Chapter 2

Representations, Models, and Constraints

Jeffrey Heinz and James Rogers

2.1 Logic and Constraints in Phonology

In this chapter, we show how to use logic and model-theoretic representations to define constraints on the well-formedness of those representations. The power in this kind of computational analysis comes from the framework's flexibility in both the kind of logic used and the choice of representation.

As will be explained, those choices provides a "Constraint Definition Language" (CDL) in the sense of (de Lacy, 2011). Each CDL has psychological, typological, and learnability ramifications which can be carefully studied. Conversely, the psychological, typological, and learnability considerations provide evidence for the computational nature of phonological generalizations on well-formedness.

This is not the first instance logic has been used in phonological theory. In fact, there is considerable history. **Example: definitions of constraints pre-OT and DP.**

A notable turning point occurred in the early 1990s with the developments of two theories: Declarative Phonology and Optimality Theory.

Declarative Phonology made explicit use of logical statements in describing the phonology of a language. **Example of a DP constraint.**

In Optimality Theory, first-order logic was often used implicitly to define

constraints.**Example: definitions of constraints in OT.**

Unlike Optimality Theory, the CDLs in this book will provide language-specific, inviolable constraints. For a representation to be well-formed it must not violate any constraint. This is a feature the CDLs here have in common with Declarative Phonology. Scobbie et al. explain:

> The actual model of constraint interaction adopted is maximally simple: the declarative model. In such a model, all constraints must be satisfied. The procedural order in which constraints are checked (or equivalently, in which they apply) is not part of the grammar, but part of an implementation of the grammar (as a parser, say) which cannot affect grammaticality. (Scobbie *et al.*, 1996, p. 692)

2.2 Chapter Outline

In the remainder of this chapter, we informally introduce model-theoretic representations of strings and different logics. Most mathematical details for the models and logical languages discussed in this chapter are provided in Appendix A to Part I of this book. Some readers may benefit by consulting Appendix A in parallel with this chapter. Readers for whom this does not satisfy their appetite are referred to the textbooks on logic and model theory provided in the Further Reading section below.

We focus on strings because they are widely used and well-understood. Most importantly, they are sufficient to illustrate how different CDLs can be defined and how these CDLs have consequences for psychological models, typology, and learnability. Several chapters later in the book provide concrete examples of non-string representations motivated by phonological theory. A mathematical treatment of representations and logic is given in the appendix of part I of this book. Concepts and definitions introduced here are presented there precisely and unambiguously.

First, we introduce the canonical word model, which is known as the successor model. This is followed by an informal treatment of First-Order (FO) logic. This yields the first CDL (FO with successor) and we show how to define a constraint like *NT—voiceless obstruents are prohibited from occuring immediately after nasals—in this CDL.

Next we alter the successor model so that the representations makes use of phonological features. This yields another CDL (FO with successor and

features). We comment on some notable points of comparison between the two CDLs, again using the *NT constraint.

The narrative continues by discussing one typological weakness the aforementioned CDLs: they are unable to describe long-distance constraints which are arguably part of the phonological competence of speakers of some languages. This provides some motivation for a CDL defined in terms of a more powerful logic, Monadic Second Order (MSO) logic. The CDL we call 'MSO with successor and features' and we explain how it is able to define such long-distance constraints. The key is that with MSO logic it is possible to deduce that one element in a string *precedes* another element, no matter how much later the second element occurs. The availability of the precedence relation makes it possible to define long-distance constraints.

We continue to evaluate the MSO with successor CDL from a typological perspective. We argue that there are significant classes of constraints definable in this CDL that are bizarre from a phonological perspective. In other words, we motivate seeking a more restrictive CDL capable of describing local and long-distance constraints in phonology.

One solution is to make the precedence relation part of the representation. This model of words is called the precedence model, which stands in contrast to the successor model. We show how the CDL "FO with precedence" is also able to describe both local and long-distance constraints of the kind found in the phonologies of the world's languages.

The remaining sections introduces logics that are more restrictive than First Order logic, which permits defining more restrictive CDLs. The motivation here primarily comes from psychological, typological, and learnability considerations. Typologically, typical constraints in phonology are definable with in terms of these more restrictive logics and CDLs. Psychologically, they have clear cognitive interpretations in terms of memory. And from a learning perspective, constraints in these restrictive CDLs can be feasibly learned from positive evidence.

Finally, the chapter concludes with a high-level discussion seeking to explain the following points. First, there is a tradeoff between representations and logical power. Second, as mentioned, the choice of representation and the choice of logic has consequences for typology, psychological reality, memory, and learnability. These consequences are fully reviewed in this section. Third, the representations and logics discussed in this chapter are only the tip of the iceberg. Readers undoubtedly will have asked themselves "What is possible with this representation?" and "Why don't we consider this variety

of logic?" Some chapters in this book address such questions. Comprehensively answering such questions, however, is beyond the scope of this book. But it is not beyond the scope of phonological theory. If some readers of this book pose and answer such questions, then this book will have succeeded in its goals.

2.3 The Successor Model

This section introduces the central ideas of model-theoretic representations with a concrete example. The concrete example comes from the "successor" model, which is arguably the canonical model for strings.

Model-theoretic representations provide a uniform framework for representing all kinds of objects. Here the objects under study are strings. We need to be clear about two things: what the objects are, and what counts as a successful model-theoretic representation of a set of objects.

Strings are sequences of events. If we are talking about words, the events could be given as speech sounds from the International Phonetic Alphabet. Strings are typically defined inductively. Each event corresponds is assigned some symbol. The set of symbols in use is called the **alphabet**. Each symbol on its own is a string, and if w is a string and a is a symbol then the concatenation of w and a, written wa, is also a string. This inductive definition yields a set of objects: all logically possible sequences of symbols of the alphabet of finite length.

A successful model theoretic-representation of a set of objects must provide a representation for each object and must provide distinct representations for distinct objects. It may be strange to ask the question "How can we represent strings?" After all, if we are talking about the string *tent* isn't *tent* itself a representation of it? It is, but the information carried in such representations is implicit. Model-theoretic representations make the information explicit.

Model-theoretic representations for objects of finite size like strings contain two parts. The first is a finite set of elements called the **domain**. The second is a finite set of relations. The relations provide information about the domain elements. The **model signature** summarizes two parts and serves to define the nature of model in terms of the information in the representation. In this book, it is written like this: $\langle D \mid R_1, R_2, \ldots R_n \rangle$.

We first show a model-theoretic representation of a word and then we

explain it. While this may seem backwards to some, it seems to work better pedagogically. It can be helpful to refer to the end-product as one goes about explaining how one got there.

Figure 2.1 shows the successor model for the word *tent* in addition to a graphical diagram of it on its right. The graphical diagram puts the domain elements in circles. Edges labeled with $\triangleleft$ indicate the binary relation called "successor." Finally, the unary relations, one for each symbol in the alphabet, are shown in typewriter face above the domain elements that belong to them. Throughout this book we will often use graphical diagrams instead of displaying the literal mathematical representation on the left. The order of the relations in the signature is fixed but it is also arbitrary.

$$
\begin{aligned}
&\mathcal{M}_{tent} \\
&= \langle D \mid \mathtt{t}, \mathtt{e}, \mathtt{n}, \mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}, \triangleleft \rangle \\
&= \Big\langle \{1,2,3,4\} \mid \{1,4\}, \{2\}, \{3\}, \\
&\qquad \varnothing, \varnothing, \ldots \varnothing, \\
&\qquad \{(1,2),(2,3),(3,4)\} \Big\rangle
\end{aligned}
$$

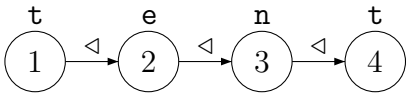

Figure 2.1: At left, the successor model of the word *tent*. At right, a graphical diagram of this model.

In the case of strings, the number of domain elements matches the length of the string. So a model-theoretic representation of a word like *tent* would have a domain with four elements, one for each event in the sequence. We can represent these domain elements with the suits in a deck of cards ($\heartsuit, \diamondsuit, \clubsuit, \spadesuit$) or we could use numbers $(1, 2, 3, 4)$ as we did in Figure 2.1. We will usually use numbers because as strings get longer we can always find new numbers. However, keep in mind that the numbers are just names of elements in the model in the same way the suits would have been. They get their meaning from the relationships they stand in, not from anything inherent in the numbers themselves.

In the successor word model, there is a unary relation $\mathtt{a}$ for each symbol a in the alphabet. We use the typewriter font to distinguish the relations from the symbols. It is customary to denote the alphabet with Σ. We write $(\mathtt{a})_{a \in \Sigma}$ to mean this finite set of relations. If a domain element belongs to the unary relation $\mathtt{a}$ then it means this element has the property of being a. So for the

word *tent*, two elements will belong to t, a different element will belong to e and the remaining element will belong to n. For every other symbol a in the alphabet the relation a will be empty. When we write $x \in \mathtt{a}$ and/or $\mathtt{a}(x)$ we mean that domain element x belongs to the unary relation a.

There is also a single binary relation called "successor". A domain element x stands in the successor relation to y if the event y corresponds to comes in fact immediately after the event x corresponds to. In this book, we use the symbol $\lhd$ to indicate the successor relation. For the word *tent*, if $2 \in R_e$ and $3 \in R_n$ then $(2, 3)$ would be in the successor relation. We will write $(2, 3) \in \lhd$, $\lhd(2, 3)$, and/or $2 \lhd 3$ to mean that domain elements 2 and 3 stand in the successor relation.

The model signature for the successor model is thus $\langle D \mid (\mathtt{a})_{a \in \Sigma}, \lhd \rangle$. The successor model is not the only way to represent words. From a phonological perspective, it is arguably a strange model. We will consider more phonologically natural models of words below.

It is easy to see that there is a general method for constructing a unique model for each logically possible string. Given a string of w of length n we can always construct the successor model as follows. Since w is a sequence of n symbols, we let $w = a_1 a_2 \ldots a_n$. Then set the domain $D = \{1, 2, \ldots n\}$. For each symbol $a \in \Sigma$ and i between 1 and n inclusive, $i \in \mathtt{a}$ if and only if $a_i = a$. And finally, for each i between 1 and $n - 1$ inclusive, let the only elements of the successor relation be $(i, i + 1)$. This is summarized in Table 2.8. This

D	$\stackrel{\text{def}}{=}$	$\{1, 2, \ldots n\}$
a	$\stackrel{\text{def}}{=}$	$\{i \in D \mid a_i = a\}$ for each unary relation a
$\lhd$	$\stackrel{\text{def}}{=}$	$\{(i, i + 1) \subseteq D \times D\}$

Table 2.1: Creating a successor model for any word $w = a_1 a_2 \ldots a_n$.

construction guarantees the model's soundness: each string has a model and distinct strings will have distinct models. It is also important to recognize that removing any one of the unary or binary relations will result in a model which does not guarantee that models of distinct strings are distinct.

Model-theoretic representations provide an ontology and a vocabulary for talking about objects. They provide a primitive set of facts from which we can reason. For instance in the word *rent*, we know that the t occurs sometime after the r. However this fact is not immediately available from the successor model. It can be deduced, but that deduction requires some

computation. Measuring the cost of such computations is but one facet of what model theory accomplishes. On the other hand, the successor model makes immediately available the information that t occurs immediately after the n. As will hopefully be clear by the end of this chapter, this distinction can shed light on differences between local and long-distance constraints in phonology.

From a psychological perspective, the primitive set of facts can be thought of as the primitive psychological units. In its strongest form, the model-theoretic representation of words as embodied in its signature makes a concrete claim about the psychological reality of the ways words are represented.

2.4 First Order Logic

Now that the models provide representations, what do we do with them? Logic provides a language for talking about these representations. First Order logic is a well-understood logical language which we introduce informally here. For those already familiar with FO logic, you will see take advantage of things like prenex normal form without discussion.

In addition to the Boolean connectives such as conjunction, disjunction, implication, and negation, FO logic also includes existential and universal quantification over variables that range over domain elements. These variables are called **first order variables**. Apart from these "logical connectives" and quantified variables, the basic vocabulary of FO logic comes from the *relations in the model signature*. Thus each model-theoretic representation supplies the ingredients for the logical language. Table 2.2 summarizes the vocabulary of FO logic with an arbitrary model $\langle D \mid R_1, R_2, \ldots R_n \rangle$. Model vocabulary are also called **atomic formulas** because they are the primitive terms from which larger logical expressions are built. As will be explained they play a special role in the ontology of model-theoretic linguistic theories.

Since the appendix defines FO logic formally, here we define valid sentences and formulas of FO logic ostensively. Below we give examples of three types of expressions: sentences of FO logic, formulas of FO logic, and syntactically ill-formed expressions.

Example 1 (Sentences of FO logic.)**.** Sentences of FO logic are complete sentences that can be interpreted with respect to a model. Below are five sentences of FO logic with English translations below.

Logical Connectives	
$\wedge$	conjunction
$\vee$	disjunction
$\neg$	negation
$\rightarrow$	implication
$\leftrightarrow$	biconditional
Syntactic Elements	
(	left parentheses
)	right parentheses
,	comma for separating variables
Variables, Quantifiers, and Equality	
x, y, z	variables which range over elements of the domain
$\exists$	existential quantifier
$\forall$	universal quantifier
$=$	equality between variables
Model Vocabulary	
$\mathtt{R}(x)$	for each unary relation R in $\{R_1, R_2, \ldots R_n\}$
$\mathtt{R}(x, y)$	for each binary relation R in $\{R_1, R_2, \ldots R_n\}$
$x\mathtt{R}y$	for each binary relation R in $\{R_1, R_2, \ldots R_n\}$
...	
$\mathtt{R}(x_1, x_2 \ldots x_n)$	for each n-ary relation R in $\{R_1, R_2, \ldots R_n\}$

Table 2.2: Symbols and their meaning in FO logic. Certain sequences of these symbols are valid FO sentences and formulas. Note we write binary relations in one of two ways.

1. Sentences of FO logic.

   (a) $\exists x, y, z\ (\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z))$
   (b) $\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
   (c) $\neg\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
   (d) $\forall x, y\ (\ \neg(\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y))$
   (e) $\forall x \exists y\ (\mathtt{n}(x) \rightarrow (\mathtt{t}(y) \wedge x \lhd y))$

2. English translation (in terms of the models).

(a) There are three distinct domain elements.
(b) There are two domain elements in the successor relation; the former has the property of being n; the latter has the property of being t.
(c) It is not the case that there exists two domain elements in the successor relation of which the former has the property of being n and the latter has the property of being t.
(d) For every pair of domain elements that stand in the successor relation, it is not the case that the former has the property of being n and the latter has the property of being t.
(e) For all domain element which have the property of being n, it is succeeded by a domain element which has the property of being t.

3. English translation (in terms of the strings the models represent).

(a) There are at least three symbols.
(b) There is a substring nt.
(c) There is no substring nt.
(d) There is no substring nt.
(e) If there is n then there is a t immediately following it.

Sentences of FO logic are **interpreted** with respect to models. Models for which the sentence is true are said to **satisfy** the sentence. If a model $\mathcal{M}$ of string w satisfies a sentence ϕ we write $\mathcal{M}_w \models \phi$. Consequently, every FO sentence ϕ divides the objects being modeled into two classes: those that satisfy ϕ and those that do not. In this way, logical sentences define **constraints**. The strings whose models satisfy the sentence do not violate the constraint; strings whose models do not satisfy the constraint do violate it. (Chapter **??** shows how logical sentences can define different types of weighted constraints.)

Table 2.3 provides examples of strings whose models satisfy the formulas in Example 1 and examples of strings whose models do not. An important feature of FO logic is that there are algorithmic solutions to the problem of deciding whether a given model satisfies a given sentence. This algorithm works because the syntactic rules that build up larger sentences from smaller ones have clear semantic interpretations with respect to the model under consideration. In short, it is an unambiguous and compositional system. For instance, $\mathcal{M} \models \phi \wedge \psi$ if and only if $\mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$. The interpretation of quantifiers is discussed after introducing formulas below.

ϕ	$\mathcal{M}_w \models \phi$	$\mathcal{M}_w \not\models \phi$
(a)	*too, tent, ttt*	*to, a*
(b)	*tent, rent, ntnt*	*ten, to, phobia*
(c)	*ten, to, phobia*	*tent, rent, ntnt*
(d)	*ten, to, phobia*	*tent, rent, ntnt*
(e)	*rent, antler*	*ten, nantucket*

Table 2.3: Some strings whose models satisfy the formulas in Example 1 and some whose models do not.

Example 2 (Formulas of FO logic.)**.** Formulas of FO logic are incomplete sentences in the sense that they contain variables that are not **bound**. A variable is bound only if it is has been introduced with a quantifier and is within that quantifier's scope. Variables that are not bound are called **free**. The formulas below are only interpretable with respect to a model $\mathcal{M}$ if the free variables are assigned some interpretation as an elements of the domain of $\mathcal{M}$.

1. Formulas of FO logic.
   (a) $\mathtt{n}(x) \vee \mathtt{m}(x) \vee \mathtt{ŋ}(x)$
   (b) $\exists y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \lhd y)$
   (c) $\neg\exists y\ (x \lhd y)$
   (d) $\neg\exists y\ (y \lhd x)$
   (e) $\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z)$
   (f) $x \lhd y \wedge y \lhd z$
2. English translation.
   (a) x has the property of being n, m, or $ŋ$.
   (b) x has the property of being n and coming immediately before an element which has the property of being t.
   (c) There is no element which succeeds x.
   (d) There is no element which x succeeds.
   (e) x, y and z are distinct.
   (f) x is succeeded succeeded by y which is succeeded by z.

The difference between formulas and sentences is that sentences admit no free variables. Because these formulas can only be interpreted in terms

of one or more un-instantiated variables, formulas are often used to define **predicates**. Predicates are essentially abbreviations for formulas with the unbound variables serving as parameters. Below we repeat the formulas from above, but use them to define new predicates.

$$\begin{aligned}
\texttt{nasal}(x) &\stackrel{\text{def}}{=} \texttt{n}(x) \vee \texttt{m}(x) \vee \texttt{ŋ}(x) \\
\texttt{nt}(x) &\stackrel{\text{def}}{=} \exists y\ (\texttt{n}(x) \wedge \texttt{t}(y) \wedge x \lhd y) \\
\texttt{last}(x) &\stackrel{\text{def}}{=} \neg\exists y\ (x \lhd y) \\
\texttt{first}(x) &\stackrel{\text{def}}{=} \neg\exists y\ (y \lhd x) \\
\texttt{distinct3}(x,y,z) &\stackrel{\text{def}}{=} \neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z) \\
\texttt{string3}(x,y,z) &\stackrel{\text{def}}{=} x \lhd y \wedge y \lhd z
\end{aligned}$$

These predicates can then be used to define new sentences. For example, the sentence $\forall x(\neg\texttt{nt}(x))$ is equivalent to (1d) in Example 1 above. In the same way that programmers write functions which encapsulate snippets of often-used programming code, predicates generally help writing and reading complex logical sentences.

Since sentences have no free variables, they must begin with quantifiers. Determining whether a model satisfies a sentence is compositional. It also depends on the **assignment** of variables to elements in the domain. For instance, to determine whether $\mathcal{M}$ satisfies $\phi = \exists x(\psi(x))$, we must find an element of the domain of $\mathcal{M}$, which if assigned to x, means that ϕ evaluates to `true`. If no such element exists, then $\mathcal{M}$ does not satisfy ϕ. Similarly, $\mathcal{M}$ satisfies $\phi = \forall x(\psi(x))$ if and only if every element of the domain $\mathcal{M}$, when assigned to x, results in ϕ evaluating to `true`.

Finally we give some examples of syntactically ill-formed sequences. The following expressions are junk; they are not interpretable at all.

Example 3 (Syntactically ill-formed sequences)**.** 1. Syntactically ill-formed sequences.

   (a) $x\exists\ )x($
   (b) $\forall\exists\ (\texttt{n} \vee \texttt{t})$
   (c) $\neg\exists(\texttt{n} \lhd \texttt{t})$

2. Comments.

(a) Quantifiers always introduce variables to their left and parentheses are used normally.

(b) No quantifier can be introduced without a variable and n-ary relations from the model vocabulary must always include n variables.

(c) Many beginning students make this sort of error when trying to express a logical sentence which forbids *nt* sequences. This expression breaks the same rules as the one before it.

We conclude this section by providing an example of a logical sentence defining a constraint which bans voiceless obstruents after nasals. This is constraint in the literature is often abbreviated *NT. Since the model signature does not include relations for concepts like nasals and voiceless consonants, we first define predicates for these notions. We assume the alphabet is limited to the following IPA symbols: *a,b,d,e,g,i,k,l,m,n,o,p,r,s,t,u,z*.

Example 4 (The constraint *NT defined under the FO with successor model.)**.**

$$\texttt{nasal}(x) \stackrel{\text{def}}{=} \texttt{n}(x) \vee \texttt{m}(x) \tag{2.1}$$

$$\texttt{voiceless}(x) \stackrel{\text{def}}{=} \texttt{p}(x) \vee \texttt{t}(x) \vee \texttt{k}(x) \vee \texttt{s}(x) \tag{2.2}$$

$$^*\texttt{NT} \stackrel{\text{def}}{=} \neg\exists x, y(x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y)) \tag{2.3}$$

It is easy to see that models of words like *tent* and *lampoon* do not satisfy $^*\texttt{NT}$ but models of words like *ten* and *moon* do. For example, in the model of *tent*, the expression $\exists x, y(x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y))$ is true when $x = 3$ and $y = 4$. Hence, $^*\texttt{NT}$ evaluates to `false`. On the other hand, in the model of the word *moon*, every value assigned x and y results in the sentence $\exists x, y(x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y))$ evaluating to `false`. Hence the sentence $^*\texttt{NT}$ evaluates to `true` and so $\mathcal{M}_{moon} \models {}^*\texttt{NT}$.

This section has presented the first CDL: FO with successor. The FO with successor model has been studied carefully and it is known precisely what kinds of constraints can and cannot be expressed with this CDL (Thomas, 1982), as will be discussed below.

2.5 Feature-based Word Models

One way in which the successor model above is strange from a phonological perspective is its absence of phonological features. The properties associated

