Phono User's Manual

Revised June 21, 2018

1. Introduction
    - 1.1. What is Phono?
    - 1.2. The accompanying models

2. Getting started

3. Operating an existing model
    - 3.1. Interactive mode
        - 3.1.1. Keyboard input
        - 3.1.2. Processing
        - 3.1.3. Screen output
    - 3.2. Batch mode
    - 3.3. Input Alphabets and Interpretive Rules
    - 3.4. Testing the Reflex Alphabet and Reflex Rules

4. Creating and editing models
    - 4.1. Editing the Input Alphabets
    - 4.2. Editing rule order
    - 4.3. Editing rules: If-lines and then-lines
        - 4.3.1. Feature names
        - 4.3.2. Location expressions
        - 4.3.3. The seven types of lines
            - 4.3.3.1. The Branching line
            - 4.3.3.2. The COUNT line
            - 4.3.3.3. The Absolute line
            - 4.3.3.4. The Relative line
            - 4.3.3.5. The DELETE line
            - 4.3.3.6. The INSERT line
            - 4.3.3.7. The SWAP line
        - 4.3.4. Clicking the buttons of the Rule Editor
            - 4.3.4.1. Comment buttons
            - 4.3.4.2. The name button
            - 4.3.4.3. If-line labels
            - 4.3.4.4. Then-line labels
            - 4.3.4.5. Location expressions
            - 4.3.4.6. Feature names, signed and unsigned
            - 4.3.4.7. Toggling and rotating
            - 4.3.4.8. INSERT, DELETE, and SWAP lines
            - 4.3.4.9. Inert buttons
    - 4.4. Editing the Vocabulary for Batch mode
5. From conventional rule notation to Phono's notation

1. Introduction

1.1. What is Phono?

Phono is a software tool for developing and testing models of regular historical sound change in natural languages. It is programmed in Visual Basic (Version 5) to run under Windows. For this purpose, a model consists essentially of an ordered series of sound-change rules. Phono operates on the assumption that a large portion of the vocabulary of any language is regular in its historical development, meaning that these words are subject to the same series of changes of pronunciation in the same chronological order. In Phono's Interactive mode, the user types an ancestor word (the **etymon**) on the keyboard and the program applies the rules of change to it in order, displaying on the screen the successive stages of development and the final descendant form (the **reflex**).

Input words are received as strings of keyboard characters. These are encoded as arrays of binary feature values based essentially on the features defined by Chomsky and Halle in *The Sound Pattern of English*. The successive changes of the derivation are carried out according to rules expressed entirely in terms of binary feature values. After each change, the feature values are decoded into phonetic symbols for display.

The rule notation follows an if/then structure which—together with the reduction of phonological segments to arrays of feature values—makes it possible to write rules not only for simple changes such as assimilation, dissimilation, syncope, metathesis, etc., but also for rules of greater complexity.

The rules of historical change are called **Diachronic rules** (literally "across time"). In addition to the rules with a diachronic function, a model may make use of formally similar rules for an interpretive purpose. **Interpretive rules** serve to provide precise phonetic detail in the encoding from input characters to feature values. As detailed in Section 3.3 below, there are two "kinds" of Interpretive rules, based on their respective functions: **Etymon Interpretive Rules** and **Reflex Interpretive Rules**. These rules fine-tune the feature values of, respectively, etymon input and—when running in the Batch mode—reflex input. (For more about the Batch mode, see Section 3.2.)

In referring to Diachronic, Etymon Interpretive, and Reflex Interpretive Rules, we are distinguishing only their respective *functions*; there is no difference of *form* among them. They are stored together in the "Supply" of rules, and in fact it is possible for one rule to serve more than one of the three functions. For example, in the Spanish model (described below) the rule of H_DELETION serves twice as a Diachronic rule—once for an early sound change in Latin and again for a change in late medieval Spanish—and then again as a Reflex Interpretive rule for dealing with the silent orthographic *h* of Modern Spanish.

The etymon may be a documented form (as in the case of the Latin input for the Spanish model), or it may be a hypothetical, reconstructed form for a model of a language without a documented ancestor (as in the case of Proto-Algonquian, for the Shawnee model). Phono tests the logic of the presumed sound-change rules and their presumed chronological order.

The program deals with just one line of descendancy at a time—rather than deriving sister languages simultaneously. It simulates only "downstream" derivation—that is, Phono cannot put a sound-change model into "reverse" and project upstream to generate ancestor words. Thus the program does not carry out directly the process of comparative reconstruction (one ancestor word from many descendants), although it may indirectly help with that enterprise by testing hypotheses.

It is my hope that others can make use of Phono to construct models of sound change for a variety of languages, or to improve on the bundled model for Spanish. Please send suggestions for improvement of the program, or of the Spanish model, or additional models for other languages, to the author, Lee Hartman <lhartman@siu.edu>.

1.2. The accompanying models

Phono is bundled with four models:

• "span15" is a model for Spanish, with some 120 rules, based largely on C.-P. Otero's *Evolución y revolución en romance.*

• "shawnee1", with 32 rules, was constructed to derive words of Shawnee from their (reconstructed) etymons in Proto-Algonquian. It was first written by Towhid Bin Muzaffar in 1997, and was adapted by me, Lee Hartman, to be operated by the 2018 version of Phono. In the Rule Editor, the comment on the name line of each rule includes a number or numbers that refer to the corresponding rules in Muzaffar's thesis (pp. 24-25), which is available online at <http://www.nlc-bnc.ca/obj/s4/f2/dsk3/ftp05/mq25868.pdf>.

• "igpay" is a simple two-rule model for deriving Pig Latin (children's "secret" language) from English words.

• "laws1" is not functionally a model, but rather a repository of 80 sound-change rules that have been proposed for a broad variety of languages, ranging from Anglo-Frisian to Winnebago. Most of these "laws" are based on rules described in R. L. Trask's *The Dictionary of Historical and Comparative Linguistics* and N. E. Collinge's *The Laws of Indo-European,* and they are included here mainly to demonstrate how Phono's unique rule notation portrays a variety of phonological rules.

2. Getting started

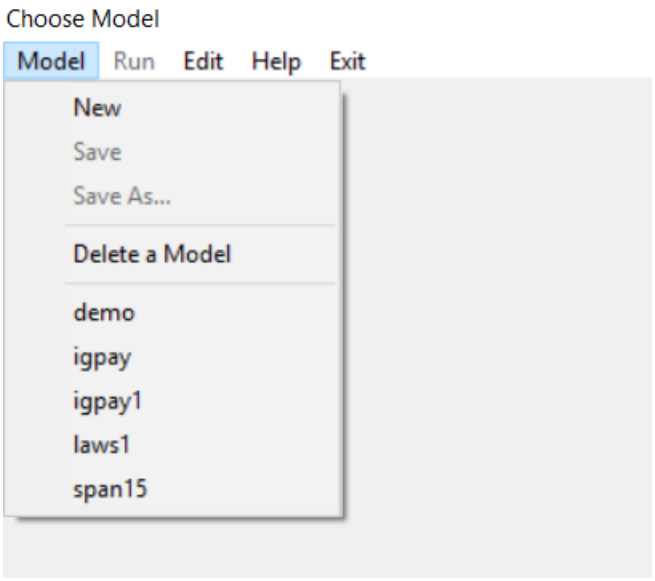
When you download <phono.zip> it will first be saved in your Downloads folder. Create a new, dedicated folder for Phono on your disk and transfer (drag and drop) <phono.zip> there. Once it is installed in that location, click on the zip file to deploy its contents: (1) the program <phono.exe>; (2) a dynamic-link library called <msvbvm50.dll>, which is a necessary auxiliary to the program; (3) the phonetic font <silmipa.ttf>; and (4) the four models (<span15.pho>, <igpay.pho>, <shawnee1.pho>, and <laws1.pho>). Finally, you need also to move the font, <silmipa.ttf>, to the folder where your other fonts reside; often this is <C:\Windows\Fonts\>. You are now ready to activate Phono by clicking on <phono.exe>.

3. Operating an existing model

Once you have booted Phono, your first action is to choose a model. In the menu on the Home Page, click "Model" to open the list of existing models. These include the four models that are bundled with the program, plus any models you have created and saved (see Figure 1). Choose a model by clicking on its name. You may then operate the model in either the **Interactive mode** (one word at a time) or the **Batch mode** (several words in quick succession) by choosing, in the menu under "Run", either "Interactive" or "Batch". I suggest you first get acquainted with the Interactive mode.

The third menu option under "Run"—namely "Reflex"—serves to test the accuracy of the model's Reflex Alphabet and Reflex Interpretive Rules (see Section 3.4 below)

Figure 1: Choosing a Model



3.1. Interactive mode

In the Interactive mode you type an etymon in the text box and click "Derive" or press the <Enter> key to see the derivation of the word. Click "Clear" or press Alt-C to clear the screen for another word.

3.1.1. Keyboard input

The etymon must be entered using the keyboard characters of the model's Etymon Alphabet. For the Latin input of the Spanish model, this consists of the 26 lowercase letters of the Latin alphabet, plus the colon (:) to follow a vowel character as an indicator of a long vowel. Spaces are not allowed in the etymon. If you accidentally type a character that is not in the alphabet, an error message will result. You can view (and edit) the Etymon Alphabet by choosing it under "Edit" in the menu.

As an alternative to typing the etymon, you can copy an etymon from the Vocabulary and paste it in the etymon box. Under "Edit" in the menu, click "Vocabulary". On the Vocabulary list, click the etymon you wish to derive. On the pop-up menu, click "Copy Etymon to Derive". This action takes you to the deriving page, with the chosen etymon in the etymon box. Click the "Derive" button.

3.1.2. Processing

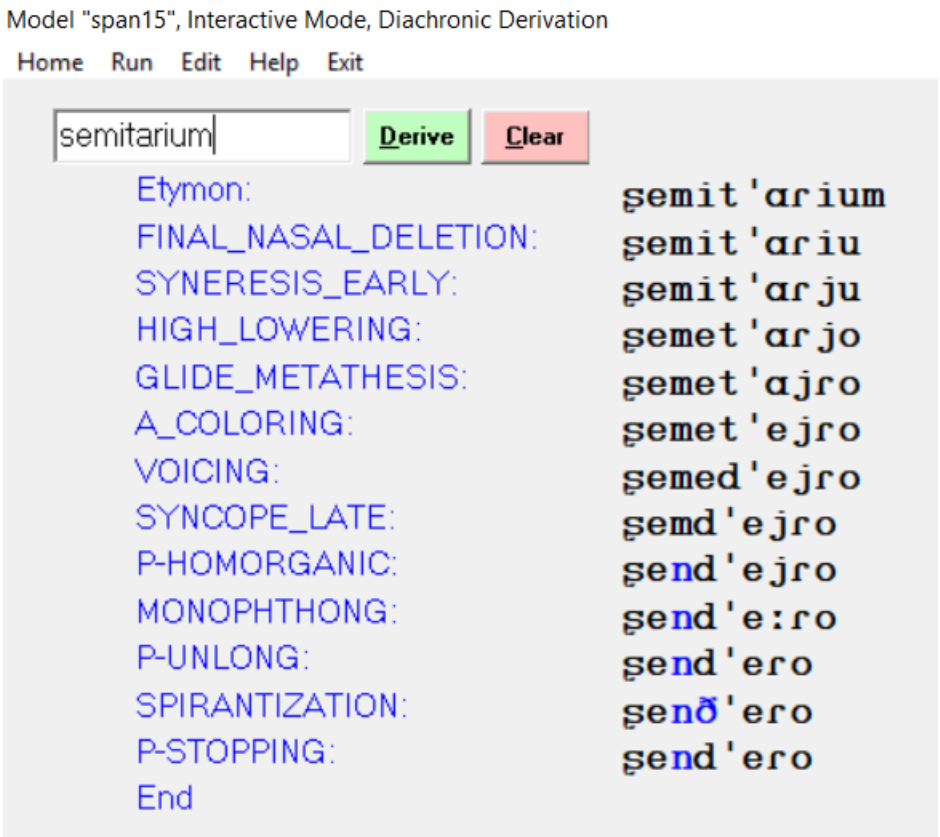
In the derivation of a word, Phono traverses the list of Diachronic rules in order. For each rule, the program traverses the length of the word from left to right, making each segment, momentarily, the **focus segment** for purposes of the rule. For each segment, the program tests for the truth value of the rule's if-conditions. If the test produces a value of "true", then the changes indicated by the rule's "then" section are carried out.

3.1.3. Screen output

The output of a derivation is displayed typically as a series of successive stages in the development of the word, with each change represented by a form in phonetic transcription alongside the name of the rule that has effected the change (see Figure 2). The phonetic font is a product of SIL International <www.sil.org> ©1993, used with permission. The phonetic alphabet—the Output Alphabet—is internal to the program and cannot be edited: it is the same for all models. Its characters are those of the International Phonetic Alphabet, with the single exception of the palatal affricates, [ʧ] and [ʤ], as these digraphs are not provided in the phonetic font. I replace them with the "wedged" symbols [č] and [ǰ] respectively. Unfortunately, Visual Basic 5 does not have direct access to the extended character set of Unicode.

The word is decoded from binary feature values to phonetic symbols by comparing the feature values of each segment with those of each symbol in the Output Alphabet and choosing the nearest equivalent symbol from that Alphabet. If the match is perfect, with all corresponding feature values equal, the phonetic symbol is displayed in black. But if any features disagree, the symbol is displayed in blue. Touching the cursor to a blue symbol will bring up a display indicating which features disagree.

Figure 2: A Sample Derivation



In each line of the derivation, the string of phonetic symbols is accompanied by the name of the rule that effected the change. The letter "P-" prefixed to a rule name indicates a **persistent rule**. This is a rule that remains active throughout the derivation, acting whenever its conditions are met. The Latin and Spanish rule of nasal assimilation is a persistent rule. Most sound-change rules are not persistent, but rather **transient**, meaning that they act once in the history and are not manifested again.

3.2. Batch mode

During development of a model, after each alteration it should be tested in the Batch mode—otherwise, alterations made for one set of words may cause errors for some other set. Such false steps in the development of models should be detected early, so that they can be corrected.

In the Batch mode, Phono runs the derivations of several words in quick succession (several *hundred* words in the case of the Spanish model), in order to compare their computed results with the known, documented reflexes, thus testing the accuracy of the model. For this purpose, the model includes a Vocabulary of word pairs: etymon and reflex (Latin and Spanish, for example).

The same as in the Interactive mode, the etymons are submitted to the series of sound changes, but in the Batch mode they are not decoded into phonetic symbols. Instead, the comparison between the final computed result and the known reflex is carried out on the basis of the binary feature values of both forms. For this purpose, the known reflex is also encoded as an array of feature values, using the values specified in the Reflex Alphabet, with possible modification by the Reflex Interpretive Rules.

Batch results are displayed in two lists: the "good" matches and the "bad" ones. The bad matches appear with indication of the segment and feature of the first discrepancy encountered, as an aid to diagnosing the problem. Text from either list can be copied with Ctrl-C and pasted elsewhere with Ctrl-V.

3.3. Input Alphabets and Interpretive Rules

Every model includes two alphabets for input: the Etymon Alphabet and the Reflex Alphabet. In the Interactive mode, only the Etymon Alphabet is used. But in the Batch mode, both input alphabets are used, since etymons and reflexes may be encoded (from keyboard characters to feature values) according to different conventions.

The etymons that are input to the Interactive mode, as well as both the etymons and the reflexes that are input to the Batch mode—all these input words must be expressed in terms of characters that are available on the keyboard: upper- or lowercase letters, numerals, punctuation, etc. (Accented letters such as *é*, accessed by keying Alt-0233, are also accepted.) The input words must be spelled in a consistent "orthography" that represents their phonological makeup unambiguously.

However, it is possible that the ancestor or descendant language includes sounds other than those represented in the International Phonetic Alphabet by the lowercase letters of the Latin alphabet. In this case it is necessary to devise a keyboard representation for such sounds, and to include that convention in the respective input alphabet of the model. Suppose for example that the ancestral

phonology includes the voiced interdental fricative [ð] (as well as the stop [d]). Since [ð] is not available on the keyboard, one solution would be to represent the fricative as uppercase “D” and to install this character in the Etymon Alphabet with the feature values [+continuant], [+delayed release], etc. to contrast it with [d]. Another solution would be to represent the fricative as the sequence “dh” (provided this is not being used also for an aspirated stop). In this case, the proper encoding as a single fricative segment could be accomplished by means of an interpretive rule, to the effect that whenever “d” is followed by “h” the “d” is rendered fricative and the “h” is deleted.

Interpretive rules can also be used to adjust allophonic details of input, making it unnecessary to include every allophone in the input alphabet.

In the case of the Spanish model, the standard orthographies of both Latin and Spanish are sufficiently unambiguous that they can be used for input directly, with only slight modifications. The Latin long vowels are represented by a following colon—thus for example *vīta* is entered as “vi:ta” (and an Etymon Interpretive Rule lengthens the vowel and deletes the colon segment). Similarly for Spanish, various environmentally conditioned phonetic details can be derived from the standard orthography by means of Reflex Interpretive Rules: the placement of stress; the spirantization of *b, d,* and *g*; the pronunciation of *c* as either [k] or [θ] , etc.

3.4. Testing the Reflex Alphabet and Reflex Rules

The third choice under “Run” in the menu, namely “Reflex”, serves to test the Reflex Alphabet and Reflex Interpretive Rules. Its input is the known reflex of a word, typed according to the designated “orthography”. The Reflex Interpretive Rules are applied in order, the same as the Diachronic Rules, but in this case only the final result is decoded and displayed in the phonetic symbols of the Output Alphabet. For the Spanish model, the input “ceja” produces the output “ θ′exa” (the vertical stroke for stress directly precedes the stressed vowel; Phono does not recognize syllable boundaries).

4. Creating and editing models

To create a new model, click on “New” under “Model” in the menu. You will be prompted to type the name of the new model. Model names must be between 3 and 20 characters in length. When you press the <Enter> key, you will be offered a textbox entitled “Notes”, where you can record a description or any comments about the model.

In addition to the Notes, a model consists of seven components:

- a supply of rules (their names and contents), which can be called by any of the three Order lists.
- the Diachronic Order list, an ordered list of rule names that controls the order of historical changes.
- the Etymon Order list, an ordered list of rule names that controls the order of application of the Etymon Interpretive rules.
- the Reflex Order list, an ordered list of rule names that controls the order of application of the Reflex Interpretive rules, for Batch testing.

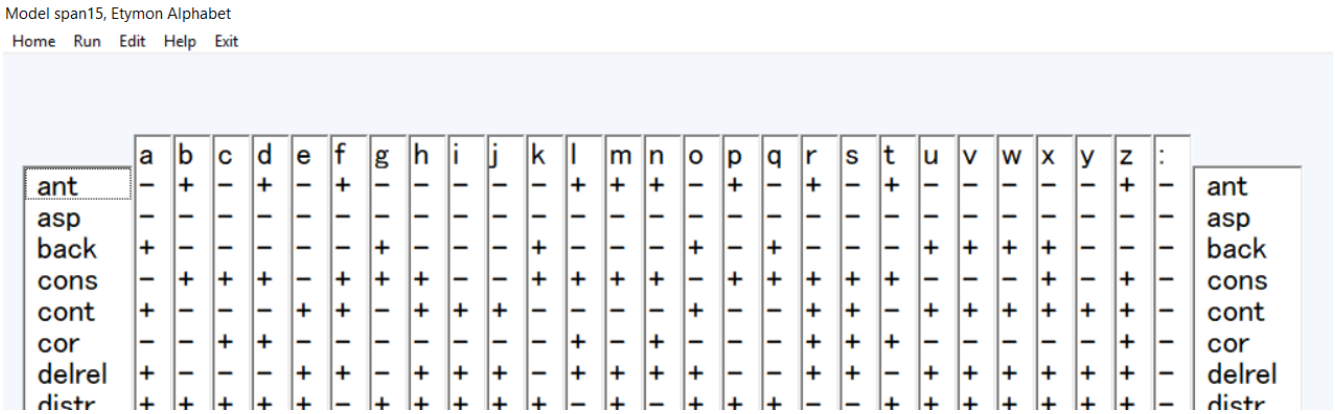
- the Etymon Alphabet, consisting of a set of keyboard characters and their assigned feature values, used for encoding etymon input.
- the Reflex Alphabet, consisting of a set of keyboard characters and their assigned feature values, used for encoding reflex input for Batch testing.
- the Vocabulary, a list of word pairs—etymon and reflex—for Batch testing.

4.1. Editing the Input Alphabets

Both input alphabets—for etymon and reflex alike—consist of a set of keyboard characters and the binary feature values of the sounds they represent. These alphabets are likely to be different from each other in some ways, and certain to be different for different models. To edit an input alphabet, choose “Edit” in the menu, then “Alphabet”, and finally either “Etymon” or “Reflex”. (The third choice under “Alphabet” opens a display of the Output Alphabet, which is read-only.)

The display of either input alphabet consists of a row of keyboard characters across the top, the list of binary features in columns down both sides, and the binary values—“+” or “–”—in the body of the display. Figure 3 shows part of the Etymon Alphabet for the Spanish model . Changing the value of a feature for any character is a simple matter of clicking on the plus or minus symbol to toggle it to the opposite value.

Figure 3: Part of an Etymon Alphabet



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	:	
ant	–	+	–	+	–	+	–	–	–	–	–	+	+	+	–	+	–	+	–	+	–	–	–	–	–	+	–	ant
asp	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	asp
back	+	–	–	–	–	–	+	–	–	–	+	–	–	–	+	–	+	–	–	–	+	+	+	+	–	–	–	back
cons	–	+	+	+	–	+	+	+	–	–	+	+	+	+	–	+	+	+	+	+	–	–	–	+	–	+	–	cons
cont	+	–	–	–	+	+	–	+	+	+	–	–	–	–	+	–	–	+	+	–	+	+	+	+	+	+	–	cont
cor	–	–	+	+	–	–	–	–	–	–	–	+	–	+	–	–	–	+	+	+	–	–	–	–	–	+	–	cor
delrel	+	–	–	–	+	+	–	+	+	+	–	+	+	+	+	–	–	+	+	–	+	+	+	+	+	+	–	delrel
distr	+	+	+	+	+	–	+	+	+	+	+	–	+	–	+	+	+	–	–	+	+	+	+	+	+	+	–	distr

A segment cannot be simultaneously [+high] and [+low], nor can it be [+front] and [+back]. The alphabet editors enforce this entailment automatically: any editing that gives one of these four features a plus sign automatically gives its corresponding opposite feature a minus sign. (On the other hand, the combinations [–high, –low] and [–front, –back] are allowed.)

To add a character to the alphabet, select the column whose feature values are most similar to those of the new character (for example “d” if the new character is to represent [ð]). Click on the keyboard character at the top of the selected column and press the <Insert> key. This creates a new column with feature values identical to those of the selected column and opens a textbox in which to type the new character. After entering the new character, edit its feature values, clicking to toggle them, to mark how the new character differs phonetically from the one on which it was based.

To delete a character from the alphabet, select its column by clicking on the character at the top, and then press the <Delete> key.

To move a character and its column to a different position in the alphabet, select the column by clicking on the character, and then move it by means of the left and right arrow keys on your keyboard.

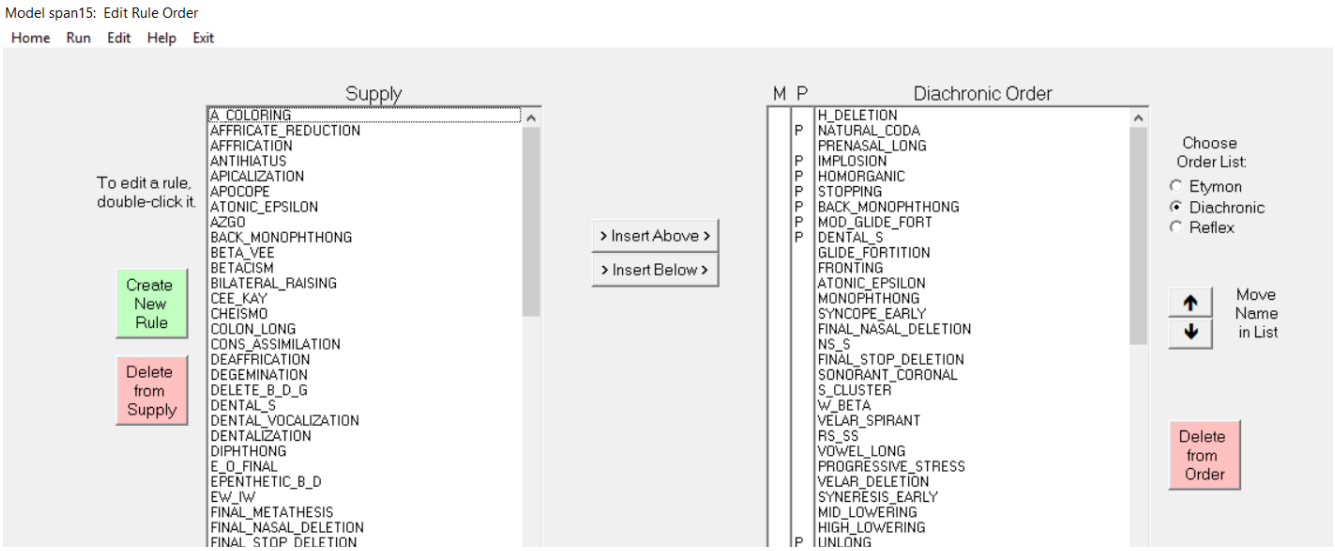
To replace a character (keeping the same feature values), select it and its column with a click, and then press the desired keyboard key. For this purpose, any letter or numeral key, shifted or not, is permitted, but duplication of an existing character is not allowed.

4.2. Editing rule order

As noted above, there are three lists that control the order of application of rules: the Diachronic Order list, the Etymon Order list, and the Reflex Order list. The Etymon and Reflex Order lists control the application of the interpretive rules—those rules that refine the precision of the input alphabets. And the Diachronic Order list controls the order of historical sound changes.

Click “Edit > Rules” on the menu to open the Order page for editing. There, you are presented with a list labeled “Supply”, on the left, with the names of all rules available in the model, in alphabetical order. And on the right is the list of Diachronic rules in chronological order (see Figure 4). To the right of this list is an array of three radio buttons where you can choose to display (and edit) the list of Etymon Interpretive rules or the list of Reflex Interpretive rules, or return to the list of Diachronic rules.

Figure 4: The Order Page



A rule from the Supply list can be inserted in the currently displayed Order list by clicking to select the chosen rule in Supply, clicking to select the rule in the Order list above or below which you wish to insert the new rule, and finally clicking, between the two lists, the button to “Insert Above” or “Insert Below”.

A rule can be removed from an Order list by clicking to select its name and then clicking the pink “Delete from Order” button to the right of the list.

The order of a rule in an Order list can be altered by clicking to select the rule and then clicking the buttons with up and down arrows to the right of the list, labeled “Move Name in List”.

The Diachronic Order list—unlike the other two Order lists—has two narrow columns along its left edge, labeled “M” and “P”. You may click in the M column alongside a rule name to temporarily “mask”, or disable, the rule. Masked rules are skipped in the sequence when deriving a word. A rule marked “P” is treated as “persistent”, meaning that it is applied throughout the derivation whenever its conditions are met. When you save a model, the P is preserved, but the M is not.

To examine or edit an individual rule in the Supply list, double-click on its name. This will take you to the Rule Editor, with the display of the contents of the rule.

To create a new rule, click on the green button labeled “Create New Rule”, to the left of the Supply list. This also will take you to the Rule Editor, with the display of a rule template to edit as a new rule.

To remove a rule from the Supply list, click to select its name, and then click the pink “Delete from Supply” button to the left of the Supply list. Caution: A rule deleted from Supply cannot be recovered.

In either of the two displayed lists, you can search for a particular rule by typing the first letter of its name.

4.3. Editing rules: If-lines and then-lines

Phono uses a unique notation for rules, based on an if/then structure and the use of binary phonological feature values. Most phonologists are familiar with a conventional format of rules based on a schema such as the following—

$A \rightarrow B / C \_ D$

—meaning that element A changes into B in the environment following element C and preceding element D. In other words, each occurrence of the sequence CAD (known as the rule’s **structural description**) is changed into CBD (and the change of A to B is called the **structural change**). This notation tends to portray sound change as the *replacement* of one segment by another. But often a single sound change affects an entire “natural class” of segments. If, for example, [p] becomes [b], [t] becomes [d], and [k] becomes [g], rather than three rules of replacement, it is in some respects more economical to express the change by specifying the affected natural class (perhaps [–continuant, –voice]) and stating the change as a toggle from [–voice] to [+voice].

Phono’s rule notation portrays the structural description as a set of one or more “**if-lines**”, labeled in order by letters (A, B, C…), and the structural change as a set of one or more “**then-lines**”,

labeled in order by numerals (1, 2, 3…). In the event that the if-lines number more than one, they are interrelated as a hierarchy, with the truth value of line A dependent on that of lines B and C, the value of B dependent on D and E, and so on. If line A is true for the current **focus segment** in the scan of the word, then the changes indicated by the then-lines are carried out.

In each rule, the if-lines and then-lines are sandwiched between the **name line** (the name of the rule) and the **end line** (the word "END:" followed by the rule name). This structure is demonstrated in the rule "FRONTING" from the Spanish model. This rule changes Latin /aj/ (orthographic *ae*) and /oj/ (orthographic *oe*) to /ɛj/ and /ej/ respectively.

FRONTING
A: B and C
B: +syll (*)
C: –cons (*+1) –syll (*+1) +front (*+1)
1: +front (*) –round (*)
2: tense (*) <> low (*)
3: –low (*)
END: FRONTING

The rule is interpreted as follows:

A: The line is true if both B and C are true.

B: The line is true if the focus segment is [+syllabic] (namely a vowel).

C: The line is true if the segment following the focus segment is [–consonantal, –syllabic] (namely a glide) and [+front], namely the glide /j/.

Line "1:" makes the focus segment [+front, –round] (namely a front, unrounded vowel). The change to [+front] is accompanied automatically by a change to [–back] to avoid the forbidden combination [+front, +back].

Line "2:", with reference to the focus segment, makes the sign of the feature [tense] opposite to that of the feature [low]. The effect (together with lines "1:" and "3:" in this rule) is to turn the originally low (and tense) vowel /a/ into the lax vowel /ɛ/, and (vacuously) to conserve the [+tense] value of the non-low vowel /o/ in its resulting /e/.

Line "3:" completes the process of turning the low /a/ to the non-low /ɛ/, while vacuously conserving the non-low status of the original /o/ in its resulting /e/. (See Section 4.3.1 below for more on the interaction of [high], [low], and [tense].)

Both if-lines and then-lines are composed mainly of **feature names**—with or without a plus or minus sign—and **location expressions**, which identify a specific segment in the word.

4.3.1. Feature names

Phono encodes each segment of the word as an array of binary feature values, using a set of 21 features based mainly on those defined by Chomsky and Halle in *The Sound Pattern of English*. Exceptionally, instead of the *SPE* feature [sonorant], Phono uses its opposite, namely [obstruent]. Any [+sonorant] segment is [−obstruent], and vice versa.

For vowels, in addition to the features [high] and [low], Phono uses the feature [tense] to produce six degrees of vowel height. Vowels that are [+high, +tense] are higher than their [−tense] counterparts; vowels that are [+low, +tense] are lower than their [−tense] counterparts; and mid vowels are characterized as [−high, −low], with the [+tense] mid vowels being higher than their [−tense] counterparts. Since the combination [+high, +low] is forbidden, Phono accompanies a change to [+high] with a change to [−low] automatically; and conversely, a change to [+low] automatically brings about a change to [−high].

The feature inventory offered in *The Sound Pattern of English* provides only the feature [back] to differentiate two degrees of frontness/backness for vowels. But Phono, in order to accommodate the three degrees of the IPA alphabet—front, central, and back—utilizes both the features [front] and [back]. Central vowels are characterized as [−front, −back]. Since the combination [+front, +back] is forbidden, Phono accompanies a change to [+front] with a change to [−back] automatically, and conversely, a change to [+back] automatically entails a change to [−front].

Table 1 on the next page shows the basis for decoding feature value combinations for vowels.

4.3.2. Location expressions

Location expressions refer to where in the word—in which segment—conditions for change are sought (in if-lines) or changes are carried out (in then-lines). All location expressions are enclosed in parentheses. A location expression can specify a segment in the word in any one of three ways: (1) by counting segments from the beginning or end of the word; (2) by counting segments from the **focus segment** in the left-to-right scan of the word performed by each rule; or (3) by feature value (in COUNT-type if-lines only—see Section 4.3.3.2 below).

Location expressions based on counting segments from the beginning or end of the word consist simply of a number—positive if counting from the beginning, negative if from the end: “(1)” for the initial segment, “(−2)” for the second-to-last segment, and so on. Location expressions based on relation to the focus segment contain an asterisk to represent the focus and optionally a number for distance from the focus: positive if to the right, or negative to the left: “(*)” for the focus segment, “(*−1)” for the first segment to its left, “(*+2)” for the second segment to its right, and so on.

.

Table 1: Decoding Vowels

Decoding from feature values to phonetic symbols		+front		–front, –back		+back	
		–round	+round	–round	+round	–round	+round
+high	+tense	i	y	ɨ	ʉ	ɯ	u
	–tense	ɪ	ʏ	Ï	Ü	**ɯ**[1] –tense	ʊ
–high, –low	+tense	e	ø	**e**[2] –front	ɵ	ɤ	o
	–tense	ɛ	œ	ə[3]	**ɔ**[4] –back	ʌ	ɔ
+low	–tense	æ	**œ**[5] –tense	**ä**[6] –tense	**ɒ** –back –tense	**ɑ** –tense	**ɒ** –tense
	+tense	a	ɶ	ä	**ɒ** –back	ɑ	ɒ

[1] Phonetic symbols in blue are as they appear in output. In output, touch them with the cursor to see which of their feature values differ from those of the default (black) version of the same symbol.

[2] For the "close-mid" (–high, –low, +tense) unrounded central vowel, a "reversed *e*" is "[n]ot officially sanctioned but suggested" by the IPA (Pullum & Ladusaw, p. 47); but the SIL font does not offer reversed *e*.

[3] The schwa symbol [ə] is decoded from feature values indicating a lax ([–tense]), non-high, non-low (therefore "mid") non-front, non-back (thus "central"), unrounded vowel. Therefore it appears in this table on the level of the vowels called "open-mid" by the IPA, and consequently Phono does not differentiate between schwa (the "mid" central unrounded vowel) and reversed epsilon (the "*open*-mid" central unrounded vowel). Strictly speaking, the IPA's chart places schwa uniquely at a level that is labeled "mid" without being either "open-mid" or "close-mid".

[4] For the open-mid central rounded vowel, the use of a "closed reversed epsilon" is "[n]ot officially sanctioned, but considered" by the IPA (Pullum & Ladusaw, p. 55). Closed reversed epsilon is not offered by the SIL font.

[5] The IPA has no symbol for the "near-open" front rounded vowel.

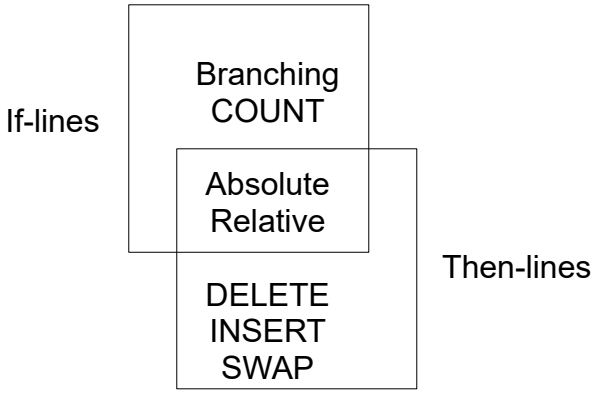
[6] The SIL font does have [ɐ], "turned a", for the lax low central unrounded vowel, but Visual Basic 5 doesn't recognize it.

4.3.3. The seven types of lines

There are four types of if-lines and five types of then-lines, with the "Absolute" and "Relative" types used as both if- and then-lines, for a total of seven types, as shown in Figure 5 on the following page.

Many rules have more than one if-line, and in that case, the first if-line, labeled A, must be a Branching line that states the relationship between lines B and C. Then B and C, in turn, may be of other line-types, or either of them may also branch into D and E, and so on. Line A, as the top element of a hierarchy, must represent *all* the if-lines of the rule combined, and each if-line below line A must be represented in a Branching line somewhere above itself.

Figure 5: Types of If-lines and Then-lines



4.3.3.2. The COUNT line

A COUNT-type if-line carries out a left-to-right subscan from one segment to another in the word, and it counts the number of occurrences of a given feature value between and including them.

The COUNT-type if-line consists of the following six items in order:

- the keyword COUNT
- a signed feature name, the **target feature value**
- a location expression for the **head** of the subscan
- a location expression for the **foot** of the subscan
- a **sign** of comparison (“>”, “<”, or “=”)
- a numerical **standard**

The example COUNT line below means “If the count of [+syllabic] segments from the focus segment to the word’s final segment, inclusive, is greater than two,...” That is, the rule applies from the beginning up to, but not including the second-to-last vowel in the word:

B: COUNT +syll (*) (–1) > 2

In a COUNT line, the abstract feature value [+segment] may be used to count segments *per se*, regardless of their concrete feature values. The following example, by insuring that the number of segments from focus to final position is 1, means, in effect, “If the focus segment is word-final,...”:

C: COUNT +segment (*) (–1) = 1

In a COUNT line, you have the option of specifying the head or foot of the subscan (or both) as locations **by feature value**. When the *head* of the subscan is located by feature value, it refers to the first instance of the specified value encountered in moving to the *left* from the focus segment. And a location by feature value for the *foot* of the subscan refers to the first instance *following* the focus. The following example, from the Spanish model, helps to locate the *post-tonic* vowel of the word—that is,

the first vowel following the stressed vowel. The line counts vowels ([+syll]) in a sequence reaching from the first stressed segment (necessarily a vowel) found to the left of the focus, to the focus itself. If the number is exactly 2 (and if the focus segment is a vowel) then the focus is on the first vowel after the stressed vowel:

D: COUNT +syll (+stress) (*) = 2

COUNT lines have a broad variety of uses, such as to determine whether the focus segment is or is not initial or final, to identify a vowel as belonging to the penultimate syllable, to identify a word as polysyllabic, or as unstressed, or to insure that no vowel intervenes between a marked agent of change and the focus segment.

4.3.3.3. The Absolute line

The line called "**Absolute**" is the simplest type of line. It specifies one or more feature values for one or more locations (segments) in the word. I'm calling this an Absolute line (for want of a better concise term), based on its contrast with the **Relative** type of line (see Section 4.3.3.4). Both if-lines and then-lines can be of the Absolute type. An Absolute line consists of one or more "**units**", each unit consisting of a signed feature name followed by a location expression, as in the following example:

–nas (1)

This unit, in an if-line, means "If the initial segment is nonnasal,..."; or, in a then-line, "…, then the initial segment becomes nonnasal." Up to eight such units can be concatenated in a single Absolute line, with the implication that they are connected by "and", as in the following example:

E: +syll (*) +low (*) –cons (*–1) –syll (*–1)

This if-line means "If the focus is a low vowel preceded by a glide,..."

If an Absolute line contains contradictory units such as "+high (*)" and "+low (*)", Phono, reading left to right, heeds only the last such unit.

4.3.3.4. The Relative line

The **Relative** type of if- or then-line corresponds to the Greek-letter variable signs (the so-called "alpha" device) introduced by Chomsky & Halle in *The Sound Pattern of English* (p. 83). A Relative line refers to the dependence of one feature value in the word upon another, as either equal or opposite. Used as an if-line, the Relative line searches for various conditions of interdependence among feature values. As a then-line, it appears in rules of assimilation (signs equal) or dissimilation (signs opposite). The Relative line consists of five elements, as follows:

- an *unsigned* feature name, the **dependent feature**
- a location expression, the **dependent location**
- a **sign** of equality (=) or inequality (<>)

• a second unsigned feature name, the **standard feature**
• a second location expression, the **standard location**

The following example is a Relative line that specifies the natural class of the back-rounded and front-unrounded vowels together—[α back, α round]:

F: back (*) = round (*)

A Relative then-line for dissimilation uses the sign of inequality (<>). The following then-line makes the nasality of the focus segment opposite to that of the following segment:

2: nas (*) <> nas (*+1)

4.3.3.5. The DELETE line

The DELETE-type then-line deletes a specified segment. It consists of the keyword DELETE and a location expression. The following example would delete the word-final segment:

3: DELETE (–1)

4.3.3.6. The INSERT line

An INSERT-type then-line inserts a lowercase letter from the Roman alphabet at a specified location in the word. It consists of three elements:

• the keyword INSERT
• the character to be inserted, the **new segment**
• a location expression

The following example inserts the segment /e/ immediately after the focus segment:

4: INSERT e (*+1)

The new segment initially has the feature values of its IPA counterpart in the Output Alphabet. These values can be adjusted by means of additional then-lines, of the Absolute or Relative types, in the same rule.

In INSERT lines, the location expression referring to the focus segment alone (*) means the focus segment is *replaced by* the new segment. Otherwise, expressions with the focus plus or minus a number—"(*+2)", "(*–1)", etc.—mean insertion at the indicated segment *boundary*, counting right or left of focus respectively.

4.3.3.7. The SWAP line

A SWAP-type then-line interchanges the positions of two segments in the word (metathesis). Its format consists of three elements: the keyword SWAP and the location expressions of the two segments to be interchanged. The following example line interchanges the focus segment with the segment to its left:

5: SWAP (*) (*–1)

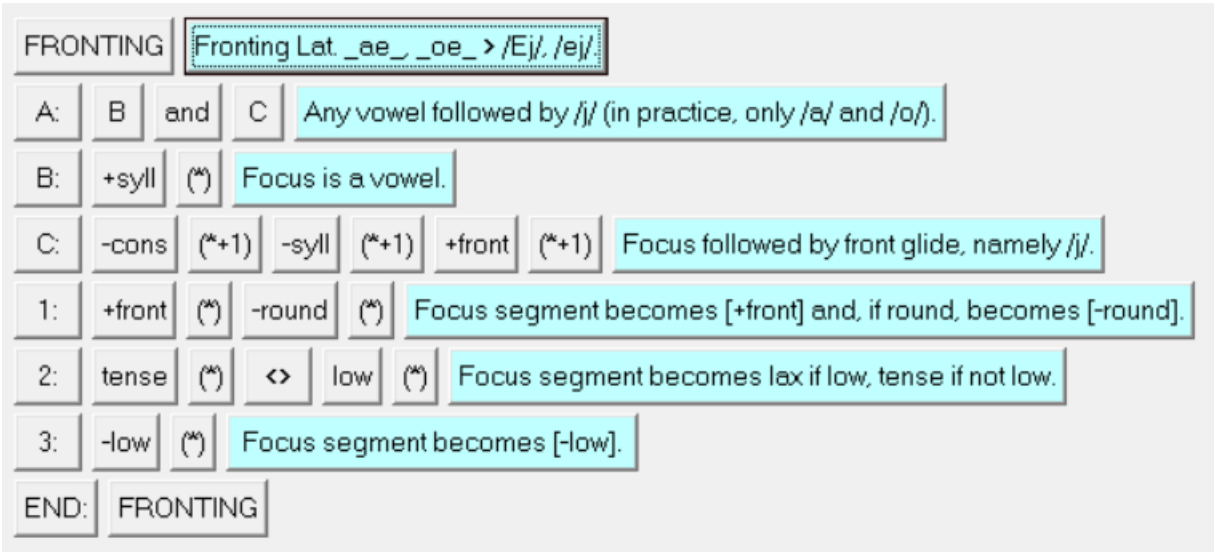
4.3.4. Clicking the buttons of the Rule Editor

In the main menu, click “Edit > Rules” to display the Order screen. To edit a particular rule, find its name in the Supply list on the left side of the screen and double-click on that name. This action opens the Rule Editor, a screen where the chosen rule appears as a display of buttons, one for each constituent of the rule (see Figure 6). Editing is done by clicking on the buttons and the pop-up menus that they evoke.

4.3.4.1. Comment buttons

At the end of each line of a rule there is a button, tinted light blue, for comments. Clicking this button opens a field where you can type a comment. When finished, press the <Enter> key. Comments can be useful as a quick way to grasp the meaning of a line of notation.

Figure 6: The rule FRONTING in the Rule Editor.



4.3.4.2. The name button

To edit the **rule name**, click on its button to open a textbox, type the new name, and press <Enter>. A rule name must begin with an uppercase letter and contain no spaces. Phono enforces this capitalization, and the space bar will insert an underscore.

4.3.4.3. If-line labels

Clicking on the label of an if-line ("A:", "B:", "C:", etc.) opens a pop-up menu with options to change the type of line or to delete the line.

For non-Branching if-lines—namely those of the Absolute, Relative, and COUNT types—the pop-up menu offers options to change to another one of these three types. The new line, appearing at first with arbitrary contents, can then be edited for the specific contents desired. In addition, the option to delete the line is offered, if it is not the rule's one and only if-line. Line "A:" cannot be deleted.

The Branching option is offered only for line "A:", but that line can be branched repeatedly until the maximum permitted number of if-lines (25) is reached. The restriction of branching to line "A:" insures that all pairs of conjoinees consist of letters that are adjacent in the alphabet (for example "B and C", rather than "B and E"). This makes the structure of the rule easier to read, and it does not reduce the versatility of the notation.

A Branching line cannot be changed to another type of if-line by direct editing, since that would leave its daughter lines as orphans. A Branching line can be changed only by deleting one of its (non-Branching) daughter lines. In that event, the body of the remaining daughter line is assigned to the label of the previously Branching line, and the lines below it are accordingly relabeled to close the gap.

Given the above restrictions, Branching lines other than "A:" have *no* options for direct editing. But line "A:", once it it is branched into "B and C", has the option to branch further, proceeding (instantaneously) in the phases shown in the following example, with the if-lines of a hypothetical rule:

- Phase 1: Original form of the rule—
  A: B and C
  B: [Original body of line B]
  C: [Original body of line C]
  etc.

- Phase 2: Opening a gap—
  A: B and C
  B:
  C:
  D: [Original body of line B]
  E: [Original body of line C]
  etc., subsequent labels bumped up by two places in the alphabet.

• Phase 3: Filling the gap while maintaining hierarchical structure and inserting a new line—arbitrarily of the Absolute type, "+cons (*)"—which can then be edited or replaced by a line of another type—

A: B and C
B: D and E
C: +cons (*)
D: [Original body of line B]
E: [Original body of line C]
etc.

4.3.4.4. Then-line labels

A click on the label of a then-line ("1:", "2:", etc.) opens a pop-up menu that offers options (1) to replace the line with a then-line of another type, (2) to insert a new then-line above the clicked line, or—if there is more than one then-line in the rule—(3) to delete the clicked line. (You cannot delete a rule's one and only then-line.) To add a new then-line *below* the last then-line, click the "END:" button in the rule's last line. Caution: On the then-line pop-up menu, do not confuse the two options containing the word "insert"—the option to *change* the line to an INSERT line vs. the option to *insert* a new line. Similarly, do not confuse the two options containing the word "delete": the keyword DELETE to change the line to a DELETE line vs. the option to "Delete This Line" from the rule.

A rule's then-lines are limited to a maximum of nine.

4.3.4.5. Location expressions

Location expressions, always in parentheses, appear in all if- and then-lines except if-lines of the Branching type. Clicking on a location expression opens a pop-up menu that offers options to toggle the "+" or "–" sign (if any) to its opposite, or to choose between formats with or without an asterisk—that is, formats relative to the focus segment, on one hand, or relative to the beginning or end of the word, on the other hand, respectively.

4.3.4.6. Feature names, signed and unsigned

Signed feature names appear in Absolute and COUNT lines, and unsigned feature names appear in Relative lines. Additionally, a signed feature name may appear as a location expression (in parentheses) for the head or foot of the subscan of a COUNT line. Clicking on a feature name opens a pop-up menu that offers options to choose a different feature, to toggle the "+" or "–" sign (if any) to its opposite, or—in an Absolute line—to insert a new "unit" (signed feature name and location). Additionally, in an Absolute line with more than one unit, the menu offers the option to delete the clicked unit.

4.3.4.7. Toggling and rotating

Some rule elements are subject to a small range of only two to four options: the conjunction of a Branching line ("and" or "or"), the sign of comparison in a COUNT line (">", "<", or "="), the numerical standard in a COUNT line (0, 1, 2, or 3), or the sign ("=" or "<>") in a Relative line. In these cases, a click on the button toggles between two options or rotates among three or four.

4.3.4.8. INSERT, DELETE, and SWAP lines

In INSERT, DELETE, and SWAP lines, the keyword button is inert. The location expressions in these three kinds of lines are edited as described in Section 4.3.4.5, above. In an INSERT line, to change the letter to be inserted, click on the letter. This opens a pop-up menu that offers the letters from *a* to *z* from which to choose a replacement. The feature values of the new segment, at first based on those of the corresponding IPA symbol in the Output Alphabet, can be adjusted by subsequent then-lines.

4.3.4.9. Inert buttons

The buttons for the conjoinees in a Branching line and those for the keywords COUNT, INSERT, DELETE, and SWAP are inert: they have no effect. The final button in the rule—the repetition of the rule name after "END:"—is likewise inert.

4.4. Editing the Vocabulary for Batch mode

Running in the Batch mode requires a **Vocabulary** of several (or several hundred) word pairs—etymon and reflex—to compare computed reflexes with the known reflexes. To edit the Vocabulary, click "Vocabulary" under "Edit" in the main menu.

To edit or delete an individual item (word pair), click on the item to select it and open the pop-up menu. On the menu, click "Delete Item" or "Edit Item"; "Edit Item" opens a light blue editing box where you can edit the word pair. When finished, press <Enter> to install the new form of the word pair in the Vocabulary List.

To add a word pair, click "Add Item" on the pop-up menu; this opens the light blue editing box, where you can type the new word pair, and then press <Enter> to install it in the list.

Click "Edit All" in the pop-up menu to copy the entire Vocabulary List into the light blue editing box, or to open the box to type several word pairs at a time. In this mode you can copy (Ctrl-C) word pairs from another source, such as a word processor or a spreadsheet, and paste them (Ctrl-V) into the editing box. When finished, click the button above the editing box to install its contents in the Vocabulary List.

The list forces items into alphabetical order.

The fifth option on the pop-up menu, “Copy Etymon to Derive” enables you to copy the etymon of the selected pair into the etymon box on the deriving page and, with a click of the “Derive” button, to run its individual derivation.

5. From conventional rule notation to Phono’s notation

In order to compose a rule in Phono’s notation, first think of its structural description (i.e. the segment to change and the environmental conditions necessary for the change) as one great if-clause, to be represented by the first if-line, labeled “A:”. Then subdivide those conditions successively by means of Branching if-lines until you reach conditions that are simple enough to be represented by Absolute, Relative, or COUNT-type if-lines. These bifurcating divisions may be between the “triggers” and the “target” of the change (or the “agents” of change and the “mutand”—that which is to be changed); between the left and right sides of the environment; between consonantal and vocalic aspects of the environment; between front and back vowels as the focus of change; or between any other pair of complementary conditions. In writing a complex rule, it is often easier to write comments on the light blue comment buttons *before* writing the rule in formal notation. And although a line of Absolute units *can* include references to different locations, the rule is easier to read if you use a Branching line to separate, for example, the units with “(*)” from those with “(*+1)”.

In the event that an actual rule is too complex to be expressed as a single rule in Phono’s notation, it is possible to notate it as a sequence of two rules in the model, by means of an abstract feature named “mark”. The first notational rule can search for one set of conditions and label a segment [+mark] when they are found. Then the second notational rule can search for the remainder of the conditions, and if they are true and the sign of [mark] is plus, then the structural change can be carried out. This sequence is normally followed by the application of a rule with a name like “UNMARK”, to return the value of [mark] to minus after it has served its purpose.

Use the Rule Editor to view example rules in the models “span15”, “igpay”, “shawnee1”, and “laws1” to see how a variety of notational problems have been handled.

In the Rule Editor, you can copy a rule’s contents with Ctrl-C and paste them into another rule with Ctrl-V, either as a template for a similar rule or to copy a rule from one model into another. These operations affect only the if- and then-lines of the rule, not its name line or end line. In moving from one model to another, you will be prompted to save (or not) the model that you have edited.

After editing rules or alphabets, when you exit the editing screen—for example to run a derivation—you will be asked if you wish to “keep” the changes made in editing. A “Yes” answer preserves the changes only for the current session with the model. To record them permanently before you exit the program, go to the Home page and click “Save” under the heading “Model”. Otherwise, when you click “Exit” to quit the program, you will be asked if you wish to save the model as edited.

=o0o=