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. 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. For instance (Scobbie *et al.*, 1996, p. 688) expressed a general principle of theories of syllables which prohibit ambisyllabicity this way: $\forall x \neg(\mathrm{onset}(x) \wedge \mathrm{coda}(x))$, which in English reads "For all segments x, it is not the case that x is both a onset and a coda."

In Optimality Theory, first-order logic was often used implicitly to define constraints. For example, the definition of the constraint MAX-IO in OT given by McCarthy and Prince (1995, p. 16) is "Every segment of the input has a correspondent in the output." On page 14, they define the correspondence relation: "Given two strings S_1 and S_2, correspondence is a relation R from the elements of S_1 to those of S_2. Elements $\alpha \in \mathrm{S}_1$ and $\beta \in \mathrm{S}_2$ are referred to as **correspondents** of one another when $\alpha R \beta$." As will be clear by the end of this chapter, this definition of MAX-IO is essentially a statement in First Orer Logic: For all $\alpha \in \mathrm{S}_1$ there exists $\beta \in \mathrm{S}_2$ such that $\alpha R \beta$.

Unlike Optimality Theory, the CDLs introduced in this chapter provide language-specific, inviolable constraints. For a representation to be well-formed it must not violate any constraint. This is a property the CDLs in this chapter 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)

What Scobbie et al. are emphasizing is that logical specifications of grammar specify *what is being computed* as opposed to *how it is being computed.* We agree with Scobbie *et al.* (1996) that this is an attractive property of logical languages.

While this chapter, and others in this book, assume the constraints are language-specific and inviolable, it is a mistake to conclude that this line of work only applies to grammars that make binary distinctions between well-formed and ill-formed structures. In fact, *weighted* logical languages allow one to specify what is being computed when structures are going to be assigned natural numbers (for instance in the case of counting the number of times a a structure violates a constraint) or real numbers (for instance in the case of assigning some probability to a structure) (Droste and Gastin, 2009). We review the basics and provide some examples in Chapter **??**.

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.

Finally, the chapter concludes with a high-level discussion seeking to emphasize 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. 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.

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

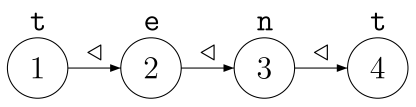

Figure 2.1: At left, the successor model of the word *tent*. At right, a graphical diagram of this model.

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 $\mathtt{t}$, a different element will belong to $\mathtt{e}$ and the remaining element will belong to $\mathtt{n}$. For every other symbol a in the alphabet the relation $\mathtt{a}$ 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 $\mathtt{a}$.

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 $w = a_1a_2 \ldots a_n$. 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 construction guarantees the model's soundness: each string has a model and

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

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

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 $\langle D \mid R_1, R_2, \dots R_n\rangle$. Model vocabulary are also called **atomic formulas** because they are the

Boolean 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 $\{R_1, R_2, \dots R_n\}$
$\mathtt{R}(x, y)$	for each binary relation R in $\{R_1, R_2, \dots R_n\}$
$x\mathtt{R}y$	for each binary relation R in $\{R_1, R_2, \dots R_n\}$
$\dots$	
$\mathtt{R}(x_1, x_2 \dots x_m)$	for each m-ary relation R in $\{R_1, R_2, \dots R_n\}$

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.

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.

1. Sentences of FO logic.
   (a) $\exists x, y, z\ (\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z))$
   (b) $\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
   (c) $\neg\exists x, y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
   (d) $\forall x, y\ (\ \neg(\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y))$
   (e) $\forall x \exists y\ (\mathtt{n}(x) \rightarrow (\mathtt{t}(y) \wedge x \triangleleft y))$

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.

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

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.

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.

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) $\exists y\ (\mathtt{n}(x) \wedge \mathtt{t}(y) \wedge x \triangleleft y)$
(c) $\neg\exists y\ (x \triangleleft y)$
(d) $\neg\exists y\ (y \triangleleft x)$
(e) $\neg(x = y) \wedge \neg(x = z) \wedge \neg(y = z)$
(f) $x \triangleleft y \wedge y \triangleleft z$

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. We write predicates in sans serif font.

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

These predicates can then be used to define new sentences. For example, the sentence $\forall x(\neg\mathsf{nt}(x))$ 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) $\forall\exists\ (\mathtt{n} \vee \mathtt{t})$
(c) $\neg\exists(\mathtt{n} \triangleleft \mathtt{t})$

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.).

$$\mathsf{nasal}(x) \overset{\text{def}}{=} \mathtt{n}(x) \vee \mathtt{m}(x) \quad (2.1)$$

$$\mathsf{voiceless}(x) \overset{\text{def}}{=} \mathtt{p}(x) \vee \mathtt{t}(x) \vee \mathtt{k}(x) \vee \mathtt{s}(x) \quad (2.2)$$

$$\mathsf{*NT} \overset{\text{def}}{=} \neg\exists x, y(x \triangleleft y \wedge \mathsf{nasal}(x) \wedge \mathsf{voiceless}(y)) \quad (2.3)$$

It is easy to see that models of words like *tent* and *lampoon* do not satisfy *NT but models of words like *ten* and *moon* do. For example, in the model of *tent*, the expression $\exists x, y(x \triangleleft y \wedge \mathsf{nasal}(x) \wedge \mathsf{voiceless}(y))$ is true when $x = 3$ and $y = 4$. Hence, *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 $\exists x, y(x \triangleleft y \wedge \mathsf{nasal}(x) \wedge \mathsf{voiceless}(y))$ evaluating to false. Hence the sentence *NT evaluates to true and so $\mathcal{M}_{moon} \models \mathsf{*NT}$.

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 with the elements of the domain are whole segments. However, nothing in model theory itself prohibits domain elements from having more than one property. It is a consequence of the construction in Table 2.8 that each domain element will satisfy exactly one of the unary relations a, no more and no less. We can formalize this statement of the successor model in Remark 1 as follows.

Remark 1 (The successor model entails disjoint unary relations)**.** For all successor models $\mathcal{M} = \langle D \mid (\mathtt{a})_{a\in\Sigma}, \triangleleft \rangle$, and for all $\mathtt{a},\mathtt{b} \in (\mathtt{a})_{a\in\Sigma}$, it is the case that $\mathtt{a} \cap \mathtt{b} = \varnothing$.

Therefore it is possible to design different models of words, where the unary relations do not represent segments like *a, b*, or *n* but phonological features such as *vocalic, labial*, or *nasal.* Crucially, in these models would not entail disjoint unary relations: a domain element could be both *voiced* and *labial* for instance.

In this part of the chapter, we give one example of such a model. There are many others, as many as there are theories of phonological features. The model we give here is primarily for pedagogical reasons; we are not stating particular beliefs or arguments regarding the nature of feature systems. We are only choosing a simple system that illustrates some key points.

We set up a feature system with **privative** features for the simple alphabet Σ discussed earlier *a,b,d,e,g,h,i,k,l,m,n,o,p,r,s,t,u,z*. The use of privative features contrasts with the typical assumption in phonological theory that features are binary. We choose not to pick a minimal nor maximal set of features for distinguishing this set. Instead we choose somewhat arbitrarily a middle ground based on standard descriptive phonetic terms used for describing the manner, place and laryngeal qualities in articulating sounds. We call this model the "successor model with features." Its signature is shown below.

$$\langle D \mid \texttt{vocalic}, \texttt{low}, \texttt{high}, \texttt{front}, \texttt{stop}, \texttt{fricative}, \texttt{nasal}, \texttt{lateral} \\ \texttt{rhotic}, \texttt{voiced}, \texttt{voiceless}, \texttt{labial}, \texttt{coronal}, \texttt{dorsal}, \lhd \rangle \tag{2.4}$$

Table 2.4 shows how to construct a successor model with features for any string in Σ^*. Again this model ensures that distinct strings from Σ^* have different models and that every string has some model.

Figure 2.2 shows the successor model with features of the word *tent*.

The successor model with features contrasts sharply with the successor model with features in an important way. To see how, first consider the constraint *NT. Under the successor model with features, this constraint would be defined as in Example 2.5

Example 5 (The constraint *NT defined under the FO with successor model with features.)**.**

$$\texttt{*NT} \quad \overset{\text{def}}{=} \quad \neg\exists x, y(x \lhd y \wedge \texttt{nasal}(x) \wedge \texttt{voiceless}(y)) \tag{2.5}$$

This looks similar to the definition of *NT under the successor model (Example 2.1), but there is a critical difference. The predicates above in Example 2.5 are *atomic* formula and not user-defined predicates as they are in Example 2.1.

This is an important ontological difference between these two models. In the successor model with features there is no primitive representational concept that corresponds to a sound segment like *t* like there is in the successor

D	$\overset{\text{def}}{=}$	$\{1, 2, \ldots n\}$
`vocalic`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{a, e, i, o, u\}\}$
`low`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i = a\}$
`high`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{i, u\}\}$
`front`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{e, i\}\}$
`stop`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, d, g, k, p, t\}\}$
`fricative`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{h, s, z\}\}$
`nasal`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{m, n\}\}$
`lateral`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i = l\}$
`rhotic`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i = r\}$
`voiced`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, d, g, z\}\}$
`voiceless`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{k, p, s, t, h\}\}$
`labial`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{b, p, m\}\}$
`coronal`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{d, s, t, z\}\}$
`dorsal`	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i \in \{d, g, k\}\}$
$\triangleleft$	$\overset{\text{def}}{=}$	$\{(i, i+1) \mid 1 \leq i < n\}$

Table 2.4: Creating a successor model with features for any word $w = a_1 a_2 \ldots a_n$.

model without features. Conversely, in the successor model without features there is no primitive representational concept that corresponds to a phonological like *voiceless* like there is in the successor model with features. In the successor model with features we can write user-defined predicates that define properties of domain elements that we can interpret to mean "being t".

$$\mathsf{is\_t}(x) \overset{\text{def}}{=} \texttt{stop}(x) \wedge \texttt{coronal}(x) \wedge \texttt{voiceless}(x) \tag{2.6}$$

Other sound segments would be defined similarly.

One way to put this difference is that in the successor model with features one can immediately determine whether a domain element is voiced or not, but in the successor model without features one cannot immediately determine this fact. Instead one can deduce it by checking the appropriate user-defined predicate. Likewise, in the successor model with features one cannot immediately determine whether a domain element is t or not. With

$$
\begin{aligned}
&\mathcal{M}_{tent} = \\
&\Big\langle \{1,2,3,4\} \mid \texttt{vocalic} = \{2\}, \\
&\qquad \texttt{front} = \{2\}, \\
&\qquad \texttt{stop} = \{1,4\}, \\
&\qquad \texttt{nasal} = \{3\}, \\
&\qquad \texttt{voiceless} = \{1,4\}, \\
&\qquad \texttt{coronal} = \{1,3,4\} \\
&\qquad \triangleleft = \{(1,2),(2,3),(3,4)\} \Big\rangle
\end{aligned}
$$

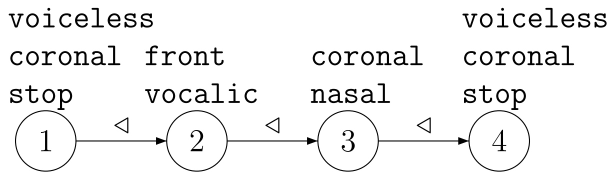

Figure 2.2: At left, the successor model with features of the word *tent*. Unary relations which equal the empty set are omitted for readability. At right, a graphical diagram of this model.

the featural representations, such a fact must be deduced with a user-defined predicate like the one above.

Also, the fact that such user-defined predicates exist should not be taken for granted. They exist here because the only logical system discussed so far is FO. With FO logic, it is possible to define a predicate for any subset of the alphabet Σ for both successor models with and without features. If the logical system was restricted in some further way then some user-defined predicates may not be possible to define. For example, if the logical system only permitted conjunction and no other Boolean connective then it would not be possible to define a predicate for voiceless stops in the successor model without features. This interplay between representations and logical power with respect to expressivity is an important theme of this chapter. It will be discussed at length with respect to the successor relation, and we will return to it in the context of features when restricted logics are introduced towards the end of the chapter.

As a consequence of FO logic then, any constraint definable with one of the representations discussed so far is definable in the other. This leads to the conclusion that there are no typological distinctions between the FO with successor theory and the FO with successor with features theory. Both admit exactly the same class of constraints.

However, while the two models do not make different typological predictions, they do make different psychological ones. In regard to phonological theory, the signature of the model is an ontological commitment to the psychological reality of the model vocabulary. Taken seriously, the successor model with features says that the mental representations of words carries only the information shown in Figure 2.2. Thus, taken seriously, the successor model with features says that the segments in the word *tent* are not perceived as such but are instead perceived in terms of their features. Clever psycholinguistic experiments might be able to bring evidence to bear on which model more accurately resembles them actual mental representations of words.

2.6 Monadic Second-Order Logic

This section introduces Monadic Second-Order (MSO) logic. This logic is strictly *more expressive* than FO logic. We motivate the discussion of MSO logic from a linguistic perspective by showing that FO with successor, both with and without features, is not sufficient to account for long-distance phonotactic constraints.

What are long-distance phonotactic constraints? Odden (1994) draws attention to an unbounded nasal assimilation in Kikongo whereby underlying /ku-kinis-il-a/ becomes [kukinisina] 'to make dance for.' From one perspective, this assimilation could be said to be driven by a phonotactic constraint that forbids laterals from occuring after nasals. Similar long-distance constraints have been posited for a variety of long-distance assimilation and dissimilation processes (Hansson, 2010).

We first show that the phonotactic constraint which bans laterals from occuring *anywhere* after nasals cannot be expressed in the FO with successor model. As we hope to make clear, the problem is that the notion of *precedence* is not FO-definable from successor. To illustrate, in Kikongo, [kukinisila] is an ill-formed string. The nasal has only one successor [i], but [n] *precedes* many segments including the second and third [i]s and the [s,l] and [a]. It is the fact that [n] precedes [l] which makes [kukinisila] ill-formed according to the phonotactic constraint which bans laterals from occuring *anywhere* after nasals. We refer to this constraint as *N..L.

Constraint *N..L is not FO definable with successor. To prove this we use an abstract characterization of the constraints definable with FO and successor due to Thomas (1982) and reviewed in Rogers and Pullum (2011).

Theorem 1 (Characterization of FO-definable constraints with successor)**.** *A constraint is FO-definable with successor if and only if there are two natural numbers k and t such that for any two strings w and v, if w and v contain the same substrings x of length k* the same number of times counting only up to t*, then either both w and v violate the constraint or neither does.*

Essentially, this theorem says constraints that are FO-definable with successor cannot distinguish among strings that are composed of the same number and type of substrings of some length k, where substrings can be counted only up to some threshold t.

We can use this theorem to show that *N..L is not FO definable with successor by presenting two strings which *N..L distinguishes but which are not distinguishable according to the criteria in Theorem 1. This would prove that *N..L is not LTT and thus not FO-definable with successor. Importantly, for any k and t we have to present two strings. These strings can depend on k and t.

We use notation a^k to mean the string consisting of k consecutive as. So $a^3 = aaa$. For any numbers k and t larger than 0, consider the words $w = a^kna^k\ell a^k$ and $v = a^k\ell a^kna^k$. Table 2.6 below shows the substrings up to length k, and their number of occurrences. Each word has the same substrings and the same number of them. Note the left and right word boundaries ($\rtimes$ and $\ltimes$ respectively) are customarily included as part of the strings.

In the discussion below, the following concept will prove useful. For every number t and every number n let the t-number of n equal n if $n < t$ otherwise let it be t. So if n is our count than the t-number of n is just the count of n up to the threshold t.

As can be seen from the above table, the two strings have exactly the same number of occurrences of each k-long substring. Consequently, the t-numbers of each k-long substring is also the same for any t. It follows, from Theorem 1 that these two strings cannot be distinguished by any constraint which is FO-definable with successor. More precisely, *any constraint which is FO-definable with successor is unable to distinguish in strings w and v whether n precedes ℓ or whether ℓ precedes n.* As such, no FO-definable constraint with successor can be violated by w but not by v and vice versa. It follows that *N..L is not FO definable with successor because for the reason that it this is precisely the distinction it makes.

Having established that linguistically motivated long-distance phonotac-

count	$w = \rtimes a^k n a^k \ell a^k \ltimes$	Notes
1	$\rtimes a^{k-1}$	
3	a^k	
1	$a^i n a^j$	(for each $0 \leq i, j \leq k-1$, $i+j = k-1$)
1	$a^i \ell a^j$	(for each $0 \leq i, j \leq k-1$, $i+j = k-1$)
1	$a^{k-1} \ltimes$	
count	$v = \rtimes a^k \ell a^k n a^k \ltimes$	**Notes**
1	$\rtimes a^{k-1}$	
3	a^k	
1	$a^i n a^j$	(for each $0 \leq i, j \leq k-1$, $i+j = k-1$)
1	$a^i \ell a^j$	(for each $0 \leq i, j \leq k-1$, $i+j = k-1$)
1	$a^{k-1} \ltimes$	

Table 2.5: The k-long substrings with their number of occurrences in the strings $w = a^k n a^k \ell a^k$ and $v = a^k \ell a^k n a^k$ with word boundaries.

tic constraints are not FO-definable with successor, we turn to the question of how such constraints can be defined from the logical perspective offered here. Essentially, there are two approaches. One is to increase the power of the logic. The other is to change the model—the representation—of strings. This section examines the first option and the next section examines the second option. This interplay between logical power and representations and how it affects the expressivity of the linguistic system is a running theme of this book.

Monadic Second Order (MSO) logic is a logical language that is strictly more powerful than FO logic. Constraints that are MSO-definable with successor include every constraint which is FO-definable with successor because every sentence and formula in FO logic with successor is also a sentence and formula in MSO logic with successor and is interpreted in the same way. In addition to first order variables, MSO comes with **second order variables**. Generally, variables that are second order are allowed to vary over n-ary relations. The restriction to monadic second order variables means the variables in this logic can only vary over unary relations, which corresponds to *sets* of domain elements. This contrasts with first order variables, which recall vary only over elements of the domain.

MSO logic is defined formally in the appendix to Part I, so here we introduce it informally with examples. In MSO logic, the MSO variables are expressed with capital letters such as X, Y, and Z to distinguish them from first order variables which use lowercase letters like x, y, and z. Observe that $x \in X$ and $X(x)$ are synonyms. As with first order variables, second order variables are introduced into sentences and formula with quantifiers.

Additional Symbols in MSO logic	
X, Y, Z	variables which range over sets of elements of the domain
$x \in X$	checks whether an element x belongs to a set of elements X
$X(x)$	checks whether an element x belongs to a set of elements X

Table 2.6: Together with the symbols of FO logic shown in Table 2.2, these symbols make up MSO logic.

With MSO logic over successor, it is now possible to define the precedence relation as shown below.

$$\mathsf{closed}(X) \overset{\text{def}}{=} (\forall x, y)\big[(x \in X \wedge x \triangleleft y) \rightarrow y \in X\big] \tag{2.7}$$

$$x < y \overset{\text{def}}{=} (\forall X)\big[(x \in X \wedge \mathsf{closed}(X) \rightarrow y \in X\big] \tag{2.8}$$

Intuitively, a set of elements X in the domain of a model of some word w satisfies $\mathsf{closed}(X)$ only if every successor of every element in X is also in X. In short, $\mathsf{closed}(X)$ is true only for sets of elements X which are transitively closed under successor. Then x precedes y only if for every closed set of elements X which x belongs to, y also belongs to X.

Figure 2.3 below illustrates these ideas. The successor model for the string *aℓaana* is shown. Six ellipses are shown, which represent the six nonempty sets of domain elements which are closed under successor and thus satisfy $\mathsf{closed}(X)$.

We can conclude that ℓ precedes n because every closed set which element 2 (which corresponds to ℓ) belongs to (X_1 and X_2) also includes the element 5 (which corresponds to n). Similarly, we can conclude that n does not precede ℓ because it is not the case that all closed sets which contain element 5 (which corresponds to n) also include element 2 (which corresponds to ℓ). Set X_4 for instance contains element 5 but not element 2.

Once the binary relation for precedence ($<$) has been defined, it is now straightforward to define the constraint *N..L with features.

$$\mathsf{*N..L} \overset{\text{def}}{=} \neg(\exists x, y)[x < y \wedge \mathtt{nasal}(x) \wedge \mathtt{lateral}(y)] \tag{2.9}$$

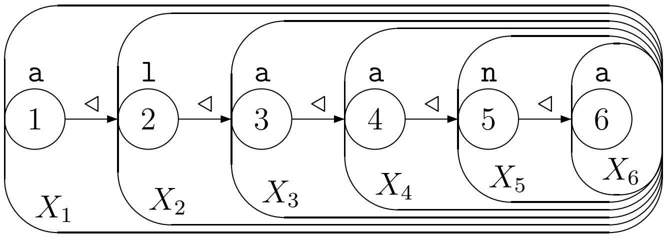

Figure 2.3: The successor model for the word *aɫaana*. Rectangular regions indicate the sets of domain elements (X_i) which are closed under successor.

The sentence above may look like a sentence of FO logic since no second order variables are present. However, it is important to remember that the precedence relation ($<$) is just an abbreviation for a longer formula, which is defined in MSO logic, and not within FO logic. Often whether a predicate is atomic or derived is not something that can be determined from inspecting a sentence or formula since the notation does not distinguish them. Usually one must be being acutely aware of the model signature to know whether a predicate is atomic or derived.

At this point, we have established that the linguistically motivated long-distance phonotactic constraint is not definable with FO logic with successor but it is definable with MSO logic with successor. We thus ask: What other kinds of constraints are MSO-definable with successor?

Another constraint that is not FO-definable with successor but is MSO-definable constraint with successor is a constraint that requires words to have an even number of nasals. Words like *man* and *neonatology* obey this constraint since they have two nasals but words like *mannequin* and *nanotechnology* do not since they have three nasals.

To see that this constraint is not FO-definable with successor, we use Theorem 1 as before. For any nonzero numbers k and t, consider the words $w = a^k(na^k)^{2t}$ and $v = a^k(na^k)^{2t}na^k$. Observe that w obeys the constraint since it contains $2t$ nasals and $2t$ is an even number. On the other hand, v contains $2t+1$ nasals and therefore violates the constraint. However, as Table 2.7 shows, these words have the same substrings of length k, and the same t-numbers of each substring.

However, this constraint is expressible with MSO logic with successor. We make use of some additional predicates, including general precedence ($<$) defined in Equation 2.8. The predicate firstN is true of x only if x is the

$w = \rtimes a^k(na^k)^{2t}a^k\ltimes$			
count	t-number	k-long substrings	notes
1	1	$\rtimes a^{k-1}$	
$2t+2$	t	a^k	
$2t+2$	t	$a^i n a^j$	(for each $0 \leq i,j \leq k-1$, $i+j=k-1$)
1	1	$a^{k-1}\ltimes$	

$v = \rtimes a^k(na^k)^{2t}na^k\ltimes$			
count	t-number	k-long substrings	notes
1	1	$\rtimes a^{k-1}$	
$2t+2$	t	a^k	
$2t+2$	t	$a^i n a^j$	(for each $0 \leq i,j \leq k-1$, $i+j=k-1$)
1	1	$a^{k-1}\ltimes$	

Table 2.7: The k-long substrings and the t-numbers of their counts in $w = a^k(na^k)^{2t}$ and $v = a^k(na^k)^{2t}na^k$ with word boundaries.

first nasal occuring in the word (Equation 2.10). The predicate lastN is true of x only if x is the last nasal occuring in the word (Equation 2.11). Also, two variables x and y stand in the $\lhd_{\mathsf{N}}$ only if y is the first nasal to occur after x (Equation 2.12). So $\lhd_{\mathsf{N}}$ is a a successor relation relativized to nasals.

$$\begin{aligned} \mathsf{firstN}(x) &\overset{\text{def}}{=} \texttt{nasal}(x) \wedge \neg(\exists y)[\texttt{nasal}(y) \wedge y < x] && (2.10)\\ \mathsf{lastN}(x) &\overset{\text{def}}{=} \texttt{nasal}(x) \wedge \neg(\exists y)[\texttt{nasal}(y) \wedge x < y] && (2.11)\\ x \lhd_{\mathsf{N}} y &\overset{\text{def}}{=} \texttt{nasal}(x) \wedge \texttt{nasal}(y) \wedge x < y \\ &\qquad \wedge \neg(\exists z)[\texttt{nasal}(z) \wedge x < z < y] && (2.12) \end{aligned}$$

Note we use the shorthand $x < y < z$ for $x < z \wedge z < y$.

With these predicates in place, we write Even-N as in Equation 2.13.

$$\begin{aligned} \textsc{Even-N} \overset{\text{def}}{=} (\exists X)\Big[\quad & (\forall x)[\mathsf{firstN}(x) \rightarrow X(x)] \\ & \wedge\ (\forall x)[\mathsf{lastN}(x) \rightarrow \neg X(x)] \ \Big] \\ & \wedge\ (\forall x,y)\big[x \lhd_{\mathsf{N}} y \wedge \big(X(x) \leftrightarrow \neg X(y)\big)\big] \qquad (2.13) \end{aligned}$$

In English, this says that a model of word w satisfies Even-N provided there is a set of domain elements X that includes the first nasal (if one occurs), does not include the last nasal (if one occurs) and for all pairs of successive nasals (if they occur), exactly one belongs to X. Consequently, words containing zero nasals satisfy the EVEN-N because the empty set of domain elements vacuously satisfies these three conditions. Words containing exactly one nasal do not satisfy EVEN-N because the first nasal and the last nasal are the same element x and it cannot both belong and not belong to X. However, words with exactly two nasals do satisfy EVEN-N because the first nasal belongs to X (satisfying the first condition), the last nasal does not (satisfying the second condition), and these two nasals are successive nasals and so are subject to the third condition, which they satisfy because exactly one of them (the first nasal) belongs to X. A little inductive reasoning along these lines lets one conclude that only words with an even number of nasals will satisfy EVEN-N as intended.

It is natural to wonder whether there is an abstract characterization of constraints that are MSO-definable with successor in the same way that Thomas (1982) provided an abstract characterization of constraints that are FO-definable with successor. In fact there is. Büchi (1960) showed that these constraints are exactly the ones describable with finite-state automata.

Theorem 2 (Characterization of MSO-definable constraints with successor)**.** *A constraint is MSO-definable with successor if and only if there is a finite-state automata which recognizes the words obeying the constraint.*

From the perspective of formal language theory, they are exactly the regular languages. Informally, these are formal languages for which the membership problem can be solved with a constant, finite amount of memory.

In this section we showed that FO-definable constraints with successor are not sufficiently powerful to express long-distance phonotactic constraints. One approach is to then increase the power of the logic. One logical system extends FO by adding quantification over monadic second order variables. This logic—MSO logic with successor—is able to express long-distance phonotactic constraints. However, MSO logic with successor also is also sufficiently expressive as a CDL to express constraints like EVEN-N.

Another way of putting it is like this. In the successor model, the information that in the word *aℓaana* the ℓ precedes the n is not immediately available from the representation. That information can be *deduced* but the deduction requires some computational effort. From the logical perspective taken here, this deduction requires MSO power and not FO power. Furthermore, once

MSO power is admitted then it becomes possible to similarly deduce whether or not there are even numbers of elements with certain properties.

Another approach to developing a CDL which can express long-distance phonotactic constraints but not EVEN-N is to change the representation of strings; that is, to change the model signature. This is precisely the topic of the next section.

2.7 The Precedence Word Model

So far, the logics we have considered have been defined with respect to the successor model of words. However, as we have seen with phonological features vis a vis atomic letters, there are different models of strings. In this section, we consider the *precedence* model of strings. Simply, this model contains the precedence relation instead of the successor relation in its signature.

As with the successor model, there is a general construction for the determining the precedence model for any string. Given a string of w of length n the precedence model is constructed as follows. Since w is a sequence of n symbols, we let $w = a_1a_2 \dots a_n$. Then set the domain $D = \{1, 2, \dots n\}$. 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 and j between 1 and n inclusive, the only elements of the precedence relation are (i, j) so long as $i < j$. This is summarized in Table 2.8. This construction guarantees the model's soundness: each string

D	$\overset{\text{def}}{=}$	$\{1, 2, \dots n\}$
$\mathtt{a}$	$\overset{\text{def}}{=}$	$\{i \in D \mid a_i = a\}$ for each unary relation $\mathtt{a}$
$<$	$\overset{\text{def}}{=}$	$\{(i, j) \subseteq D \times D \mid i < j\}$

Table 2.8: Creating a successor model for any word $w = a_1a_2 \dots a_n$.

has a model and distinct strings will have distinct models.

Figure 2.4 shows the precedence model for the word *tent* in addition to a graphical diagram of it on its right.

The difference between the precedence model and the successor model is how the order of segments in the word are represented. In the precedence model, the fact that the n is preceded by t in the word *tent* is immediately available because the element corresponding to t is in the precedence relation with the element corresponding to the first t. Under the successor model,

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

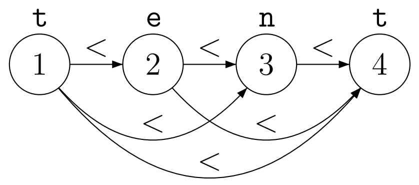

Figure 2.4: At left, the precedence model of the word *tent*. At right, a graphical diagram of this model.

this information was not immediately available as it was not part of the representation. However, under the precedence model it is.

Take seriously from a psychological perspective, the precedence model can be taken to mean that as words are perceived, information about the precedence relations is being stored in memory as part of the lexical representation of the word.

Also, in the same way that we considered the successor model both with and without features, we can also consider a precedence model with and without features. The precedence model introduced above was without features, but it is a simple matter to replace the unary relations in that model with the ones in Table 2.4.

It is straightforward to now write the constraint *N..L in the CDL which we call "FO with precedence with features."

$$\textsf{*N..L} \quad \overset{\text{def}}{=} \quad \neg\exists x, y(x < y \wedge \mathtt{nasal}(x) \wedge \mathtt{lateral}(y)) \tag{2.14}$$

Equation 2.14 looks identical to Equation 2.9. However, there is critical difference. In Equation 2.14, the precedence relation is an atomic formula but in Equation 2.9 it is a user-defined predicate in MSO logic.

It is natural to ask of course whether a constraint like *NT is expressible in this CDL. The answer is Yes because successor is FO-definable from precedence. Equation 2.15 shows how. Essentially, x is succeeded by y only if x precedes y and there is no element z such that $z < y$ and $x < z$.

$$x \triangleleft y \quad \overset{\text{def}}{=} \quad x < y \wedge \neg(\exists z)[x < z < y] \tag{2.15}$$

It is a striking fact that successor is FO-definable from precedence but precedence is MSO-definable from successor. This is a considerable asymmetry between the successor and precedence models of strings.

There are two important consequences. The first is the CDL "FO with precedence" properly subsumes the CDL "FO with successor." Not only is every constraint expressible with the CDL "FO with successor" also expressible with the CDL "FO with precedence", but there are constraints like *N..L above that expressible with the CDL "FO with precedence" but not with the CDL "FO with successor."

Another important consequence is that the CDL "MSO with precedence" is equivalent in expressive power to the CDL "MSO with successor" discussed in the previous section. This is because with MSO logic, precedence can be defined from successor as shown previously. Likewise because MSO logic properly extends FO logic, successor can also be defined from precedence. So at the level of MSO, these two models make no distinctions among the kinds of constraints that can be expressed. Constraints in each CDL correspond to exactly the regular stringsets.

There is also an abstract characterization of the FO-definable constraints with precedence due to McNaughton and Papert (1971).

Theorem 3 (Characterization of FO-definable constraints with precedence)**.** *A constraint is FO-definable with precedence if and only if there is a poistive integer n such that for all strings x, y, z if xy^nz obeys the constraint then for all $k > n$, xy^kz obeys the constraint too.*

This characterization says that FO-definable constraints with precedence can only distinguish iterations within strings up to some finite n. Two strings xy^iz and xy^jz with both $i, j > n$ but $i \neq j$ cannot be distinguished by any FO-definable constraint with precedence. As McNaughton and Papert (1971) amply document, there are other independently-motivated characterizations of this class as well.

The above characterization can be used to show that Even-N is not FO-definable with precedence. Again, the strategy is to consider any n and then to find strings w, v, x, y, z and numbers $i, j > n$ such that $w = xy^iz$ and $v = xy^jz$ where Even-N distinguishes w and v in the sense that one violates Even-N and the other does not. If the constraint were FO-definable with precedence such strings could not exist by Theorem 3. In this case, one solution is to set $x = z = \lambda$ (the empty string), $y = ma$, $i = 2n$ and $j = 2n + 1$. Then $w = (ma)^{2n}$ and $v = (ma)^{2n+1}$. Clearly, w has an even number of nasals since it has $2n$ [m]s but v has an odd number since it has

$2n+1$ [m]s. Thus Even-N distinguishes these strings and thus by Theorem 3 it cannot be FO-definable with precedence.

In this section, we considered a model of words where order is represented with the precedence relation instead of the successor relation. It was shown that long-distance constraints can readily be expressed in the CDL "FO with precedence." Furthermore, local phonotactic constraints like *NT can also be expressed because successor is FO-definable from precedence. However, the converse is not true. This asymmetry means that FO with precedence is strictly more expressive than "FO with successor." It was also shown that Even-N is not expressive in this system. Finally, it was noted that "MSO with precedence" is equally expressive as "MSO with successor". Once there is MSO power, successor and precedence are each definable from the other. Which constraints can be expressed by which CDLs is summarized in Figure 2.5.

MSO	*N..L, Even-N	Even-N
FO	*NT	*NT, *N..L
	$\triangleleft$	$<$

Figure 2.5: Classifying the constraints *NT, *N..L, and Even-N.

More generally, this section established the following. Although one way to increase the expressivity of a CDL is to increase the power of the logic, another way is to change the representations underlying the models. This speaks directly to the interplay between representations and computational power, one of the themes of this chapter.

We conclude that the only CDL discussed so far that can express both local and long-distance phonotactic constraints (like *NT and *N..L) and fails to express constraints like Even-N is the CDL "FO with precedence."

2.8 Discussion

2.8.1 Tradeoffs between representations and power

2.8.2 Typology, learnability, and psychological reality

2.8.3 Well-formedness and Transformations

2.9 Further Reading