Language (2026), **102**:1, 198–222
doi:10.1017/S0097850726000056

Linguistic Society of America

TEACHING LINGUISTICS

Phonology as coding: An online tool for teaching and developing analyses

Daniel A. Kaufman[1,2], Raphael Finkel[3], and Cynthia Gan[4]

[1]Queens College, LCD, The City University of New York, Queens, NY, USA
[2]Endangered Language Alliance, New York, NY, USA
[3]Department of Computer Science, University of Kentucky, Lexington, KY, USA
[4]Hunter College, CUNY, New York, NY, USA

Corresponding author: Daniel A. Kaufman; Email: dkaufman@qc.cuny.edu

Received: 18 November 2022; **Revised:** 12 October 2025; **Accepted:** 16 December 2025

Keywords: pedagogy; computational phonology; generative phonology; phonological features

Abstract

The Phonomaton, a public web-facing facility, computes phonological derivations based on a user's underlying representations and rules. The tool allows a formal implementation of phonological analyses using familiar methods and lets students interactively explore the mechanics of feature systems and serial derivations. We demonstrate a number of the program's features and end with a discussion of its implementation in the classroom.

1. Introduction

We introduce here the Phonomaton, a public web-facing facility (https://www.kratylos.org/ph) designed to assist students in introductory phonology courses as well as more advanced phonologists in modeling and comparing complex derivations. The program produces step-by-step derivations based on user-provided representations and rules that can make full use of features and autosegmental tiers.[1] We then discuss an implementation of this tool in a semester-long trial and consider the ways in which such tools can facilitate the instruction of theoretical linguistics more generally.

The Phonomaton aims both to be powerful enough to model the full phonology of any language and also to serve as a pedagogical tool for introducing fundamental notions of phonology in an interactive manner that encourages exploration. The tool facilitates teaching underlying representations, rule notation (including alpha notation, segment indexing, and regular expressions), feature systems, natural classes, serial derivations, morphological levels, and autosegmental phonology. The Phonomaton allows a user to formulate hypotheses and immediately test them by manipulating an analysis and seeing its output. We also envision the tool as a gateway to computer science. It instantiates a high-level programming language (albeit one that looks nearly identical to familiar rule formalism) and brings to life the

[1] A companion program, the Optimaton (https://www.kratylos.org/op), which is not discussed here for reasons of space, produces evaluations of violable constraints in the style of **Optimality Theory** tableaux, with user-provided underlying representations, candidates, and constraints.

© The Author(s), 2026. Published by Cambridge University Press on behalf of the Linguistic Society of America. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike licence (http://creativecommons.org/licenses/by-nc-sa/4.0), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the same Creative Commons licence is used to distribute the re-used or adapted article and the original article is properly cited. The written permission of Cambridge University Press or the rights holder(s) must be obtained prior to any commercial use.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

procedural algorithms that have been kept until now in the province of the written page, even among introductions with a strong focus on formalization (e.g. Bale & Reiss 2018).

In the following subsection, we contextualize the program in the current landscape of linguistic software for professional and pedagogical use. In Section 2, we give an overview of the core functions of the software, focusing on how it handles segments and features (Section 2.1) and phonological rules (Section 2.2). This latter section aims to give a feel of what the program is capable of by examining not only the basic rule formalism (Section 2.3), but also underspecification (Section 2.4), alpha notation (Section 2.5), morphological boundaries (Section 2.6), segment indexing (Section 2.7), quantifiers and regular expressions (Section 2.8), types of rule application (Section 2.9), and morphological paradigms (Section 2.10). In Section 3, we discuss how we have employed the program in the classroom and the results of a student survey on its implementation. We conclude in Section 4 with a note on future pedagogical and technological prospects. Readers who are less interested in the mechanics of the program and more concerned with its pedagogical application should be able to skip to Section 3 without getting lost.

1.1. The state of the field

At present, few computational tools are available for teaching phonology. Pheatures Spreadsheet (van Vugt 2012), which supersedes the earlier FeaturePad (Zuraw 2004) and PhonologyPad (Albro 1999), allows users to explore natural classes within inventories. The program also shows the result of altering a phonological feature and can check answers to questions involving natural classes. Given its narrower focus, however, Pheatures Spreadsheet lacks the power to model phonological derivations/evaluations. Phonology Assistant (Summer Institute of Linguistics (SIL) 2005) is even more distant, as it is aimed not at students but rather at linguists seeking to describe a phonological system. Phonology Assistant works in tandem with other software produced by SIL (FLEx and Toolbox) to help the user derive as many phonological generalizations as possible from the lexicon (including stochastic tendencies). But like the previous programs, it lacks the ability to model alternations and more resembles a corpus tool. The program closest to the one presented here is Derive! (Steel & Jurgec 2017), an interactive browser-based program that calculates serial derivations using SPE-style rules (*Sound pattern of English*; Chomsky & Halle 1968) and a customizable set of features. Despite these similarities, the Phonomaton is, to our knowledge, the first publicly available program designed to model derivational phonology in its full breadth, using feature geometry, underspecification, autosegmental representations, and morphological categorization, and including as well as the ability to download and upload analyses.[2]

We believe that formalization has broader implications from a field-wide perspective and that our efforts address two unfortunate trends: increasing incommensurability across analyses and frameworks, and the marginalization of theoretical linguistics in applied fields. With the proliferation of theories, frameworks, and proposals, the field has reached a point where no two linguists play by all the same rules, with disagreements both in details and fundamental concepts. One factor in this trend, we believe, has been the informal nature in which arguments and analyses are customarily presented. Generative linguistics was long distinguished by a solid mathematical foundation with a paradoxical antagonism toward computational implementation, especially puzzling considering the early and widespread recognition that generative linguistics shares much with the algorithmic style of programming.[3] Halle had already stated in the forward to his 1959 *The sound pattern of Russian*,

[2] Software with a similar goal has long been used to good effect in semantics and logic (e.g. Larson et al. 1998, Barker-Plummer et al. 1999, Champollion et al. 2007, inter alia) as well as morphology (Finkel 2016). However, nearly all of the computational modeling of serial derivations has centered on historical sound change, as summarized by Piwowarczyk (2022), rather than synchronic phonology.

[3] Kyle Gorman (p.c.) believes that Johnson (1972) solved the question of SPE's computability, but that the computational power to implement such a formalism with a full feature system was not widely available until the 1990s. In Gorman's view, computational implementations of phonological theory did not suffer from an anti-implementation ethos, as suggested here, but rather for the average working linguist the advantages of implementation were insufficiently compelling.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

> I have assumed that an adequate description of a language can take the form of a set of rules—analogous perhaps to a program of an electronic computing machine—which when provided with further special instructions, could in principle produce all and only well-formed (grammatical) utterances in the language in question. (Halle 1971 [1959]:12)

By the early 1970s, however, much of the substantive formal foundations of Chomsky's earliest work, including SPE (Chomsky & Halle 1968), had largely fallen away, leaving a residue of what critics often refer to as 'pseudomath'.[4] The consequences of this aversion have been both scientific and sociological. First, practical applications of computational linguistics are now dominated by statisticians and engineers rather than experts in language structure. Consequently, the field of theoretical linguistics has boxed itself out of what could have been its most lucrative applications.[5] This is not to say that a field devoted to understanding the human cognitive system should be driven by economic concerns, but now, in the eighth decade of generative linguistics, it seems quite likely that the intuition-based (and, arguably, compu-phobic) approach that has defined mainstream linguistics in the US has made pursuing a profession in the field a gamble that only few can afford. Put more bluntly, a field of study that should have wide-ranging practical applications throughout society has largely settled for reproducing linguists.[6] Given the persistent socioeconomic inequities along racial and ethnic lines in the US and elsewhere, the field's long-standing inclination toward 'pure' theory could have only negative implications for diversity by narrowing practical applications and career paths.[7] Needless to say, the promotion of diversity and equity faces multiple obstacles requiring multiple solutions; here, we only seek to nudge linguistics pedagogy toward a formal rigor with wider application.

Within the pedagogical context, the Phonomaton aims to promote computational thinking more generally, that is, to develop algorithmic problem-solving techniques that a computer can carry out (Papert 1990, Wing 2006) and that are generalizable to any number of professional fields. Computational thinking is often defined as having the following steps.[8]

- Decomposition: reducing complex problems to simpler components
- Pattern recognition: identifying similarities within and between problems
- Abstraction: using idealized, abstract representations to simplify patterns
- Designing algorithms: creating a sequence of instructions to solve a problem
- Evaluation: comparing the advantages and disadvantages of alternative solutions

Any linguist will likely recognize that these are precisely the steps taken implicitly or explicitly in the analysis of natural language. Considering that problem solving in computer science is so naturally akin to problem solving in linguistics, it appears as even more of a lost opportunity that formal links between the two fields have until recently been restricted to computational linguistics proper and have not permeated

[4] There is likely a direct link between Chomsky's increasing emphasis at the time on the 'Galilean/Cartesian method' of scientific inquiry and the decreasing attention to 'computer realizability' in theoretical linguistics, but this discussion is clearly beyond our scope here.

[5] This situation was summed up semi-jokingly by Fred Jelinek, a pioneer in automatic speech recognition: 'Every time I fire a linguist, the performance of our speech recognition system goes up'.

[6] The 2018 NSF Survey of Doctorate Recipients indicates that 78% of Linguistics Ph.D.s are employed by educational institutions, while 20% are employed by business/industry and 2% by government.

[7] Charity Hudley, Mallinson, and Bucholtz (2020:e202) point out that, as of 2018, '[t]he population of ethnic minorities with advanced degrees in linguistics is so low in the U.S. that few federal agencies report data for these groups' (quoting from the Linguistic Society of America 2019:28). But this is not to say that the path from formalization to diversity is either straight or guaranteed. On the contrary, fields such as mathematics, computer science, and physics face similar struggles to diversify without the additional hurdle described here. It must be noted, though, that many measures of ethnic/racial diversity in Linguistics lag behind the average in the sciences (Einaudi et al. 2022).

[8] For whatever reason, the final element, evaluation, is left out in descriptions of computational thinking more than it is included. In the field of linguistics, comparison of alternative analyses and evaluation against economic metrics have long held pride of place, arguably far more so than in computer science.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

significantly into the core subdisciplines of phonology, morphology, syntax, and semantics.[9] One of the pedagogical goals of the Phonomaton is thus to make explicit the very steps laid out above by presenting phonology as coding. Although we believe this approach provides transferable skills, we emphasize that the aim is not to train students to become programmers; it is to use the computer to improve our own understanding of a phenomenon. In the words of Alan Perlis, 'Whereas we **think** we know something when we learn it, and are **convinced** we know it when we can teach it, the fact is that we don't **really** know it until we can code it for an automatic computer!' (quoted by George Forsythe in his 1958 address to the Mathematical Association of America; see e.g. Denning & Tedre 2019).

In the following, we introduce the main features of the Phonomaton in some detail but refer the reader to the documentation for a more complete description of all of the program's possibilities. For reasons of space and scope, we do not discuss autosegmental phonology or the use of the morphological tier.

2. Creating a derivation in the Phonomaton

Upon opening the Phonomaton in a web browser, we see the three fields that contain the principle components of an analysis:

- **Phonological rules:** Contains a preamble, where the user can declare an inventory, custom phonemes, custom feature geometries, and (inviolable) surface filters, followed by a series of SPE-style phonological rules, which comprise the derivation proper.
- **Underlying representations (URs):** A set of abstract phonological forms (provided by the user in IPA), which will serve as the input to the serial rules in the previous field. This field may optionally contain the output target for each UR, which the program will use to check against the actual output as derived by the user's rules.
- **Morphological rules:** An optional field containing morphological rules written in the same style as phonological ones (e.g. ∅ → -ɪŋ / _>), but with a different application. Each rule in this field applies to each UR independently to create a new stem that is then fed into the phonological rules. For example, if there are three rules in this field, three additional derivational columns will be created for each UR, representing three morphological derivations. This feature is designed for solving problems that involve morphological paradigms.

In addition to the above fields, we find the following ancillary fields:

- **Metadata:** Contains subfields for the provenance of the data, author of the analysis, and analysis version, among other types of information. This metadata, supplied by the user, is meant to organize downloaded analyses and to make analyses easily searchable if they are uploaded to the Phonomaton's library.
- **Sample rules:** A set of sample phonological rules that can be inserted into a derivation with a click of a button.
- **Library:** A collection of implemented analyses and classic phonology problems, which the user can open and experiment with.
- **Historical data sets:** Contains two large data sets of Austronesian historical data: Blust et al.'s (2023) *Austronesian comparative dictionary*, and Edwards's (2021a,b) *Rote-Meto comparative dictionary.* Clicking on a particular language from one of these resources sets the user up to derive attested forms in that language from reconstructed etyma.
- **IPA chart:** A full IPA chart that displays associated feature sets for each segment and allows the user to compare segments and test for natural classes.

[9] There is, for instance, nary a mention of computers nor of computer science in Anderson's (2021) overview of phonology in the twentieth century, although a note appears in the final paragraph of the book regarding recent interest in statistical and corpus approaches. While the blossoming of such approaches may bode well for the field, computational methods should be understood to range far beyond stochastic frameworks to include classic algorithmic thinking.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

- **Segments and features:** Contains the full feature matrix for each IPA segment, following Hayes (2009). As in the IPA chart, clicking on multiple segments highlights their similarities and differences.

2.1. Segments, features, and inventories

Figure 1 shows the use of the interactive IPA chart. Here, the user has clicked on four IPA symbols [θ s ʃ ʂ], which are now highlighted, and the program displays their differences in the matrix shown in the lavender pop-out window. In this case, the segments differ only in the features [anterior], [distributed], and [strident], so only these features are presented in the matrix. Below that, it lists all of the features these segments have in common. By default, the Phonomaton employs the feature set of Hayes 2009, although the user can modify the feature sets of segments as well as introduce new segments (see Section 2.1).

The Phonomaton can also compare segments in conjunction with an inventory. An inventory is declared by listing a set of phones in the preamble (e.g. 'Inventory: a i u p t k') or simply by selecting segments in the IPA chart and clicking the 'Make inventory' button.

When an inventory has been declared, the segments it contains are highlighted in the chart with a blue outline, as seen in Figure 2. At this point, clicking any group of segments shows their distinctive features if they constitute a natural class within the inventory, in addition to the standard feature comparison. In Figure 2, the user has clicked three segments, [t θ s], and the lavender window shows that this is a natural class within the inventory with distinctive features [−voice, +anterior].

If no set of features uniquely distinguishes the group of selected segments, the software notes that fact and suggests the closest match containing all of the selected segments. For instance, given the same inventory as in Figure 2, the user has clicked [d n θ ð s] in Figure 3. The lavender box now informs us that this collection of segments does not form a natural class and suggests the closest match, which in this case involves adding [z] to the set.

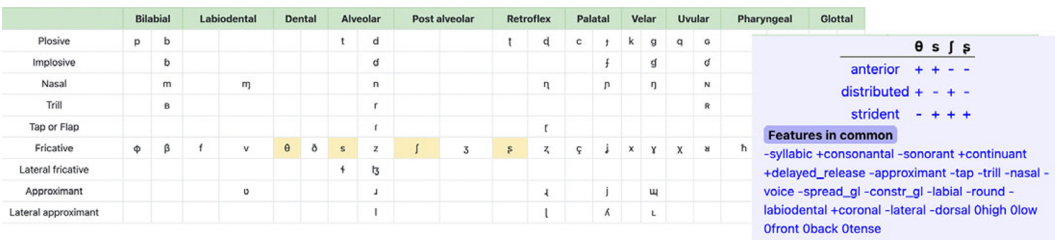


Figure 1. *Comparison of segments without an inventory.*

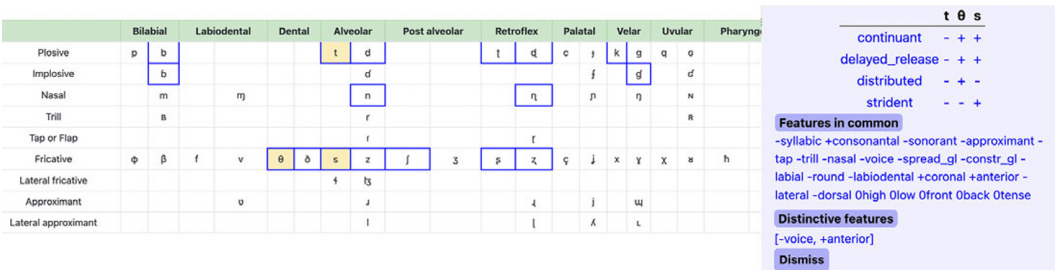


Figure 2. *Comparing segments within an inventory that form a natural class.*

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press
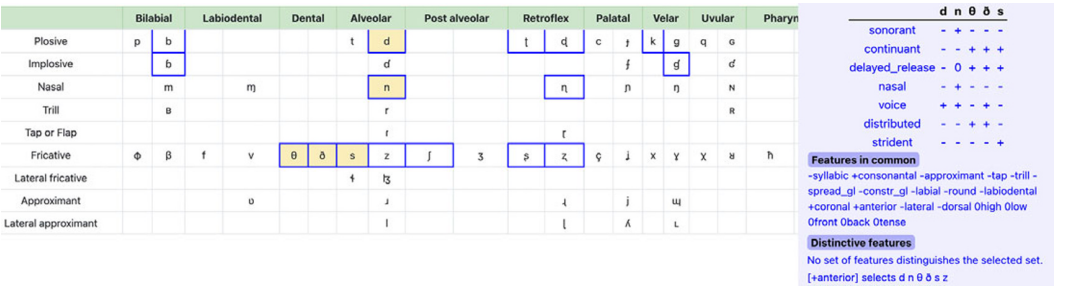


Figure 3. *Comparing segments within an inventory that do not form a natural class.*



Figure 4. *Selecting segments by feature.*

In addition to selecting segments directly on the IPA chart, the user can select segments by their features using the interface shown in Figure 4. Upon selection of a set of feature specifications, the relevant segments are highlighted in the IPA chart.

2.1.1. Custom segments

The Phonomaton also allows the user to create new segments based on a custom set of feature specifications. This facility can also be used to create segments that are underspecified for particular features. To take a concrete example, obstruent voicing in Turkish suggests an analysis in which stops in the underlying stratum are underspecified for voice, whereas those in loan strata are specified. Inkelas (1995) posits the set of stops in 1, where the 'archiphonemes' in 1a lack a voice feature altogether, and the fully specified sets in 1b and 1c have a negative and positive value for the [voice] feature, respectively.

(1) a. /P T K/ [ ]
b. /p t k/ [−voice]
c. /b d g/ [+voice]

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

The user can define segments in the preamble of the rules using the syntax shown in 2.[10]

(2) Define: T[-syllabic,+consonantal,-sonorant,-continuant,-delayed_release,-approximant,-tap,-trill, -nasal,-spread_gl,-constr_gl,-labial,-round,-labiodental,+coronal,+anterior,-distributed,-strident, -lateral,-dorsal,0high,0low,0front,0back,0tense]

The declaration in 2 creates a new segment /T/ that is identical to /t/ except that it lacks the [voice] feature. 'Define' can create completely novel segments with any feature set, and one can also redefine the feature sets of existing segments by using 'Redefine'. For a language without contrastive voicing in stops, we could thus redefine stops such as /p t k/ as lacking a [voice] feature entirely.[11] As discussed below in Section 2.4, 'feature-filling' rules can subsequently target segments with underspecified features to provide them with a value in a particular environment.[12]

2.2. Phonological rules

Beyond functioning as a feature calculator, the strength of the Phonomaton lies in its ability to execute ordered rules and to derive surface forms from underlying representations. Phonological rules are written in the traditional SPE notation: target → change / environment (optionally preceded by a rule name), and the software executes them in the order they are written. The user can type rules from scratch in the phonological rules field or make use of the sample rules, shown in Figure 5, which are automatically pasted into a derivation when clicked and can be modified manually from there. The sample rule collection contains simple segmental rules as well as more complex ones for syllabification, mora assignment, footing, and stress, which can be experimented with freely.

A simple derivation with two of the sample rules applying to user-generated representations is shown in Figure 6.

Figure 6 shows two ordered rules that operate on the right edge of the word (symbolized by '>'; see Section 2.6 for the full set of boundary symbols), with three underlying representations. The resulting derivation is shown at the top, where light blue cells show intermediate representations and blank cells represent the failure of a rule to apply.

2.3. The target and change

The user can specify the target, change, and environment of a rule directly in IPA symbols and the common abbreviations C (for [−syllabic] segments), V (for [+syllabic] segments), and X (for all segments, excluding boundary symbols), as in 3a. All parts of a rule can also refer to features, as in 3b.

(3) a. k → ɣ / V꞉_V
b. [-continuant,-voice] → [+continuant,+voice,+delayed_release] / [+syllabic,+long]_[+syllabic]

Rules can make use of traditional feature notation, exemplified in 3b, but the Phonomaton also introduces several novel methods for relating to features that go beyond textbook treatments. The symbol

[10] The user need not type the specified features manually. Clicking on any segment in the IPA chart opens the aforementioned lavender box, which allows one to copy the entire feature matrix into the clipboard.

[11] In keeping with Hayes 2009, the feature matrix also contains zero specifications, which represent unlicensed features. For instance, [strident] is licensed by [+coronal], and the tongue-position features [low, high, front, back] are licensed by [+dorsal]. There are thus four polarities for a feature: a positive specification, a negative specification, underspecification, and a zero specification for segments where it is unlicensed.

[12] The program allows a more sweeping modification of the underlying features through the Delete command, which entirely removes a feature from the feature matrix. For the sake of simplification, the Phonomaton enforces certain automatic implications inherent to the standard definition of segments. For example, [+nasal] implies [+sonorant, 0delayed_release], which allows an oral stop to be nasalized with the minimal rule: C → [+nasal].

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press



Figure 5. *The library of sample rules.*

± for a feature in the change means 'change only if necessary to match a phone in the inventory' (or, if there is no specified inventory, in the entire IPA). This specification is useful in cases in which a single phonological process affecting one or more features implies an incidental change in a different feature only in a subset of the target segments. For instance, a process of intervocalic lenition may effect the following set of changes: p → ɸ, t → s, k → x. Such lenition in featural terms is a change to [+continuant, +delayed_release]. However, for the second segment, /t/, there is an additional change from [−strident] to [+strident], a feature that is incompatible with the bilabial and velar places of articulation. In this situation, we could write the rule as in 4a, which produces the derivation in 4b.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

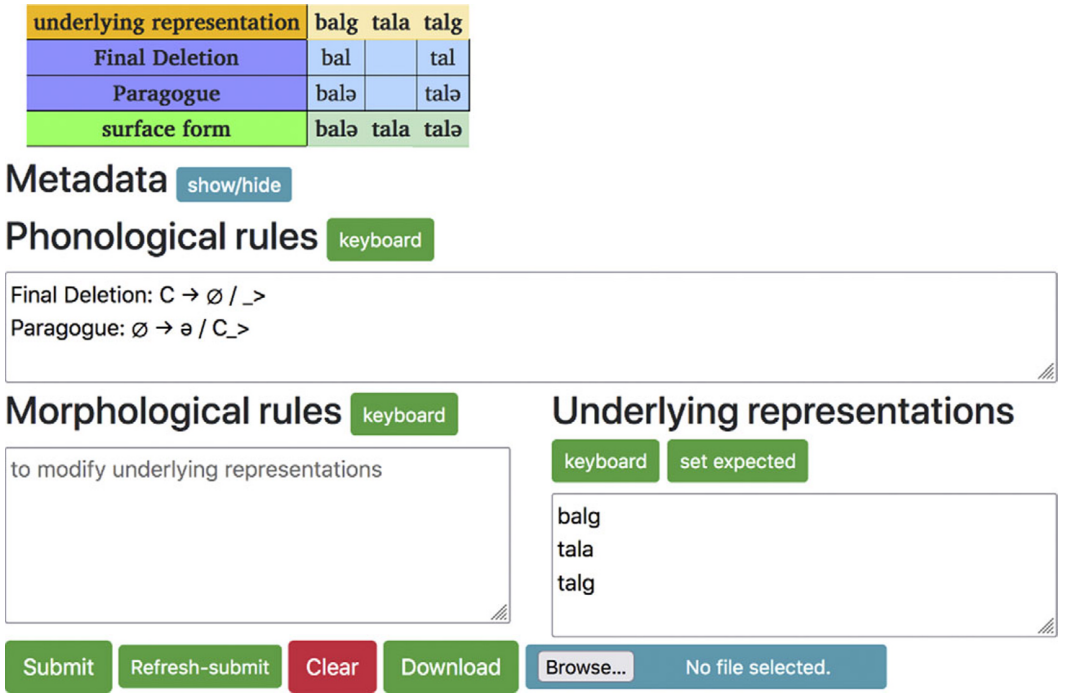


Figure 6. *A sample derivation.*

(4) a. Intervocalic lenition: C → [+continuant,+delayed_release,±strident] / V_V

b. Underlying representation	/apa/	/ata/	/aka/	/aqa/
Intervocalic lenition	aɸa	asa	axa	aχa
Surface form	[aɸa]	[asa]	[axa]	[aχa]

The change to [+strident] is necessary only in the case of /t/, because no segment differs from /t/ only in being [+continuant, +delayed_release] without also being [+strident]. (The other candidate for a continuant here is [θ], but this segment differs in being [+distributed].)

Although ± causes a change only when **necessary**, a feature change can also be defeasible, taking place only when **possible**, given a particular inventory. Defeasible changes are signaled by the symbols ⊕ and ⊖ preceding a feature, which mean 'change to plus or minus, respectively, if the resulting segment is contained in the inventory'.[13] This restriction is the default behavior when a rule makes reference to only a single feature. For instance, the rule in 5a affects only the segments /s/ and /t/ in 5b because the other segments targeted have no counterpart differing only in being [−anterior]. We do not expect the change to create an impossible segment in these other cases; rather, we expect that the rule simply fails to apply.

(5) a. Retroflex assimilation: C → [-anterior] / _[-anterior]

b. Underlying representation	/apʂa/	/aθʂa/	/asʂa/	/atʂa/	/akʂa/	/aqʂa/
Retroflex assimilation			aʂʂa	aʈʂa		
Surface form	[apʂa]	[aθʂa]	[aʂʂa]	[aʈʂa]	[akʂa]	[aqʂa]

[13] Another way of conceiving of these operations is as a combination of disjunctive rules. A rule that contains [voice] as part of the change would first be executed without altering [voice]. Only if this failed would it then attempt the rule with a change in voice. By contrast, [⊕voice] and [⊖voice] first attempt to make the specified change to [voice], and only if this attempt fails is the rule executed without making any change to [voice]. Compare Bale, Papillon, & Reiss 2014 and Reiss 2022 for alternative approaches to defeasibility in derivational phonology.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

However, asserting a defeasible change becomes useful when some partial set of specified changes still takes place even when the full set of changes invoked cannot apply to all of the relevant segments. For instance, a process of lenition turns voiceless stops into voiced fricatives, as shown in 6, but if the inventory language lacks a voiced uvular fricative [ʁ], lenition to a fricative takes place without voicing when applying to uvular stops.

(6) /p/ → [β] /k/ → [ɣ] /q/ → [χ]

Such a process can often be modeled by two rules, in this case, a spirantization rule and a subsequent voicing rule, the second of which would not apply to the uvular because of the gap in the inventory. But such a two-step approach can fail by incorrectly merging segments that should remain distinct. For instance, the language may have underlying /ɸ/ and /x/, which do not voice intervocalically under the same process, yet the underlying spirants would not be differentiated from derived ones after the spirantization rule. In this scenario, the rule in 7 obtains exactly the correct result for 6, effecting the change from stops to fricatives across the board but adding [+voice] only when possible, as signaled by [⊕voice] in the change.

(7) Lenition: C → [+continuant,+delayed_release,⊕voice] / V_V

Another approach to facts such as these makes use of feature deletion. Noncontrastive features can be removed from the calculation entirely using the 'Delete' command, as in 8. Here, surface segments may differ in [voice], but rules cannot refer to this feature, nor are differences in [voice] visible to the grammar.

(8) Inventory: a p β k ɣ q χ
Delete: [voice]
Lenition: C → [+continuant,+delayed_release] / V_V

Although there are both voiced and voiceless segments in the surface inventory in 8, noncontrastive voicing can now be treated as simply being concomitant with [+continuant] or [+delayed release]. Deleting the voice feature can therefore also derive the alternation in 6 above. This case exemplifies one way in which the Phonomaton facilitates comparison between competing solutions.

2.4. Underspecification and feature-filling rules

As discussed in Section 2.1 above, the user can declare underspecified segments as part of the preamble. Now we demonstrate how rules can target underspecified values so that they provide feature values only where none previously exist. First, we introduce an underspecified segment using Define, as shown in 9a, by copying the features of /t/ and deleting the voice specification. In 9b, we formulate a rule that is strictly 'feature filling', as signaled by ⊗ in the target.[14] The polarity ⊗ matches only an underspecified feature value (that is, neither + nor −).

(9) a. Define: T[-syllabic,+consonantal,-sonorant,-continuant,-delayed_release,-approximant,-tap, -trill,-nasal,-spread_gl,-constr_gl,-labial,-round,-labiodental,+coronal,+anterior,-distributed, -strident,-lateral,-dorsal,0high,0low,0front,0back,0tense]
b. Intervocalic voicing: [-syllabic,⊗voice] → [+voice] / V_V
Final devoicing: [-syllabic,⊗voice] → [-voice] / _>

[14] Bale, Papillon, and Reiss (2014:244) entertain such an approach but ultimately do not pursue it. Targeting underspecified features is an approach that has a checkered history in linguistic theory, because underspecification has often been understood to imply invisibility. This question is outside the scope of the present discussion, but we hope that the Phonomaton itself can be used to demonstrate the correctness or incorrectness of such an approach.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

underlying representation	ata	ada	aTa	at	ad	aT
Intervocalic voicing			ada			
Final devoicing						at
surface form	ata	ada	ada	at	ad	at

Figure 7. *Derivation with a feature-filling rule.*

Figure 7 shows a derivation based on 9. Here, intervocalic voicing applies only to /T/, which is underspecified for [voice], and not to its voiceless counterpart /t/. Similarly, final devoicing applies only to /T/, and not to its voiced counterpart /d/. Underspecification in conjunction with ⊗ provides us with an elegant way to model alternations of the type seen in Turkish voicing, discussed in Section 2.1, where loan strata contain the contrastively voiced stops /t/ and /d/, but where voicing is underspecified (and therefore predictable) in /T/.

2.5. Alpha notation and abbreviations

Alpha notation is available through the use of three Greek letters α, γ, and δ, which may also be specified negatively. (We omit β because it is identical to the IPA symbol for the bilabial fricative.) A simple example is given in 10, with a resulting derivation shown in Figure 8. The rule in 10 uses the variable α to assert that a consonant should match the voicing of an immediately following consonant.[15] The result is that the /g/ in /agta/ assimilates in voice to the following /t/ to yield [akta] and the /p/ in /apdu/ assimilates in the same way to yield [abdu]. Alpha notation is thus able to express feature assimilation as a single process, whether it involves a change to a positive or negative specification of a feature.

(10) Voice assimilation: C → [αvoice] / _[-syllabic,αvoice]

We can also use variables to enforce agreement between the target and the environment. For instance, 11 asserts that a vowel should agree with the following vowel in the feature [round] when both vowels match for height features [high] and [low]. Here we explicitly state the labial feature as ± in the target, so the target's [labial] value can change if necessary to satisfy rounding assimilation.

(11) Rounding assimilation: [+syllabic,γhigh] → [αround,±labial] / _[+syllabic,αround,γhigh]

Figure 9 shows a derivation resulting from this rule. Because the derivation is decomposed into subrules containing all of the combinatorial possibilities, the exact condition under which the change takes place is displayed clearly. In the first subrule, all instances of α and γ (in the target, change, and

underlying representation	agta	apdu	agda	aptu
Voice assimilation (α+)		abdu		
Voice assimilation (α-)	akta			
surface form	akta	abdu	agda	aptu

Figure 8. *Voice assimilation.*

[15] The software decomposes each use of a variable into two separate operations: one in which the variable is taken to have a positive value, and one in which it is taken to have a negative value. Regressive assimilation is triggered by a voiceless consonant in the first form but by a voiced stop in the second form; the changes take place in distinct stages.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

underlying representation	bui	biu	bue	beu
Voice assimilation (α+ γ+)		byu		
Voice assimilation (α+ γ-)				
Voice assimilation (α- γ+)	bwi			
Voice assimilation (α- γ-)				
surface form	bui	byu	bue	beu

Figure 9. *Variables in the target and environment.*

environment) are replaced by +; in the second subrule, all instances of α are replaced by +, all instances of γ are replaced by −, and so forth.

The first two forms in Figure 9, /bui/ and /biu/, trigger regressive assimilation because the two adjacent vowels are of similar height. The last two forms, /bue/ and /beu/, however, do not undergo rounding harmony, because the adjacent vowels are of different heights.

2.6. Morphological boundaries

The Phonomaton recognizes several morphological boundaries in the underlying forms, employing the standard symbolic notation of the Leipzig Glossing Rules: - for prefix and suffix boundaries, ~ for reduplicant boundaries, ⟨ ⟩ for infix boundaries, = for clitic boundaries, and white space for word boundaries. Departing from tradition, we provide a boundary symbol ⇒ when an analysis needs to differentiate prefix or proclitic boundaries from suffix or enclitic boundaries. The symbol ≣ is also provided for indicating special morphosyntactic junctures.[16] Phonological rules can reference generic word boundaries by # and left and right word boundaries by < and >, respectively. Therefore, one can write a word-final devoicing rule as in 12.

(12) Final devoicing: [-syllabic] → [-voice] / _>

A rule that raises mid vowels before enclitic boundaries and word-finally can be written as in 13. Disjunction is written with the elements separated by the vertical bar character and enclosed in parentheses (see Section 2.8 below).

(13) Final raising: [+syllabic,-low] → [+high] / _(=|>)

2.7. Segment indexing

Certain processes, like metathesis, require indexing segments in a rule's target and change. The Phonomaton employs superscripts (from ¹ to ⁴) to index segments. The rule in 14 matches a sequence of two consonants in which the first is coronal and the second is labial. The rule defines variables by labeling the relevant segments with a superscript; these indices appear again in the change but without any preceding features or segments. That is, instead of repeating redundant information in the change, as in the common notation $C^1C^2 \rightarrow C^2C^1$ for metathesis, the Phonomaton expects $C^1C^2 \rightarrow {}^{21}$, with the redundancy removed (but see 18 below for an example where features modify an indexed segment).

[16] For one use of the special prefix/proclitic boundary, see the implementation of the Sanskrit *nati* rule in the Phonomaton's sample derivations. For an example of the ≣ boundary, see the implementation of Japanese compound accent, where ≣ represents the morphosyntactic juncture between the elements of a compound. None of the -, =, ≣, or ⇒ symbols bears any phonological features, and each can be deployed in whatever way the user finds convenient. As much as possible, we have restricted symbols, including the various types of brackets, to single functions in order to provide a consistent interpretation in the programming environment.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

underlying representation	apta	atpa	akpa	apka
metathesis		apta		
surface form	apta	apta	akpa	apka

Figure 10. *A derivation showing metathesis of coronal and labial stops.*

(14) metathesis: [-syllabic,+coronal]^{1}[-syllabic,+labial]^{2} → ^{21}

The result of 14 is to swap a coronal consonant with a following labial consonant, as shown in Figure 10.

Indexing also assists in rules for total assimilation, reduplication, gemination, and degemination. The Phonomaton requires that indices be associated with a segment or class of segments before they are used as referents. For example, the rule of total assimilation in 15a associates the superscripted index 1 with a segment that matches [+consonantal] in the target and then refers back to that segment in the change with the plain superscript numeral. This would have the effect of changing /nk/ to [kk], /mb/ to [bb], and so forth. A formulation of the same rule as in 15b would fail because it invalidly employs an index in the change before the environment where it is defined.

(15) Total assimilation of a nasal to a following consonant
  a. ✓ [+nasal][+consonantal]^{1} → ^{11}
  b. ✗ [+nasal] → ^{1} / _[+consonantal]^{1}

Similarly, a rule of vowel copying, often written in a form like 16b, runs afoul of the requirement that the target define each index. Such rules can easily be reformulated as in 16a, where the entire environment is targeted and copied with a modification. Despite being common practice, it is redundant, and thus disallowed, to repeat the Cs and Vs in the change.

(16) Vowel copying
  a. ✓ V^{1}C^{2}C^{3} → ^{1213} / _#
  b. ✗ ∅ → ^{1} / V^{1}C_C#

One can write complex rules of reduplication (a morphological operation) using indices in conjunction with simple regular expressions. The rule in 17 reduplicates a word-initial onset followed by the vowel /a/, an example of reduplication with 'fixed segmentalism'. Figure 11 shows the resulting derivation for three underlying forms.[17]

(17) Reduplication: C^{1}V^{2} → ^{1}a^{12} / <_

underlying representation	binsu	tuka	koko
Reduplication	babinsu	tatuka	kakoko
surface form	babinsu	tatuka	kakoko

Figure 11. *A derivation showing reduplication with fixed segmentalism.*

[17] Clearly, indexing individual segments has the power to vastly overgenerate patterns of reduplication. However, once an underlying form has been syllabified via rules like those described in Section 2.9, reduplication can also be modeled as a more constrained prosodic operation, in which indexing refers to units delimited by syllable boundaries and possibly moras rather than segments, although this is beyond the scope of the current article. In general, overgeneration has not been a major concern in the design of the program; the program allows users to generate patterns in various ways and thus formulate precise arguments for whether certain types of operations are necessary or egregiously powerful.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

underlying representation	gibo	gubo	gɨbo	gabo
Reduplication	gɛgibo	gɔgubo	gəgɨbo	gəgabo
surface form	gɛgibo	gɔgubo	gəgɨbo	gəgabo

Figure 12. *Reduplication with partial neutralization.*

Indexed segments can also undergo featural modifications. Any features specified before an index in the change component of a rule apply to the indexed segment in the output. In 18, the features [−high, −low, −tense] apply to the first instance of a segment indexed by ², that is, the first vowel in the stem. Figure 12 shows sample derivations.

(18) Reduplication: $C^{1}V^{2} \rightarrow {}^{1}[\text{-high,-low,-tense}]^{212}$ / <_

2.8. Regular expression syntax

The Phonomaton allows regular expressions in the target and environment of a rule, the most important component of which are quantifiers. In 19 we present a comparison between the regular expressions for quantifiers used by the Phonomaton (based on Perl's regex notation) and the familiar SPE notation (Chomsky & Halle 1968).

(19)

Quantifier		Example	SPE
*	zero or more	C*	C_0
?	zero or one	C?	C_0^1 or (C)
+	one or more	C+	C_1
{x,y}	range min to max	C{2,4}	C_2^4 (from 2 to 4)

Thus, a rule of vowel lowering that applies at the end of the word can be specified to occur regardless of the presence of a word-final coda, as in 20a, with either a simple word-final coda or no coda at all, as in 20b, with a simple or complex coda, as in 20c, or with a complex coda of two to four consonants, as in the unlikely case of 20d.

(20) a. V → [-high] / _C*>
b. V → [-high] / _C?>
c. V → [-high] / _C+>
d. V → [-high] / _C{2,4}>

These quantifiers can combine with indices to handle complex cases of reduplication. In 21a, we see word-initial CV-reduplication. In 21b, we see reduplication of a word-initial unit containing two vowels and their preceding consonants (a disyllable, in the common case). The index in the target captures the entire expression within the parenthesized group, which is then duplicated in the change. The environment ensures that this process is anchored to the left edge of the word.[18]

[18] Cases where, for instance, the reduplicant diverges from the base, as in Tagalog *pa~plato*, can be modeled by brute force with a rule such as $C^{1}(C^{*})^{2}V^{3} \rightarrow {}^{13123}$ / <_, but are more insighfully handled by targeting the reduplicant for a particular phonological process that does not apply to the base. For instance, the (unreduced) reduplication rule in (i) introduces a special boundary symbol (~), which is then referred to by a second rule making a phonotactic simplification.

(i) Reduplication: $(C^{*})^{1}V^{2} \rightarrow {}^{12\sim12}$ / <_
(ii) Deletion: C* → ∅ / <C_X*~

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

(21) a. $(C^*V)^1 \rightarrow {}^{11} / <_$
b. $(C^*VC^*V)^1 \rightarrow {}^{11} / <_$

Regular expressions also work in conjunction with various feature specifications and underspecification to model locality and blocking effects. Consider the case of retroflex harmony, where /ʃ/ assimilates to the [−distributed] feature of a retroflex when the retroflex precedes it. We can model this process as in 22, which says that a [−anterior] consonant becomes [−distributed] (retroflex) when preceded anywhere in the word by a [−distributed, −anterior] segment. The symbol X stands for any segment and X* stands for zero or more segments, so the process operates long-distance.

(22) Retroflex assimilation:
[-syllabic,-anterior] → [-distributed] / [-distributed,-anterior]X*_

We can also model blocking effects for rules such as these. If we modify the above environment as in 23, the rule will now apply only when the segments intervening between the target and the preceding trigger are [−coronal].

(23) … / [-distributed,-anterior][-coronal]*_

Figure 13 shows a derivation based on this rule; the dorsal consonant /k/ in the first form is transparent for retroflex harmony, but the intervening /s/ in the second form, being coronal, is opaque and thus blocks assimilation.

Another relevant feature of regular expressions is grouping and alternation (signaled by brace notation in SPE). If a rule must make reference to segments that cannot be generalized by features, it can express alternation by the vertical bar symbol. For instance, a change that takes place only before the phonemes /s d ɣ/ at the end of a word is expressed as in 24.

(24) a → ə / _(s|d|ɣ)>

2.9. Types of rule application and a brief look at syllabification

The Phonomaton allows for four flavors of rule application. The default is for a rule to apply wherever it can, scanning an underlying or intermediate representation from left to right. But rules can also apply iteratively, in which case the output of the rule can feed another application of the same rule. The default application of a nasalization rule is shown in 26a, while its iterative counterpart is shown in 25b.

(25) a. V → [+nasal] / _C*[+nasal]
b. V → [+nasal] / _C*[+nasal] // iterate

The rule in 25a would apply to a representation like /balabon/ to derive [balabõn], whereas that in 25b would derive [bãlãbõn]. The iterative rule in 25b takes the output of its application as its input until no more changes can be made, at which point the derivation proceeds to the next rule.

Other cases require a rule to apply only to its maximal environment, as per the Pāṇini principle (also known as the 'subset principle' or 'elsewhere condition'). Among other cases, this is crucial for syllabifying strings algorithmically, by rules that insert onset and coda boundaries, as exemplified in 26.

underlying representation	ʂakaʃa	ʂasaʃa
Retroflex assimilation	ʂakaʂa	
surface form	ʂakaʂa	ʂasaʃa

Figure 13. *Retroflex harmony blocking.*

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

(26) Onset: ∅ → ⟨ / _(([-sonorant]?[+approximant,-syllabic]?)|C)V // exclusive
Coda: ∅ → ⟨ / VC?_ // exclusive

The onset rule in 26 states: insert an onset boundary (⟨) preceding a vowel, optionally preceded by (i) an approximant (i.e. trills, taps, laterals, glides), in turn optionally preceded by a nonsonorant, or (ii) a singleton consonant of any type. The embedded parenthesis group captures certain types of CC clusters, such as the ones listed below option I in 27, while excluding others (e.g. ml, rl, nt) that do not fit this template.

(27)

Option I		**Option II**
(([-sonorant]?[+approximant,-syllabic]?)		| C)
t	r	t
p	j	r
k	l	n
s	w	m

Because each part of the embedded group is optional, this part of the pattern could also capture singleton onsets that fit the given feature specifications. However, we would still need option II for a C of any type if there are singleton onsets that do not participate in clusters. For instance, our onset rule in 26 describes a language that allows nasal onsets but does not allow nasals in either component of a CC cluster, because a [+sonorant, −approximant] segment such as *m*, *n*, *ŋ* is not captured by either component of option I.

The coda rule in 26 is simpler and allows for a singleton coda consonant of any type. Crucially, the specification 'exclusive' at the end of both rules ensures that the boundaries will be inserted only once at the edge of the maximal string containing the optional segments rather than at every possible point. Otherwise, a UR such as /blabla/ would result in ⟨b⟨l⟨a⟩⟨b⟨l⟨a⟩ instead of the correct ⟨bla⟩⟨bla⟩.[19]

2.10. Morphological rules

Exercises in phonology courses commonly present a paradigm, which the student must analyze into a set of roots and affixes or other morphophonological processes. One example of such an exercise, based on Serbo-Croatian adjectives, is shown in Table 1 (Kenstowicz & Kisseberth 1979:74).

Coopting for simplicity's sake the [tone_high] feature to indicate stress (as the Phonomaton does not treat stress as a segmental feature), we can derive the forms in Table 1 with the phonological rules in 28a and the morphological rules in 28b.

(28) a. Phonological rules
Stress assignment: V → [+tone_high] / _C*>
Epenthesis: ∅ → a / C_C>
L-vocalization: l → o / _>
b. Morphological rules
FEM: 0 → a / _>
NEUT: 0 → o / _>
PL: 0 → i / _>

[19] In rarer cases, it is necessary to restrict the application of a rule even more strictly so that it applies only once per input. This can be done with the specification 'once' following the rule. Not discussed here is the question of directionality. Environments are scanned from left to right in the Phonomaton, but there is a workaround if the reverse is required. If the rule environment begins with <X* (left word boundary followed by any number of segments), the quantifier will first expand the pattern as far as possible before advancing to the next element in the environment, and thus yield what is essentially right-to-left scanning.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

MASC	FEM	NEUT	PL	Gloss
mlád	mladá	mladó	mladí	‘young’
zelén	zelená	zelenó	zelení	‘green’
púst	pustá	pustó	pustí	‘empty’
lédan	ledná	lednó	lední	‘frozen’
dóbar	dobrá	dobró	dobrí	‘good’
jásan	jasná	jasnó	jasní	‘clear’
debéo	debelá	debeló	debelí	‘fat’
béo	belá	beló	belí	‘white’
mío	milá	miló	milí	‘dear’
nágao	naglá	nagló	naglí	‘abrupt’
óbao	oblá	obló	oblí	‘round’
pódao	podlá	podló	podlí	‘base’

Table 1. *Serbo-Croatian adjective paradigm.*

underlying representation	ledn	ledna	ledno	ledni	debel	debela	debelo	debeli	nagl	nagla	naglo	nagli
Stress Assignment	lédn	ledná	lednó	lední	debél	debelá	debeló	debelí	nágl	naglá	nagló	naglí
Epenthesis	lédan								nágal			
L-vocalization					debéo				nágao			
surface form	lédan	ledná	lednó	lední	debéo	debelá	debeló	debelí	nágao	naglá	nagló	naglí

Figure 14. *Serbo-Croatian derivation with morphological rules.*

The Phonomaton applies each morphological rule to each underlying representation to create an input form, as shown in Figure 14 for the underlying representations /ledn/, /debel/, and /nagl/. We see the bare underlying representation, as well, preceding the forms that have undergone morphological rules.[20]

The Phonomaton accepts morphological rules in the same format as phonological rules. For instance, a rule introducing a simple affix converts a null to the phonological form of the affix in a word-edge environment, namely <_ for prefixes and _> for suffixes. This approach also facilitates modeling nonconcatenative processes such as lenition, ablaut, and infixation, as any process that can be written as a phonological rule can be treated as a morphological rule.

With this, we conclude our review of the program’s features and proceed to how it has been implemented in the classroom.

3. The Phonomaton as a teaching tool

3.1. Pedagogically oriented features

Crucial to its pedagogical function is the Phonomaton’s ability to save and upload analyses. Saving an analysis downloads the data found in each component (underlying representation, phonological rules, and morphological rules) as a single, human-readable text file, with all of the contents of the analysis appearing precisely as in the web interface. Analyses are downloaded and uploaded with a single click without any need to log in or register. Students can use these downloaded analyses as submissions to class exercises for the instructor to check.

[20] This facility is strictly for producing paradigms, as shown; each morphological rule derives a single column in the table. More complex derivations based on abstract morphological input are also possible but are not discussed here.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

An advantage to treating phonological derivations as code is that an analysis can be interspersed with comments that are not executed by the program, following Knuth's (1992) popular concept of 'literate programming', whereby the programmer writes code as an expository text for a human audience interspersed with executable lines directed at the software. The program explains itself as it goes, preceding the expository comments with a character that tells the program not to execute that particular line. This approach has become routine in reproducible research and can be easily applied to executable linguistic analysis of the type presented here. The following brief example demonstrates the idea with the complex vowel-length alternations of Chimwiini (Kisseberth & Abasheik 1974). The comments, not executed by the program, are preceded by %.

```
% Epenthesize a vowel to stems ending in a consonant:
Final vowel attachment: ∅ → a / C_>
% Lengthen all word-final vowels:
Word final lengthening: V → [+long] / _>
% Shorten all vowels that precede the third syllable from the end. This is implemented here by referring
to vowels rather than syllables. The following rule's environment looks for three vowels in the forward
context that can be interrupted by any number of segments or boundary symbols (word boundaries, in
this case):
Preantepenultimate shortening: V → [-long] / _.*V.*V.*V
% Shorten a vowel at the end of a phrase, represented by the end of the line $:
Phrase final shortening: V → [-long] / _$
% Shorten a vowel that precedes a long vowel. The environment specifies that any number of
consonants (C*) can intervene between the long vowel trigger and the target:
Pre-long shortening: V → [-long] / _C*[+syllabic, +long]
```

When the code above is executed it yields the derivation shown in Figure 15, with the given underlying representations.

Another pedagogically relevant feature of note in Figure 15 is the final row in the table headed by 'expected'. In the field where underlying representations are entered, the user can optionally add the expected surface form following the representation separated by a double backslash (e.g. kama mp^{h}aka // kama: mp^{h}aka). This inclusion has the effect of displaying the expected form at the bottom of each column and checking it against the actual derived form.[21] If the two match, the expected form is displayed in green, as seen in Figure 15. If they do not match, the expected form is highlighted in red. This feature is also convenient for creating problem sets, which an instructor can present as a blank Phonomaton analysis with expected surface forms; the students' task is then to provide the underlying representations and rules.

underlying representation	kure:beɭan	xte:ka	xteka	bo:zeɭ	kuna kahawa	kama mp^{h}aka
Final vowel attachment	kure:beɭana			bo:zeɭa		
Word final lengthening	kure:beɭana:	xte:ka:	xteka:	bo:zeɭa:	kuna: kahawa:	kama: mp^{h}aka:
Preantepenultimate shortening	kurebeɭana:				kuna kahawa:	
Phrase final shortening	kurebeɭana	xte:ka	xteka	bo:zeɭa	kuna kahawa	kama: mp^{h}aka
Pre-long shortening						
surface form	**kurebeɭana**	**xte:ka**	**xteka**	**bo:zeɭa**	**kuna kahawa**	**kama: mp^{h}aka**
expected	kurebeɭana	xte:ka	xteka	bo:zeɭa	kuna kahawa	kama: mp^{h}aka

Figure 15. *Chimwiini vowel alternations.*

[21] When used in conjunction with morphological rules, as described in Section 2.10, the various forms expected by each morphological rule are separated by a single slash following the double slash after the UR. For instance, the UR and expected output for the first set in the Serbo-Croatian example in Figure 14 would be given as: ledn // lédan / ledná / lednó / lední.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

3.2. Implementation

We have received constructive input from two introductory phonology courses where an earlier version of the Phonomaton was employed. We report here from a survey of nineteen students who used the program in the Spring 2024 semester at Queens College (City University of New York), where the first author was the instructor and the third author was the teaching assistant. In addition to the positive comments, we highlight here some of the critical ones as well, since we believe these uncover important issues in the use of technology in the classroom more broadly.

For context, Queens College is part of the City University of New York, a public university with a mission of making higher education accessible to all.[22] The campus is among the most diverse in the country, and over a third of the students are the first generation in their families to attend college. Within the undergraduate linguistics program, there is also a relatively wide range in the students' levels of preparation. Some students could compete handily with their peers in the most prestigious departments, while others are not yet comfortable with the technical aspects of linguistic analysis. Optimally, the Phonomaton would lift all boats, aiding both advanced students and those who did not yet have a strong grasp of the fundamentals.

In the semester-long trial, the students submitted all of their work as Phonomaton analyses, so they already knew whether their solutions were formatted correctly and generated the desired results. The instructor's role in evaluating assignments could then focus on questions of economy, insight, and naturalness, which are higher-level matters often occluded by the effort to get the mechanics right. The instructor also made frequent use of the Phonomaton during lectures in order to show the various effects of rules and their interactions. Overall, the majority of the students reported finding the Phonomaton helpful, as seen in Figure 16, although a considerable number of students had mixed feelings, and four students responded negatively.

When asked about their favorite aspect of the program, eight students responded with reference to finding and calculating features, six students chose the Phonomaton's ability to check their work, and another six students chose the ability to calculate and display derivations. Several remarks in response to the question, 'Did you find the Phonomaton program helpful in learning how phonological rules and derivations work? Why or why not? What were the best and worst parts of it?', made reference to verifying the output of rules and their interactions:

> I found the Phonomaton helpful and the best part of the Phonomaton for me was seeing each rule in action. I could see exactly how each rule changed the underlying representation.
>
> Seeing exactly how my rules affect the underlying forms helped me think of them algorithmically.
>
> I really enjoyed how it showed how the UF [underlying forms] were changed step by step by the different rules. Having a program perform the rules for you was much more efficient, especially across large data sets. It was very salient to see where rules did and did not work to create a correct SF [surface form], and that was very helpful.
>
> I really liked seeing you use it in class to demonstrate rule ordering. I found that very helpful to understand how underlying forms can change and how rules can feed/counterfeed/bleed other rules.

Other comments noted how the program helped them understand feature systems, which involve a steep learning curve, as students have to unlearn descriptive categories, such as stop, fricative, and affricate, in order to learn more abstract features that do not always have easily discernible physical or acoustic correlates (such as [distributed] and [dorsal]):

[22] In the relevant period, enrollment was just under 14,000 students with an acceptance rate of approximately 70%. In the same time period, the student body was reported to be 74% minority, and 51% of the students received Pell grants.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

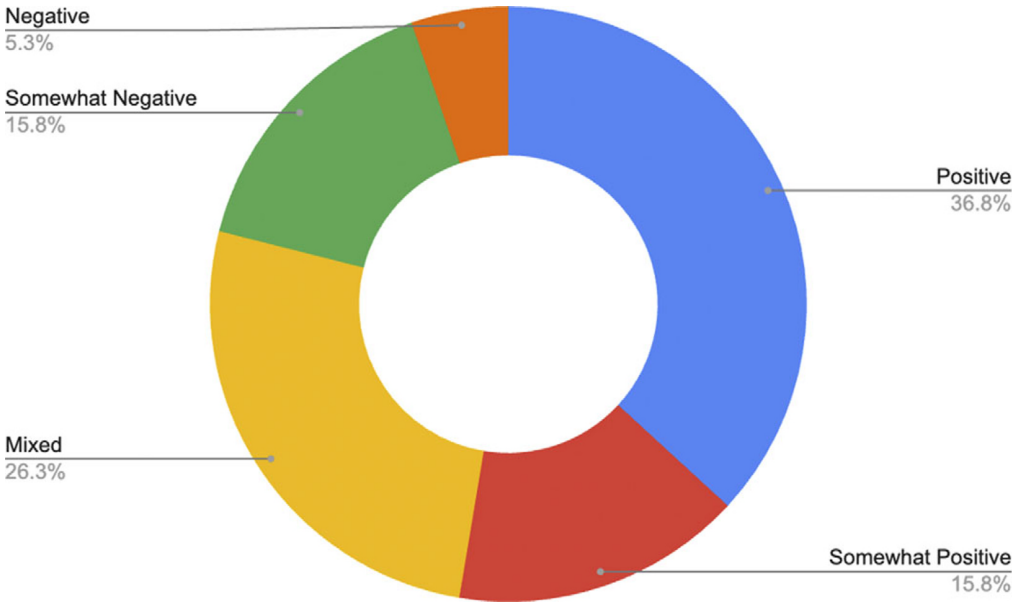


Figure 16. *Responses to 'Was the Phonomaton helpful?'.*

> My favorite part of the phonomaton was showing how IPA sound[s] share certain features, as well as what features change as you go from one sound to another. It really helped me understand the concept of features to begin with, and how much languages tend to care more about features than individual phonemes.

As for the worst parts of the program, students largely referred to its sensitivity to small errors and the lack of helpful error messages. In the most critical remark to this effect, the student stated that the program actually detracted from their learning because they were so concerned about getting the program to work as they expected. We return to these points below.

During the course, early exercises on features and on the mechanics of rule writing were assigned in two stages. In the first stage, the assignment was to be done by hand and submitted in the normal manner. In the second stage, the students had to complete the same problem using the Phonomaton, from which they downloaded and submitted their analysis. The exercises were not discussed in class until after the second stage. In the exercise on applying rules, students were given a rule (e.g. ə → ∅ / VC _ CV) and asked to apply it to a number of underlying forms. In the exercise on writing rules, they were given three underlying forms with their corresponding surface forms and asked to devise a rule that accounted for the alternations. The before and after assessments for both assignments are shown in Figure 17, where the vertical axis represents the scores out of 100.

The scores improved significantly, especially in relation to rule writing, when the students had the opportunity to manipulate and check their analyses using the Phonomaton. Whereas many students struggled with consistent formalization of rules in the first stage, through experimenting with different possibilities, they were able to self-correct to a large extent during the second stage.

A different exercise, adapted from Hayes 2009:101, presented the students with statements like that in 29a, which they were expected to translate into feature-based rules like that in 29b, given a particular inventory of vowels and consonants.

(29) a. [i y ɯ u] become [e ø ɤ o] before [q ɴ].
b. Vowel lowering: [+syllabic,+high] → [-high] / _[-continuant,+dorsal,-high]

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

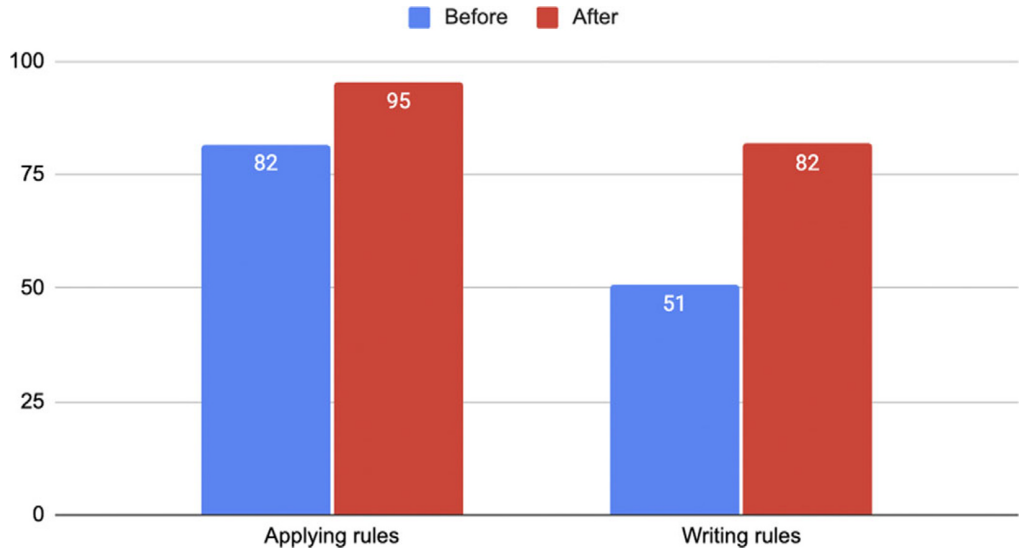


Figure 17. *Assessment scores before and after using the Phonomaton.*

The Phonomaton led to a 30% improvement in the second attempt over the first one, where students had the aid of only a static feature chart. In this case, however, it can be argued that the problem becomes trivial with the help of the Phonomaton, as the program calculates natural classes and featural differences within any subset of segments. The user must only enter the relevant inventory and check each pair to ensure that a particular change in features accounts for the entire alternation. To this point, an anonymous referee asks, 'What happens if the student needs Phonomaton to identify a natural class? Do they learn this concept if the program does it for them?'. Overall improvements in classroom-based assessments, where the students did not have access to the program, suggest that the program helped them overcome the considerable initial difficulties that the stage-one scores revealed. However, our evaluation was not comprehensive enough to compare classroom-based assessments on exactly the same type of material before and after the students worked with the Phonomaton. Our future evaluations of the program will attempt to pinpoint how the Phonomaton aids progress in independent work, as opposed to overall improvements. We note, though, that the program does not aim to explain phonology any more than a calculator is meant to explain concepts in math.[23] For a well-motivated student who independently explores the various possibilities of rule formulation and interaction, we believe the program could serve a broader function, but the Phonomaton is not designed to replace instruction.[24] One student noted this explicitly in the survey:

[23] Research in math education has shown that the appropriate use of calculators actually bolsters skills in unaided calculation (Hembree & Dessart 1992, inter alia) but that the tool itself is insufficient to yield substantive change without being deployed thoughtfully (Ruthven 1996).

[24] An anonymous referee asks whether mastering the various formalisms of rule writing in the Phonomaton actually facilitates learning of concepts. We would argue that adopting any formalism in and of itself only encourages precision and consistency. It is, rather, the process of independent exploration and tinkering with derivations that allows for a far more thorough appreciation of the concepts, more thorough, in all likelihood, than what can be expected from absorbing the concept through a lecture or readings. The central role of the instructor in this case becomes motivating students to explore and tinker. The approach we are currently developing in this regard involves a set of open-ended exercises without fixed solutions. For instance, students may be tasked with constructing their own cases of bleeding, feeding, counterfeeding, counterbleeding, and so forth or with devising a feature geometry that could handle various types of assimilations of their making. We leave a discussion of these methods to further work.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

> I don't know if I would say it was helpful for learning HOW rules and derivations work, it was helpful moreso for checking rules and derivations and also troubleshooting rules that were slightly wrong. I feel like the logic behind the rules had to be learned without the Phonomaton.

The two biggest challenges encountered in applying the software pedagogically were (i) the lack of guidance on malformed rules, and (ii) an unevenness in how the tool was taken up by students at different levels of proficiency. We believe these two challenges are linked. The most common complaint in the evaluations of the program had to do with the unhelpful error messages that malformed rules would trigger. Perhaps in our eagerness for the Phonomaton to handle as much phonological theory as possible, including autosegmental phonology and optimality theory (not discussed here for reasons of space), we skimped on what turned out to be the most important element for novices: detailed guidance on the mechanics of rule writing. Based on the evaluations, we have remedied this problem to a large extent. The program now pinpoints which rule and what part of that rule contains an error, calling out specific problems such as a missing underscore, a missing arrow, a missing bracket, or invoking a feature that doesn't exist. Most likely as a result of this lacuna in the original trial semester, we found a strong correlation between the students' overall grades in the course and their assessment of the Phonomaton. Those who achieved higher scores in assessments and in the course as a whole consistently reported finding the Phonomaton more useful. We address this problem with urgency because, if left untreated, it could yield precisely the opposite effect of what we had originally set out to do, namely, to lift all boats, and instead simply exacerbate the gap between those who are already comfortable with mathematical formalism and algorithmic thinking upon entering the course and those who are not. Now that the program offers friendly guidance in rule formulation and allows for a bit more flexibility, we expect a more even response across students at varying levels of proficiency. In particular, we aim for this aspect of the program to have a broader impact in opening the doors to computational thinking for those students who may not take naturally to rule syntax and mathematical formalization.[25]

There are several facilities that have not yet been integrated into the Phonomaton but that would be useful to students in optimizing a working analysis. The program could check not only whether a derivation achieves its desired target, as it does now, but also whether there are redundancies in the rules (e.g. reference to features that do not need to be invoked, or overspecified rule environments). The program could also gauge the complexity of a given analysis and assign a numeric score based on the number of rules and specifications for each rule. Although formal economy metrics of this type have never been widely adopted in theoretical phonology, they could serve an important pedagogical function. We continue to explore these and other potential extensions of the program.

4. Conclusion

Bird (2005:45) outlines some of the challenges in producing a 'phonologist's workbench' program:

> A phonologist's workbench should help people to 'debug' their analyses and spot errors before going to press with an analysis. Developing such tools is much more difficult than it might appear. First, there is no agreed method for modelling non-linear representations, and each proposal has shortcomings. Second, processing data sets presents its own set of problems, having to do with tokenization, symbols which are ambiguous as to their featural decomposition, symbols marked as uncertain or optional, and so on. Third, some innocuous looking rules and constraints may be surprisingly difficult to model, and it might only be possible to approximate the desired behavior. Additionally, certain universal principles and tendencies may be hard to express in a formal manner.

[25] In response to an anonymous referee's query regarding how much time it takes to learn the interface, we emphasize that the rule syntax is only a minor elaboration of the classic SPE formalism. There is very little special syntax that one has to learn to operate the program. As can be seen in the various examples here, the Phonomaton 'code' that a user produces does not look very different from the familiar derivational analyses that can be found in any introductory textbook.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

Similarly, Piwowarczyk (2022:40) notes in his review of computational models of historical sound change that:

> … very few have actually ventured to take historical sound change rules from textbooks of well studied languages and develop a working computer model. And anyone who HAS ventured into this territory has quickly realized that there is a world of difference between the rules as they are written in standard linguistic notation and as they need to be written in computer models.

The Phonomaton has taken up this challenge, and we can attest to all of the difficulties mentioned above. However, the value of a user-friendly phonological rule engine is well worth the trouble. Such a program can help students grasp the more daunting formal aspects of phonology by encouraging experimentation. Furthermore, the tool can serve as a bridge to precise thinking about language and an interactive proof that a theory produces the expected results, in contrast to the typically underformalized presentations found in textbooks. On a higher level, a program such as the one presented here leads the way to reconfiguring linguistics pedagogy as 'contructionism', in the sense of Papert (1990), in which students build concrete versions of abstract concepts and have the pleasure of seeing their creations produce results with novel input. In this scenario, students adopt precise formulations as they learn to communicate with the program, not simply because the instructor demands it. In other words, it is constructionism rather than instructionism that ultimately guides the students.

The rigor enforced by computability also pays dividends to professional linguists. The reader may have noticed that, despite our opening with a jeremiad on the fragmentation of linguistic theory, many of the Phonomaton's capabilities presented here are novel (e.g. the use of ±, ⊕, ⊖, ⊗, feature deletion, and the use of phonetic inventories, discussed in Section 2.2). To a large extent, these innovations were brought about by our failure to implement solutions to classic problems as typically presented in textbooks. In this case, the standard of computability has suggested new ways of interacting with features and underspecification. The ethos of the project, however, aims to provide multiple strategies to solve problems, in the spirit of exploring advantages and drawbacks to different approaches. We now have a formal framework from which we can argue for or against such innovations.

In closing, we point out some further directions for the Phonomaton and their potential impacts. The current norm in teaching any introductory theoretical linguistics course, including Phonology, involves a rapidly moving semester-long carousel of concepts exemplified by disembodied snippets from a wide variety of languages. As theories have developed in their complexity, there has been a waning interest in producing works on the scale of the *Sound pattern of English* (Chomsky & Halle 1968), the *Sound pattern of Russian* (Halle 1971 [1959]), or *Spanish phonology* (Harris 1969), among many others of that era, which attempt to tackle entire phonological grammars. We hope that the Phonomaton will rekindle an appetite for holistic projects of this nature by enforcing consistency and opening the analysis to public inspection and testing. The possibility of collaboration on a large-scale phonological grammar is even more tantalizing.[26]

We see a certain urgency to the program presented here. In the foundational stories of the generative revolution, Chomsky's algorithmic approach was built atop the ruins of structuralism and behaviorism. Ironically, seventy years after behaviorism had been all but vanquished in mainstream linguistics, it has made its return through the back door of natural language processing, to devastating effect. In particular, ChatGPT, the first publicly available language model to convincingly pass the Turing test, appears to make no use whatsoever of the facts, solutions, and general wisdom accumulated over nearly a century of generative explorations of the human language faculty.[27] Mainstream linguistics, having staked so much

[26] Although not discussed here, we also aim for interaction with the burgeoning field of computational phonology proper (see Bird 2005, Daland 2014, and Chandlee & Heinz 2017 for recent summaries). This will naturally require laying bare the inner workings of the program for a comparison with current (mostly finite-state) models. As this is of little relevance to our current focus on pedagogy at introductory levels, we leave it to further work.

[27] For what the Turing test is worth in the current context, see Bender & Koller 2020. Regardless of the deeper questions and ChatGPT's use of pure probabilities in place of meaningful structures, it is an engineering feat on par with the defeat of the world's

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

in abstract structure and symbolic manipulation, now appears to be in a pitched battle with a type of 'deep learning' that is largely opaque and illegible to human observers. So complete is the circle that Chomsky et al. (2023) rehearse many of the same arguments against 'deep learning' language models that Chomsky (1959) famously deployed against Skinner over sixty years earlier. Surely it will not be a phonological rule engine that saves algorithmic approaches from the depredations of 'deep learning', if that indeed materializes as an existential threat. We hope, however, that it can serve as a small bridge between computer science and the cognitive approaches to language that have come to define the field from Pāṇini to the present day. Most immediately, however, it is our aim to make these approaches more accessible and easier to learn for new generations of students and to help them pivot to broader applications by framing phonology as coding.

Data availability statement. All linguistic data herein has been previously published in the cited sources. The Phonomaton program is freely available for use at www.kratylos/ph. For access to the code, please contact the authors.

Acknowledgments. We thank the Computer Science Department of the University of Kentucky at Lexington for providing computing resources as well as the audiences at the University of Kentucky, CUNY Graduate Center, Cornell, and elsewhere for their feedback. We especially thank Kie Zuraw, Christian DiCanio, the section editors and anonymous referees, whose extensive comments led to significant revisions and improvements. All remaining errors are ours alone. [Full editorial history: Received 18 November 2022; revision invited 04 December 2022; revision received 14 July 2023; revision invited 21 December 2023; revision received 20 January 2025; accepted pending revisions 30 September 2025; revision received 12 October 2025; accepted 15 December 2025.]

Competing interests. The authors declare that they have no competing interests.

Ethics statement. The authors declare that this manuscript is original research, has not been published previously, and is not currently under consideration for publication elsewhere. All authors have read and approved the final manuscript and have no conflicts of interest to disclose. This article does not contain any studies with human or animal participants performed by any of the authors. Anonymized reviews of the software reported here were provided on an entirely voluntary basis by students at the first author's institution.

References

Albro, Daniel. 1999. PhonologyPad [Computer program]. Los Angeles: Department of Linguistics, University of California, Los Angeles. https://brucehayes.org/120a/PhonologyPad.htm.

Anderson, Stephen R. 2021. *Phonology in the twentieth century*. 2nd edn, revised and expanded. Berlin: Language Science. https://doi.org/10.5281/zenodo.5509618.

Bale, Alan; Maxime Papillon; and Charles Reiss. 2014. Targeting underspecified segments: A formal analysis of feature-changing and feature-filling rules. *Lingua* 148.240–53. https://doi.org/10.1016/j.lingua.2014.05.015.

Bale, Alan, and Charles Reiss. 2018. *Phonology: A formal introduction*. Cambridge, MA: MIT Press.

Barker-Plummer, David; John Barwise; and John Etchemendy. 1999. *Language, proof and logic*. Stanford, CA: CSLI Publications.

Bender, Emily M., and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, 5185–98. https://doi.org/10.18653/v1/2020.acl-main.463.

Bird, Steven. 2005. Phonology. *The Oxford handbook of computational linguistics*, ed. by Ruslan Mitkov, 3–24. Oxford: Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199276349.013.0001.

Blust, Robert; Stephen Trussel; and Alexander D. Smith. 2023. CLDF dataset derived from Blust's 'Austronesian Comparative Dictionary' (v1.2). https://doi.org/10.5281/zenodo.7741197.

Champollion, Lucas; Joshua Tauberer; and Maribel Romero. 2007. The Penn lambda calculator: Pedagogical software for natural language semantics. *Proceedings of the Grammar Engineering across Frameworks (GEAF) 2007 Workshop*, 106–27. https://web.stanford.edu/group/cslipublications/cslipublications/GEAF/2007/geaf07.pdf.

Chandlee, Jane, and Jeffrey Heinz. 2017. Computational phonology. *Oxford research encyclopedia of linguistics*. Oxford: Oxford University Press. https://doi.org/10.1093/acrefore/9780199384655.013.116.

Charity Hudley, Anne H.; Christine Mallinson; and Mary Bucholtz. 2020. Toward racial justice in linguistics: Interdisciplinary insights into theorizing race in the discipline and diversifying the profession. *Language* 96.e200–e235. https://doi.org/10.1353/lan.2020.0074.

Chomsky, Noam. 1959. Review of *Verbal behavior*, by B. F. Skinner. *Language* 35.26–58. https://doi.org/10.2307/411334.

top chess and Go players that is sure to attract even greater attention and funding in the coming years. Even Chomsky, Roberts, and Watumull (2023) refer to ChatGPT and its ilk as 'marvels of machine learning'.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press

Chomsky, Noam, and Morris Halle. 1968. *The sound pattern of English*. New York: Harper and Row.
Chomsky, Noam; Ian Roberts; and Jeffrey Watumull. 2023. The false promise of ChatGPT. *New York Times*, March 8, 2023. https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html.
Daland, Robert. 2014. What is computational phonology? *Loquens* 1.1–23. http://doi.org/10.3989/loquens.2014.004.
Denning, Peter J., and Matti Tedre. 2019. *Computational thinking*. Cambridge, MA: MIT Press. https://doi.org/10.7551/mitpress/11740.001.0001.
Edwards, Owen. 2021a. *Rote-Meto comparative dictionary*. Canberra: ANU Press. http://doi.org/10.22459/RMCD.2021.
Edwards, Owen. 2021b. Supplementary material for 'Rote-Meto Comparative Dictionary'. https://doi.org/10.22459/RMCD.2021.
Einaudi, Peter; Jonathan Gordon; and Kelly Kang. 2022. Baccalaureate origins of underrepresented minority research doctorate recipients. NSF 22-335. Alexandria, VA: National Center for Science and Engineering Statistics. https://ncses.nsf.gov/pubs/nsf22335.
Finkel, Raphael. 2016. Computer-based tools for word and paradigm computational morphology. *Oxford research encyclopedia of linguistics*. Oxford: Oxford University Press. https://doi.org/10.1093/acrefore/9780199384655.013.162.
Halle, Morris. 1971 [1959]. *The sound pattern of Russian: A linguistic and acoustical investigation*. 2nd printing. The Hague: Mouton.
Harris, James W. 1969. *Spanish phonology*. Cambridge, MA: MIT Press.
Hayes, Bruce. 2009. *Introducing phonology*. Malden, MA: Wiley-Blackwell.
Hembree, Ray, and Donald Dessart. 1992. Research on calculators in mathematics education. *Calculators in mathematics education*, ed. by James Taylor Fey and Christian R. Hirsch, 23–32. Reston, VA: National Council of Teachers of Mathematics.
Inkelas, Sharon. 1995. The consequences of optimization for underspecification. *North East Linguistic Society (NELS)* 25.287–302. https://hdl.handle.net/20.500.14394/36784.
Johnson, C. Douglas. 1972. *Formal aspects of phonological description*. The Hague: Mouton. https://doi.org/10.1515/9783110876000.
Kenstowicz, Michael, and Charles W. Kisseberth. 1979. *Generative phonology: Description and theory*. New York: Academic Press. https://doi.org/10.1016/C2013-0-10970-1.
Kisseberth, Charles, and Mohammad Abasheik. 1974. Vowel length in Chi-Mwi:ni—A case study of the role of grammar in phonology. *Chicago Linguistic Society (Parasession on natural phonology)* 10(1).193–209.
Knuth, Donald E. 1992. *Literate programming*. Stanford, CA: CLSI Publications.
Larson, Richard K.; David S. Warren; Juliana Freiré de Lima e Silva; D. O. Patricia Gomez; and Konstantinos Sagonas. 1998. *Semantica*. Cambridge, MA: MIT Press.
Linguistic Society of America. 2019. *Annual report 2018: The state of linguistics in higher education*. Washington, DC: Linguistic Society of America. http://files.eric.ed.gov/fulltext/ED607313.pdf.
Papert, Seymour. 1990. *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
Piwowarczyk, Dariusz. 2022. Computational approaches to linguistic chronology and subgrouping. *The Indo-European language family: A phylogenetic perspective*, ed. by Thomas Olander, 33–50. Cambridge: Cambridge University Press. https://doi.org/10.1017/9781108758666.003.
Reiss, Charles. 2022. Priority union and feature logic in phonology. *Linguistic Inquiry* 53.199–209. https://doi.org/10.1162/ling_a_00400.
Ruthven, Kenneth. 1996. Calculators in the mathematics curriculum: The scope of personal computing technology. *International handbook of mathematics education, part 1*, ed. by Alan J. Bishop, Ken Clements, Christine Keitel, Jeremy Kilpatrick, and Colette Laborde, 435–68. Dordrecht: Kluwer. https://doi.org/10.1007/978-94-009-1465-0_13.
Steel, George, and Peter Jurgec. 2017. Derive! An online tool for rule derivation in phonology. https://phonology.us/derive.
Summer Institute of Linguistics. 2005. Phonology Assistant [Computer software]. https://software.sil.org/phonologyassistant/.
van Vugt, Floris. 2012. Pheatures Spreadsheet [Computer program]. Los Angeles: Department of Linguistics, University of California, Los Angeles. https://brucehayes.org/120a/Pheatures/.
Wing, Jeanette M. 2006. Computational thinking. *Communications of the ACM* 49.33–35. https://doi.org/10.1145/1118178.1118215.
Zuraw, Kie. 2004. FeaturePad [Computer software]. Los Angeles: Department of Linguistics, University of California, Los Angeles. https://brucehayes.org/120a/FeaturePad.htm.

Daniel A. Kaufman is an associate professor in the Department of Linguistics and Communication Disorders at Queens College, CUNY, and founding co-director of the Endangered Language Alliance. Email: dkaufman@qc.cuny.edu.

Raphael Finkel is an emeritus professor in the Department of Computer Science at the University of Kentucky. Email: raphael@cs.uky.edu.

Cynthia Gan is at Hunter College, CUNY. Email: cynthia.gan42@stu-mail.hunter.cuny.edu.

Cite this article: Kaufman, Daniel A.; Raphael Finkel; and Cynthia Gan. 2026. Phonology as coding: An online tool for teaching and developing analyses. *Language* 102, 198–222. https://doi.org/10.1017/S0097850726000056.

https://doi.org/10.1017/S0097850726000056 Published online by Cambridge University Press