Modeling Phonological Change

Lee Hartman
Southern Illinois University
lhartman@siu.edu

1. Overview.

In the following pages I present some desiderata discovered, as well as some questions raised, by the experience of developing, over a period of some twenty years, a program, named Phono, for creating and operating models of regular historical sound change. Phono is now in the beta-testing stage of its fourth version (Hartman 2003a and 2003b). The approach is directed toward producing a practical tool for testing given hypotheses about sound-change rules and their order, and (in the case of languages with undocumented ancestor forms) for testing given hypotheses of reconstructed forms—rather than, say, for generating such hypotheses of rules or forms computationally. Given an ordered set of rules, the model performs a single line of derivation, for one word at a time, downstream from etymon to reflex.

The Phono project has evolved through several incarnations. Version 1 was written around 1980 in PL/I to run on an IBM 360 mainframe whose output was limited to the uppercase letters of the Roman alphabet. A model for deriving Spanish words from Latin was hard-coded in the program. (The Spanish model is based mainly on Otero 1971 and Hartman 1974.) Versions 2 and 3 were written in DOS-based Pascal for microcomputers. In these versions the rules of the Spanish model were extracted from the program code and recast as data for the program to read and interpret, thus opening the possibility for Phono to operate models for other languages. The output notation gained access to both upper- and lowercase letters, as well as the rest of the extended ASCII character set, which was used as an ad hoc phonetic alphabet. And finally, Version 4 (in Visual Basic, for Windows, released in 2003) displays output in a phonetic font, essentially the alphabet of the International Phonetic Association (IPA) (©1993 by SIL International, www.sil.org, used with permission), thus solving the main problem cited by Becker (1996) in his review of Version 3.2.

The main practical issues encountered have been those of notation (see Hartman 1993a and 1993b): notation of data words as input, as output, and during the derivation; and notation of sound-change rules. Based on the experience with Phono, I recommend, below, forms of notation (1) for the input of etymon words, (2) for the internal representation of words during derivation, and (3) for displaying the output of derived forms with both readability (the conventional IPA alphabet) and preservation of unexpected details in the results ("feature-based diacritics", explained below). I also present a versatile form of notation for rules as data and consider ways in which this rule notation might be made more user-friendly to the historical phonologist.

Additionally, I recommend mechanisms to handle some deviations from linear rule order (Chafe's concept of the "persistent" rule, and a provision for temporarily

"masking" a rule). I consider some additional complications of rule order and ways in which the program might be modified to handle them.

I recommend a procedure for processing many pairs of words—etymon and known reflex—together in a "batch" mode, in order to test and maintain the integrity of a model during its development.

Finally, I describe a "rule trace" procedure for monitoring the functional load of each rule.

2. Theoretical assumptions.

In order to make Phono potentially useful to a broad variety of researchers on language history, I have attempted to keep the program as nearly theory-neutral as possible. Nevertheless, the project is based on a set of arguable assumptions, including the following: (1) that sound change is regular *in some degree*; (2) that regular sound change can be modeled as a chronologically ordered set of rules, each of which acts on the output of its predecessor; and (3) that the chronological order of the rules is the same for most words in the language. Even while adopting these assumptions, it is important to bear in mind that *regular* change is only one part of language history. Exceptions to regular change do occur, either explicably—through paradigmatic analogy, dialect mixing and other borrowing, "lexical diffusion" (i.e. gradual and sometimes incomplete passage of a change through the vocabulary—see Wang 1969), influence of written language, or other phenomena—or, let's confess, inexplicably.

"Rules" of sound change, formerly called "laws", are of course not *prescriptive* rules: no explicit authority has ever encouraged sound change. But what then *is* the relationship between the rules and the sound-change phenomena? Phono makes no claim as to whether rules "describe", "explain", "cause", or "reflect" the evolution of sounds in people's speech. As a neutral term, we may say that a rule "corresponds to" a set of changes, until such time as other investigators decide what cause-and-effect relations may be at work, or what "psychological reality" a rule may have. And Phono makes no claim about how or why sound changes are propagated through social and geographical space. These questions are not unimportant; they are merely beyond the intended scope of the program.

3. Internal notation: binary features.

From its beginning, the Phono program has been based on the expression of sound-change rules in terms of binary feature values. Binary features were favored by the early proponents of (synchronic) generative phonology for theoretical reasons: certain combinations of feature values efficiently define various "natural classes" of sounds, i.e. groups of sounds that tend to behave similarly (for example, p > b tends to be accompanied by t > d and k > g, all three of which can be expressed collectively as [–voice] > [+voice]). This aspect of features also partially motivates their use by Phono, but the feature mode of notation is chosen here mainly for its versatility and precision.

Phono has adopted, with minimal modification, most of the features defined in *The Sound Pattern of English* (Chomsky and Halle 1968:298-329)—known as "*SPE*"—(20 of them in the present version) because that work represents a unique moment of near-consensus in the development of phonological theory. Since its publication, the role of features in phonological theory has been discussed vigorously—with the case being made variously for replacement of some of the *SPE* features, "feature geometry" (i.e. hierarchical relationships among the features), "unary" features, multi-valued features, underspecified features, feature markedness, etc. But it is not a goal of the Phono project to enter into the debates on synchronic phonological theory. The *SPE* features are fully adequate for Phono's needs of precision and versatility, but their use is not intended to make any further claim about their "psychological reality" or theoretical significance. Perhaps future versions of Phono and computational models of synchronic phonology can share insights with mutual benefit.

4. Encoding, change, and decoding.

While on one hand the sound-change rules are expressed entirely in terms of binary feature values, on the other hand the typical user deals more easily with character strings. As a result, there is a need for translation of character strings into and out of feature notation. In this regard the interactive derivation consists of five phases: input, encoding, diachronic changes, decoding, and display. (1) The user feeds the etymon word to the program as a string of keyboard characters. (2) The program encodes the word, translating the character string to a set of feature values. (3) The program performs the derivation, changing the word's feature values according to the ordered series of sound-change rules. (4-5) After each diachronic change, the program decodes the word's feature values back to a character-string notation for display on the screen.

4.1. Input and encoding.

In practice, the encoding process for etymon input is more complex than a mere lookup procedure in a table of correspondences between characters and feature values. Of the 81 phonetic symbols of the IPA alphabet, only 26 are directly represented on the keyboard (i.e. the lowercase letters of the Roman alphabet). In the likely event that the ancestor language's phonological inventory includes sounds that differ in some respect from those represented in the IPA alphabet by the 26 letters, the user must devise an unambiguous "orthography" for etymon input from the keyboard. This input notation need not be restricted to a one-character-to-one-phoneme correspondence; for example, it may use digraphs (such as <bh> for the bilabial fricative beta), or, conversely, it could be made to allow Latin orthographic <x> to represent the series /ks/. These devices of input notation can be given their phonological interpretation by "etymon input adjustment rules". (The adjustment from <bh> to beta, for example, would be carried out by a rule that makes /b/ fricative before /h/ and then deletes /h/.) In other words, the apparatus for etymon input consists of (1) an alphabet of keyboard characters with feature values fully specified, held in a lookup table, and (2) a series of adjustment rules that can serve to interpret digraphs, assign context-sensitive feature values (such as nasal assimilation or predictable stress), or otherwise fine-tune the feature configuration of the input word before the derivation begins.

4.2. Diachronic change.

The internal representation of the word as a bundle of feature values is subjected to the series of changes specified by the ordered set of rules. Each rule (consisting of "IF-lines" and "THEN-lines", as explained below), in chronological order, performs a scan of the word, from right to left, examining each segment to see if it and its environment fit the pattern of conditions necessary for the change to occur. In the event that the conditions are met—that the IF-lines are collectively "true"—then the set of THEN-lines is executed, making the prescribed changes to the feature values of the segment on which the scan is currently focused. Each time a change takes place, the feature representation of the word is decoded to display that successive stage in the word's evolution.

4.3. Decoding and display.

While an input alphabet must be limited to the characters available on the keyboard, the output display in the present version of the program has access to the 81 characters of the IPA font. And yet, 20 binary features offer the prospect, mathematically, of the 20th power of 2, or more than a million different combinations. Of course not all these combinations are phonetically possible, but even in practical terms, the feature notation for a segment potentially carries more information than an unmodified IPA phonetic symbol can. So the challenge for decoding is not only to translate from features to phonetic symbols, but also to preserve somehow any information that is lost in the process.

Each phonetic symbol in the output alphabet is "fully specified" with feature values; that is, every feature is assigned either a "plus" or a "minus" value. Additionally, in the alphabet's lookup table, some of these positive and negative values are marked with "double" signs (#, =), while others have "single" signs (+, –). The double signs of a particular phonetic symbol designate the minimal set of features sufficient to differentiate that symbol from any other symbol in the alphabet, and it is only these essential, defining, double signs that are used in pattern-matching to select a phonetic symbol for output. After this selection is made, the values of the nonessential, single-sign features are compared, and if any of these values of the segment differ from the default values in the alphabet, the output word is highlighted in blue in the display. (Unfortunately Visual Basic does not permit individual symbols in the character string to be colored in contrast to others in the same string.) Specification about the discrepancy—which features of which segments differ from those segments' nearest equivalents in the alphabet—can be displayed as "feature-based diacritics" in a box at the top of the screen (see Burton-Hunter 1976 and Hartman 1981).

5. Rule notation.

Most linguists are familiar with a form of rule notation that can be represented schematically as "A > B / C _ D", meaning that element A becomes B in the environment

following C and preceding D, or, in other words, each instance of CAD becomes CBD. CAD is known as the "structural description", and the change of A to B is called the "structural change". Chomsky and Halle (1968) used this template extensively, along with signed feature names, to write (synchronic) phonological rules. Superficially, this notation seems simple, but it must be noted that many of Chomsky and Halle's rules are supplemented by additional devices such as parentheses, curly braces, angled brackets, "diacritic" features, category symbols with subscripts (e.g. $C_1 \ldots C_n$ for different consonants), "truth-functional conditions", etc. (1968:390-399). In order to capture these additional complexities that occur, Phono portrays the rules in a unique notation that is based on *if*-clauses and *then*-clauses that are composed mainly of binary feature values and expressions of locations in the word.

Specifically, Phono's rule notation system is based on four types of "IF-lines" (Branching, COUNT, Constant, and Variable) and five types of "THEN-lines" (Constant, Variable, DELETE, INSERT, and SWAP). (The COUNT, DELETE, INSERT, and SWAP line types are identified in the notation by these respective keywords in all-uppercase letters.) Since the Constant and Variable line types are able to carry out either IF or THEN functions, there are seven types altogether.

The Branching-type IF-line is the basis of the *hierarchy* of IF-lines: it joins the labels of two following IF-lines with a conjunction, either "and" or "or". If the rule has more than one IF-line, the first of them, 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 member of the 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.

The "Constant" and "Variable" line types, which can do double duty as both IF-lines and THEN-lines, are composed of feature values and references to locations in the word. As IF-lines they *detect*, and as THEN-lines they *alter*, selected feature values in the word. In a Constant-type line the feature names appear with specific values ("+" or "–"), while in a Variable-type line they detect or set the value of one feature equal (or opposite) to that of another — thus corresponding to the "Greek-letter variable" signs (the so-called "alpha" device) introduced in *SPE* (pp. 177-178).

Rules can express the location of a segment in the word either as an "absolute" location (i.e. with reference to the word-initial or -final positions) or as a "relative" location (i.e. with reference to the current focus segment of the scan). A Constant-type IF-line can be used to express conditions such as "If the word-initial segment (absolute location) is [+nasal]", or "If the segment immediately following the 'focus' segment (relative location) is [–continuant]". A Constant-type THEN-line, similarly, can be used to express changes such as "The word-final segment (absolute) becomes [–voice]", or "The focus segment (relative) becomes [+strident]".

A Variable-type IF-line seeks a feature value defined with regard to (i.e., the same as, or the opposite of) another feature value in the word. It can be used, for example, to

express a condition such as "If the [back] and [round] features (of a vowel) agree" — in order to restrict the change to the set of back-rounded and front-unrounded vowels. Used as a THEN-line, the Variable-type line appears in rules of assimilation (as well as in those of dissimilation), where the change of one feature may be to "+" or to "–" according to the value of a neighboring feature. So, for example, in a rule of voicing assimilation, it would be a Variable-type line that sets the value of the feature [voice] for one consonant equal to that of the same feature in the following consonant.

Most of the conditions for change (IF-lines) can be expressed by some combination of these Constant- and Variable-type lines based on feature values. But some changes depend additionally on some characteristic of the entire word (for example whether it has one syllable or more than one; or whether it contains a stressed syllable or not), or on a characteristic of some "subscan" of segments within the word (for example whether a certain feature value occurs in a segment located between the focus segment and a potential agent of change; or whether a vowel in focus is the last vowel in the word, regardless of how many consonants may follow it). These conditions are detected by means of the COUNT-type line, which specifies the "head" and the "foot" of the subscan, the feature value being sought, and the number of finds necessary to make the line "true".

Finally, some changes, rather than affecting merely some feature value(s) of the focus segment, may delete the entire segment, change its location in the word (metathesis), or insert an entire new segment. These whole-segment changes are brought about by the DELETE-, SWAP-, and INSERT-type THEN-lines, respectively.

Metathesis is treated as an interchange, a "swap", of positions of two segments (rather than as a movement by the focus segment some number of positions to the left or right) based on real changes such as that of Spanish *miraglo* > *milagro* ('miracle'), in which two non-adjacent segments are relocated simultaneously.

This system of rule notation has proved adequate to express the more-than-130 rules of the Spanish model. It has also supported a model to derive Shawnee from Proto-Algonquian, consisting of 21 rules (bin Muzaffar 1996 and 1997). Its main disadvantage arises from its unconventionality: it constitutes a considerable learning task for the new user. One challenge for future versions of Phono will be to alleviate the unfamiliarity of this rule-notation system. In this regard it remains to be seen whether it will be possible to devise algorithms for translating bi-directionally between the present if/then notation and something more akin to the standard "A > B / C _ D" rule format. If the standard notation proves not to be possible for all rules, then at least the writing of rules in the if/then notation can perhaps be facilitated by providing some automated guidance in the form of a structured questionnaire; and, conversely, the reading of this notation can be facilitated by means of a "prosifier", as was done in Version 3.2, which translates individual lines of rule notation into *if*- or *then*-clauses in English.

6. Deviations from linear rule order.

Although diachronic changes seem archetypally to follow one another in a constant order, the realism of the model can be enhanced by providing the possibility of exceptions to simple linear order of rules. The present version has the capability to label any rule as "persistent" and to temporarily "mask" other rules. Worth considering for incorporation in future versions would be the possibility of marking data words, either for exemption from specified rules, or for reversals of rule order.

6.1. Persistent rules.

In spite of the goal of remaining theory-neutral insofar as possible, I have adopted from the framework of generative phonology the notion of "persistent" rules. Sound-change rules are generally considered "transient", meaning that they act at one specific time in the chronology and never again. But experience with the Spanish model has supported the notion that some rules repeat their changes from the moment they are acquired for the remainder of the derivation, whenever their conditions occur. For this kind of rule, the term "persistent" was proposed and defined by Chafe (1968:131). Specifically, for example, the rule of nasal assimilation (to a following consonant) is apparently active at all moments throughout the history of Spanish, not only in Latin words (as partially reflected in the spellings *-mp-* and *-mb-*), but also centuries later in consonant clusters brought together by the deletion of vowels. For this and other such recurring rules, Phono provides the possibility to mark any diachronic rule as persistent. As the program traverses the diachronic list of rules in the derivation, it records the identities of the persistent rules in a separate list; then, after each subsequent rule that changes the word, it re-traverses the entire list of persistent rules.

6.2. Rule masking.

Phono's newest version provides the capability to temporarily disable, or "mask" any individual rule in a model, in order to observe how the output differs. In the case of the Spanish model, this tactic has helped to reveal that the form of some apparently exceptional words—those traditionally labeled "semi-learnèd", with implied attribution to the influence of written language—is due to their failure to undergo just one regular sound change. Being exempted from a single rule can set a word on a much different path from that of other words that previously were similar to it (see Hartman 1986).

6.3. Marking words for variable order?

Some investigators—beginning with Wang 1969—have suggested that sound changes may spread through the vocabulary gradually (by "lexical diffusion"), and that some changes may lose their momentum and cease to act, leaving part of the vocabulary unaffected. Whether for this reason or for the implied written influence on the semi-learnèd words noted above, it may be useful for sound-change modelers to have the capability, not merely to mask a rule temporarily, but rather to mark individual words permanently for exemption from specific rules.

Additionally, lexical diffusion—by acknowledging that the same change may affect different words in different historical epochs—opens the possibility of rules A and B affecting some words in AB order and others in BA order. If it is found that large numbers of words require variable order of application for certain pairs of rules, then it will be necessary in future versions of sound-change modelers to provide for marking data words individually for exceptional rule orders.

7. Batch testing.

One of the reasons for modeling sound change computationally is to observe it as a system, rather than just one or two rules at a time. And yet, during the development of a model, the experimenter may alter the model to accommodate one set of words only to find—eventually—that the alterations have ruined the accuracy of the model with regard to some other set of words. In order to monitor the integrity of the model throughout its development, it is important to test it at each step against a large vocabulary of pairs of words—etymon and known reflex—in order to insure that the derived results continue to coincide with the known reflexes. For this purpose, Phono is equipped with the capability to run in a "batch" mode, drawing the etymon/reflex pairs from the data file of the model. In the batch mode, the *etymon* words are encoded—translated from character strings to feature values—in the same way as for the interactive mode. Additionally, the input of known *reflexes* is, like etymon input, originally formed as strings of keyboard characters, and needs to be translated to feature values for comparison (the batch mode compares feature values, not character strings). So, analogously with the apparatus for etymon input, the reflex input for the batch mode is expressed in its own alphabet, and it may be adjusted by a set of "reflex adjustment rules" similar in function to the adjustment rules for etymon input.

In summary, the complete model includes three alphabets (keyboard alphabets for etymon input and for reflex input, and the phonetic alphabet for the output display); three rule sequencers (for etymon input adjustment, for reflex input adjustment, and for diachronic change); and one set of rules (a common "supply", upon which each of the three sequencers may draw).

8. Trace procedures.

Version 3 of Phono had the capability to run "rule trace" and "word trace" procedures based on the outcome of the batch test, and these will be incorporated into Version 4 in the near future. The rule trace keeps, for each rule, a record of all the words in which the rule effects a change. Such a record can be useful for measuring the value of each rule, insofar as this is based on its "functional load". Conversely, the word trace summarizes the history of each word as the list of rules that affect the word.

9. Limitations and future development.

The flexibility of Phono's rule-notation system is a double-edged sword. Its advantage is that it probably can portray virtually all sound changes of all natural languages (although it has yet to be tested with tone languages, vowel-harmony phenomena, or languages with discontinuous morphemes such as those of the Semitic family). On the other hand, a potential danger in such versatility is that the notation imposes no limits on the kinds or the complexity of rules that can be written: the investigator must still use human intuition to decide which rules are "natural" or plausible enough to be considered valid, and it is still a human decision how few example words (vs. how many counterexamples) are necessary to justify the formulation of a rule. Hopefully Phono and other programs like it can provide a basis for making these human judgments with more consistency and confidence.

References

Becker, Donald A. 1996. "Historical Linguistics as a Hacker's Paradise: Review of Phono 3.2". *Glot International*, 2:22.

Burton-Hunter, Sarah K. 1976. "Romance Etymology: A Computerized Model". *Computers and the Humanities*, 10:217-220.

Chafe, Wallace. 1968. "The Ordering of Phonological Rules". *International Journal of American Linguistics*, 34:115-136.

Chomsky, Noam, and Morris Halle. 1968. *The Sound Pattern of English*. New York: Harper & Row.

Hartman, Lee. 2003a. Phono: Historical Sound Change Modeler. Version 4.0-VB. Downloadable from http://mypage.siu.edu/lhartman.

Hartman, Lee. 2003b. "Phono (Version 4.0): Software for Modeling Regular Historical Sound Change". In Leonel Ruiz Miyares, Celia E. Álvarez Moreno, and María Rosa Álvarez Silva (eds.), *Actas: VIII Simposio Internacional de Comunicación Social: Santiago de Cuba, 20-24 de Enero del 2003*, I, 606-609.

Hartman, Steven Lee. 1974. "An Outline of Spanish Historical Phonology". *Papers in Linguistics*, 7:123-191.

_____. 1981. "A Universal Alphabet for Experiments in Comparative Phonology". *Computers and the Humanities*, 15:75-82.

_____. 1986. "Learnèd Words, Popular Words, and 'First Offenders'". In Oswaldo Jaeggli and Carmen Silva-Corvalán (eds.), *Studies in Romance Linguistics* (Dordrecht: Foris), pp. 87-98.

_____. 1993a. "Three Problems of Notation in Modeling Sound Change". Paper presented at Round Table on Computer Applications in Historical Linguistics, Brussels, Belgium, December 8.

_____. 1993b. "Writing Rules for a Computer Model of Sound Change". In *Southern Illinois Working Papers in Linguistics and Language Teaching*, 2:31-39.

Muzaffar, Towhid bin. 1996b. "Computer Simulation of Shawnee Historical Phonology". In *Canadian Linguistic Association Annual Conference Proceedings* (Calgary: Calgary Working Papers in Linguistics), pp. 293-303.

_____. 1997. "Computer Simulation of Shawnee Historical Phonology". M.A. thesis, Memorial University of Newfoundland.

Otero, Carlos-Peregrín. 1971. *Evolución y revolución en romance.* Barcelona: Seix Barral.

Wang, William S-Y. 1969. "Competing Changes as a Cause of Residue". *Language*, 45:9-25.