a formalization for derivational histories and morphotactics.

(693)   *Sketch of an interaction model with an Operation stage*
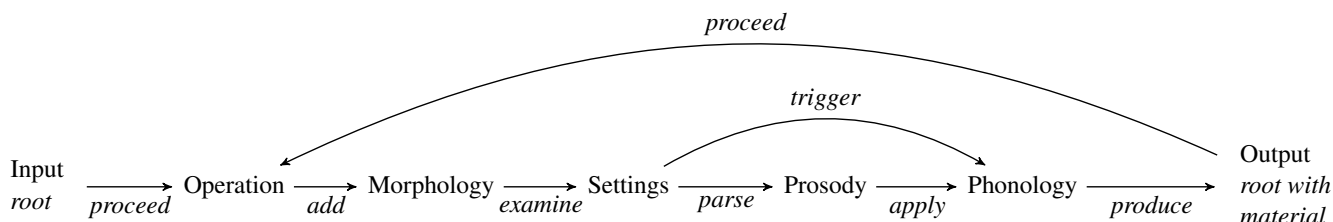
In sum, the takeaway is the unification of concepts present in theoretical and computational linguistics. On the one hand, linguistic theory posits multiple hidden structures, including morphological, prosodic, and phonological rule domains. These structures interact in a system of representations and rules (Anderson 1985). An often-assumed assertion in linguistics is that these structures or representations are somehow 'easy' or 'simple' to define and use: *if the representations are right, then the rules will follow* (McCarthy 1988:84). On the other hand, the dissertation shows that these assertion is *computationally true*. We showed that the hidden structure (representations) and the rules which operate over these representations are computationally simple and local.

10.3   Open questions

This dissertation tried to answer the three core questions Q1-Q3. Although we made significant strides in answering these questions, some aspects are still unanswered. On the empirical side, there are more areas in Armenian where we see morphophonological interactions, including problems in language contact, verbal conjugation classes, schwa epenthesis, and reduplication. These topics were hinted at in different parts of the dissertation. They are left for future work.

On the theoretical side, the computational formalism gives precise definitions for what counts as a local vs. non-local process. It likewise shows that the interactionist model which is assumed in much current work in morphophonology is empirically viable and computational definable. The next step is analyzing how much cross-linguistic variation we find in morphological, prosodic, and phonological domains. Given an adequate typology of hidden structures, we can then determine how much of it is still a) definable within the interactionist system, and b) computationally local. I did not formalize some areas such as moras and feet because I could not find enough evidence for them in Armenian.

On the computational side, defining the problem space of morphophonology is the first step to developing a software implementation and learning algorithm for morphophonology. For implementation, a possible initial step is to develop a package which can process graph-to-graph logical transductions using Prolog or Python. Prolog was previously used to define a working implementation for Lexical Phonology (Williams 1993), while Python has ample resources (Aksënova 2020b). This can be used to fact-check the analyses. As for learning, there is work on developing provably correct learning algorithms for simpler types of phonological structure which are couched within formal logic (Strother-Garcia et al. 2017; Vu et al. 2018;

Chandlee et al. 2019). The next step is to see how they scale up to learning the type of hierarchical structures and processes present in morphophonology. As repeatedly shown, most of these processes over these structures are computational local. The computational locality of the interface makes the task of learning easier (cf. Heinz et al. 2015).

Finally, this dissertation looked at the morphology-phonology interface and asked different descriptive, theoretical, and computational questions. We can also look at the syntax-phonology interface in the same way. The answers found in this dissertation on the morphology-phonology interface are a stepping stone to answering questions on the syntax-phonology interface. Many of the problems found in formalizing the morphology-phonology interface are also found in the syntax-phonology interface. These problems include the use of morphosyntactic trees, the generation of prosodic structures, and problems in cyclicity and look-ahead (Selkirk 1986, 2011; Inkelas and Zec 1990; Elordieta 1997, 2008; Seidl 2000; Pak 2008; Cheng and Downing 2016; Kalivoda 2018; Miller 2018; Bennett and Elfner 2019). I speculate that the logical formalism for the syntax-phonology interface will need minimal adjustments. Some routes to modeling all of these issues are using Minimalist Grammars which include information on prosodic boundaries (Yu and Stabler 2017; Yu 2019) and then defining a logical transformation from syntactic trees to prosodic trees (Ashton 2012). The former approach has the benefit of providing an interface for computational implementations of other linguistic modules, especially when using tree transducers (Morawietz 2003). The latter approach has the benefit of combining results in the computational formalization of syntactic trees (Rogers 1998) and prosodic trees (Bird 1995).

Bibliography

Abeġyan, M. (1933). Hayoc' lezvi taġačap'owt'yown metri [Metrics of the Armenian language]. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Abrahamyan, A. A. (1959). Bayakan krknavor baṙerë žamanakakic' hayerenowm [Reduplicated verbs in Armenian]. HSSṘ GA Teġekagir hasarakakan gitowt'yownneri 4, 41–54.

Ackema, P. and A. Neeleman (2004). Beyond morphology: Interface conditions on word formation. Number 6 in Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.

Ackerman, F. (1998). Constructions and mixed categories: Determining the semantic interpretation of person/number marking. In M. Butt and T. Holloway-King (Eds.), Proceedings of the LFG98 Conference, Stanford: CA. CSLI Publications.

Ackerman, F. and I. Nikolaeva (1997). Identity in form, difference in function: The person/number paradigm in W. Armenian and N. Ostyak. In Proceedings of the LFG97 Conference, Stanford: CA. CSLI Publications.

Ackerman, F. and I. Nikolaeva (2014). Descriptive Typology and Linguistic Theory: A study in the morphosyntax of relative clauses. Stanford, CA: CSLI Publications.

Ackerman, F., I. Nikolaeva, and R. Malouf (2004). Possessive relatives and cooperating constructions. In 11th International Conference on Head-Driven Phrase Structure Grammar, Leuven.

Ačaṙyan, H. (1971). Liakatar k'erakanowt'yown hayoc' lezvi [Complete Grammar of the Armenian Language], Volume 6. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Adjarian, H. (1909). Classification des dialectes arméniens. Paris: Librairie Honoré Champion.

Aksënova, A. (2020a, January). Sigmapie for subregular and subsequential grammar induction. Tutorial at SCIL 2020, FLT workshop. New Orleans, LA.

Aksënova, A. (2020b). Tool-assisted induction of subregular languages and mappings. Ph. D. thesis, Stony Brook University.

Aksënova, A. and A. De Santo (2018). Strict locality in morphological derivations. In Proceedings of the 53rd meeting of Chicago Linguistics Society (CLS 53).

Aksënova, A. and S. Deshmukh (2018). Formal restrictions on multiple tiers. In Proceedings of the Society for Computation in Linguistics, Volume 1, pp. 64–73.

Aksënova, A., T. Graf, and S. Moradi (2016). Morphotactics as tier-based strictly local dependencies. In Proceedings of the 14th sigmorphon workshop on computational research in phonetics, phonology, and morphology, pp. 121–130.

Al-Bataineh, A. (2015). Cent ans après: Politiques scolaires et la vitalité des langues en danger le cas de l'arménien occidental. Ph. D. thesis, Sorbonne Paris Cité.

Allen, M. R. (1979). Morphological investigations. Ph. D. thesis, University of Connecticut, Storrs, CT.

Anderson, S. R. (1985). Phonology in the twentieth century: Theories of rules and theories of representations. Chicago: University of Chicago Press.

Anderson, S. R. (1992). A-morphous morphology, Volume 62 of Cambridge Studies in Linguistics. Cambridge: Cambridge University Press.

Andersson, S., H. Dolatian, and Y. Hao (2020). Computing vowel harmony: The generative capacity of search & copy. In Proceedings of the Annual Meetings on Phonology, Volume 8.

Andonian, H. (1999). Beginner's Armenian. Hippocrene Books.

Anttila, A. (2002). Morphologically conditioned phonological alternations. Natural Language & Linguistic Theory 20(1), 1–42.

Anttila, A. (2006). Variation and opacity. Natural Language & Linguistic Theory 24(4), 893–944.

Aṙak'elyan, V. (1955). Žamanakakic' hayereni hnčyownabanowt'yown [Modern Armenian phonology]. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Aronoff, M. (1976). Word formation in generative grammar. Number 1 in Linguistic Inquiry Monographs. Cambridge, MA: The MIT Press.

Aronoff, M. (1988). Head operations and strata in reduplication: A linear treatment. In G. Booij and J. van Marle (Eds.), Yearbook of Morphology, Volume 1, pp. 1–15. Dordrecht: Foris.

Aronoff, M. (1994). Morphology by itself: Stems and inflectional classes. Number 22 in Linguistic Inquiry Monographs. London/Cambridge: MIT press.

Aronoff, M. and S. N. Sridhar (1983). Morphological levels in English and Kannada, or atarizing Reagan. In J. F. Richardson, M. Marks, and A. Chukerman (Eds.), Papers from the Parasession on the Interplay of Phonology, Morphology, and Syntax, Chicago, pp. 3–16. Chicago Linguistic Society.

Arregi, K., N. Myler, and B. Vaux (2013). Number marking in Western Armenian: A non-argument for outwardly-sensitive phonologically conditioned allomorphy. In 87th Linguistic Society of America Annual Meeting, Boston. Available online at `https://works.bepress.com/bert_vaux/4/`. Accessed 1 July 2019.

Ashton, N. (2012). Second position clitics and monadic second-order transduction. In Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing, pp. 31–41.

Athanasopoulou, A., I. Vogel, and H. Dolatian (2017). Acoustic properties of canonical and non-canonical stress in French, Turkish, Armenian and Brazilian Portuguese. Proceedings of Interspeech 2017, 1398–1402.

Avcu, E. (2018). Experimental investigation of the subregular hypothesis. In W. G. Bennet, L. Hracs, and D. R. n Storoshenko (Eds.), Proceedings of the 35th West Coast Conference on Formal Linguistics, Somerville, MA, pp. 77–86. Cascadilla Proceedings Project.

Avcu, E. (2019). Using Cognitive Neuroscience to Understand Learning Mechanisms: Evidence from Phonological Processing. Ph. D. thesis, University of Delaware.

Avcu, E., C. Shibata, and J. Heinz (2017). Subregular complexity and deep learning. In S. Dobnik and S. Lappin (Eds.), CLASP Papers in Computational Linguistics: Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017), Gothenburg, 12 –13 June, pp. 20–33.

Avetisyan, Y. S. (2007). Arewelahayereni ew arewmtahayereni zowgadrakan k'erakanowt'yown [Comparative grammar of Eastern and Western Armenian]. Yerevan: Yerevani petakan hamalsaran.

Avetisyan, Y. S. (2011). Arewelahayereni ew arewmtahayereni zowgadrakan hnčyownabanowt'yown [Comparative phonology of Eastern and Western Armenian]. Yerevan: Hayastani Hanrapetowt'yan Sp'yowrk'i Naxararowt'yown.

Bach, E. and D. Wheeler (1981). Montague phonology: A first approximation. In University of Massachusetts occasional papers in linguistics, Volume 7, pp. 27–45. Amherst, MA: Graduate Linguistic Student Association.

Baek, H. (2018). Computational representation of unbounded stress: Tiers with structural features. In Proceedings of the Chicago Linguistics Society CLS 2017, Volume 53, pp. 13–24.

Balabanian, G. (2019). A co-phonological approach to Persian loanwords within Armenian phonology. Master's thesis, University of Ottawa.

Bale, A., M. Gagnon, and H. Khanjian (2011). On the relationship between morphological and semantic markedness. Morphology 21(2), 197–221.

Bale, A. and H. Khanjian (2008). Classifiers and number marking. In Semantics and Linguistic Theory, Volume 18, pp. 73–89.

Bale, A. and H. Khanjian (2014). Syntactic complexity and competition: The singular-plural distinction in Western Armenian. Linguistic Inquiry 45(1), 1–26.

Bale, A. C., M. Gagnon, and H. Khanjian (2010). Cross-linguistic representations of numerals and number marking. In Semantics and Linguistic Theory, Volume 20, pp. 582–598.

Bardakjian, K. B. and R. W. Thomson (1977). A Textbook of Modern Western Armenian. Delmar: NY: Caravan Books.

Baronian, L. (2017). Two problems in Armenian phonology. Language and Linguistics Compass 11(8), e12247.

Baronian, L. V. (2006). North of phonology. Ph. D. thesis, Stanford University.

Barton, G. E. (1986). Computational complexity in two-level morphology. In Proceedings of the 24th annual meeting on Association for Computational Linguistics, pp. 53–59. Association for Computational Linguistics.

Barton, G. E., R. C. Berwick, and E. S. Ristad (1987). Computational complexity and natural language. Cambridge, MA: MIT press.

Beard, R. (1991). Decompositional composition: The semantics of scope ambiguities and 'bracketing paradoxes'. Natural Language & Linguistic Theory 9(2), 195–229.

Becker, M., A. Nevins, and J. Levine (2012). Asymmetries in generalizing alternations to and from initial syllables. Language 88(2), 231–268.

Becker, T. (1993). Back-formation, cross-formation, and 'bracketing paradoxes' in paradigmatic morphology. In G. Booij and J. v. Marle (Eds.), Yearbook of Morphology 1993, pp. 1–25. Dordrecht: Kluwer.

Beckman, J. N. (1997). Positional faithfulness, positional neutralisation and Shona vowel harmony. Phonology 14(1), 1–46.

Beekes, R. S. P. (2003). Historical phonology of Classical Armenian. See Kortlandt (2003), pp. 133–211.

Beesley, K. and L. Karttunen (2003). Finite-state morphology: Xerox tools and techniques. Stanford, CA: CSLI Publications.

Belk, Z. (2019). LF bracketing paradoxes: A new account. Syntax 22(1), 24–65.

Bennett, R. (2018). Recursive prosodic words in Kaqchikel (Mayan). Glossa: A journal of general linguistics 3(1).

Bennett, R. and E. Elfner (2019). The syntax–prosody interface. Annual Review of Linguistics 5, 151–171.

Benua, L. (1997). Transderivational identity: Phonological relations between words. Ph. D. thesis, Rutgers University, New Brunswick, NJ.

Bermúdez-Otero, R. (1999). Constraint interaction in language change: Quantity in English and Germanic. Ph. D. thesis, University of Manchester.

Bermúdez-Otero, R. (2008). Lexical phonology crippled by the legacy of SPE. University of Manchester.

Bermúdez-Otero, R. (2011). Cyclicity. See van Oostendorp et al. (2011), pp. 2019–2048.

Bermúdez-Otero, R. (2012). The architecture of grammar and the division of labour in exponence. In J. Trommer (Ed.), The morphology and phonology of exponence, Number 41 in Oxford Studies in Theoretical Linguistics, pp. 8–83. Oxford: Oxford University Press.

Bermúdez-Otero, R. (2014). Amphichronic explanation and the life cycle of phonological processes. In P. Honeybone and J. C. Salmons (Eds.), The Oxford handbook of historical phonology, pp. 374–399. Oxford: Oxford University Press.

Bermúdez-Otero, R. (2016). We do not need structuralist morphemes, but we do need constituent structure. See Siddiqi and Harley (2016), pp. 385–428.

Bermúdez-Otero, R. (2018). Stratal phonology. In S. Hannahs and A. R. K. Bosch (Eds.), The Routledge handbook of phonological theory, pp. 382–410. Abingdon: Routledge.

Bermúdez-Otero, R. (in prep). Stratal Optimality Theory. Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.

Bermúdez-Otero, R. and A. McMahon (2006). English phonology and morphology. In B. Aarts and A. McMahon (Eds.), The handbook of English linguistics, pp. 382–410. Oxford: Blackwell.

Bermúdez-Otero, R. and G. Trousdale (2012). Cycles and continua: On unidirectionality and gradualness in language change. In T. Nevalainen and E. C. Traugott (Eds.), The Oxford handbook of the history of English. New York: Oxford University Press.

Bezrukov, N. (2016). Number marking mismatches in Modern Armenian: A Distributed Morphology approach. Master's thesis, University of Chicago.

Bezrukov, N. and H. Dolatian (2020). Mobile affixes across Western Armenian: Conflicts across modules. In University of Pennsylvania Working Papers in Linguistics, Volume 26.

Bhaskar, S., J. Chandlee, A. Jardine, and C. Oakden (2020). Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In A. Leporati, C. Martín-Vide, D. Shapira, and C. Zandron (Eds.), Proceedings of the 14th International Conference on Language and Automata Theory and Applications (LATA 2020).

Bird, S. (Ed.) (1991a). Declarative perspectives on phonology. University of Edinburgh: Centre for Cognitive Science.

Bird, S. (1991b). Feature structures and indices. Phonology 8(1), 137–144.

Bird, S. (1991c). Focus and phrasing in unification categorial grammar. See Bird (1991a), pp. 139–165.

Bird, S. (1992). Finite-state phonology in HPSG. In Proceedings of the 14th conference on Computational linguistics-Volume 1, pp. 74–80. Association for Computational Linguistics.

Bird, S. (1994). Introduction to computational phonology. Computational Linguistics 20(3), 1–8.

Bird, S. (1995). Computational phonology: A constraint-based approach. Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Bird, S. and P. Blackburn (1991). A logical approach to Arabic phonology. In Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics, pp. 89–94. Association for Computational Linguistics.

Bird, S., J. S. Coleman, J. Pierrehumbert, and J. M. Scobbie (1992). Declarative phonology. In Proceedings of the XVth International Congress of Linguistics, University Laval, Quebec.

Bird, S. and T. M. Ellison (1994). One-level phonology: Autosegmental representations and rules as finite automata. Computational Linguistics 20(1), 55–90.

Bird, S. and E. Klein (1990). Phonological events. Journal of linguistics 26(1), 33–56.

Bird, S. and E. Klein (1994). Phonological analysis in typed feature systems. Computational linguistics 20(3), 455–491.

Bjorkman, B. and E. Dunbar (2016). Finite-state phonology predicts a typological gap in cyclic stress assignment. Linguistic Inquiry 47(2), 351–363.

Bobaljik, J. and S. Wurmbrand (2013). Suspension across domains. See Matushansky and Marantz (2013), pp. 185–198.

Bobaljik, J. D. (2000). The ins and outs of contextual allomorphy. In K. K. Grohmann and C. Struijke (Eds.), University of Maryland working papers in linguistics, Volume 10, pp. 35–71. College Park: University of Maryland.

Bobaljik, J. D. (2012). Universals in comparative morphology: Suppletion, superlatives, and the structure of words. Number 50 in Current Studies in Linguistics. Cambridge, MA: MIT Press.

Bojańczyk, M. (2014). Transducers with origin information. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias (Eds.), Automata, Languages, and Programming, Berlin, Heidelberg, pp. 26–37. Springer.

Bojanczyk, M., L. Daviaud, B. Guillon, and V. Penelle (2017). Which classes of origin graphs are generated by transducers. In I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl (Eds.), 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), Volume 80 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, pp. 114:1–114:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Bonet, E. and M.-R. Lloret (2005). More on alignment as an alternative to domains: The syllabification of Catalan clitics. Probus 17(1), 37–78.

Booij, G. (1980). Rule ordering, rule application and the organization of grammars. In W. Dressler, O. Pfeiffer, and J. Rennison (Eds.), Phonologica 1980, Volume 1981, pp. 45–56. Innsbruck: Institut für Sprachwissenschaft.

Booij, G. (1985). The interaction of phonology and morphology in prosodic phonology. In E. Gussmann (Ed.), Phono-morphology: Studies in the interaction of phonology and morphology, pp. 23–34. Lublin: Katolicki Universytet Lubelski.

Booij, G. (1987). Lexical phonology and the organization of the morphological component. In E. Gussmann (Ed.), Rules and the Lexicon, pp. 43–65. Lublin: Catholic University of Lublin.

Booij, G. (1988a). Lexical phonology and prosodic phonology. In W. Dressler, H. Luschützky, P. Oskar, and J. Rennison (Eds.), Phonologica 1988, pp. 49–62. Cambridge: Cambridge University Press.

Booij, G. (1988b). On the relation between lexical and prosodic phonology. In P. M. Bertinetto and M. Loporcaro (Eds.), Certamen phonologicum, pp. 63–76. Torino: Rosemberg and Sellier.

Booij, G. (1994). Lexical phonology: A review. Lingua e stile 29(4), 525–555.

Booij, G. (1997). Non-derivational phonology meets lexical phonology. See Roca (1997), pp. 261–288.

Booij, G. (2000). The phonology-morphology interface. In The First Glot International State of the Art Book. The Latest in Linguistics, pp. 287–306. Walter de Gruyter.

Booij, G. and R. Lieber (1993). On the simultaneity of morphological and prosodic structure. See Hargus and Kaisse (1993), pp. 23–44.

Booij, G. and J. Rubach (1984). Morphological and prosodic domains in lexical phonology. Phonology 1, 1–27.

Booij, G. and J. Rubach (1987). Postcyclic versus postlexical rules in lexical phonology. Linguistic Inquiry 18(1), 1–44.

Booij, G. E. (1996). Cliticization as prosodic integration: The case of Dutch. The Linguistic Review 13, 219.

Boyacioglu, N. (2010). Hay-Pay: Les Verbs de l'arménien occidental. France: L'Asiatheque.

Božič, J. (2019). Constraining long-distance allomorphy. The Linguistic Review 36(3), 485–505.

Bozsahin, C. (1999). Categorial morphosyntax: Transparency of the morphology-syntax-semantics interface.

Brame, M. K. (1974). The cycle in phonology: Stress in Palestinian, Maltese, and Spanish. Linguistic Inquiry 5(1), 39–60.

Broe, M. (1991a). Paradox lost: A non-derivational approach to Grassmann's law. See Bird (1991a), pp. 1–19.

Broe, M. (1991b). A unification-based approach to prosodic analysis. See Bird (1991a), pp. 27–43.

Bromberger, S. and M. Halle (1989). Why phonology is different. Linguistic inquiry 20(1), 51–70.

Büchi, J. R. (1960). Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly 6(1-6), 66–92.

Buckley, E. (2017). Global effects in Kashaya prosodic structure. See Gribanova and Shih (2017), pp. 113–139.

Burness, P. and K. McMullin (2019). Efficient learning of output tier-based strictly 2-local functions. In Proceedings of the 16th Meeting on the Mathematics of Language, pp. 78–90.

Burness, P. A. and K. McMullin (2020). Modelling non-local maps as strictly piecewise functions. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Burzio, L. (1998). Multiple correspondence. Lingua 104(1-2), 79–109.

Burzio, L. (2011). Derived environment effects. See van Oostendorp et al. (2011), pp. 2089–2114.

Caha, P. (2013). Explaining the structure of case paradigms by the mechanisms of nanosyntax. Natural Language & Linguistic Theory 31(4), 1015–1066.

Calder, J. and S. Bird (1991). Defaults in underspecification phonology. See Bird (1991a), pp. 107–125.

Carden, G. (1983). The non-finite-state-ness of the word formation component. Linguistic Inquiry 14(3), 537–541.

Cardinaletti, A. and L. Repetti (2009). Phrase-level and word-level syllables: Resyllabification and prosodization of clitics. See Grijzenhout and Kabak (2009), pp. 79–104.

Carson-Berndsen, J. (1998). Time map phonology: Finite state models and event logics in speech recognition, Volume 5 of Text, Speech and Language Technology. Dordrecht: Springer Science & Business Media.

Casali, R. F. (1997). Vowel elision in hiatus contexts: Which vowel goes? Language 73(3), 493–533.

Chae, H.-R. (1990). Function argument structure, category raising, and bracketing paradoxes. In Y. No and M. Libucha (Eds.), Proceedings of the Seventh Eastern States Conference on Linguistics (ESCOL), Volume 90, pp. 28–39. ERIC.

Chae, H.-R. (1993). A categorial analysis of bracketing paradoxes in English. Language Research 29(2), 163–187.

Chandlee, J. (2014). Strictly Local Phonological Processes. Ph. D. thesis, University of Delaware, Newark, DE.

Chandlee, J. (2017). Computational locality in morphological maps. Morphology 27(4), 1–43.

Chandlee, J. (2019). A computational account of tone sandhi interaction. In Proceedings of the Annual Meetings on Phonology, Volume 7.

Chandlee, J., A. Athanasopoulou, and J. Heinz (2012). Evidence for classifying metathesis patterns as subsequential. In J. Choi, E. A. Hogue, J. Punske, D. Tat, J. Schertz, and A. Trueman (Eds.), The Proceedings of the 29th West Coast Conference on Formal Linguistics, Somerville, MA, pp. 303–309. Cascillida Press.

Chandlee, J., R. Eyraud, and J. Heinz (2014). Learning strictly local subsequential functions. Transactions of the Association for Computational Linguistics 2, 491–503.

Chandlee, J., R. Eyraud, and J. Heinz (2015). Output strictly local functions. In 14th Meeting on the Mathematics of Language, pp. 112–125.

Chandlee, J., R. Eyraud, J. Heinz, A. Jardine, and J. Rawski (2019, 18–19 July). Learning with partially ordered representations. In Proceedings of the 16th Meeting on the Mathematics of Language, Toronto, Canada, pp. 91–101. Association for Computational Linguistics.

Chandlee, J. and J. Heinz (2012). Bounded copying is subsequential: Implications for metathesis and reduplication. In Proceedings of the 12th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, SIGMORPHON '12, Montreal, Canada, pp. 42–51. Association for Computational Linguistics.

Chandlee, J. and J. Heinz (2017). Computational phonology. In M. Aronoff (Ed.), Oxford Research Encyclopedia of Linguistics. Oxford University Press.

Chandlee, J. and J. Heinz (2018). Strict locality and phonological maps. Linguistic Inquiry 49(1), 23–60.

Chandlee, J., J. Heinz, and A. Jardine (2018). Input strictly local opaque maps. Phonology 35(2), 171–205.

Chandlee, J. and A. Jardine (2014). Learning phonological mappings by learning strictly local functions. In Proceedings of the Annual Meetings on Phonology, Volume 1.

Chandlee, J. and A. Jardine (2019a). Autosegmental input strictly local functions. Transactions of the Association for Computational Linguistics 7, 157–168.

Chandlee, J. and A. Jardine (2019b). Quantifier-free least fixed point functions for phonology. In Proceedings of the 16th Meeting on the Mathematics of Language (MoL 16), Toronto, Canada. Association for Computational Linguistics.

Chandlee, J., A. Jardine, and J. Heinz (2015). Learning repairs for marked structures. In Proceedings of the Annual Meetings on Phonology, Volume 2.

Chandlee, J. and C. Koirala (2014). Learning local phonological rules. In Proceedings of the 37th Penn Linguistics Conference.

Chelliah, S. L. (1995). An autolexical account of voicing assimilation in Manipuri. In E. Schiller, E. Steinberg, and B. Need (Eds.), Autolexical Theory: Ideas and Methods, Number 85 in Trends in Linguistics: Studies and Monographs, pp. 11–30. Berlin: Mouton De Gruyter.

Cheng, L. L.-S. and L. J. Downing (2016). Phasal syntax= cyclic phonology? Syntax 19(2), 156–191.

Chew, P. (2003). A computational phonology of Russian. Universal-Publishers.

Cho, Y.-m. Y. (2009). A historical perspective on non-derived environment blocking: The case of Korean palatalization. See Hanson and Inkelas (2009), pp. 461–486.

Chomsky, N. (1956). Three models for the description of language. IRE Transactions on information theory 2(3), 113–124.

Chomsky, N. (1957). Syntactic Structures. Berlin: Mouton de Gruyter.

Chomsky, N. and M. Halle (1968). The sound pattern of English. Cambridge, MA: MIT Press.

Chomsky, N., M. Halle, and F. Lukoff (1956). On accent and juncture in English. In For Roman Jakobson, pp. 65–80. The Hague: Mouton.

Church, K. W. (1983). Phrase-structure parsing: A method for taking advantage of allophonic constraints. Ph. D. thesis, Massachusetts Institute of Technology.

Clackson, J. (1994). The linguistic relationship between Armenian and Greek. Oxford: Blackwell.

Cohen-Sygal, Y. and S. Wintner (2006). Finite-state registered automata for non-concatenative morphology. Computational Linguistics 32(1), 49–82.

Cohn, A. C. (1989). Stress in Indonesian and bracketing paradoxes. Natural language & linguistic theory 7(2), 167–216.

Cole, J. (1990). Arguing for the phonological cycle: A critical review. In D. Meyer, S. Tomioka, and L. Zidani-Eroglue (Eds.), Proceedings of the Formal Linguistics Society of Midamerica, Madison, WI, pp. 51–67. Linguistics Student Association: University of Wisconson.

Cole, J. (1995a). The cycle in phonology. In J. Goldsmith (Ed.), The Handbook of Phonological Theory (1 ed.)., pp. 70–113. Cambridge, MA: Blackwell Publishers.

Cole, J. (1995b). Eliminating cyclicity as a source of complexity in processing phonology. See Cole et al. (1995), pp. 255–280.

Cole, J. and J. Coleman (1992). No need for cyclicity in generative phonology. In C. P. Canakis, G. P. Chan, and J. M. Denton (Eds.), Proceedings of the 28th Regional Meeting of the Chicago Linguistics Society, Volume 2, Chicago, IL, pp. 36–50. University of Chicago.

Cole, J., G. M. Green, and J. L. Morgan (Eds.) (1995). Linguistics and Computation. Number 52 in CSLI Lecture Notes. Stanford, CA: CSLI Publications.

Coleman, J. (1992). The phonetic interpretation of headed phonological structures containing overlapping constituents. Phonology 9(1), 1–44.

Coleman, J. (1993). English word-stress in unification-based grammar. See Ellison and Scobbie (1993), pp. 97–106.

Coleman, J. (1995a). Declarative lexical phonology. In J. Durand and F. Katamba (Eds.), Frontiers of phonology: Atoms, structures, derivations, pp. 333–383. London: Longman.

Coleman, J. (1995b). Phonology and computational linguistics: A personal overview. See Cole et al. (1995), pp. 223–254.

Coleman, J. (1996). Declarative syllabification in Tashlhit Berber. In J. Durand and B. Laks (Eds.), Current trends in phonology: Models and methods, Volume 1, pp. 175–216. Salford: European Studies Research Institute, University of Salford.

Coleman, J. (1998). Phonological representations: Their names, forms and powers. Cambridge University Press.

Coleman, J. (2000). Candidate selection. The Linguistic Review 17(2-4), 167–180.

Coleman, J. (2006). Declarative approaches to phonology. In K. Brown (Ed.), Encyclopedia of Language & Linguistics (2 ed.)., pp. 374–375. Elsevier.

Coleman, J. (2011). Phonology as computation. See Goldsmith et al. (2011), pp. 596–630.

Coleman, J. and J. Local (1991). The "no crossing constraint" in autosegmental phonology. Linguistics and Philosophy 14(3), 295–338.

Coleman, J. and J. Local (1992). Monostratal phonology and speech synthesis. In P. Tench (Ed.), Studies in systemic phonology, pp. 183–193. London: Pinter Publishers.

Coleman, J. and J. Pierrehumbert (1997). Stochastic phonological grammars and acceptability. In Third meeting of the ACL special interest group in computational phonology: Proceedings of the workshop, East Stroudsburg, PA, pp. 49–56. Association for computational linguistics.

Coleman, J. S. (1991). Prosodic structure, parameter-setting and ID/LP grammar. See Bird (1991a), pp. 65–78.

Collie, S. (2007). English stress preservation and stratal optimality theory. Ph. D. thesis, University of Edinburgh.

Collie, S. (2008). English stress preservation: The case for 'fake cyclicity'. English Language & Linguistics 12(3), 505–532.

Comon, H., M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi (2007). Tree automata techniques and applications. http://www.grappa.univ-lille3.fr/tata.

Comrie, B. (1984). Some formal properties of focus in Modern Eastern Armenian. Annual of Armenian Linguistics 5, 1–21.

Connolly, M. (1972). A descriptive normalization of the Classical Armenian nominal system. Revue des Études Arméniennes NS 9, 3–45.

Courcelle, B. (1994). Monadic second-order definable graph transductions: A survey. Theoretical Computer Science 126(1), 53–75.

Courcelle, B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1, pp. 313–400. World Scientific.

Courcelle, B. and J. Engelfriet (2012). Graph Structure and Monadic Second-Order Logic, a Language Theoretic Approach. Cambridge University Press.

Crosswhite, K. (2001). Vowel reduction in optimality theory. Outstanding dissertations in linguistics. New York & London: Routledge.

Crysmann, B. (1999). Morphosyntactic paradoxa in Fox. In G. Bouma, E. W. Hinrichs, G.-J. M. Kruijff, and R. T. Oehrle (Eds.), Constraints and resources in natural language syntax and semantics, Number 3 in Studies in Constraint-Based Lexicalism, Stanford, pp. 41–61. CSLI.

Culy, C. (1985). The complexity of the vocabulary of Bambara. Linguistics and philosophy 8, 345–351.

Czaykowska-Higgins, E. (1993). Cyclicity and stress in Moses-Columbia Salish (Nxa'amxcin). Natural Language & Linguistic Theory 11(2), 197–278.

Czaykowska-Higgins, E. (1997). The morphological and phonological constituent structure of words in Moses-Columbia Salish (Nxa'amxcín). Trends in linguistics studies and monographs 107, 153–196.

Dai, H. and R. Futrell (2020). Information-theoretic characterization of the sub-regular hierarchy. In Proceedings of the Society for Computation in Linguistics, Volume 3.

d'Alessandro, R. and T. Scheer (2015). Modular PIC. Linguistic Inquiry 46(4), 593–624.

Daniel, M. and V. Khurshudian (2015). Valency classes in Eastern Armenian. In A. Malchukov and B. Comrie (Eds.), Valency Classes in the World's Languages, pp. 483–540. Berlin & Boston: De Gruyter Mouton.

Danis, N. and A. Jardine (2019). Q-theory representations are logically equivalent to autosegmental representations. In Proceedings of the Society for Computation in Linguistics, Volume 2, pp. 29–38.

Davidian, R. D. (1987). The maintenance of the Armenian language and culture in the United States among: Immigrant Armenian high school students. Ph. D. thesis, Northwestern University.

De Santo, A. (2018). Extending TSL to account for interactions of local and non-local constraints. In Proceedings of the Society for Computation in Linguistics, Volume 1.

De Santo, A. and T. Graf (2019). Structure sensitive tier projection: Applications and formal properties. In Formal Grammar. Lecture Notes in Computer Science, Volume 11668, Berlin/Heidelberg. Springer.

Deal, A. R. (2016). Plural exponence in the Nez Perce : A DM analysis. Morphology 26(3-4), 313–339.

Deal, A. R. and M. Wolf (2017). Outwards-sensitive phonologically-conditioned allomorphy in Nez Perce. See Gribanova and Shih (2017), pp. 29–60.

DeLisi, J. (2018). Armenian prosody in typology and diachrony. Language Dynamics and Change 8(1), 108–133.

DeLisi, J. L. (2015). Epenthesis and prosodic structure in Armenian: A diachronic account. Ph. D. thesis, University of California, Los Angeles, Los Angeles, CA.

Di Sciullo, A.-M. and E. Williams (1987). On the definition of word. Number 14 in Linguistic inquiry monographs. Cambridge, MA: MIT Press.

Dirksen, A. (1993). Phrase structure phonology. See Ellison and Scobbie (1993), pp. 81–96.

Dixon, R. M. and A. Y. Aikhenvald (2003). Word: A cross-linguistic typology. Cambridge: Cambridge University Press.

Dolatian, H. (2019a). Armenian prosody: A case for prosodic stems. In Proceedings of the 53rd Annual Meeting of the Chicago Linguistics Society. Chicago: Chicago Linguistics Socieity.

Dolatian, H. (2019b). Cyclicity and prosody in Armenian stress-assignment. In University of Pennsylvania Working Papers in Linguistics, Volume 25.

Dolatian, H. and J. Heinz (2018a, September). Learning reduplication with 2-way finite-state transducers. In O. Unold, W. Dyrka, , and W. Wieczorek (Eds.), Proceedings of Machine Learning Research: International Conference on Grammatical Inference, Volume 93 of Proceedings of Machine Learning Research, Wroclaw, Poland, pp. 67–80.

Dolatian, H. and J. Heinz (2018b). Modeling reduplication with 2-way finite-state transducers. In Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Brussells, Belgium. Association for Computational Linguistics.

Dolatian, H. and J. Heinz (2019a). Redtyp: A database of reduplication with computational models. In Proceedings of the Society for Computation in Linguistics, Volume 2. Article 3.

Dolatian, H. and J. Heinz (2019b). Reduplication with finite-state technology. In Proceedings of the 53rd Annual Meeting of the Chicago Linguistics Society. Chicago: Chicago Linguistics Socieity.

Dolatian, H. and J. Heinz (forthcoming). Computing and classifying reduplication with 2-way finite-state transducers. Journal of Language Modeling.

Dolatian, H. and J. Rawski (2020a). Finite-state locality in Semitic root-and-pattern morphology. In University of Pennsylvania Working Papers in Linguistics.

Dolatian, H. and J. Rawski (2020b). Multi input strictly local functions for templatic morphology. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Don, J. (2004). Categories in the lexicon. Linguistics 42, 931–956.

Don, J. (2005). On conversion, relisting and zero-derivation. SKASE Journal of Theoretical Linguistics 2(2), 2–16.

Donabédian, A. (1991). L'article dans l'économie des catégories nominales en arménien occidental moderne. Ph. D. thesis, Université Paris III.

Donabédian, A. (1993). Le pluriel en arménien moderne. Faits de langues 1(2), 179–188.

Donabédian, A. (1997). Neutralisation de la diathèse des participes en *-ac* de l'arménien moderne occidental. Studi italiani di linguistica teorica ed applicata 26(2), 327–339.

Donabédian, A. (2000). De l'arménien classique à l'arménien moderne: Typologie, ordre des mots et contact linguistique. Cahiers de Linguistique de l'INALCO 3, 34–54.

Donabédian, A. (2001a). Tabou linguistique en arménien occidental: 'gor' progressif est-il 'turc'? In A. Donabédian (Ed.), Langues de diaspora, languages en contact. Faits de langue 18, pp. 201–210. Paris: Ophrys.

Donabédian, A. (2001b). Towards a semasiological account of evidentials: An enunciative approach of-er in modern Western Armenian. Journal of pragmatics 33(3), 421–442.

Donabédian, A. (2004). Arménien. In P. J. Arnaud (Ed.), Le nom composé: Données sur seize langues, pp. 3–20. Lyon: Presses Universitaires de Lyon.

Donabédian, A. (2016). The aorist in Modern Armenian: Core value and contextual meanings. In Z. Guentchéva (Ed.), Aspectuality and temporality: Descriptive and theoretical issues, Volume 172, pp. 375. John Benjamins Publishing Company.

Donabédian, A. (2018). Middle east and beyond-Western Armenian at the crossroads: A sociolinguistic and typological sketch. In C. Bulut (Ed.), Linguistic minorities in Turkey and Turkic-speaking minorities of the periphery, pp. 89–148. Wiesbaden: Harrazowitz Verlag.

Downing, L. J. (1998a). On the prosodic misalignment of onsetless syllables. Natural Language & Linguistic Theory 16(1), 1–52.

Downing, L. J. (1998b). Prosodic misalignment and reduplication. In G. Booij and J. van Marle (Eds.), Yearbook of Morphology 1997, pp. 83–120. Dordrecht: Kluwer Academic Publishers.

Downing, L. J. (1999a). Prosodic stem ≠ prosodic word in Bantu. See Hall and Kleinhenz (1999), pp. 73–98.

Downing, L. J. (1999b). Verbal reduplication in three Bantu languages. See Kager et al. (1999), pp. 62–89.

Downing, L. J. (2005). Morphological complexity and prosodic minimality. Catalan journal of linguistics 4, 83–106.

Downing, L. J. (2006). Canonical forms in prosodic morphology. Number 12 in Oxford studies in Theoretical Linguistics. Oxford University Press.

Downing, L. J. (2016). The prosodic hierarchy in Chichewa: How many levels? Studies in Prosodic Grammar 1, 5–33.

Downing, L. J., T. A. Hall, and R. Raffelsiefen (Eds.) (2005). Paradigms in phonological theory. Oxford: Oxford University Press.

Downing, L. J. and M. Kadenge (to appear). Re-placing PStem in the prosodic hierarchy. The Linguistic Review.

Duanmu, S. (1999). Alignment and the cycle are different. In B. Hermans and M. van Oostendorp (Eds.), The Derivational Residue in Phonological Optimality Theory, Volume 28, pp. 129–152. John Benjamins Publishing.

Dum-Tragut, J. (2009). Armenian: Modern Eastern Armenian. Number 14 in London Oriental and African Language Library. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Elfner, E. (2015). Recursion in prosodic phrasing: Evidence from Connemara Irish. Natural Language & Linguistic Theory 33(4), 1169–1208.

Ellison, T. M. (1993). Concatenation vs conjunction in constraint based phonology. See Ellison and Scobbie (1993), pp. 1–18.

Ellison, T. M. (1994). The machine learning of phonological structure. Ph. D. thesis, University of Western Australia.

Ellison, T. M. and J. Scobbie (Eds.) (1993). Computational Phonology. University of Edinburgh: Centre for Cognitive Science.

Elordieta, G. (1997). Morphosyntactic feature chains and phonological domains. Ph. D. thesis, University of Southern California.

Elordieta, G. (2008). An overview of theories of the syntax-phonology interface. Anuario del Seminario de Filología Vasca "Julio de Urquijo" 42(1), 209–286.

Elordieta, G. (2014). The word in phonology. In I. I.-A. nano and J.-L. Mendívil-Giró (Eds.), To be or not to be a Word. New Reflections on the Definition of Word, pp. 6–65. Newcastle upon Tyne: Cambridge Scholars Publishing.

Ēloyan, S. A. (1972). Hodakapov ew anhodakap bardowt'yownnerë žamanakakic' hayerenowm [The compound words with and without copulative particles in modern Armenian]. Lraber Hasarakakan Gitowt'yownneri 7, 77–85.

Embick, D. (2010). Localism versus globalism in morphology and phonology, Volume 60 of Linguistic Inquiry Monographs. Cambridge, MA: MIT Press.

Embick, D. (2013). Morphemes and morphophonological loci. See Matushansky and Marantz (2013), pp. 151–166.

Embick, D. (2014). Phase cycles, φ-cycles, and phonological (in) activity. In S. Bendjaballah, N. Faust, M. Lahrouchi, and N. Lampitelli (Eds.), The form of structure, the structure of forms: Essays in honor of Jean Lowenstamm, pp. 271–86. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Embick, D. (2015). The morpheme: A theoretical introduction, Volume 31. Boston and Berlin: Walter de Gruyter.

Embick, D. and R. Noyer (2001). Movement operations after syntax. Linguistic Inquiry 32(4), 555–595.

Enderton, H. (2001). A mathematical introduction to logic. Academic Press.

Engelfriet, J. and H. J. Hoogeboom (2001, April). MSO definable string transductions and two-way finite-state transducers. ACM Trans. Comput. Logic 2(2), 216–254.

Ermolaeva, M. and D. Edmiston (2018). Distributed morphology as a regular relation. In Proceedings of the Society for Computation in Linguistics, Volume 1, pp. 178–181.

Ezekyan, L. K. (2007). Hayoc' lezow [Armenian language]. Yerevan: Yerevani Petakan Hamalsarani Hratarakčowt'yown.

Fabb, N. (1988). English suffixation is constrained only by selectional restrictions. Natural Language & Linguistic Theory 6(4), 527–539.

Fairbanks, G. H. (1948). Phonology and morphology of modern spoken West Armenian. Ph. D. thesis, University of Wisconsin-Madison, Madison, WI.

Falk, Y. N. (1991). Bracketing paradoxes without brackets. Lingua 84(1), 25–42.

Filiot, E. and P.-A. Reynier (2016, August). Transducers, logic and algebra for functions of finite words. ACM SIGLOG News 3(3), 4–19.

Finley, S. (2008a). Formal and cognitive restrictions on vowel harmony. Ph. D. thesis, Rutgers University.

Finley, S. (2008b). The interaction of vowel harmony and epenthesis. In Proceedings from the Annual Meeting of the Chicago Linguistic Society, Volume 44, pp. 95–109. Chicago Linguistic Society.

Finley, S. (2010). Exceptions in vowel harmony are local. Lingua 120(6), 1549–1566.

Fitzpatrick, J., A. Nevins, and B. Vaux (2004). Exchange rules & feature-value variables. Handout for 3rd North American Phonology Conference.

Fitzpatrick-Cole, J. (1994). The prosodic domain hierarchy in reduplication. Ph. D. thesis, Stanford University, Stanford, CA.

Frota, S. and M. Vigário (2013). Review of *Prosody matters: Essays in honor of Elisabeth Selkirk* by Borowsky et al. (2012). Phonology 30(1), 165–172.

Fukushima, K. (1999). Bound morphemes, coordination and bracketing paradox. Journal of Linguistics 35(2), 297–320.

Fukushima, K. (2014). Morpho-semantic bracketing paradox and compositionality: The implications of the sized inalienable possession construction in japanese. Journal of Linguistics 50(1), 91–141.

Fukushima, K. (2015). Compositionality, lexical integrity, and agglutinative morphology. Language Sciences 51, 67–85.

Fulmer, S. L. (1991). Dual-position affixes in Afar: An argument for phonologically driven morphology. In A. L. Halpern (Ed.), Proceedings of the Ninth West Coast Conference on Formal Linguistics., Stanford, pp. 189–203. CSLI.

Fulmer, S. L. (1997). Parallelism and planes in Optimality Theory: Evidence from Afar. Ph. D. thesis, University of Arizona, Tucson, AZ.

Gainor, B., R. Lai, and J. Heinz (2012). Computational characterizations of vowel harmony patterns and pathologies. In J. Choi, E. A. Hogue, J. Punske, D. Tat, J. Schertz, and A. Trueman (Eds.), The Proceedings of the 29th West Coast Conference on Formal Linguistics, Somerville, MA, pp. 63–71. Cascillida Press.

Galstyan, L. N. (2004). Grabari soskaçanc'avor bayeri hamapatasxanowt'yownnerë arewmtahayerenowm [Correspondences of derivative verbs of Classical Armenian in Western Armenian]. Lraber Hasarakakan Gitowt'yownneri 2, 178–185.

Ġaragyowlyan, T. (1979). Hayereni gaġtnavankayin ë-i hnčman himnakan aṙanjnahatkowt'yownnerë [Primary characteristics of epenthetic schwas]. Lraber Hasarakakan Gitowt'yownneri 12, 35–45.

Ġaragyowlyan, T. A. (1974). Žamanakakic' hayereni owġġaxosowt'yownë [Moderrn Armenian orthoepy]. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Garrett, A. (1998). Adjarian's law, the glottalic theory, and the position of Armenian. In Annual Meeting of the Berkeley Linguistics Society, Volume 24, pp. 12–23.

Gibbon, D. (1987). Finite state processing of tone systems. In Third Conference of the European Chapter of the Association for Computational Linguistics, Copenhagen, Denmark, pp. 291–297. Association for Computational Linguistics.

Gibbon, D. (2001a). Finite state prosodic analysis of African corpus resources. In P. Dalsgaard, B. Lindberg, H. Benner, and Z. Tan (Eds.), EUROSPEECH 2001 Scandinavia, 7th European Conference on Speech Communication and Technology, 2nd INTERSPEECH Event, Aalborg, Denmark, September 3-7, 2001, pp. 83–86. ISCA.

Gibbon, D. (2001b). Preferences as defaults in computational phonology. In K. Dziubalska-Kolaczyk (Ed.), Constraints and Preferences, pp. 143–200. Mouton De Gruyter.

Giegerich, H. J. (1999). Lexical strata in English: Morphological causes, phonological effects. Number 89 in Cambridge studies in linguistics. Cambridge: Cambridge University Press.

Gildea, D. and D. Jurafsky (1996). Learning bias and phonological-rule induction. Computational Linguistics 22(4), 497–530.

Giorgi, A. (2011). Remarks on temporal anchoring: The case of Armenian aorist. University of Venice Working Papers in Linguistics 21, 89–110.

Giorgi, A. and S. Haroutyunian (2016). Word order and information structure in Modern Eastern Armenian. Journal of the Society for Armenian Studies 25, 185–200.

Giorgi, A. and S. Haroutyunian (2019). Indirect reports in Modern Eastern Armenian. In Indirect reports and pragmatics in the world languages, pp. 277–298. Springer.

Godel, R. (1975). An introduction to the study of Classical Armenian. Belgium: Reichert Wiesbaden.

Godson, L. I. (2004). Phonetics of language attrition: Vowel production and articulatory setting in the speech of Western Armenian heritage speakers. Ph. D. thesis, University of California, San Diego.

Goldsmith, J. (2012). Theory, kernels, data, methods. In A. Beltrama (Ed.), Proceedings from the Annual Meeting of the Chicago Linguistic Society, Volume 48, pp. 263–277. Chicago Linguistic Society.

Goldsmith, J., J. Riggle, and A. C. L. Yu (Eds.) (2011). The Handbook of Phonological Theory (2 ed.). Oxford: Blackwell.

González, C. (2005). Phonologically-conditioned allomorphy in Panoan: Towards an analysis. In J. Heinz, A. Martin, and K. Pertsova (Eds.), UCLA Working Papers in Linguistics: Papers in phonology, Volume 6, pp. 39–56. Los Angeles: UCLA, Department of Linguistics.

Gordon, M. (2002). A factorial typology of quantity-insensitive stress. Natural Language & Linguistic Theory 20(3), 491–552.

Gouskova, M. (2010). The phonology of boundaries and secondary stress in Russian compounds. The Linguistic Review 27(4), 387–448.

Gouskova, M. and K. Roon (2008). Interface constraints and frequency in Russian compound stress. In Proceedings of FASL, Volume 17, pp. 49–63.

Graf, T. (2010a). Comparing incomparable frameworks: A model theoretic approach to phonology. In University of Pennsylvania Working Papers in Linguistics, Volume 16.

Graf, T. (2010b). Logics of phonological reasoning. Master's thesis, University of California, Los Angeles.

Graf, T. (2013). Local and transderivational constraints in syntax and semantics. Ph. D. thesis, University of California, Los Angeles.

Graf, T. (2017). The power of locality domains in phonology. Phonology 34(2), 385–405.

Graf, T. (2019a). Monotonicity as an effective theory of morphosyntactic variation. Journal of Language Modelling 7(2), 3–47.

Graf, T. (2019b). A subregular bound on the complexity of lexical quantifiers. In J. J. Schlöder, D. McHugh, and F. Roelofsen (Eds.), Proceedings of the 22nd Amsterdam Colloquium, pp. 455–464.

Graf, T. and A. De Santo (2019). Sensing tree automata as a model of syntactic dependencies. In Proceedings of the 16th Meeting on the Mathematics of Language, Toronto, Canada, pp. 12–26. Association for Computational Linguistics.

Graf, T. and C. Mayer (2018, October). Sanskrit n-retroflexion is input-output tier-based strictly local. In Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology, Brussels, Belgium, pp. 151–160. Association for Computational Linguistics.

Graf, T. and N. Shafiei (2019a). C-command dependencies as TSL string constraints. In Proceedings of the Society for Computation in Linguistics, pp. 205–215.

Graf, T. and N. Shafiei (2019b). C-command dependencies as TSL string constraints. In G. Jarosz, M. Nelson, B. O'Connor, and J. Pater (Eds.), Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pp. 205–215.

Greppin, J. A. C. and A. A. Khachaturian (1986). Handbook of Armenian Dialectology. Delmar, NY: Caravan Books.

Gribanova, V. (2010). Composition and locality: The morphosyntax and phonology of the Russian verbal complex. Ph. D. thesis, University of California, Santa Cruz.

Gribanova, V. (2015). Exponence and morphosyntactically triggered phonological processes in the Russian verbal complex. Journal of Linguistics 51(3), 519–561.

Gribanova, V. and B. Harizanov (2017). Locality and directionality in inward-sensitive allomorphy: Russian and Bulgarian. See Gribanova and Shih (2017), pp. 61–90.

Gribanova, V. and S. S. Shih (Eds.) (2017). The morphosyntax-phonology connection: Locality and directionality at the interface. Oxford University Press.

Grijzenhout, J. and B. Kabak (Eds.) (2009). Phonological domains: Universals and deviations. Number 16 in Interface explorations. Berlin: Mouton de Gruyter.

Gross, T. (2011a). Bracketing paradoxes in dependency morphology. In Language and Culture, Volume 24, pp. 1–28.

Gross, T. (2011b). Transformational grammarians and other paradoxes. In 5th International Conference on Meaning-Text Theory, pp. 88–97.

Guekguezian, P. A. (2017a). Prosodic recursion and syntactic cyclicity inside the word. Ph. D. thesis, University of Southern California.

Guekguezian, P. A. (2017b). Templates as the interaction of recursive word structure and prosodic well-formedness. Phonology 34(1), 81–120.

Gulakian, B. (1965). K voprosu o fonologicheskoi sisteme armianskikh glasnykh [on the phonological system of Armenian vowels]. Patma-banasirakan handes 1, 219–230.

Gulian, K. H. (1902). Elementary modern Armenian grammar. Heidelberg: Julius Groos.

Güneş, G. (2015). Deriving prosodic structures. Ph. D. thesis, University of Groningen.

Guzzo, N. B. (2018). The prosodic representation of composite structures in Brazilian Portuguese. Journal of Linguistics 54(4), 683–720.

Hacopian, N. (2003). A three-way vot contrast in final position: Data from Armenian. Journal of the International Phonetic Association 33(1), 51–80.

Haghverdi, V. (2016). Armenian schwa: A phonetic and phonological analysis. Master's thesis, Rutgers University, New Brunswick, NJ.

Hagopian, G. (2005). Armenian for everyone: Western and Eastern Armenian in parallel lessons. Ann Arbor, MI: Caravan Books.

Haig, H. A. (1980). Temporal adverbial clauses in Modern Western Armenian. Master's thesis, University of California, Los Angeles.

Hall, T. A. (1999). The phonological word: A review. See Hall and Kleinhenz (1999), pp. 1–22.

Hall, T. A. and U. Kleinhenz (Eds.) (1999). Studies on the phonological word, Volume 174. Amsterdam/Philadelphia: John Benjamins Publishing.

Halle, M. and M. Kenstowicz (1991). The free element condition and cyclic versus noncyclic stress. Linguistic inquiry 22(3), 457–501.

Halle, M. and A. Marantz (1993). Distributed morphology and the pieces of inflection. In K. Hale and S. J. Keyser (Eds.), The view from Building 20: Studies in linguistics in honor of Sylvaln Bromberger, pp. 111–176. Cambridge, MA: MIT Press.

Halle, M. and A. Nevins (2009). Rule application in phonology. See Raimy and Cairns (2009), pp. 355–383.

Halle, M. and B. Vaux (1998). Theoretical aspects of Indo-European nominal morphology: The nominal declensions of Latin and Armenian. In J. Jasanoff, H. C. Melchert, and L. Oliver (Eds.), Mír Curad: Studies in honor of Calvert Watkins, pp. 223–240. Innsbruk, Austria: Innsbrucker Beiträge zur Sprachsissenschaft.

Halle, M., B. Vaux, and A. Wolfe (2000). On feature spreading and the representation of place of articulation. Linguistic Inquiry 31(3), 387–444.

Halle, M. and J.-R. Vergnaud (1987a). An essay on stress. Cambridge, MA: MIT Press.

Halle, M. and J.-R. Vergnaud (1987b). Stress and the cycle. Linguistic Inquiry 18(1), 45–84.

Hammalian, S. J. (1984). A generative phonology of Old Armenian. Ph. D. thesis, New York University, New York City.

Hammond, M. (1992). Morphemic circumscription. In G. Booij and J. van Marle (Eds.), Yearbook of Morphology 1991, pp. 195–209. Springer.

Hammond, M. (1993). On the absence of category-changing prefixes in English. Linguistic Inquiry 24(3), 562–567.

Hammond, M. (2009). One-level finite-state phonology. In W. D. Lewis, S. Karimi, H. Harley, and S. O. Farrar (Eds.), Time and Again: Theoretical Perspectives on Formal Linguistics: In Honor of D. Terence Langendoen, Volume 135, pp. 209–226. John Benjamins Publishing.

Han, E. (1995). Prosodic structure in compounds. Ph. D. thesis, Stanford University, Stanford, CA.

Hanson, K. and S. Inkelas (Eds.) (2009). The Nature of the Word: Studies in Honor of Paul Kiparsky. Current Studies in Linguistics. Cambridge, MA: The MIT Press.

Hao, Y. (2020). Metrical grids and generalized tier projection. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Hao, Y. and S. Andersson (2019). Unbounded stress in subregular phonology. In Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, Florence, Italy, pp. 135–143. Association for Computational Linguistics.

Hao, Y. and D. Bowers (2019). Action-sensitive phonological dependencies. In Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, Florence, Italy, pp. 218–228. Association for Computational Linguistics.

Harðarson, G. R. (2016). Peeling away the layers of the onion: On layers, inflection and domains in icelandic compounds. The Journal of Comparative Germanic Linguistics 19(1), 1–47.

Harðarson, G. R. (2017). Cycling through grammar: On compounds, noun phrases and domains. Ph. D. thesis, University of Connecticut.

Harðarson, G. R. (2018). Forming a compound and spelling it out. In University of Pennsylvania Working Papers in Linguistics, Volume 24, pp. 11.

Hargus, S. (1985). The lexical phonology of Sekani. Ph. D. thesis, UCLA.

Hargus, S. (1993). Modeling the phonology–morphology interface. See Hargus and Kaisse (1993), pp. 45–74.

Hargus, S. and E. M. Kaisse (Eds.) (1993). Studies in lexical phonology, Volume 4 of Phonetics and Phonology. San Diego: Academic Press.

Harley, H. (2009). Compounding in distributed morphology. In R. Lieber and P. Štekauer (Eds.), The Oxford handbook of compounding, pp. 129–143. Oxford/New York: Oxford University Press.

Harley, H. (2014). On the identity of roots. Theoretical linguistics 40(3/4), 225–276.

Haroutyunian, S. (2011). An analysis of Dante's tenses in the Armenian translations of the Divina Commedia. Ph. D. thesis, Università Ca'Foscari Venezia.

Harris, J. (1987). Non-structure-preserving rules in lexical phonology: Southeastern Bantu harmony. Lingua 72(4), 255–292.

Haspelmath, M. (2020). The morph as a minimal linguistic form. Morphology 30(2), 117–134.

Haugen, J. D. (2016). Readjustment: Rejected. See Siddiqi and Harley (2016), pp. 303–342.

Haugen, J. D. and H. Harley (2013). Head-marking inflection and the architecture of grammatical theory. In S. T. Bischoff, D. Cole, A. V. Fountain, and M. Miyashita (Eds.), The Persistence of Language: Constructing and confronting the past and present in the voices of Jane H. Hill, Volume 8, pp. 133. Amsterdam/Philadelphia: John Benjamins Publishing.

Heinz, J. (2007). The Inductive Learning of Phonotactic Patterns. Ph. D. thesis, University of California, Los Angeles.

Heinz, J. (2009). On the role of locality in learning stress patterns. Phonology 26(2), 303–351.

Heinz, J. (2010). Learning long-distance phonotactics. Linguistic Inquiry 41(4), 623–661.

Heinz, J. (2014). Culminativity times harmony equals unbounded stress. In H. van der Hulst (Ed.), Word stress: Theoretical and typological issues, Volume 8, pp. 255–275. Cambridge: Cambridge University Press.

Heinz, J. (2018). The computational nature of phonological generalizations. In L. Hyman and F. Plank (Eds.), Phonological Typology, Phonetics and Phonology, Chapter 5, pp. 126–195. De Gruyter Mouton.

Heinz, J. (Ed.) (in prep). Doing Computational Phonology. Oxford: Oxford University Press.

Heinz, J., C. de la Higuera, and M. van Zaanen (2015). Grammatical Inference for Computational Linguistics. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Heinz, J. and W. Idsardi (2013). What complexity differences reveal about domains in language. Topics in Cognitive Science 5(1), 111–131.

Heinz, J. and C. Koirala (2010). Maximum likelihood estimation of feature-based distributions. In Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, pp. 28–37.

Heinz, J. and R. Lai (2013). Vowel harmony and subsequentiality. In A. Kornai and M. Kuhlmann (Eds.), Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13), Sofia, Bulgaria, pp. 52–63. Association for Computational Linguistics.

Heinz, J., C. Rawal, and H. 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: Short papers-Volume 2, pp. 58–64. Association for Computational Linguistics.

Hockett, C. F. (1942). A system of descriptive phonology. Language 18, 3–21.

Hodgson, K. (2019). Relative clauses in colloquial Armenian: Syntax and typology. Ph. D. thesis, Université Sorbonne Paris.

Hoeksema, J. (1984). Categorial morphology. van Denderen: Groningen.

Hoeksema, J. (1987). Relating word structure and logical form. Linguistic inquiry 18(1), 119–126.

Hoeksema, J. (1988). Head-types in morpho-syntax. Yearbook of morphology 1, 123–137.

Hoeksema, J. (1992). The head parameter in morphology and syntax. Language and cognition 2, 119–132.

Hoeksema, J. and R. D. Janda (1988). Implications of process-morphology for categorial grammar. In R. T. Oehrle, E. Bach, and D. Wheeler (Eds.), Categorial grammars and natural language structures, pp. 199–247. Springer.

Hovakimyan, K. (2016). Eastern Armenian consonant clusters. Bachelor's thesis, Reed College, Portland, OR.

Hudson, G. (1986). Arabic root and pattern morphology without tiers. Journal of Linguistics 22(1), 85–122.

Hulden, M. (2006). Finite-state syllabification. In A. Yli-Jyrä, L. Karttunen, and J. Karhumäki (Eds.), Finite-State Methods and Natural Language Processing. FSMNLP 2005. Lecture Notes in Computer Science, Volume 4002. Berlin/Heidelberg: Springer.

Hulden, M. (2009a). Finite-state machine construction methods and algorithms for phonology and morphology. Ph. D. thesis, The University of Arizona, Tucson, AZ.

Hulden, M. (2009b). Regular expressions and predicate logic in finite-state language processing. In J. Piskorski, B. Watson, and A. Yli-Jyrä (Eds.), Finite-State Methods and Natural Language Processing. Post-Proceedings of the 7th International Workshop (FSMNLP) 2008, Volume 191, pp. 82–97. Amsterdam, the Netherlands: IOS Press.

Hulden, M. (2009c). Revisiting multi-tape automata for Semitic morphological analysis and generation. In Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages, pp. 19–26. Association for Computational Linguistics.

Hulden, M. (2017). Formal and computational verification of phonological analyses. Phonology 34(2), 407–435.

Hulden, M. and S. T. Bischoff (2009). A simple formalism for capturing reduplication in finite-state morphology. In J. Piskorski, B. Watson, and A. Yli-Jyrä (Eds.), <u>Proceedings of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008</u>, Amsterdam, pp. 207–214. IOS Press.

Hurch, B. (Ed.) (2005). <u>Studies on reduplication</u>. Number 28 in Empirical Approaches to Language Typology. Berlin: Walter de Gruyter.

Hwangbo, H. (2015). Learnability of two vowel harmony patterns with neutral vowels. In <u>Proceedings of the The third Annual Meeting on Phonology (AMP 2015)</u>.

Hyde, B. (2002). A restrictive theory of metrical stress. <u>Phonology</u> 19(3), 313–359.

Hyman, L. M. (2008). Directional asymmetries in the morphology and phonology of words, with special reference to Bantu. <u>Linguistics</u> 46(2), 309–350.

Idsardi, W. J. (2009). Calculating metrical structure. See Raimy and Cairns (2009), pp. 191–211.

Ikawa, S., A. Ohtaka, and A. Jardine (2020). Quantifier-free tree transductions. In <u>Proceedings of the Society for Computation in Linguistics</u>, Volume 3, pp. 145–153.

Inkelas, S. (1989). <u>Prosodic constituency in the lexicon</u>. Ph. D. thesis, Stanford University, Stanford, California.

Inkelas, S. (1993). Deriving cyclicity. See Hargus and Kaisse (1993), pp. 75–100.

Inkelas, S. (1998). The theoretical status of morphologically conditioned phonology: A case study of dominance effects. In G. Booij and J. van Marle (Eds.), <u>Yearbook of Morphology 1997</u>, pp. 121–155. Dordrecht: Kluwer Academic Publishers.

Inkelas, S. (2008). The morphology-phonology connection. In S. Berson, A. Bratkievich, D. Bruhn, A. Campbell, R. Escamilla, A. Giovine, L. Newbold, M. Perez, M. Piqueras-Brunet, and R. Rhomieux (Eds.), <u>Proceedings of the 34th meeting of the Berkeley Linguistics Society</u>, Berkeley, CA, pp. 145–162. Berkeley Linguistics Society.

Inkelas, S. (2014). <u>The interplay of morphology and phonology</u>. Oxford: Oxford University Press.

Inkelas, S. and C. O. Orgun (1995). Level ordering and economy in the lexical phonology of Turkish. <u>Language</u> 71(4), 763–793.

Inkelas, S. and C. O. Orgun (2003). Turkish stress: A review. <u>Phonology</u> 20(1), 139–161.

Inkelas, S., C. O. Orgun, and C. Zoll (1996). Exceptions and static phonological patterns: Cophonologies vs. prespecification. ROA-124, Rutgers Optimality Archive, http://roa.rutgers.edu/.

Inkelas, S., O. Orgun, and C. Zoll (1997). The implications of lexical exceptions for the nature of grammar. See Roca (1997), pp. 393–417.

Inkelas, S. and D. Zec (1990). <u>The phonology-syntax connection</u>. University of Chicago Press.

Inkelas, S. and C. Zoll (2005). <u>Reduplication: Doubling in Morphology</u>. Cambridge: Cambridge University Press.

Inkelas, S. and C. Zoll (2007). Is grammar dependence real? A comparison between cophonological and indexed constraint approaches to morphologically conditioned phonology. Linguistics 45, 133–171.

Itô, J. and A. Mester (1999). The phonological lexicon. In N. Tsujimura (Ed.), The handbook of Japanese linguistics, pp. 62–100. Oxford: Blackwell.

Ito, J. and A. Mester (2009). The extended prosodic word. See Grijzenhout and Kabak (2009), pp. 135–194.

Ito, J. and A. Mester (2012). Recursive prosodic phrasing in Japanese. In T. Borowsky, S. Kawahara, S. Takahito, and M. Sugahara (Eds.), Prosody matters: Essays in honor of Elisabeth Selkirk, pp. 280–303. London: Equinox Publishing.

Ito, J. and A. Mester (2013). Prosodic subcategories in Japanese. Lingua 124, 20–40.

Jäger, G. and J. Rogers (2012). Formal language theory: Refining the Chomsky hierarchy. Philosophical Transactions of the Royal Society B: Biological Sciences 367(1598), 1956–1970.

Jardine, A. (2014). Logic and the generative power of autosegmental phonology. In Proceedings of the Annual Meetings on Phonology, Volume 1.

Jardine, A. (2016a). Computationally, tone is different. Phonology 33(2), 247–283.

Jardine, A. (2016b). Learning tiers for long-distance phonotactics. In Proceedings of the 6th Conference on Generative Approaches to Language Acquisition North America (GALANA 2015), pp. 60–72.

Jardine, A. (2016c). Locality and non-linear representations in tonal phonology. Ph. D. thesis, University of Delaware, Newark, DE.

Jardine, A. (2017a). The local nature of tone-association patterns. Phonology 34(2), 363–384.

Jardine, A. (2017b). On the logical complexity of autosegmental representations. In Proceedings of the 15th Meeting on the Mathematics of Language, pp. 22–35.

Jardine, A. (2019). The expressivity of autosegmental grammars. Journal of Logic, Language and Information 28(1), 9–54.

Jardine, A. (2020). Melody learning and long-distance phonotactics in tone. Natural Language & Linguistic Theory, 1–51.

Jardine, A., J. Chandlee, R. Eyraud, and J. Heinz (2014). Very efficient learning of structured classes of subsequential functions from positive data. In International Conference on Grammatical Inference, pp. 94–108.

Jardine, A., N. Danis, and L. Iacoponi (to appear). A formal investigation of Q-theory in comparison to autosegmental representations. Linguistic Inquiry.

Jardine, A. and J. Heinz (2016a). Learning tier-based strictly 2-local languages. Transactions of the Association for Computational Linguistics 4, 87–98.

Jardine, A. and J. Heinz (2016b). Markedness constraints are negative: An autosegmental constraint definition language. In Proceedings of the 51st Annual Meeting of the Chicago Linguistics Society (CLS 2015).

Jardine, A. and K. McMullin (2017). Efficient learning of tier-based strictly k-local languages. In International Conference on Language and Automata Theory and Applications, pp. 64–76. Springer.

Ji, J. and J. Heinz (2020). Input strictly local tree transducers. In A. Leporati, C. Martín-Vide, D. Shapira, and C. Zandron (Eds.), Language and Automata Theory and Applications, Cham, pp. 369–381. Springer International Publishing.

Johnson, C. D. (1972). Formal aspects of phonological description. The Hague: Mouton.

Johnson, E. W. (1954). Studies in East Armenian Grammar. Ph. D. thesis, University of California, Berkeley, Berkeley, CA.

Kabak, B. and A. Revithiadou (2009). An interface approach to prosodic word recursion. See Grijzenhout and Kabak (2009), pp. 105–133.

Kager, R. (1996). On affix allomorphy and syllable counting. In U. Kleinhenz (Ed.), Interfaces in Phonology, pp. 155–171. Berlin: Akademie-Verlag.

Kager, R., H. van der Hulst, and W. Zonneveld (Eds.) (1999). The prosody-morphology interface. Cambridge: Cambridge University Press.

Kahnemuyipour, A. and K. Megerdoomian (2011). Second-position clitics in the vP phase: The case of the Armenian auxiliary. Linguistic Inquiry 42(1), 152–162.

Kahnemuyipour, A. and K. Megerdoomian (2017). On the positional distribution of an Armenian auxiliary: Second-position clisis, focus, and phases. Syntax 20(1), 77–97.

Kaisse, E. M. and S. Hargus (1993). Introduction. See Hargus and Kaisse (1993), pp. 1–19.

Kaisse, E. M. and S. Hargus (1994). When do linked structures evade structure preservation. See Wiese (1994), pp. 185–204.

Kaisse, E. M. and A. McMahon (2011). Lexical phonology and the lexical syndrome. See van Oostendorp et al. (2011), pp. 2236–2257.

Kaisse, E. M. and P. A. Shaw (1985). On the theory of lexical phonology. Phonology 2(1), 1–30.

Kalivoda, N. (2018). Syntax-prosody mismatches in Optimality Theory. Ph. D. thesis.

Kang, B.-m. (1993). Unhappier is really a "bracketing paradox". Linguistic inquiry 24(4), 788–794.

Kaplan, R. M. and M. Kay (1994). Regular models of phonological rule systems. Computational linguistics 20(3), 331–378.

Karakaş, A. (2020). An IBSP description of Sanskrit /n/-retroflexion. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Karapetian, S. (2014). "How Do I Teach My Kids My Broken Armenian?": A Study of Eastern Armenian Heritage Language Speakers in Los Angeles. Ph. D. thesis, University of California, Los Angeles.

Karapetyan, S. H. (2016). Goyakanakan himk'erov baġadrowt'yownnerë hayereni ew anglereni baṙakazmakan hamakargowm [Noun-based structural patterns in the Armenian and English word-formation systems]. Ph. D. thesis, HH GAA Hračya Ačaṙyani anvan lezvi institowt.

Karst, J. (1901). Historische Grammatik des Kilikisch-Armenischen [Grammar of Cilician Armenian]. Strassburg: Verlag von Karl Trübner.

Karttunen, L. (1993). Finite-state constraints. In J. Goldsmith (Ed.), The last phonological rule: Reflections on constraints and derivations, pp. 173–194. University of Chicago Press.

Karttunen, L. (2003). Computing with realizational morphology. In International Conference on Intelligent Text Processing and Computational Linguistics, pp. 203–214. Springer.

Karttunen, L. (2006a). A finite-state approximation of optimality theory: The case of Finnish prosody. In Advances in Natural Language Processing, pp. 4–15. Springer.

Karttunen, L. (2006b). The insufficiency of paper-and-pencil linguistics: The case of Finnish prosody. In M. Butt, M. Dalrymple, , and T. H. King (Eds.), Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan, Number 179 in CSLI Lecture Notes, pp. 287–300. Stanford, CA: CSLI.

Kassabian, H. (1971). A contrastive analysis of stress in colloquial west Armenian and American English. Master's thesis, Department of Education, AUB.

Katvalyan, V. L. (1989). Hnčyownneri hamakc'akan p'op'oxowt'yownnerë ew dranc' haraberowt'yownë myows hnčyownap'oxowt'yownnerin [Combinatory modifications and their relationship with other types of sound changes]. Lraber Hasarakakan Gitowt'yownneri 4, 64–69.

Kay, M. (1987, April). Nonconcatenative finite-state morphology. In Third Conference of the European Chapter of the Association for Computational Linguistics, Copenhagen, Denmark. Association for Computational Linguistics.

Kean, M.-L. (1974). The strict cycle in phonology. Linguistic Inquiry 5(2), 179–203.

Keane, E. (2005). Phrasal reduplication and dual description. See Hurch (2005), pp. 239–261.

Kenstowicz, M. (1996). Base-identity and uniform exponence: Alternatives to cyclicity. In J. Durand and B. Laks (Eds.), Current Trends in Phonology: Models and Methods, pp. 363–393. Salford: University of Salford.

Khachaturian, A. (1983). The nature of voiced aspirated stops and affricates in Armenian dialects. Annual of Armenian Linguistics 4, 57–62.

Khachaturian, A. (1985). The phonology of the Armenian ə vowel in modern East Armenian. Annual of Armenian Linguistics 6, 53–58.

Khachaturian, A. H. (1992). Voiced aspirated consonants in the Nor Bayazet dialect of Armenian. In J. Greppin (Ed.), Proceedings of the Fourth International Conference on Armenian Linguistics, Delmar, New York, pp. 115–128. Caravan.

Khanjian, H. (2009). Stress dependent vowel reduction. In I. Kwon, H. Pritchett, , and J. Spence (Eds.), Proceedings of the 35th Annual Meeting of the Berkeley Linguistics Society, Berkeley, CA, pp. 178–189. Berkeley Linguistics Society.

Khanjian, H. (2013). (Negative) concord and head directionality in Western Armenian. Ph. D. thesis, Massachusetts Institute of Technology.

Kilborne-Ceron, O., H. Newell, M. Noonan, and L. Travis (2016). Phase domains at pf: Root suppletion and its implications. See Siddiqi and Harley (2016), pp. 121–161.

Kim, J. J.-R. (1991). Bracketing paradoxes in natural language processing. Language Research 27(1), 99–117.

Kim, J. J.-R. (1992). Bracketing paradoxes in Korean. Korean Linguistics 7, 53 – 71.

Kim, Y. (2010). Phonological and morphological conditions on affix order in Huave. Morphology 20(1), 133–163.

Kim, Y. (2015). Mobile affixation within a modular approach to the morphology-phonology interface. In S. Manova (Ed.), Affix ordering across languages and frameworks, pp. 111–123. New York: Oxford University Press.

Kiparsky, P. (1973). Abstractness, opacity, and global rules. In O. Fujimura (Ed.), Three dimensions of linguistic theory. Tokyo: Taikusha.

Kiparsky, P. (1982a). From cyclic phonology to lexical phonology. In H. van der Hulst and N. Smith (Eds.), The structure of phonological representations, pp. 131–175. Dordrecht: Foris.

Kiparsky, P. (1982b). Lexical morphology and phonology. In I.-S. Yang (Ed.), Linguistics in the morning calm: Selected papers from SICOL-1981, pp. 3–91. Seoul: Hansin.

Kiparsky, P. (1983). Word-formation and the lexicon. In F. Ingemann (Ed.), Proceedings of the 1982 Mid-America Linguistics Conference, Lawrence, pp. 3–29. Mid-America Linguistics Conference: University of Kansas.

Kiparsky, P. (1985). Some consequences of lexical phonology. Phonology 2(1), 85–138.

Kiparsky, P. (1993). Blocking in non-derived environments. See Hargus and Kaisse (1993), pp. 277–313.

Kiparsky, P. (2000). Opacity and cyclicity. The Linguistic Review 17, 351–366.

Kiparsky, P. (2015). Stratal OT: A synopsis and FAQs. In Y. E. Hsiao and L.-H. Wee (Eds.), Capturing phonological shades within and across languages, pp. 2–44. Cambridge: Cambridge Scholars Publishing.

Kiraz, G. A. (2000). Multitiered nonlinear morphology using multitape finite automata: A case study on Syriac and Arabic. Computational Linguistics 26(1), 77–105.

Kiraz, G. A. (2001). Computational nonlinear morphology: With emphasis on Semitic languages. Cambridge University Press.

Kiraz, G. A. and B. Möbius (1998). Multilingual syllabification using weighted finite-state transducers. In The third ESCA/COCOSDA workshop (ETRW) on speech synthesis.

Kitagawa, Y. (1986). More on bracketing paradoxes. Linguistic Inquiry 17(1), 177–183.

Klein, E. (1991). Phonological data types. See Bird (1991a), pp. 127–138.

Klein, E. (1993). An HPSG approach to Sierra Miwok verb stems. See Ellison and Scobbie (1993), pp. 19–35.

Klein, J. S., B. D. Joseph, and M. Fritz (Eds.) (2017). Handbook of Comparative and Historical Indo-European Linguistics. Berlin/Boston: Walter de Gruyter.

Kobele, G. M. (2006). Generating Copies: An investigation into structural identity in language and grammar. Ph. D. thesis, University of California, Los Angeles.

Kogian, S. L. (1949). Armenian grammar (West dialect). Vienna: Mechitharist Press.

Kornai, A. (1995). Formal phonology. Garland Publishing Inc.

Kornai, A. (2007). Mathematical linguistics. Springer Science & Business Media.

Kornai, A. (2009). The complexity of phonology. Linguistic Inquiry 40(4), 701–712.

Kortlandt, F. H. H. (Ed.) (2003). Armeniaca: Comparative notes. Ann Arbor, MI: Caravan Books.

Koser, N. and A. Jardine (2019). The computational nature of stress assignment. In Proceedings of AMP 2019.

Koser, N., C. Oakden, and A. Jardine (2019). Tone association and output locality in non-linear structures. In Supplemental proceedings of AMP 2019.

Koskenniemi, K. (1983a). Two-level model for morphological analysis. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'83, San Francisco, CA, USA, pp. 683–685. Morgan Kaufmann Publishers Inc.

Koskenniemi, K. (1983b). Two-level morphology: A General Computational Model for Word-Form Recognition and Production. Ph. D. thesis, University of Helsinki.

Koskenniemi, K. (1984). A general computational model for word-form recognition and production. In Proceedings of the 10th international conference on Computational Linguistics, pp. 178–181. Association for Computational Linguistics.

Koskenniemi, K. and K. W. Church (1988). Complexity, two-level morphology and Finnish. In Proceedings of the 12th Conference on Computational Linguistics - Volume 1, COLING '88, USA, pp. 335–340. Association for Computational Linguistics.

Kozintseva, N. (1995). Modern Eastern Armenian. Number 22 in Languages of the World. München: Lincom Europa.

LaCharité, D. and C. Paradis (1993). Introduction: The emergence of constraints in generative phonology and a comparison of three current constraint-based models. Canadian Journal of Linguistics/Revue canadienne de linguistique 38(2), 127–153.

Ladd, D. R. (1986). Intonational phrasing: The case for recursive prosodic structure. Phonology 3, 311–340.

Lai, R. (2012). Domain specificity in learning phonology. Ph. D. thesis, University of Delaware.

Lai, R. (2015). Learnable vs. unlearnable harmony patterns. Linguistic Inquiry 46(3), 425–451.

Lambert, D. and J. Rogers (2019). A logical and computational methodology for exploring systems of phonotactic constraints. In Proceedings of the Society for Computation in Linguistics, Volume 2, pp. 247–256.

Lambert, D. and J. Rogers (2020). Tier-based strictly local stringsets: Perspectives from model and automata theory. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Lamont, A. (2018). Decomposing phonological transformations in serial derivations. In Proceedings of the Society for Computation in Linguistics (SCiL) 2018, pp. 91–101.

Lamont, A., C. O'Hara, and C. Smith (2019). Weakly deterministic transformations are subregular. In Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology, Florence, Italy, pp. 196–205. Association for Computational Linguistics.

Langendoen, D. T. (1981). The generative capacity of word-formation components. Linguistic Inquiry 12(2), 320–322.

Laporte, E. (1997). Rational transductions for phonetic conversion and phonology. See Roche and Schabes (1997), pp. 407–429.

Laszakovits, S. (2018). Case theory in minimalist grammars. In International Conference on Formal Grammar 2018, Berlin/Heidelberg, pp. 37–61. Springer.

Lemus, J. E. (1996). Phonology at two levels: A new model of Lexical Phonology. Ph. D. thesis, The University of Arizona.

Levelt, W. (1974). Formal grammars in linguistics and psycholinguistics. The Hague: Mouton.

Liberman, M. Y. (1975). The intonational system of English. Ph. D. thesis, Massachusetts Institute of Technology.

Libkin, L. (2013). Elements of finite model theory. Springer Science & Business Media.

Lieber, R. (1980). On the organization of the lexicon. Ph. D. thesis, Massachusetts Institute of Technology.

Lieber, R. (1989). On percolation. Yearbook of morphology 2, 95–138.

Light, M. (1991). Taking the paradoxes out of bracketing in morphology. In Proceedings of the Second Formal Linguistics Society of Mid-America Conference. University of Michigan.

Lochbihler, B. (2017). Syntactic domain types and PF effects. See Newell et al. (2017), pp. 74–99.

Łubowicz, A. (2002). Derived environment effects in optimality theory. Lingua 112(4), 243–280.

Lubowicz, A. (2003). Local conjunction and comparative markedness. Theoretical linguistics 29(1), 2.

Luo, H. (2017). Long-distance consonant agreement and subsequentiality. Glossa: A journal of general linguistics 2(1), 1–25.

Macak, M. J. (2016). Studies in Classical and Modern Armenian Phonology. Ph. D. thesis, University of Georgia.

Macak, M. J. (2017). The phonology of Classical Armenian. See Klein et al. (2017), pp. 1037–1079.

Mamadou, T. and A. Jardine (2020). Representation and the computation of long distance tone processes. In Proceedings of the Annual Meetings on Phonology, Volume 8.

Mansfield, J. (2017). Prosodic words in cyclic derivation: The strange case of Murrinhpatha compound verbs. Morphology 27(3), 359–382.

Marantz, A. (1988). Clitics, morphological merger, and the mapping to phonological structure. In Theoretical morphology, pp. 253–270. Academic Press.

Marantz, A. (1997). No escape from syntax: Don't try morphological analysis in the privacy of your own lexicon. In A. Dimitriadis and L. Siegel (Eds.), University of Pennsylvania Working Papers in Linguistics, Volume 4, pp. 201–25.

Marantz, A. (2007). Phases and words. In S.-H. Choe, D.-W. Yang, Y.-S. Kim, S.-H. Kim, and A. Marantz (Eds.), Phases in the theory of grammar, pp. 191–222. Seoul: Dong-In Publishing Co.

Margaryan, A. S. (1997). Žamanakakic' hayoc' lezow: Hnčyownabanowt'yown [Contemporary Armenian language: Phonology]. Yerevan: Yerevani Petakan Hamalsarani Hratarakčowt'yown.

Marowt'yan, A. A. (2003). Miavank baġadričov bard goyakanneri hognakii kazmowt'yownë žamanakakic' hayerenowm [Plural formation of monosyllabic-final compounds in contemporary Armenian]. Lraber Hasarakakan Gitowt'yownneri 2, 52–62.

Martí, L. (2020). Numerals and the theory of number. Semantics and Pragmatics 13.

Martirosyan, H. (2009). Etymological dictionary of the Armenian inherited lexicon. Brill.

Martirosyan, H. (2019). The Armenian dialects. In G. Haig and G. Khan (Eds.), The Languages and Linguistics of Western Asia: An Areal Perspective, Volume 6, pp. 46. Berlin/Boston: Walter de Gruyter GmbH & Co KG.

Marvin, T. (2002). Topics in the stress and syntax of words. Ph. D. thesis, Massachusetts Institute of Technology.

Mascaró, J. (1976). Catalan phonology and the phonological cycle. Ph. D. thesis, Massachusetts Institute of Technology.

Matasović, R. (2009). A grammatical sketch of Classical Armenian. Unpublished ms. University of Zagreb. Available online: `http://mudrac.ffzg.unizg.hr/~rmatasov/ARMENIAN2.pdf`. Accessed 1 July 2019.

Matushansky, O. and A. Marantz (Eds.) (2013). Distributed Morphology Today. MIT Press.

Mayer, C. and T. Major (2018). A challenge for tier-based strict locality from Uyghur backness harmony. In International Conference on Formal Grammar 2018, Berlin/Heidelberg, pp. 62–83. Springer.

McCarthy, J. J. (1988). Feature geometry and dependency: A review. Phonetica 45(2-4), 84–108.

McCarthy, J. J. (2003). Comparative markedness. Theoretical linguistics 29(1-2), 1–51.

McCarthy, J. J. and A. Cohn (1998). Alignment and parallelism in Indonesian phonology. In N. Adisasmito-Smith, B. Ham, and L. Lavoie (Eds.), Working Papers of the Cornell Phonetics Laboratory, Volume 12, pp. 53–137. Ithaca, NY: Cornell University.

McCarthy, J. J. and A. Prince (1993). Generalized alignment. In G. Booij and J. van Marie (Eds.), Yearbook of morphology 1993, pp. 79–153. Dordrecht: Kluwer Academic Publishers.

McCarthy, J. J. and A. Prince (1995). Faithfulness and reduplicative identity. In J. N. Beckman, L. W. Dickey, and S. Urbanczyk (Eds.), Papers in Optimality Theory. Amherst, MA: Graduate Linguistic Student Association, University of Massachusetts.

McMullin, K., A. Aksënova, and A. De Santo (2019). Learning phonotactic restrictions on multiple tiers. In Proceedings of the Society for Computation in Linguistics, Volume 2, pp. 377–378.

McMullin, K. and G. O. Hansson (2016). Long-distance phonotactics as tier-based strictly 2-local languages. In Proceedings of the Annual Meetings on Phonology, Volume 2.

McMullin, K. and G. O. Hansson (2019). Inductive learning of locality relations in segmental phonology. Laboratory Phonology: Journal of the Association for Laboratory Phonology 10(1).

McMullin, K. J. (2016). Tier-based locality in long-distance phonotactics: Learnability and typology. Ph. D. thesis, University of British Columbia.

McNaughton, R. and S. A. Papert (1971). Counter-Free Automata. The MIT Press.

McPherson, L. and B. Hayes (2016). Relating application frequency to morphological structure: The case of Tommo So vowel harmony. Phonology 33(1), 125–167.

Megerdoomian, K. (2009). Beyond words and phrases: A unified theory of predicate composition. VDM, Verlag Dr. Müller.

Merchant, J. (2015). How much context is enough? two cases of span-conditioned stem allomorphy. Linguistic Inquiry 46(2), 273–303.

Meyer, R. (2013). Classical Armenian Relative Clause Syntax. A comparative study of relative clauses in the Armenian and Greek NT and other 5th-c. Armenian texts. Ph. D. thesis, University of Oxford.

Miller, G. A. and N. Chomsky (1963). Finitary models of language users. In R. D. Luce, R. R. Bush, and E. Galanter (Eds.), Handbook of Mathematical Psychology, Volume II, pp. 419–491. New York: John Wiley.

Miller, P. H. (1999). Strong generative capacity: The semantics of linguistic formalism. Stanford: CSLI publications.

Miller, T. L. (2018). The phonology-syntax interface and polysynthesis: A study of Kiowa and Saulteaux Ojibwe. Ph. D. thesis, University of Delaware.

Miller, T. L. (2020). Navigating the phonology-syntax interface and Tri-P mapping. In Proceedings of the Annual Meetings on Phonology, Volume 8.

Minassian, M. (1980). Grammaire d'arménien oriental. Delmar, NY: Caravan Books.

Mkrtchyan, H. M. (1952). Karno Barbaṙë (Hnčyownabanowt'yown, jevabanowt'yown, baṙaran) [The Karin dialect (Phonology, Morphology, and Dictionary)]. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Mkrtčyan, G. (2015). Hayereni ë hodi çagman ew gorčaṙowt'yan harc'i šowrǰ [on the origin and function of the Armenian article]. Banber Erewani hamalsarani. Banasirowt'yown 2(17), 35–43.

Mkrtčyan, R. (1972). Bard baři dasakargowmë hayerenowm [The classification of Armenian compounds. <u>Lraber Hasarakakan Gitowt'yownneri</u> 3, 100–106.

Mkrtčyan, R. (1973). Orošič-orošyali haraberowt'yamb baġadričnerov bardowt'yownnern ardi hayerenowm [Compound words of the type definer-defined in the modern Armenian language]. <u>Lraber Hasarakakan Gitowt'yownneri</u> 4, 45–50.

Mkrtčyan, R. (1977). Bard baři dasakargowmë ew nranc' baġadričneri imastayin p'oxharaberowt'yownë žamanakakic' hayerenowm [The classification of complex words and the meaning of their components in contemporary Armenian]. <u>Lezvi ew oči harc'er</u> 4, 77–158.

Mkrtčyan, R. (1980). Bard baři baġadričneri hamadreliowt'yan sahmanap'akič hangamank'nerë [Factors limiting the compatibility of components in compound words]. <u>Lraber Hasarakakan Gitowt'yownneri</u> 1, 34–40.

Mohanan, K. P. (1986). <u>The Theory of Lexical Phonology</u>. Dordrecht: Reidel.

Mohri, M. (1997). Finite-state transducers in language and speech processing. <u>Computational Linguistics</u> 23(2), 269–311.

Moradi, S., A. Aksënova, and T. Graf (2019). The computational cost of generalizations: An example from micromorphology. In <u>Proceedings of the Society for Computation in Linguistics</u>, Volume 2, pp. 367–368.

Morawietz, F. (2003). <u>Two-Step Approaches to Natural Language Formalism</u>. Berlin: Mouton de Gruyter.

Moskal, B. (2015a). Limits on allomorphy: A case study in nominal suppletion. <u>Linguistic Inquiry</u> 46(2), 363–376.

Moskal, B. and P. W. Smith (2019). The status of heads in morphology. In <u>Oxford Research Encyclopedia of Linguistics</u>. Oxford University Press.

Moskal, B. A. (2015b). <u>Domains on the border: Between morphology and phonology</u>. Ph. D. thesis, University of Connecticut, Storrs, CT.

Müller, S. (2003). Solving the bracketing paradox: An analysis of the morphology of German particle verbs. <u>Journal of Linguistics</u> 39(2), 275–325.

Myers, S. (1991). Structure preservation and the strong domain hypothesis. <u>Linguistic Inquiry</u> 22(2), 379–385.

Nazarov, A. and J. Pater (2017). Learning opacity in stratal maximum entropy grammar. <u>Phonology</u> 34(2), 299–324.

Nelson, M., H. Dolatian, J. Rawski, and B. Prickett (2020). Probing rnn encoder-decoder generalization of subregular functions using reduplication. In <u>Proceedings of the Society for Computation in Linguistics</u>, Volume 3.

Nespor, M. (1999). Stress domains. In H. van der Hulst (Ed.), <u>Word prosodic systems in the langauges of Europe</u>, pp. 117–159. Berlin: Mouton de Gruyter.

Nespor, M. and A. Ralli (1996). Morphology-phonology interface: Phonological domains in Greek compounds. <u>The linguistic review</u> 13(3-4), 357–382.

Nespor, M. and I. Vogel (1986). Prosodic phonology. Dordrecht: Foris.

Nevins, A. (2011). Phonologically conditioned allomorph selection. See van Oostendorp et al. (2011), pp. 2357–82.

Nevins, A. and B. Vaux (2008). Introduction: The division of labor between rules, representations, and constraints in phonological theory. In B. Vaux and A. Nevins (Eds.), Rules, constraints, and phonological phenomena, pp. 1–19. Oxford: Oxford University Press.

Newell, H. (2005). Bracketing paradoxes and particle verbs: A late adjunction analysis. In Proceedings of ConSOLE XIII, pp. 249–272. University of Leiden Leiden.

Newell, H. (2008). Aspects of the morphology and phonology of phases. Ph. D. thesis, McGill University, Montreal, QC.

Newell, H. (2017). Nested phase interpretation and the PIC. See Newell et al. (2017), pp. 20–40.

Newell, H. (2019). Bracketing paradoxes in morphology. In Oxford Research Encyclopedia of Linguistics. Oxford University Press.

Newell, H., M. Noonan, and G. Piggott (Eds.) (2017). The structure of words at the interfaces, Volume 68. Oxford: Oxford University Press.

Newell, H. and G. Piggott (2014). Interactions at the syntax-phonology interface: Evidence from ojibwe. Lingua 150, 332–362.

Nikolou, K. (2009). The role of recursivity in the phonological word. In A. Karasimous, C. Vlachos, E. Dimela, M. Giakoumelou, M. Pavlakou, N. Koutsoukos, and D. Bougonikolou (Eds.), Proceedings of the First Patras International Conference of Graduate students in Linguistics (PICGL1), pp. 41–52. University of Patras.

Noyer, R. (1994). Mobile affixes in Huave: Optimality and morphological wellformedness. In E. Duncan, D. Farkas, and P. Spaelti (Eds.), Proceedings of the Twelfth West Coast Conference on Formal Linguistics. Stanford: CSLI, Stanford, pp. 67–82. CSLI.

Oakden, C. (to appear). Notational equivalence in tonal geometry. Phonology.

Odden, D. (1994). Adjacency parameters in phonology. Language 70(2), 289–330.

Odden, D. and M. Odden (1985). Ordered reduplication in Kihehe. Linguistic Inquiry 16(3), 497–503.

Oehrle, R. T. (1991). Prosodic constraints on dynamic grammatical analysis. See Bird (1991a), pp. 167–196.

Ogden, R. (1993). What Firthian prosodic analysis has to say to us. See Ellison and Scobbie (1993), pp. 107–127.

Ogden, R. (1999). A declarative account of strong and weak auxiliaries in English. Phonology 16(1), 55–92.

Olsen, B. (2017). The morphology of Armenian. See Klein et al. (2017), pp. 1080–1097.

Olsen, B. A. (2011). The noun in Biblical Armenian: Origin and word-formation-with special emphasis on the Indo-European heritage, Volume 119. Berlin/New York: Walter de Gruyter.

Oltra-Massuet, I. (1999a). On the constituent structure of Catalan verbs. In K. Arregi, V. Lin, C. Krause, and B. Bruening (Eds.), MIT Working Papers in Linguistics, Volume 33, pp. 279–322. Cambridge, MA: Department of Linguistics, Massachusetts Institute of Technology.

Oltra-Massuet, M. I. (1999b). On the notion of theme vowel: A new approach to Catalan verbal morphology. Master's thesis, Massachusetts Institute of Technology.

Orgun, C. O. (1994). Monotonic cyclicity. ROA-123, Rutgers Optimality Archive. Available online at http://roa.rutgers.edu/files/123-0496/roa-123-orgun-4.pdf Accessed 1 July 2019.

Orgun, C. O. (1996). Sign-based morphology and phonology with special attention to Optimality Theory. Ph. D. thesis, University of California, Berkeley, Berkeley, CA.

Orgun, C. O. (1998). Cyclic and noncyclic phonological effects in a declarative grammar. In Yearbook of Morphology 1997, pp. 179–218. Springer.

Orgun, C. O. (2002). Reconsidering bracket erasure. In G. Booij and J. v. Marle (Eds.), Yearbook of Morphology 2001, pp. 115–146. Dordrecht: Springer.

Oseki, Y. (2018). Syntactic structures in morphological processing. Ph. D. thesis, New York University.

Oseki, Y., C. Yang, and A. Marantz (2019, June). Modeling hierarchical syntactic structures in morphological processing. In Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, Minneapolis, Minnesota, pp. 43–52. Association for Computational Linguistics.

Ouwayda, S. (2017). On the DP dependence of collective interpretation with numerals. Natural Language Semantics 25(4), 263–314.

Oyer, M. W. (2017). The declensions of Modern Eastern Armenian: A paradigm function morphology approach. Master's thesis, University of Kentucky, Lexington, Kentucky.

Özçelik, Ö. (2017). The foot is not an obligatory constituent of the prosodic hierarchy: "stress" in Turkish, French and child English. The Linguistic Review 34(1), 157–213.

Padrosa Trias, S. (2011). Complex Word-Formation and the Morphology-Syntax Interface. Ph. D. thesis, Universitat Autònoma de Barcelona, Barcelona.

Pak, M. (2008). The postsyntactic derivation and its phonological reflexes. Ph. D. thesis, Unpublished doctoral dissertation, University of Pennsylvania.

Papillon, M. (2020). Precedence Relation-Oriented Phonology. Ph. D. thesis, University of Maryland.

Paster, M. (2005). Subcategorization vs. output optimization in syllable-counting allomorphy. In Proceedings of the 24th west coast conference on formal linguistics, pp. 326–333. Cascadilla Press Somerville, MA.

Paster, M. (2006). Phonological conditions on affixation. Ph. D. thesis, University of California, Berkeley, Berkeley, CA.

Paster, M. (2009). Explaining phonological conditions on affixation: Evidence from suppletive allomorphy and affix ordering. Word structure 2(1), 18–37.

Paster, M. (2019). Phonology counts. Radical: A Journal of Phonology 1.

Pater, J. (2007). The locus of exceptionality: Morpheme-specific phonology as constraint indexation. In L. Bateman, M. O'Keefe, E. Reilly, and A. Werle (Eds.), University of Massachusetts Occasional Papers in Linguistics 32: Papers in Optimality Theory III, pp. 187–207. Amherst, MA: Graduate Linguistics Student Association, University of Massachusetts.

Pater, J. (2009). Morpheme-specific phonology: Constraint indexation and inconsistency resolution. In S. Parker (Ed.), Phonological argumentation: Essays on evidence and motivation, pp. 123–154. London: Equinox Publishing Ltd.

Payne, A. (2014). Dissimilation as a subsequential process. In J. Iyer and L. Kusmer (Eds.), NELS 44: Proceedings of the 44th Meeting of the North East Linguistic Society, Volume 2, Amherst, MA, pp. 79–90. Graduate Linguistic Student Association, University of Massachusetts.

Payne, A. (2017). All dissimilation is computationally subsequential. Language: Phonological Analysis 93(4), e353–e371.

Payne, A., M. H. Vu, and J. Heinz (2017). A formal analysis of correspondence theory. In Proceedings of the Annual Meetings on Phonology, Volume 4.

Peperkamp, S. A. (1997). Prosodic words. The Hague: Holland Academic Press.

Pesetsky, D. (1979). Russian morphology and lexical theory. Unpublished manuscript.

Pesetsky, D. (1985). Morphology and logical form. Linguistic Inquiry 16(2), 193–246.

Peters, P. S. and R. W. Ritchie (1973). On the generative power of transformational grammars. Information sciences 6, 49–83.

Pierce, M. (2007). Vowel epenthesis vs. schwa lexicalization in Classical Armenian. Journal of Indo-European Studies 35(1/2), 111–119.

Pierrehumbert, J. B. (1980). The phonology and phonetics of English intonation. Ph. D. thesis, Massachusetts Institute of Technology.

Plungian, V. (2018). Notes on Eastern Armenian verbal paradigms: "temporal mobility" and perfective stems. In D. Van Olmen, T. Mortelmans, and F. Brisard (Eds.), Aspects of linguistic variation: Studies in honor of Johan van der Auwera, pp. 233–245. Berlin: De Gruyter Mouton.

Potts, C. and G. K. Pullum (2002). Model theory and the content of OT constraints. Phonology 19(3), 361–393.

Prince, A. and P. Smolensky (2004). Optimality Theory: Constraint Interaction in Generative Grammar. Blackwell Publishing.

Pullum, G. K. (2007). The evolution of model-theoretic frameworks in linguistics. Model-theoretic syntax at 10, 1–10.

Pullum, G. K. and B. C. Scholz (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In International Conference on Logical Aspects of Computational Linguistics, pp. 17–43. Springer.

Raffelsiefen, R. (1999). Phonological constraints on English word formation. In G. Booij and J. van Marle (Eds.), Yearbook of Morphology 1998, pp. 225–287. Dordrecht: Kluwer.

Raffelsiefen, R. (2005). Paradigm uniformity effects versus boundary effects. In L. J. Downing, T. A. Hall, and R. Raffelsiefen (Eds.), Paradigms in phonological theory. Oxford: Oxford University Press.

Raimy, E. (2000). The Phonology and Morphology of Reduplication. Berlin: Mouton de Gruyter.

Raimy, E. and C. E. Cairns (Eds.) (2009). Contemporary views on architecture and representations in phonology. Number 48 in Current Studies in Linguistics. Cambridge, MA: MIT Press.

Rainer, F. (1993). Head-operations in Spanish morphology. In G. Booij and J. van Marle (Eds.), Yearbook of Morphology 1992, pp. 113–128. Dordrecht: Kluwer Academic Publishers.

Ralli, A. (2008). Compound markers and parametric variation. STUF-Sprachtypologie und Universalienforschung 61(1), 19–38.

Ralli, A. (2012). Compounding in modern Greek, Volume 2. Springer Science & Business Media.

Ramsammy, M. (2015). The life cycle of phonological processes: Accounting for dialectal microtypologies. Language and Linguistics Compass 9(1), 33–54.

Ravnæs, E. (1988). The prehistory of Armenian ē and oy. In V. A. Friedman (Ed.), Studia Caucasologica 1: Proceedings of the Third Caucasian Colloquium, Oslo, July 1986, Oslo, pp. 225–238.

Ravnæs, E. (2005). The relative chronology of the sound changes from Proto-Indo-European to Classical Armenian. In D. Haug and E. Welo (Eds.), Haptačahaptāitiš: Festschrift for Fridrik Thordarson on the occasion of his 77th birthday, pp. 191–204. Oslo: Novus.

Rawski, J. (2019a). Phonological complexity is subregular: Evidence from sign language. In Proceedings of the 53rd Annual Meeting of the Chicago Linguistics Society. Chicago: Chicago Linguistics Society.

Rawski, J. (2019b). Tensor product representations of subregular formal languages. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019) workshop on Neural-Symbolic Learning and Reasoning.

Rawski, J. and H. Dolatian (2020). Multi input strictly local functions for tonal phonology. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Rawski, J. and J. Heinz (2019). No free lunch in linguistics or machine learning: Response to pater. Language 95(1), e125–e135.

Reich, P. A. (1969). The finiteness of natural language. Language 45, 831–843.

Revithiadou, A. (1999). Headmost accent wins: Head dominance and ideal prosodic form in lexical accent systems. Ph. D. thesis, Rutgers University, New Brunswick, NJ.

Rice, K. (1992). On deriving rule domains: The Athapaskan case. In D. Bates (Ed.), Proceedings of the 10th West Coast Conference on Formal Linguistics, Volume 10, Stanford, pp. 417–430. Center for the Study of Language and Information.

Rice, K. D. (1993). The structure of the Slave (Northern Athabaskan) verb. See Hargus and Kaisse (1993), pp. 145–171.

Ristad, E. S. (1990). Computational structure of human language. Ph. D. thesis, Massachusetts Institute of Technology.

Ritchie, G. (1989). On the generative power of two-level morphological rules. In Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics, pp. 51–57. Association for Computational Linguistics.

Ritchie, G. (1992). Languages generated by two-level morphological rules. Computational Linguistics 18(1), 41–59.

Ritchie, G. D., G. J. Russell, A. W. Black, and S. G. Pulman (1992). Computational morphology: Practical mechanisms for the English lexicon. Cambridge, MA: MIT press.

Roark, B. and R. Sproat (2007). Computational Approaches to Morphology and Syntax. Oxford: Oxford University Press.

Roca, I. (Ed.) (1997). Derivations and· Constraints in Phonology. Oxford: Clarendon Press.

Roche, E. and Y. Schabes (Eds.) (1997). Finite-state language processing. MIT press.

Rogers, J. (1997). "grammarless" phrase structure grammar. Linguistics and Philosophy 20(6), 721–746.

Rogers, J. (1998). A Descriptive Approach to Language-Theoretic Complexity. Stanford, CA: CSLI Publications.

Rogers, J., J. Heinz, M. Fero, J. Hurst, D. Lambert, and S. Wibel (2013). Cognitive and sub-regular complexity. In G. Morrill and M.-J. Nederhof (Eds.), Formal Grammar, Volume 8036 of Lecture Notes in Computer Science, pp. 90–108. Springer.

Rogers, J. and D. Lambert (2019a). Extracting subregular constraints from regular stringsets. Journal of Language Modelling 7(2), 143–176.

Rogers, J. and D. Lambert (2019b). Some classes of sets of structures definable without quantifiers. In Proceedings of the 16th Meeting on the Mathematics of Language, Toronto, Canada, pp. 63–77. Association for Computational Linguistics.

Rogers, J. and G. Pullum (2011). Aural pattern recognition experiments and the subregular hierarchy. Journal of Logic, Language and Information 20, 329–342.

Roon, K. (2005). Stress in Russian compound nouns: Head dominance or root control. In Formal Approaches to Slavic Linguistics (FASL-14). The Princeton Meeting, pp. 319–330.

Rubach, J. (1985). Lexical phonology: Lexical and postlexical derivations. Phonology 2(1), 157–172.

Rubach, J. (1996). Nonsyllabic analysis of voice assimilation in Polish. Linguistic Inquiry 27(1), 69–110.

Rubach, J. (1997). Extrasyllabic consonants in Polish: Derivational optimality theory. See Roca (1997), pp. 551–581.

Rubach, J. (2003). Polish palatalization in derivational optimality theory. Lingua 113(3), 197–237.

Rubach, J. (2008). An overview of lexical phonology. Language and Linguistics Compass 2(3), 456–477.

Rubach, J. and G. E. Booij (1990). Edge of constituent effects in Polish. Natural Language & Linguistic Theory 8(3), 427–463.

Russell, K. (1993). A constraint-based approach to phonology and morphology. Ph. D. thesis, University of Southern California, Los Angeles, CA.

Sadock, J. M. (1985). Autolexical syntax: A proposal for the treatment of noun incorporation and similar phenomena. Natural Language & Linguistic Theory 3(4), 379–439.

Samuels, B. (2012). Consequences of phases for morpho-phonology. In A. J. Gallego (Ed.), Phases: Developing the Framework, pp. 251–282. Berlin/Boston: Mouton De Gruyter.

Samuels, B. D. (2011). Phonological architecture: A biolinguistic perspective. Oxford Studies in Biolinguistics. Oxford University Press.

Sande, H., P. Jenks, and S. Inkelas (2020). Cophonologies by ph(r)ase. Natural Language & Linguistic Theory, 1–51.

Sande, H. L. (2017). Distributing morphologically conditioned phonology: Three case studies from Guébie. Ph. D. thesis, University of California, Berkeley, Berkeley, CA.

Sargsyan, A. (1979). Verǰin miavank baġadričov bard bařeri hognaki t'vi zowgajewerë žamanakakic' hayerenowm [Doublets in the plural of monosyllabic-final compounds in contemporary Armenian]. Lraber Hasarakakan Gitowt'yownneri 4, 35–44.

Sargsyan, A. (1984). Goyakani holovman tiperi vičakagrakan bnowt'agirë ardi hayerenowm [statistical characterization of nominal declension types in modern Armenian]. Lraber Hasarakakan Gitowt'yownneri 5, 23–30.

Sargsyan, A. (1987). Goyakanakan zowgajewowt'yownnerë žamanakakic' hayerenowm [Noun doublets in contemporary Armenian]. Lezvi ew oči harc'er 10, 123–230.

Sağ, Y. (2019). The semantics of number marking: Reference to kinds, counting, and optional classifiers. Ph. D. thesis, Rutgers University, New Brunswick, NJ.

Sayeed, O. and B. Vaux (2017). The evolution of Armenian. See Klein et al. (2017), pp. 1146–1167.

Scalise, S. and A. Fábregas (2010). The head in compounding. In S. Scales and I. Vogel (Eds.), Cross-Disciplinary Issues in Compounding, pp. 109–125. Amsterdam and Philadelphia: John Benjamins.

Scalise, S., A. Fábregas, and F. Forza (2009). Exocentricity in compounding. Gengo kenky. Journal of the Linguistic Society of Japan 135, 49–84.

Scheer, T. (2011). A guide to morphosyntax-phonology interface theories: How extra-phonological information is treated in phonology since Trubetzkoy's Grenzsignale. Berlin: Mouton de Gruyter.

Scheer, T. (2012). Chunk definition in phonology: Prosodic constituency vs. phase structure. In M. Bloch-Trojnar and A. Bloch-Rozmej (Eds.), Modules and interfaces, pp. 221–253. Lublin: Wydawnictwo KUL.

Scheer, T. (2016). Melody-free syntax and phonologically conditioned allomorphy. Morphology 26(3-4), 341–378.

Schirru, G. (2012). Laryngeal features of Armenian dialects. In B. N. Whitehead, T. Olander, B. Olsen, and J. E. Rasmussen (Eds.), The sound of Indo-European: Phonetics, phonemics, and morphophonemics, pp. 435–457. Copenhagen: Museum Tusculanum Press.

Scobbie, J. (1993a). Issues in constraint violation and conflict. See Ellison and Scobbie (1993), pp. 37–54.

Scobbie, J. M. (1991a). Attribute value phonology. Ph. D. thesis, University of Edinburgh.

Scobbie, J. M. (1991b). Towards declarative phonology. See Bird (1991a), pp. 1–25.

Scobbie, J. M. (1993b). Constraint violation and conflict from the perspective of declarative phonology. Canadian Journal of Linguistics/Revue canadienne de linguistique 38(2), 155–167.

Scobbie, J. M., J. S. Coleman, and S. Bird (1996). Key aspects of declarative phonology. In J. Durand and B. Laks (Eds.), Current Trends in Phonology: Models and Methods, Volume 2. Salford, Manchester: European Studies Research Institute.

Seidl, A. H. (2000). Minimal indirect reference: A theory of the syntax-phonology interface. Ph. D. thesis, University of Pennsylvania.

Selkirk, E. (1980). Prosodic domains in phonology: Sanskrit revisited. In M.-L. Aronoff, Mark Kean (Ed.), Juncture, pp. 107–129. Saratoga: Anma Libri.

Selkirk, E. (1986). On derived domains in sentence phonology. Phonology Yearbook 3(1), 371–405.

Selkirk, E. (1996). The prosodic structure of function words. In J. L. Morgan and K. Demuth (Eds.), Signal to syntax: Bootstrapping from speech to grammar in early acquisition, Volume 187, pp. 214. Mahwah, NJ: Lawrence Erlbaum Associates.

Selkirk, E. (2011). The syntax-phonology interface. See Goldsmith et al. (2011), pp. 435–483.

Selkirk, E. O. (1982). The syntax of words. Number 7 in Linguistic Inquiry Monographs. Cambridge, Mass: MIT Press.

Seropian, H. (1968). A sample transformational generative grammar of West-Armenian. Master's thesis, American University of Beirut.

Sevak, G. (2009). Jhamanakakic hayoc lezvi dasy'nt'ac [Course in Modern Armenian]. Yerevan: YSU Publishing House.

Seyfarth, S. and M. Garellek (2018). Plosive voicing acoustics and voice quality in Yerevan Armenian. Journal of Phonetics 71, 425–450.

Shafiei, N. and T. Graf (2020). The subregular complexity of syntactic islands. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Shaw, P. A. (2005). Non-adjacency in reduplication. See Hurch (2005), pp. 161–210.

Shaw, P. A. (2009). Inside access: The prosodic role of internal morphological constituency. See Hanson and Inkelas (2009), pp. 241–272.

Shibata, C. and J. Heinz (2016). Predicting sequential data with lstms augmented with strictly 2-piecewise input vectors. In S. Verwer, M. van Zaanen, and R. Smetsers (Eds.), Proceedings of The 13th International Conference on Grammatical Inference, Volume 57 of JMLR: Workshop and Conference Proceedings, pp. 137–142.

Shibata, C. and J. Heinz (2019). Maximum likelihood estimation of factored regular deterministic stochastic languages. In Proceedings of the 16th Meeting on the Mathematics of Language, pp. 102–113.

Shwayder, K. (2015). Words and Subwords: Phonology in a Piece-Based Syntactic Morphology. Ph. D. thesis, University of Pennsylvania, Philadelphia, PA.

Siddiqi, D. and H. Harley (Eds.) (2016). Morphological Metatheory, Volume 229. Linguistik Aktuell/Linguistics Today.

Siegel, D. C. (1974). Topics in English morphology. Ph. D. thesis, Massachusetts Institute of Technology.

Sigler, M. (1997). Specificity and agreement in standard Western Armenian. Ph. D. thesis, Massachusetts Institute of Technology.

Skopeteas, S. (2019). Prosodic structure and word-final stress in Eastern Armenian. Unpublished ms. Universität Bielefeld.

Sowk'iasyan, A. M. (2004). Žamanakakic' hayoc' lezow (Hnčyownabanowt'yown, baṙagitowt'yown, baṙakazmowt'yown) [Modern Armenian phonology, lexicology, and word-formation]. Yerevan: Yerevani Petakan Hamalsarani Hratarakčowt'yown.

Spencer, A. (1988). Bracketing paradoxes and the English lexicon. Language 64(4), 663–682.

Sproat, R. (1988). Bracketing paradoxes, cliticization and other topics: The mapping between syntactic and phonological structure. In M. Everaert, A. Evers, R. HuyBregts, and M. Trommelen (Eds.), Morphology and modularity, pp. 339–360. Dordrecht: Foris.

Sproat, R. (1992a). Unhappier is not a "bracketing paradox". Linguistic Inquiry 23(2), 347–352.

Sproat, R. W. (1985). On deriving the lexicon. Ph. D. thesis, Massachusetts Institute of Technology.

Sproat, R. W. (1992b). Morphology and computation. Cambridge:MA: MIT press.

Steddy, S. (2019). Compounds, composability, and morphological idiosyncrasy. The Linguistic Review 36(3), 453–483.

Steriade, D. (2000). Paradigm uniformity and the phonetics-phonology boundary. In M. Broe and J. Pierrehumbert (Eds.), Papers in laboratory phonology, Volume 5, pp. 313–334. Cambridge: Cambridge University Press.

Steriade, D. (2008a). Contour correspondence: The segmental evidence. Paper presented at the 11th International Symposium on Chinese Languages and Linguistics, National Chiao Tung University, Taiwan.

Steriade, D. (2008b). A pseudo-cyclic effect in Romanian morphophonology. In A. Bachrach and A. Nevins (Eds.), Inflectional identity, pp. 313–355. Oxford: Oxford University Press.

Steriade, D. (2017). ATB-shifts and ATB-blockage in vocalic plateaus. Unpublished ms., MIT.

Sterling, G. E. (2004). Armenian paradigms. Leuven, Belgium: Peeters Pub & Booksellers.

Strauss, S. L. (1982). On "relatedness paradoxes" and related paradoxes. Linguistic Inquiry 13(4), 694–700.

Strother-Garcia, K. (2018). Imdlawn Tashlhiyt Berber syllabification is quantifier-free. In Proceedings of the Society for Computation in Linguistics, Volume 1, pp. 145–153.

Strother-Garcia, K. (2019). Using model theory for grammatical inference: A case study from phonology. Ph. D. thesis, University of Delaware.

Strother-Garcia, K., J. Heinz, and H. J. Hwangbo (2017). Using model theory for grammatical inference: A case study from phonology. In International Conference on Grammatical Inference, pp. 66–78.

Stump, G. T. (1995a). Two types of mismatch between morphology and semantics. In E. Schiller, E. Steinberg, and B. Need (Eds.), Autolexical Theory: Ideas and Methods, Number 85 in Trends in Linguistics: Studies and Monographs, pp. 291–318. Berlin: Mouton De Gruyter.

Stump, G. T. (1995b). The uniformity of head marking in inflectional morphology. In Yearbook of Morphology 1994, pp. 245–296. Springer.

Stump, G. T. (2001). Inflectional morphology: A theory of paradigm structure. Number 93 in Cambridge Studies in Linguistics. Cambridge: Cambridge University Press.

Su, Y.-Y. J. (2012). The syntax of functional projections in the vP periphery. Ph. D. thesis, University of Toronto.

Szpyra, J. (1989). The phonology-morphology interface: Cycles, levels and words. London: Routeledge.

Tamrazian, A. (1994). The syntax of Armenian: Chains and the auxiliary. Ph. D. thesis, University of London.

Tat, D. (2013). Word syntax of nominal compounds: Internal and aphasiological evidence from Turkish. Ph. D. thesis, The University of Arizona.

Ten Hacken, P. (2009). Early generative approaches. In R. Lieber and P. Štekauer (Eds.), The Oxford handbook of compounding, pp. 54–77. Oxford/New York: Oxford University Press.

ter Meulen, A. G. (2012). Mathematical linguistics. Oxford Bibliographies.

Thomas, W. (1982). Classifying regular events in symbolic logic. Journal of Computer and System Sciences 25(3), 360–376.

Thomas, W. (1997). Languages, automata, and logic. In G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Volume 3, pp. 389–455. New York, NY, USA: Springer-Verlag New York, Inc.

Thomson, R. W. (1989). An introduction to Classical Armenian. Delmar, NY: Caravan Books.

Toparlak, T. (2019). Etudes phonétiques en arménien. Master's thesis, Université Paris 3 - Sorbonne Nouvelle.

T'oxmaxyan, R. M. (1971). Žamanakakic' hayereni baṙayin šeštë [Modern Armenian word stress]. <u>Lraber Hasarakakan Gitowt'yownneri</u> <u>2</u>, 60–64.

T'oxmaxyan, R. M. (1975). Žamanakakic' hayereni baṙayin šešti bnowyt'ë [The essence of the word stress of contemporary Armenian]. <u>Lezvi ew oči harc'er</u> <u>3</u>, 123–207.

T'oxmaxyan, R. M. (1983). <u>Žamanakakic' hayereni šeštabanowt'yownë [The accentology of contemporary Armenian]</u>. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

T'oxmaxyan, R. M. (1988). Hayereni hnčowyt'neri kapakc'eliowt'yownë grayin ew artasanakan makardaknerowm [Connectivity of Armenian phonemes in written and spoken language]. <u>Lezvi ew oči harc'er</u> <u>1</u>, 69–135.

Trommer, J. (2012). <u>The morphology and phonology of exponence</u>. Number 41 in Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.

Trommer, J. (2013). Stress uniformity in Albanian: Morphological arguments for cyclicity. <u>Linguistic Inquiry</u> <u>44</u>(1), 109–143.

Truckenbrodt, H. (1995). <u>Phonological phrases–their relation to syntax, focus, and prominance</u>. Ph. D. thesis, Massachusetts Institute of Technology.

Truckenbrodt, H. (1999). On the relation between syntactic phrases and phonological phrases. <u>Linguistic Inquiry</u> <u>30</u>(2), 219–255.

Tyler, M. (2019). Simplifying match word: Evidence from English functional categories. <u>Glossa: A journal of general linguistics</u> <u>4</u>(1).

Vaillette, N. (2003). Logical specification of regular relations for NLP. <u>Natural Language Engineering</u> <u>9</u>(1), 65–85.

Vaillette, N. (2004). <u>Logical specification of finite-state transductions for natural language processing</u>. Ph. D. thesis, The Ohio State University, Columbus, OH.

van der Linden, E. (1991). Accent placement and focus in categorial logic. See Bird (1991a), pp. 197–217.

van Oostendorp, M. (2004). Crossing morpheme boundaries in Dutch. <u>Lingua</u> <u>114</u>(11), 1367–1400.

van Oostendorp, M., C. Ewen, E. Hume, and K. Rice (Eds.) (2011). <u>The Blackwell companion to phonology</u>. Malden, MA: Wiley-Blackwell.

Vaux, B. (1992). Adjarian's Law and consonantal ATR in Armenian. In J. Greppin (Ed.), <u>Proceedings of the Fourth International Conference on Armenian Linguistics</u>, Delmar, New York, pp. 271–293. Caravan.

Vaux, B. (1993). Coronal fronting in the Akn dialect of Armenian. <u>Annual of Armenian linguistics</u> <u>14</u>, 15–29.

Vaux, B. (1994). Wackernagel's law in Classical Armenian. <u>Revue des études arméniennes</u> <u>25</u>, 17–42.

Vaux, B. (1995a). A problem in diachronic Armenian verbal morphology. In J. Weitenberg (Ed.), <u>New Approaches to Medieval Armenian Language and Literature</u>, pp. 135–148. Amsterdam: Rodopi.

Vaux, B. (1995b). Vowel harmony in the Armenian dialect of Karchevan. In Proceedings of the 9th Biennial Non-Slavic Languages Conference, Chicago, May 1995, Volume 9, pp. 1–9. Chicago: Chicago Linguistic Society.

Vaux, B. (1996a). The status of ATR in feature geometry. Linguistic Inquiry 27(1), 175–182.

Vaux, B. (1996b). Vowel harmony in the Armenian dialect of Marash. Association Internationale des Etudes Armeniennes, 1–11.

Vaux, B. (1997). The phonology of voiced aspirates in the Armenian dialect of New Julfa. In N. Awde (Ed.), Armenian Perspectives. 10th Anniversary Conference of the Association Internationale des Études Arméniennes, Richmond, Surrey, pp. 231–248. Curzon.

Vaux, B. (1998a). The laryngeal specifications of fricatives. Linguistic inquiry 29(3), 497–511.

Vaux, B. (1998b). The phonology of Armenian. Oxford: Clarendon Press.

Vaux, B. (2000). Notes on the Armenian dialect of Ayntab. Annual of Armenian linguistics 20, 55–82.

Vaux, B. (2001a). The Armenian dialect of Aslanbeg. Annual of Armenian linguistics 21, 31–64.

Vaux, B. (2001b). Hemshinli: The forgotten Black Sea Armenians. Journal of Armenian Studies 6, 47–71.

Vaux, B. (2002). The Armenian dialects of Jerusalem. In M. Stone (Ed.), Armenians in the Holy Land, pp. 231–254. Leuven: Peeters.

Vaux, B. (2003). Syllabification in Armenian, Universal Grammar, and the lexicon. Linguistic Inquiry 34(1), 91–125.

Vaux, B. (2006). The Armenian dialect of Digranagert/Tigranakert and Urfa. In R. G. Hovannisian (Ed.), Armenian Tigranakert/Diyarbakir and Edessa/Urfa, UCLA Armenian History And Culture Series. Historic Armenian Cities and Provinces, pp. 191–207. Costa Mesa, CA: Mazda.

Vaux, B. (2007). Homshetsma: The language of the Armenians of Hamshen. In H. H. Simonian (Ed.), The Hemshin, pp. 257–278. New York: Routledge.

Vaux, B. (2008). Zok: The Armenian dialect of Agulis. In B. D. Mugrdichian (Ed.), In between Paris and Fresno: Armenian Studies in Honor of Dickran Kouymjian, pp. 283–301. Costa Mesa, CA: Mazda Press.

Vaux, B. (2012). The Armenian dialect of Smyrna. In R. Hovannisian (Ed.), Armenian Smyrna/Izmir: The Aegean Communities, pp. 111–126. Costa Mesa, CA: Mazda Publishers.

Vaux, B. (2013). The Armenian dialect of Khodorjur. In K. A. Arkun and V. Rowe (Eds.), hodorchur: Lost Paradise. Memories of a land and its people. Monterey, CA: Mayreni Publishing.

Vaux, B., S. La Porta, and E. Tucker (1996). Ethnographic materials from the Muslim Hemshinli with linguistic notes. Annual of Armenian linguistics 17, 25–45.

Vaux, B. and B. Samuels (2005). Laryngeal markedness and aspiration. Phonology 22(3), 395–436.

Vaux, B. and A. Wolfe (2009). The appendix. See Raimy and Cairns (2009), pp. 101–144.

Vaysse, O. (1986, December). Addition molle et fonctions p-locales. Semigroup Forum 34(1), 157–175.

Vigário, M. (2010). Prosodic structure between the prosodic word and the phonological phrase: Recursive nodes or an independent domain? The Linguistic Review 27(4), 485–530.

J̌ahowkyan, G. (2010). Hayeren stowgabanakan baṙaran [Armenian etymological dictionary]. Yerevan: Asoġik Hratarakčowt'yown.

Vogel, I. (1989). Prosodic constituents in Hungarian. Acta Linguistica Hungarica 39(1/4), 333–351.

Vogel, I. (2008). The morphology-phonology interface: Isolating to polysynthetic languages. Acta Linguistica Hungarica 55(1-2), 205–226.

Vogel, I. (2009). The status of the clitic group. See Grijzenhout and Kabak (2009), pp. 15–46.

Vogel, I. (2010). The phonology of compounds. In S. Scales and I. Vogel (Eds.), Cross-Disciplinary Issues in Compounding, pp. 145–67. Amsterdam and Philadelphia: John Benjamins.

Vogel, I. (2012). Recursion in phonology? In B. Botma and R. Noske (Eds.), Phonological Explorations: Empirical, Theoretical and Diachronic Issues, pp. 41–62. Berlin/Boston: Walter de Gruyter.

Vogel, I. (2016). Life after the strict layer hypothesis: Prosodic structure geometry. In Y. Zhang, Hongming Qian (Ed.), Prosodic Studies: Challenges and Prospects. London: Routledge.

Vu, L. H. (2019). A quantifier-based approach to NPI-licensing typology: Empirical and computational investigations. Ph. D. thesis, University of Delaware, Newark, DE.

Vu, M. H. (2018). Towards a formal description of npi-licensing patterns. In Proceedings of the Society for Computation in Linguistics, Volume 1.

Vu, M. H., N. Shafiei, and T. Graf (2019). Case assignment in TSL syntax: A case study. In G. Jarosz, M. Nelson, B. O'Connor, and J. Pater (Eds.), Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pp. 267–276.

Vu, M. H., A. Zehfroosh, K. Strother-Garcia, M. Sebok, J. Heinz, and H. G. Tanner (2018). Statistical relational learning with unconventional string models. Frontiers in Robotics and AI 5, 76.

Wagner, M. (2005). Prosody and recursion. Ph. D. thesis, Massachusetts Institute of Technology.

Wagner, M. (2010). Prosody and recursion in coordinate structures and beyond. Natural Language & Linguistic Theory 28(1), 183–237.

Walther, M. (1993). Declarative syllabification with applications to German. See Ellison and Scobbie (1993), pp. 55–79.

Walther, M. (1998). Computing declarative prosodic morphology. In SIGPHON'98 The Computation of Phonological Constraints.

Walther, M. (1999). One-level prosodic morphology. arXiv preprint cs/9911011.

Walther, M. (2000). Finite-state reduplication in one-level prosodic morphology. In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, NAACL 2000, Seattle, Washington, pp. 296–302. Association for Computational Linguistics.

Wareham, H. (1999). Systematic parameterized complexity analysis in computational phonology. Ph. D. thesis, University of Victoria, Victoria, BC.

Weitenberg, J. (2017). The dialectology of Armenian. See Klein et al. (2017), pp. 1132—1146.

Weitenberg, J. J. (2002). Aspects of Armenian dialectology. In J. Berns and J. van Merle (Eds.), Present-day dialectology: Problems and findings, Volume 137 of Trends in linguistics studies and monographs, pp. 141–157. Berlin & New York: Mouton De Gruyer.

Wheeler, D. (1988). Consequences of some categorially-motivated phonological assumptions. In Categorial grammars and natural language structures, pp. 467–488. Springer.

Wheeler, D. W. (1981). Aspects of a categorial theory of phonology. Ph. D. thesis, University of Massachusetts Amherst.

Wiebe, B. (1992). Modelling autosegmental phonology with multi-tape finite state transducers. Master's thesis, Simon Fraser University.

Wiese, R. (Ed.) (1994). Recent developments in lexical phonology. Düsseldorf: Heinrich-Heine-Universität.

Williams, E. (1981). On the notions "lexically related" and "head of a word". Linguistic Inquiry 12(2), 245–274.

Williams, S., C. Biggs, and R. Brasington (1989). A computational implementation of aspects of lexical phonology. Communication & Cognition - Artificial Intelligence 6(1), 27–33.

Williams, S. M. (1993). LexPhon: A computational implementation of aspects of lexical phonology. Ph. D. thesis, University of Reading.

Williams, S. M. (1994). Lexical phonology and speech style: Using a model to test a theory. In Computational Phonology.

Wolf, M. (2008). Optimal interleaving: Serial phonology-morphology interaction in a constraint-based model. Ph. D. thesis, Universit of Massachusetts, Amherst, Amherst, MA.

Wolf, M. (2011). Exceptionality. See van Oostendorp et al. (2011), pp. 2538–2559.

Wolf, M. (2013). Candidate chains, unfaithful spell-out, and outwards-looking phonologically-conditioned allomorphy. Morphology 23(2), 145–178.

Xačatryan, A. (1988). Žamanakakic' hayereni hnčyownabanowt'yown [Phonology of Contemporary Armenian]. Yerevan: Haykakan SSH Gitowt'yownneri Akademiayi Hratarakčowt'yown.

Xačatryan, A. (2009a). Bayakan baġadričov kazmvaç bown kam iskakan bardowt'yownneri šarahyowsakan haraberowt'yownnerë [Syntactic relations of compounds made from verbs]. Kant'eġ. Gitakan hodvaçneri žoġovaçow 2, 86–91.

Xačatryan, A. (2009b). Bayakan himk'erov kazmvaç bown kam iskakan bardowt'yownneri jewabanakan kaġaparnerë [Morphological models of real or essential compounds formed with verbal stems]. Lraber Hasarakakan Gitowt'yownneri 3, 217–224.

Yap, N. T. (2006). Modeling syllable theory with finite-state transducers. Ph. D. thesis, University of Delaware.

Yeghiazaryan, L. (2010). Caso, definitude e os sintagmas nominais no armênio. Ph. D. thesis, Universidade de São Paulo.

Yli-Jyrä, A. (2013). On finite-state tonology with autosegmental representations. In Proceedings of the 11th international conference on finite state methods and natural language processing. Association for Computational Linguistics.

Yli-Jyrä, A. (2015). Three equivalent codes for autosegmental representations. In Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing 2015 (FSMNLP 2015 Düsseldorf).

Yli-Jyrä, A. (2019). Optimal kornai-karttunen codes for restricted autosegmental representations. In C. Condoravdi and T. H. King (Eds.), Tokens of Meaning: Papers in Honor of Lauri Karttunen. Center for the Study of Language and Information (CSLI).

Yu, K. (2017). Advantages of constituency: Computational perspectives on Samoan word prosody. In International Conference on Formal Grammar 2017, Berlin, pp. 105–124. Spring.

Yu, K. M. (2019). Parsing with minimalist grammars and prosodic trees. In R. C. Berwick and E. P. Stabler (Eds.), Minimalist Parsing, pp. 69–109. London: Oxford University Press.

Yu, K. M. and E. P. Stabler (2017). (in) variability in the Samoan syntax/prosody interface and consequences for syntactic parsing. Laboratory Phonology: Journal of the Association for Laboratory Phonology 8(1), 1–44.

Zec, D. (2005). Prosodic differences among function words. Phonology 22(1), 77–112.

Zhu, Y. (2020). Extending the autosegmental input strictly local framework: Metrical dominance and floating tones. In Proceedings of the Society for Computation in Linguistics, Volume 3.

Zwicky, A. M. (1985). Heads. Journal of linguistics 21(1), 1–29.