Chapter 3

Transformations, Logically

Jeffrey Heinz

This chapter explains how transformations from one representation to another can be described with the same logical tools introduced in the last chapter. Transformations are a central component of phonological theory, which posits a transformation exists between the abstract mental representations of the pronunciation of morphemes (the underlying form) to the more concrete, more directly observable, surface representation (the surface form). The mathematical and computational basis for this work is summarized in Courcelle (1994).

This chapter aims to introduce these ideas in an accessible way to linguists with a basic knowledge of phonology. However, the techniques have application beyond the theory of phonology to any other subfield of linguistics, notably morphology and syntax, in part because these methods apply equally well to trees and graphs, not just strings. Also this chapter is merely an introduction to these methods. As such, it introduces them in the context of string-to-string transformations; that is, *functions* from strings to strings. As a matter of fact, these methods have been generalized by computer scientists to describe *weighted relations* between strings . These generalizations permit one to describe and characterize optionality and exceptionality, in addition to gradient and probabilistic generalizations. Such generalizations are not discussed until the end of the book (Chapter XYZ) for two reasons. First they are unnecessarily complicate the central ideas, which are easier to first understand without them. Second, much valuable work can be done without them, as exemplified by the chapters in parts 2 and 3 of this book.

The application of these methods for phonological description and theory is what primarily distinguishes this work from One-Level Declarative Phonology developed by Bird, Coleman, and Scobbie twenty five years ago. That research, like the research in this book, emphasizes the importance of a declarative approach to phonological description and theory. The key difference is that twenty-five years ago transformations were studied within "one level." In other words, transformations were understood as *constraints* on unspecified representations. As such, those 'transformations' could only *add* (further specify) information to representations. In contrast, in this chapter we will see how logic can be used to literally add, subtract, change, or more generally *transform* one representation into another. For this reason, one could say that the Computational Generative Phonology apperoach in this book is essentially a form of *Two-level* Declarative Phonology.

3.1 Strings-to-string Transformations

A logical description of a string-to-string function uses logic to explain how to an input string is mapped to an output string. As with the constraints in the previous chapter, the logic does not operate over the strings themselves, but over the model-theoretic representation of those strings. Therefore, a logical description of a string-to-string function uses logic to convert an input *model* of a string into another *model* (of a possibly different string). Recall that the model of a string is understood in terms of its *signature*. The signature includes the relations over the domain of the model that must be specified in order to uniquely identify some string. Therefore, the logical description needs to specify the domain and relations in the *output model* in terms of the the domain and relations in the *input model*.

A logical description of a function specifies the domain of the function, and for each input, it must specify the domain of the output model and the relations over it with logical formulas, interpreted with respect to the input model.[1] The domain of the function is specified by the *domain formula*. The domain of the output model is specified by three ingredients: the *copy set*

[1]Note there are two distinct meanings of the word 'domain' in use here. The first has to do with the domain of a *function* and the second with the domain of a *model*. A function's domain is the set of elements over which the function is defined. For instance for $F : A \rightarrow B$, the domain is the set A. In contrast, the domain of a model is the elements in the 'universe' the model is describing. In finite model theory, which is used in this book, the domain of a model is a finite set of natural numbers $1, \ldots n$, representing the finitely

and the *licensing formula*. The relations over the output model are specified by *relational formulas* for each of the relations in the signature of the output model. These formulas are evaluated with respect the *input model* in a way that will be made clear below. In our first examples, we leave out the copy set and return to it in section 3.2.

3.1.1 Word-final obstruent devoicing

For concreteness, let us provide a logical description of the phonological process of word-final obstruent devoicing. This process maps strings with word-final voiced obstruents to voiceless ones. For example, this process maps the string *hauz* to *haus* and the string *bad* to *bat*. Words without word-final voiced obstruents surface faithfully so this process also maps the string *haus* to *haus*.

We choose to model this process with the feature-based successor model described in 2.5 (see Table 2.4). Strings in both the input and the output will be represented with the feature-based successor model. Note this is a choice. One can choose to model the input, the output, or both, will some other word model, such as the conventional word model with successor. It follows we want to provide a logical transformation which maps the model of *hauz* to the model of *haus*, as shown in Figure 3.1. We introduce the logical formulas one at a time and then summarize them at the end of the example.

We must specify the domain of the function f with a logical formula with no free variables $\phi_{\texttt{domain}}$. For a string w, the function f is defined if and only if its model $\mathcal{M}_w \models \phi_{\texttt{domain}}$. In this case, we want word-final obstruent devoicing to apply to every string. Hence we let $\phi_{\texttt{domain}} \overset{\text{def}}{=} \texttt{true}$.

How is the domain of the output model of $f(\mathcal{M}_w)$ determined? Logical transductions fix the domain of the output as a *copy* of the input domain. For example, as shown in Figure **??**, the domain of $\mathcal{M}_{hauz}$ is {1,2,3,4}. Therefore, the domain of $f(\mathcal{M}_{hauz})$ is also {1,2,3,4}. An immediate consequence is that it appears that functions cannot alter the size of the input upon which they are acting. However, it is precisely the copy set (section 3.2) and the licensing formula $\phi_{\texttt{license}}$ (3.1.2) which determines the size of the output model. Basically, the copy set allows transformations to relate *larger* outputs to *smaller* inputs and the licensing formula (section **??**) allows transformations to relate *smaller* outputs to *larger* inputs. Working together these ingredients

many elements in the universe.

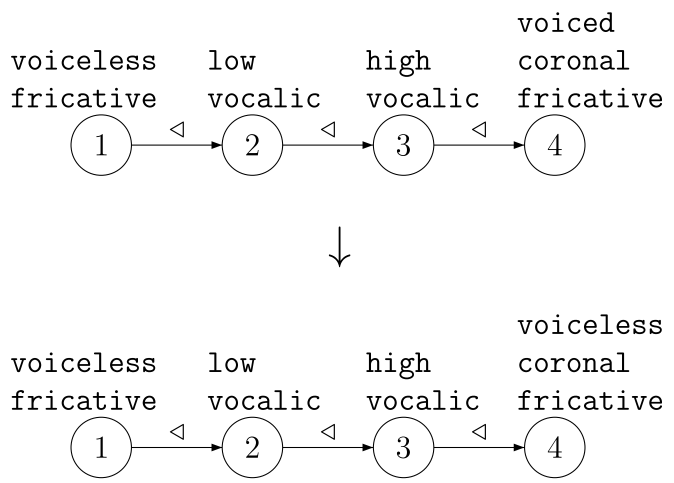


Figure 3.1: A graphical diagram of the feature-based successor model of *hauz* being mapped to the feature-based successor model of *haus*.

let one relate any input model of size n to an output model of size m, for any $n, m \in \mathbb{N}$. For now, since the word-final voiced obstruent devoicing *does preserve* the size of each input in the output, we set aside the copy-set and licensing formula for now.[2]

Thus given $\mathcal{M}_{hauz}$, f sets the domain of the output model to {1,2,3,4}. Finally, we must determine the relations which hold over these elements. For each relation R of arity n in the signature of output model, we must specify a formula $\phi_{\mathtt{R}}$ with n free variables $\phi_{\mathtt{R}}(x_1, \ldots x_n)$. Since the signature of the output model has one binary relation (the successor relation $\triangleleft$) and several unary features (the phonological features), we need a formula for each of these.

How are these formula interpreted? As follows. For any string-to-string

[2]For completeness, in this case the copy set $C = \{1\}$ and the $\phi_{\mathtt{license}} \overset{\text{def}}{=} \mathtt{true}$.

function f and input model $\mathcal{M}_w$, the elements $x_1, \ldots x_n$ in the domain of the output model $f(\mathcal{M}_w)$ stand in the n-ary relation R in the output signature if and only if $\phi_{\mathtt{R}}(x_1, \ldots x_n) \models \mathcal{M}_w$.

For example, the successor relation is a binary relation in the output signature. So we must define $\phi_{\triangleleft}(x, y)$. Since word-final obstruent devoicing does not affect the successor relations, we define this function as follows.

$$\underbrace{\phi_{\triangleleft}(x, y)}_{\text{Do x and y in the output model stand in the successor relation?}} \overset{\text{def}}{=} \underbrace{x \triangleleft y}_{\text{Evaluate with respect to the input model.}}$$

This means the following: elements x and y *in the output* stand in the successor relation if and only if corresponding elements x and y *in the input* satisfy the successor relation in the input model. Since $1 \triangleleft 2$ in the input model, it follows that elements 1 and 2 likewise stand in the successor relation in the output model. Similarly, since elements 1 and 3 *do not stand* in the successor relation in the input model, it follows that they do not stand in the successor relation in the output model. Consequently, the formula above guarantees (in fact literally says) that the successor relation in the output will be the same as the successor relation in the input.

As another example, consider the unary relation `vocalic`. As this is a unary relation, we must define a formula with one free variable $\phi_{\mathtt{vocalic}}(x)$. Let us define it as follows.

$$\underbrace{\phi_{\mathtt{vocalic}}(x)}_{\text{Does x have the feature \texttt{vocalic} in the output model?}} \overset{\text{def}}{=} \underbrace{\mathtt{vocalic(x)}}_{\text{Evaluate with respect to the input model.}}$$

It follows from this definition that domain element x in the output model is vocalic if and only if the corresponding domain element x in the input is vocalic. Thus, as we expect word final obstruent devoicing does not affect the vocalic nature of elements within a string.

As we know, the only features affected by word-final devoicing are *voicing* features, which in our model are `voiced` and `voiceless`. All other unary relations in the signature of the output model will be defined similarly to $\phi_{\mathtt{vocalic}}(x)$ (as shown in Table 3.1 on page 57). However, the voicing features are affected by this process, so how do we specify which domain elements are voiced or voiceless? The voiced elements will be the ones that were voiced in the input and are not the ones which are word-final obstruents. We can formalize this as follows. It will be useful to write some user-defined predicates.

$$\textsf{word-final}(x) \overset{\text{def}}{=} \neg\exists y\ (x \lhd y) \tag{3.1}$$

$$\textsf{obstruent}(x) \overset{\text{def}}{=} \texttt{stop}(x) \vee \texttt{fricative}(x) \tag{3.2}$$

We thus define $\phi_{\texttt{voiced}}(x)$ as follows.

$$\underbrace{\phi_{\texttt{voiced}}(x)}_{\text{Does x have the feature } \texttt{voiced} \text{ in the output model?}} \overset{\text{def}}{=} \underbrace{\texttt{voiced(x)} \wedge \neg(\textsf{word-final}(\texttt{x}) \wedge \textsf{obstruent}(\texttt{x}))}_{\text{Evaluate with respect to the input model.}}$$

Similarly, the domain elements in the output which are voiceless are those that are voiceless in the input or those that are word-final obstruents.

$$\underbrace{\phi_{\texttt{voiceless}}(x)}_{\text{Does x have the feature } \texttt{voiceless} \text{ in the output model?}} \overset{\text{def}}{=} \underbrace{\texttt{voiceless(x)} \vee (\textsf{word-final}(\texttt{x}) \wedge \textsf{obstruent}(\texttt{x}))}_{\text{Evaluate with respect to the input model.}}$$

For completeness, we show the complete logical description of word-final devoicing.

3.1.2 Word-final vowel deletion

Let us consider another example, word-final vowel deletion, which will illustrate the role played by the licensing formula. Word-final vowel deletion has been argued to be a process in Yawelmani Yokuts. It essentially maps strings like *paka* to *pak* and *pilot* to *pilot*.

As before, the domain of this function is all strings and so $\phi_{\texttt{domain}} \overset{\text{def}}{=} \texttt{true}$. Also as before, the domain of the output model is a copy of the domain elements of the input model. However, these domain elements of the output model do not automatically exist in the output model; they must be *licensed* by a formula with one free variable called the licensing formula $\phi_{\texttt{license}}(x)$. In other words, the domain elements of the output model are really the *licensed* copies of the domain elements of the input model. Since word-final

$\phi_{\texttt{domain}}$	$\overset{\text{def}}{=}$	$\texttt{true}$
C	$\overset{\text{def}}{=}$	$\{1\}$
$\phi_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\texttt{true}$
$\phi_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$x \triangleleft y$
$\phi_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\texttt{low}(x)$
$\phi_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{high}(x)$
$\phi_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{front}(x)$
$\phi_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiced}(x) \wedge \neg(\textsf{word-final}(x) \wedge \textsf{obstruent}(x))$
$\phi_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiceless}(x) \vee (\textsf{word-final}(x) \wedge \textsf{obstruent}(x))$

Table 3.1: The complete logical specification for word-final obstruent devoicing when the input and output string models are both the feature-based successor model.

vowels delete in this process, all domain elements which do not correspond to word-final vowels are licensed.

$$\underbrace{\phi_{\texttt{license}}(x)}_{\text{Does x belong to the domain of the output model?}} \overset{\text{def}}{=} \underbrace{\neg(\textsf{word-final}(x) \wedge \texttt{vocalic}(x))}_{\text{Evaluate with respect to the input model.}}$$

Since this process does not affect the phonological features in the string, each of the unary relations R in the signature of the output model can be

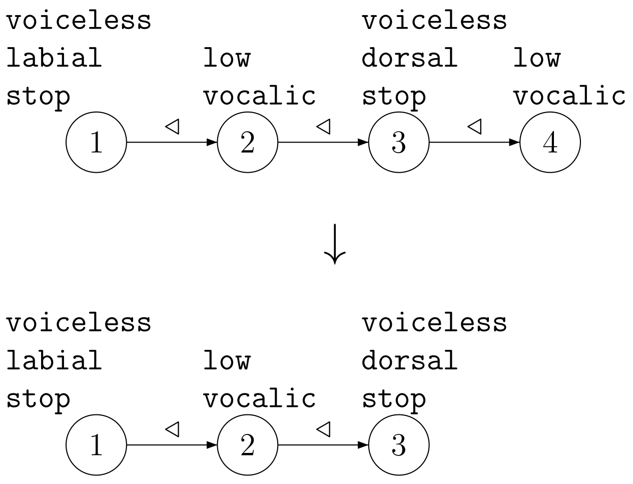


Figure 3.2: A graphical diagram of the feature-based successor model of *paka* being mapped to the feature-based successor model of *pak*.

defined as follows: $\phi_{\mathtt{R}}(x) \overset{\text{def}}{=} \mathtt{R}(x)$. In other words, $\phi_{\mathtt{vocalic}}(x) \overset{\text{def}}{=} \mathtt{vocalic}(x)$ and $\phi_{\mathtt{voiced}}(x) \overset{\text{def}}{=} \mathtt{voiced}(x)$ and so on. What about the binary successor relation? Letting $\phi_{\triangleleft}(x, y) \overset{\text{def}}{=} x \triangleleft y$ is sufficient. While it is true that $3 \triangleleft 4$ is true in the input, the fact that 4 is not licensed is sufficient to ensure that the pair $(3, 4)$ is not an element of the successor relation in the output model. The relations in the output model are always *restricted* to tuples which only contain *licensed* domain elements.

For completeness, Table 3.2 shows the complete logical description of word-final vowel deletion.

This section explained in more detail how the domain elements of the output model are determined. While these are always copies of the domain elements of the input model, it is not the case that every domain element in the input model is always copied as a domain element of the output model.

$\phi_{\texttt{domain}}$	$\overset{\text{def}}{=}$	$\texttt{true}$
C	$\overset{\text{def}}{=}$	$\{1\}$
$\phi_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\neg(\textsf{word-final}(x) \wedge \texttt{vocalic}(x))$
$\phi_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$x \triangleleft y$
$\phi_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\texttt{low}(x)$
$\phi_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{high}(x)$
$\phi_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{front}(x)$
$\phi_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiced}(x)$
$\phi_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiceless}(x)$

Table 3.2: The complete logical specification for word-final obstruent devoicing when the input and output string models are both the feature-based successor model.

Only those elements x which satisfy $\phi_{\texttt{license}}(x)$ become domain elements in the output model.

3.2 Getting Bigger

So far we have exemplified logical transductions with phonological processes that change segmental material and processes that delete segmental material.

How can logical transductions be used to define processes that *add* segmental material?

The answer to this question lies in the copy set. We have set aside this ingredient until now. In the previous examples, the copy set contained only *one* element. Thus each input element in the domain was copied exactly *once*. More generally, the copy set may contain n elements. It follows that the domain of the output model may contain n copies of *each* domain element of the input model. The copies of a domain element x in the input model are distinguished from each other using the names of the elements in the copy set. For example, consider the word *hauz* so that the domain elements of $\mathcal{M}_{hauz}$ are $\{1, 2, 3, 4\}$. If we are defining a logical transduction and define the copy set $C \overset{\text{def}}{=} \{1, 2\}$ then there are as many as *eight* domain elements in the output structure. It is customary to name these domain elements as *pairs*; the first coordinate indicates the domain element in the input model being copied and the second coordinate indicates *which* copy. Thus the pair $(1, 2)$ indicates the second copy of the first domain element of the input model and $(3, 1)$ indicates the first copy of the third element and so on. The eight possible domain elements in the output model are thus $\{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 1), (4, 2)\}$.

Whenever the copy set contains more than one element, the number of licensing formulas and relational formulas needed to describe the logical transduction multiplies as well. For each $i \in C$, there is a licensing formula $\phi^i_{\texttt{license}}(x)$. As before, this formula is evaluated with respect to the corresponding domain element in the input model. If it evaluates to true on x then the domain element (x, i) is licensed and belongs to the domain of the output model. Thus for a copy set C, there are $|C|$ licensing formulas.

Similarly, for each unary relation R in the signature of the output model, there are $|C|$ relational formulas: for each $i \in C$, we must define $R^i(x)$. The domain element (x, i) — the ith copy of x in the output model — belongs to R in the output model if and only if $R^i(x)$ evaluates to true in the input model.

For each binary relation R in the output signature, there are $|C|^2$ relational formulas $R^{i,j}(x, y)$ with $i, j \in C$. If and only if $R^{i,j}(x, y)$ evaluates to true with respect to the input model then the ith copy of x stands in the R relation to the jth copy of y in the output model. In which case, we have $((x, i), (y, j)) \in R$. If $R^{i,j}(x, y)$ evaluates to false with respect to the input model then $((x, i), (y, j))$ does not belong to R. For relations of higher arity, the licensing and relational formula multiply out similarly. Since the word models developed so far involve at most binary relations, we ignore relations of higher arity here (though they are discussed in the mathematical appendix **??**).

How the copy set works along with the additional formulas it entails are illustrated next with word-final vowel epenthesis and total reduplication. We provide complete logical descriptions of these transformations.

3.2.1 Word-final vowel epenthesis

Hindi speakers epenthesize the low vowel *a* to words which end in sonorant consonants (Shukla, 2000). We provide a logical description of this process given the the segments describable with the feature-based model. For example, this process would map the hypothetical word *pan* to *pana* as well as *pak* to *pak*. Figure 3.3 visualizes the mapping between the model structures *pan* and *pana*.

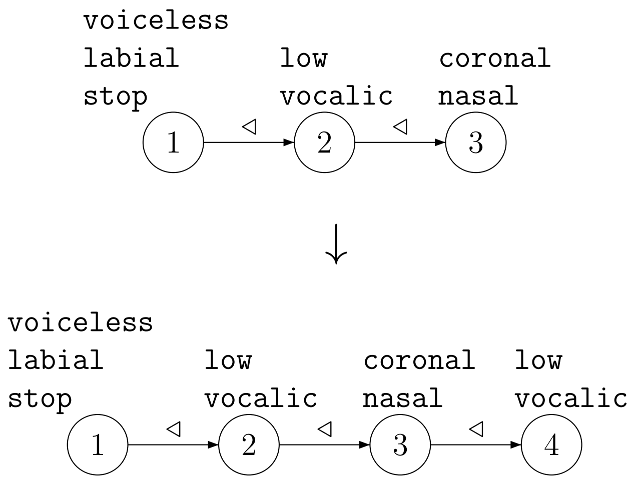


Figure 3.3: A graphical diagram of the feature-based successor model of *pan* being mapped to the feature-based successor model of *pana*.

First we can define sonorant consonants as follows.

$$\mathsf{sonorant\_C}(x) \overset{\text{def}}{=} \texttt{nasal}(x) \vee \texttt{lateral}(x) \vee \texttt{rhotic}(x) \tag{3.3}$$

Next, we need a copy set of at least size 2 and so we define $C \overset{\text{def}}{=} \{1, 2\}$. Consequently, for the input *pan* which has three domain elements $\{1, 2, 3\}$, there are maximally 6 domain elements in the output model: $\{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2)\}$. Since the copy set C has two elements, we must define two licensing formula, each with one free variable.

$$\phi^1_{\texttt{license}}(x) \overset{\text{def}}{=} \texttt{true} \tag{3.4}$$

$$\phi^2_{\texttt{license}}(x) \overset{\text{def}}{=} \mathsf{sonorant\_C}(x) \wedge \mathsf{word\text{-}final}(x) \tag{3.5}$$

$\phi^1_{\texttt{license}}(x)$ is always true so the first copy of each element is present. $\phi^2_{\texttt{license}}(x)$ is only true when $\mathsf{sonorant\_C}(x) \wedge \mathsf{word\text{-}final}(x)$ evaluates to true in the input model. For the word *pan* this occurs for $x = 3$, but for the word *pak* no x satisfies $\phi^2_{\texttt{license}}(x)$. Consequently, the output model of the process applied to *pan* has four domain elements $\{(1,1),(2,1),(3,1),(3,2)\}$ but the the output model of the process applied to *pak* has three domain elements $\{(1,1),(2,1),(3,1)\}$.

This is illustrated in Figure 3.4, where the first and second copies of the domain elements of *pan* are arranged in rows and the unlicensed elements are in gray.

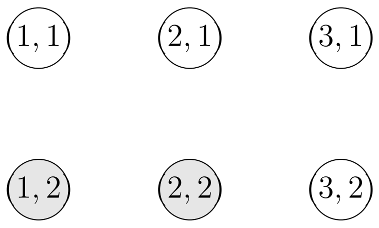


Figure 3.4: The possible domain elements of the output model for input *pan* when the copy set $C \overset{\text{def}}{=} \{1,2\}$. The unlicensed elements are colored gray.

Next, we turn to the binary successor relation in the output model. Here, we must have four formulas to specify the successor relation in the output signature. We define these as follows

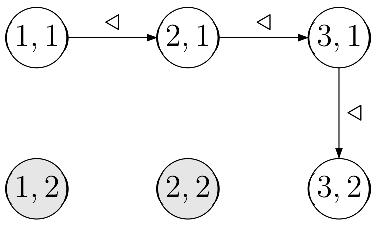
$$\phi^{1,1}_{\triangleleft}(x,y) \overset{\text{def}}{=} x \triangleleft y \tag{3.6}$$

$$\phi^{1,2}_{\triangleleft}(x,y) \overset{\text{def}}{=} \mathsf{sonorant\_C}(x) \wedge \mathsf{word\text{-}final}(x) \wedge \mathsf{word\text{-}final}(y) \tag{3.7}$$

$$\phi^{2,1}_{\triangleleft}(x,y) \overset{\text{def}}{=} \texttt{false} \tag{3.8}$$

$$\phi^{2,2}_{\triangleleft}(x,y) \overset{\text{def}}{=} \texttt{false} \tag{3.9}$$

Consequently, the successor relations are preserved among the first copy of the domain elements and the only successor from an element of the first copy to an element of the second copy is satisfied when x satisfies both $\mathsf{word\text{-}final}(x)$ and $\mathsf{sonorant\_C}(x)$.



$$\begin{tikzcd}
(1,1) \arrow[r, "\triangleleft"] & (2,1) \arrow[r, "\triangleleft"] & (3,1) \arrow[d, "\triangleleft"] \\
(1,2) & (2,2) & (3,2)
\end{tikzcd}$$

Figure 3.5: The successor relations in the output model for input *pan* when the copy set $C \overset{\text{def}}{==} \{1, 2\}$. The unlicensed elements are colored gray.

Finally, we must define two formulas for each unary relation R in the output signature, $\phi^1_{\mathtt{R}}(x)$ and $\phi^2_{\mathtt{R}}(x)$. For each unary relation R, we define $\phi^1_{\mathtt{R}}(x) \overset{\text{def}}{=} R(x)$. Thus, the first copy of the domain elements are faithful to the unary relations they satisfied in the input. For the second copy, we can also let the domain elements be faithful to the unary relations they satisfied in the input with two exceptions. In our model,the low vowel a is low and vocalic and so $\phi^2_{\mathtt{vocalic}}(x)$ and $\phi^2_{\mathtt{low}}(x)$ must be defined to be true only when x corresponds to an element in the input that satisfies $\mathsf{sonorant\_C}(x)$ and $\mathsf{word\text{-}final}(x)$. For other unary relations R, we can define $\phi^2_{\mathtt{R}}(x) \overset{\text{def}}{=} \mathtt{false}$.

For completeness, Table 3.3 shows the complete logical description of word-final vowel epenthesis.

3.2.2 Duplication

Here we provide another example of a logical transduction, total reduplication. Obviously, we set the copy set $C \overset{\text{def}}{=} \{1, 2\}$. Then we essentially make all unary relations be faithful to their input so for all unary relations R in the output signature we have $\phi^1_{\mathtt{R}}(x) = \phi^2_{\mathtt{R}}(x) \overset{\text{def}}{=} R(x)$. As for the successor relation, two elements (x, i) and (y, j) stand in the successor relation if only if either $i = j$ and $x \triangleleft y$ in the input model or $i = 1$ and $j = 2$ and x is word-final in the input and y is word-initial in the input. We define $\mathsf{word\text{-}initial}(x)$ as follows.

$$\mathsf{word\text{-}initial}(x) \overset{\text{def}}{=} \neg\exists y\ (y \triangleleft x) \tag{3.10}$$

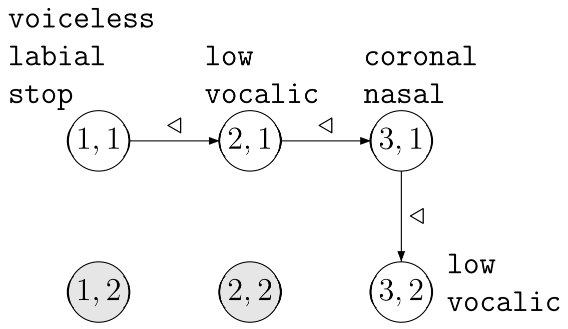


Figure 3.6: The model representing *pana* which is output for the input *pan*. The unlicensed elements are colored gray.

To illustrate, Figure 3.7 shows the output model for the input *pan*. In this way the copy set and the logical make it straightforward to define total reduplication.

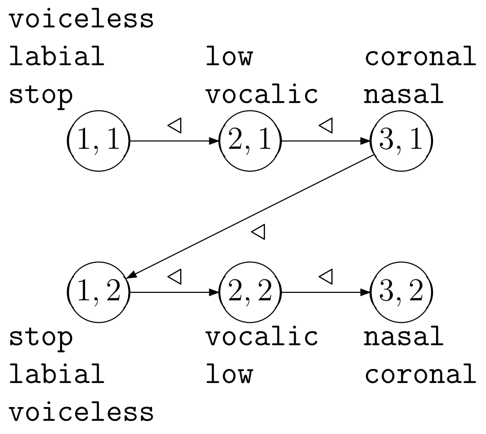


Figure 3.7: The model representing *panpan* is the output of the input *pan* to the process of reduplication.

For completeness, Table 3.4 shows the complete logical description of total reduplication.

$\phi_{\texttt{domain}}$	$\overset{\text{def}}{=}$	$\texttt{true}$	C	$\overset{\text{def}}{=}$	$\{1,2\}$
$\phi^1_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\texttt{true}$	$\phi^2_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\textsf{sonorant\_C}(x) \wedge \textsf{word-final}(x)$
$\phi^{1,1}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$x \triangleleft y$	$\phi^{1,2}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$\textsf{sonorant\_C}(x) \wedge \textsf{word-final}(x) \wedge \textsf{word-final}(y)$
$\phi^{2,1}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$\texttt{false}$	$\phi^{2,2}_{\texttt{succ}}(x,y)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{vocalic}(x)$	$\phi^2_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\textsf{sonorant\_C}(x) \wedge \textsf{word-final}(x)$
$\phi^1_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\texttt{low}(x)$	$\phi^2_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\textsf{sonorant\_C}(x) \wedge \textsf{word-final}(x)$
$\phi^1_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{high}(x)$	$\phi^2_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{front}(x)$	$\phi^2_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{stop}(x)$	$\phi^2_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{fricative}(x)$	$\phi^2_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{nasal}(x)$	$\phi^2_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{lateral}(x)$	$\phi^2_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{rhotic}(x)$	$\phi^2_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{labial}(x)$	$\phi^2_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{coronal}(x)$	$\phi^2_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{dorsal}(x)$	$\phi^2_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiced}(x)$	$\phi^2_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$
$\phi^1_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiceless}(x)$	$\phi^2_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{false}$

Table 3.3: The complete logical specification for word-final vowel epenthesis when the input and output string models are both the feature-based successor model.

3.2.3 Summary

At this point, we have covered how to define transformations logically. One remarkable aspect about these methods is that these can be used for different representations. We will see this is the case

$\phi_{\texttt{domain}}$	$\overset{\text{def}}{=}$	$\texttt{true}$	C	$\overset{\text{def}}{=}$	$\{1,2\}$
$\phi^1_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\texttt{true}$	$\phi^2_{\texttt{license}}(x)$	$\overset{\text{def}}{=}$	$\texttt{true}$
$\phi^{1,1}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$x \triangleleft y$	$\phi^{1,2}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$\mathsf{word\text{-}final}(x) \wedge \mathsf{word\text{-}initial}(y)$
$\phi^{2,1}_{\triangleleft}(x,y)$	$\overset{\text{def}}{=}$	$\texttt{false}$	$\phi^{2,2}_{\texttt{succ}}(x,y)$	$\overset{\text{def}}{=}$	$x \triangleleft y$
$\phi^1_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{vocalic}(x)$	$\phi^2_{\texttt{vocalic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{vocalic}(x)$
$\phi^1_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\texttt{low}(x)$	$\phi^2_{\texttt{low}}(x)$	$\overset{\text{def}}{=}$	$\texttt{low}(x)$
$\phi^1_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{high}(x)$	$\phi^2_{\texttt{high}}(x)$	$\overset{\text{def}}{=}$	$\texttt{high}(x)$
$\phi^1_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{front}(x)$	$\phi^2_{\texttt{front}}(x)$	$\overset{\text{def}}{=}$	$\texttt{front}(x)$
$\phi^1_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{stop}(x)$	$\phi^2_{\texttt{stop}}(x)$	$\overset{\text{def}}{=}$	$\texttt{stop}(x)$
$\phi^1_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{fricative}(x)$	$\phi^2_{\texttt{fricative}}(x)$	$\overset{\text{def}}{=}$	$\texttt{fricative}(x)$
$\phi^1_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{nasal}(x)$	$\phi^2_{\texttt{nasal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{nasal}(x)$
$\phi^1_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{lateral}(x)$	$\phi^2_{\texttt{lateral}}(x)$	$\overset{\text{def}}{=}$	$\texttt{lateral}(x)$
$\phi^1_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{rhotic}(x)$	$\phi^2_{\texttt{rhotic}}(x)$	$\overset{\text{def}}{=}$	$\texttt{rhotic}(x)$
$\phi^1_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{labial}(x)$	$\phi^2_{\texttt{labial}}(x)$	$\overset{\text{def}}{=}$	$\texttt{labial}(x)$
$\phi^1_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{coronal}(x)$	$\phi^2_{\texttt{coronal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{coronal}(x)$
$\phi^1_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{dorsal}(x)$	$\phi^2_{\texttt{dorsal}}(x)$	$\overset{\text{def}}{=}$	$\texttt{dorsal}(x)$
$\phi^1_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiced}(x)$	$\phi^2_{\texttt{voiced}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiced}(x)$
$\phi^1_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiceless}(x)$	$\phi^2_{\texttt{voiceless}}(x)$	$\overset{\text{def}}{=}$	$\texttt{voiceless}(x)$

Table 3.4: The complete logical specification of total reduplication when the input and output string models are both the feature-based successor model.

3.3 Power of MSO-definable Transformations

What other kinds of transformations can be described with logical transformations? As the astute reader may no doubt have already gathered, many phonologically or morphologically *unnatural* processes are also easy to describe with logical transformations. As explained more detail in the next chapter, this is a strength, not a weakness, of the formal methods advocated here. Basically, the formal methods do not constitute a *theory* of phonology;

rather, they constitute a *meta-language* in which theories of phonology can be stated.

In this section, however, we simply wish to establish the fact that two unnatural processes – string mirroring and sorting – also permit logical descriptions.

3.3.1 Mirroring

String mirroring is a process that takes any string w as input and outputs ww^r where w^r is the reverse of the string w. For example if the string *pan* is submitted to the mirroring process, then the output would be *pannap*. Similarly, if *paka* were input to the mirroring process, the output would be *pakaakap*.

This can be modeled with a logical transduction that is nearly identical to the one for total reduplication. The unary relations are defined in the same way. The only differences lie in two of the formulas for the successor relation in the output model. Specifically we define $\phi_{\triangleleft}^{1,1}(x,y)$ and $\phi_{\triangleleft}^{2,1}(x,y)$ as before. However, $\phi_{\triangleleft}^{2,2}(x,y) \overset{\text{def}}{=} y \triangleleft x$, which essentially reverses the successor relations in the second copies of the domain elements. Also, $\phi_{\triangleleft}^{1,2}(x,y) \overset{\text{def}}{=} \mathsf{word\text{-}final}(x) \wedge \mathsf{word\text{-}final}(y)$. Thus mirroring places the copies of the element which is word-final in the input model into the successor relation. Figure 3.8 shows the output model of the string *pannap* that is produced by this logical description of string mirroring given the input *pan*.

3.3.2 Sorting

String sorting is a process that takes any string as input and outputs a string of the same length where the symbols are sorted in leixographic order. For instance if the input string is *paka* the output string would be *aakp*. Similarly, if the input string was *banapi* the output string would be *aabinp*. While, we can do this for any word model of strings, we will assume an alphabet Σ and a conventional precedence model for strings (see section 2.7) for convenience. We also assume that the alphabet is totally ordered under some lexicographic order ($<_\ell$).

Then sorting can be modeled with a logical transduction as follows. We let the copyset $C = \Sigma$. This may seem unusual, but what we are saying is that we make as many copies as there are letters in the alphabet. Then for

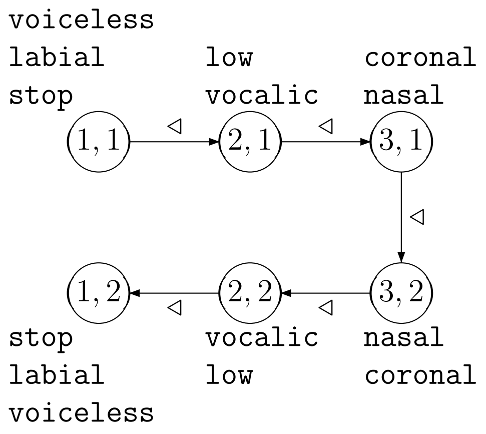


Figure 3.8: The model representing *pannap* is the output of the input *pan* to the process of mirroring.

each $a \neq b \in \Sigma$ define the licensing formulas and the relational formulas as follows.

- $\phi^{b}_{\mathtt{a}}(x) \overset{\text{def}}{=} \mathtt{a}(x)$
- $\phi^{a,a}_{<}(x,y) \overset{\text{def}}{=} x < y$
- $\phi^{a,b}_{<}(x,y) \overset{\text{def}}{=} \mathit{true}$ whenever $a <_\ell b$ and `false` otherwise
- $\phi^{a}_{\mathtt{license}}(x) \overset{\text{def}}{=} \mathtt{a}(x)$

The first item defines all the unary relations in the output (for each $a \in \Sigma$). It says that each copy of a domain element in the input which belonged to the unary relation `a` also belongs to `a` in the output.

The second item defines the binary precedence relations for domain elements in the output that belong to the same copy. In this case, domain elements (x, a) and (y, a) stand int the precedence relation in the output only if $x < y$ in the input. This ensures the familiar left-to-right ordering among elements, at least within a copy.

The third item defines the binary precedence relations for domain elements in the output that belong to different copies. The basic idea is that alhabetically earlier copies will precede alphabetically later ones. Recall we are defining multiple formulas $\phi^{a,b}_{<}(x,y)$ for each $a \neq b \in \Sigma$. Whenever $a <_\ell b$

(a is alphabetically earlier than b) we let $\phi_<^{a,b}(x,y) \overset{\text{def}}{=}$ `true`. Whenever $b <_\ell a$, we let $\phi_<^{a,b}(x,y) \overset{\text{def}}{=}$ `false`.

Finally, we get to the licensing formulas, of which there will be $|\Sigma|$. We define these formulas so that only those domain elements that belong to the unary relation a in the ath copy are licensed. Everything else is unlicensed. Recall that relations in the output model are restricted to the licensed domain elements.

Figure 3.9 illustrates this construction when the input is *paka*.

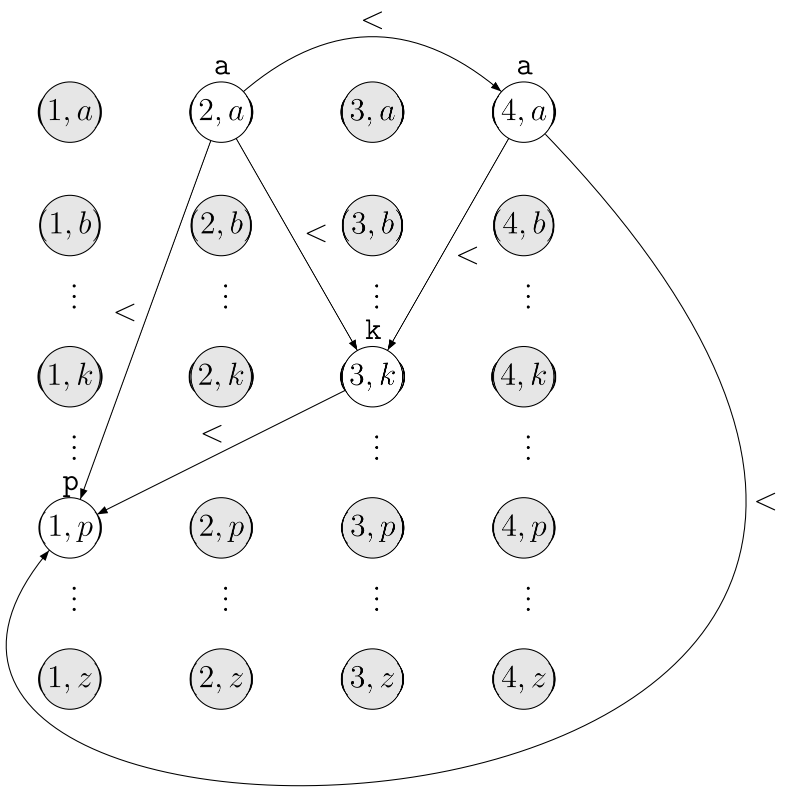


Figure 3.9: The model representing the output *aakp* of the input *paka* to the process of sorting with all potential domain elements shown. Unlicensed elements are in gray.

3.4 Conclusion

This chapter has explained how transformations can be expressed logically between model-theoretic representations. The signature of the output model determines the relational formulas that need to be defined.