A Default Inheritance Hierarchy for Computing Hebrew Verb Morphology

Raphael Finkel and Gregory Stump
University of Kentucky, USA

Abstract

We apply default inheritance hierarchies to generating the morphology of Hebrew verbs. This approach represents inflectional exponents as markings associated with the application of rules by which complex word forms are deduced from simpler roots or stems. The high degree of similarity among verbs of different conjugation classes allows us to formulate general rules; these general rules are, however, sometimes overridden by conjugation-specific rules. Similarly, a verb's form within a particular conjugation is determined both by default rules and by overriding rules specific to lexical stem peculiarities. Our result is a concise set of rules defining the morphology of Hebrew verbs in all conjugations. We express these rules in KATR, both a formalism for default inheritance hierarchies and associated software for generating the forms specified by those rules. As we describe the rules, we point out general strategies for expressing morphology in KATR. We conclude by discussing KATR's advantages over ordinary DATR for the representation of morphological systems and our plans for KATR's successor, LATR.

Correspondence:
Raphael Finkel
E-mail:
raphael@cs.uky.edu

1 Introduction

Recent research into the nature of morphology suggests that the definitions of a natural language's inflectional system must be both inferential and realizational (Stump, 2001). A definition is *inferential* if it represents inflectional exponents as markings associated with the application of rules by which complex word forms are deduced from simpler roots and stems; an inferential definition of this sort contrasts with a *lexical* definition, according to which an inflectional exponent's association with a particular set of morphosyntactic properties is listed in the lexicon, in exactly the way that the association between a lexeme's formal and contentive properties is stipulated. A definition of a language's inflectional system is *realizational* if it deduces a word's inflectional exponents from its grammatical properties; a realizational definition contrasts with an *incremental* definition, according to which words acquire morphosyntactic properties only by acquiring the morphology which expresses those properties.

The conclusion that inflectional systems should be defined realizationally rather than incrementally is favored by a range of evidence, such as the widespread incidence of extended exponence in inflectional morphology and the fact that a word's inflectional exponents often underdetermine its morphosyntactic content (Stump, 2001). Moreover, inferential–realizational definitions can avoid certain theoretically unmotivated distinctions upon which lexical or incremental definitions often depend. For instance, inferential–realizational definitions do not entail that concatenative and nonconcatenative morphology are fundamentally

Literary and Linguistic Computing, Vol. 22, No. 2, 2007.
doi:10.1093/llc/fqm004

different in their grammatical status; they do not necessitate the postulation of any relation between inflectional markings and morphosyntactic properties other than the relation of simple exponence; and they are compatible with the assumption that a word form's morphological representation is not distinct from its phonological representation.

Various means of defining a language's inflectional morphology in inferential–realizational terms are imaginable. In an important series of articles (Corbett and Fraser, 1993; Fraser and Corbett 1995, 1997), Greville Corbett and Norman Fraser proposed Network Morphology, an inferential–realizational morphological framework that makes extensive use of nonmonotonic inheritance hierarchies to represent the information constituting a language's inflectional system. Analyses in Network Morphology are implemented in DATR, a formal language for representing lexical knowledge designed and implemented by Evans and Gazdar (1989). In recent work, we have extended DATR, creating KATR, which is both a formal language and a computer program that generates desired forms by interpreting definitions in that language.

In this article, we present an inferential–realizational definition of Biblical Hebrew verb morphology in the KATR formalism. As we do so, we propose and name general strategies for exploiting the capabilities of nonmonotonic inheritance hierarchies in accounting for the properties of 'root-and-pattern' verb inflection in Hebrew.

Although we advocate here an inferential–realizational approach to morphological generation, we acknowledge that finite-state approaches to the analysis and generation of Semitic languages, such as those undertaken by Beesley (1996, 1998) and Yona and Wintner (2005), have also been very successful. Our approach is not necessarily preferable to such finite-state approaches on purely computational grounds. Nevertheless, we believe that our approach has one important advantage over finite-state approaches, namely its nonmonotonicity: It readily accommodates the straightforward formulation of defaults and overrides.

The usefulness of nonmonotonic systems for representing morpholexical information has been recognized for millennia. Defaults and overrides play a central role in the Aṣṭādhyāyī (the systematic description of the Sanskrit language composed by the grammarian Pāṇini in the 5th century BCE), and their importance in formal linguistics has been reasserted in modern times by Kiparsky (1973) and in much subsequent work. Finite-state machines, however, are intrinsically monotonic (Koskenniemi, 1990): One cannot designate some of the arcs emanating from a given state as defaults and others as overrides of those defaults. For this reason, one must resort to special means in order to emulate the effect of defaults and overrides in finite-state approaches to morphology.

For instance, Beesley and Karttunen (2003, p. 306) propose an analysis of English plurals involving three finite-state components:

(1) a finite-state grammar that generates regular plurals for English nouns (cats, dogs, horses), overgenerating in the case of nouns whose plurals are irregular (sheeps, oxes, childs);
(2) a filter that removes the overgenerating paths from (1); and
(3) a finite-state grammar that generates irregular plurals (sheep, oxen, children).

Plurals are thus defined through the union of (3) with the composition of (1) and (2). Though this analysis mimics the effect of a default/override specification, many linguists are likely to find this manner of description somewhat cumbersome, particularly if they are accustomed to thinking of morphological systems in terms of extensive defaults and overrides. In our analysis of Hebrew verb morphology, we make frequent use of the encoding strategies of overriding (Section 3) and specializing (Section 6), both of which are facilitated by the nonmonotonicity inherent in KATR's semantics. Although it would surely be possible to devise a parallel finite-state analysis in which the effects of frequent overriding and specializing are emulated by means of frequent recourse to components comparable to (1)–(3), this analysis would be more complicated to encode and harder for readers to interpret by inspection.

2 Preliminaries

To make this article readable without knowledge of the Hebrew alphabet and vocalization, we Romanize Hebrew letters and vowels according to a modified form of Lambdin's scheme (Lambdin, 1971). Most Hebrew letters can be doubled by adding a central dot (the dagesh). We deviate from Lambdin in generally indicating dotted consonants by a dot above (*ḃ*). We write *f* instead of an undotted *p*. We mark a dagesh by itself as a raised dot (·). Hebrew has two kinds of shəwa. We mark the so-called 'moving' shəwa as ə. We either omit the 'resting' shəwa (in complete words) or denote it by ':'. We use capital letters in the instances when we need to distinguish an orthographic final form from a medial form (for letters *k*, *m*, *n*, *p*, *ṣ*). We do not mark stress when it is on the ultima; in all other cases we mark stress with an acute accent (′).

Our working Hebrew KATR theory represents orthographic characters and vowels in Unicode characters (Daniels, 1993). Our KATR software is completely Unicode-compliant.

The purpose of the KATR theory described here is to generate perfect, imperfect, and participial forms of verbs belonging to various conjugations (properly called binyanim, singular binyan) in Hebrew. In particular, given a verbal lexeme L and a sequence σ of morphosyntactic properties appropriate for verbs, the theory evaluates the pairing of L with σ as an inflected-verb surface form. For instance, it evaluates the pairing of the lexeme *d-b-r* 'speak' with the property sequence `<perfect 3 sg masc>` as the surface form *ḋibēr* 'he spoke'.

Cross-cutting the categorization into binyanim is the fact that certain verb roots contain 'weak' consonants whose properties affect, often greatly, the proper surface form.[1] We have chosen to categorize verb roots primarily by binyan and secondarily by the presence of weak consonants; it seems to us that the opposite choice would lead to a much more complex theory.

Biblical Hebrew was in active use during a period of about a millennium. Vowels only began to be represented in written Hebrew under the Masoretes (Nineth and tenth century CE, Tiberias). Our KATR theory aims to faithfully represent Hebrew verb forms with the Masoretic vowels included. Verbal lexemes occasionally have multiple correct surface forms that express a given morphosyntactic property set. We have relied on Gesenius's classic text (Gesenius and Kautzsch, 1910) as our principal arbiter of surface forms. We also referred on occasion to other lists of verb forms (Bolozky, 1996; Lambdin, 1971; Pelt and Pratico, 2001). We have also tested our KATR theory against verbs in the Groves-Wheeler Westminster Hebrew Morphology (Westminster Hebrew Institute, 2005).

Hebrew verb morphology is based on a few simple facts.

- Words always begin with consonants, and consonants are invariably followed by vowels. There are no consonant clusters. (Spelling rules often remove a final shəwa vowel, and our Romanization omits the resting shəwa.)
- Hebrew verb stems generally contain three consonants. Sometimes a weak consonant (and its following vowel) are dropped from surface forms.
- Hebrew verbs generally consist of a sequence of morphological formatives, arranged in four slots:
  - Verb prefix, which realizes a verb's person, gender, and number in the imperfect.
  - Binyan prefix, which marks a verb's binyan (in some but not all binyanim), occasionally metathesizing with the first letter of the stem.
  - Stem, which comprises the three-consonant root, as well as a pattern of vocalism that is sensitive to binyan, to the presence of weak consonants, and to morphosyntactic properties.
  - Verb suffix, which realizes a verb's person, gender, and number.
- There are seven frequently encountered binyanim: **Qal**, **Nifʿal**, **Pi'ēl**, **Puʿal**, **Hifîl**, **Hofʿal**, and **Hitpaʿēl**. A single stem may appear (with different meanings) in more than one binyan. One can describe the 'spirit' of each binyan: Pi'ēl is active, Puʿal is passive, Hifîl is causative, and so forth, but these generalizations are not accurate for all verbs. We don't attempt to guess the meaning of a verb from its root and binyan.
- There are two tenses, called *perfect* and *imperfect*. It is unimportant to our purposes how these

tenses are used. Verbs in either tense can be cast in a form called the waw-consecutive, which effectively introduces two new tenses. We do not treat these latter two tenses.

- Verbs have a participial form, which is declined according to number and gender (as are all Hebrew adjectives). We only generate the masculine singular form. Verbs in the Qal have two participles, which we call *active* and *passive*. Verbs in all the other binyanim have only a single participle, which we call active in commonly active binyanim and passive in commonly passive binyanim.

3 *Ṗi'ēl* Verb *d-b-r* 'Speak'

A theory in KATR is a network of *nodes*. The network of nodes constituting our verb morphology theory is partially represented in Fig. 1. The organizational principle in this network is hierarchical: The tree structure's terminal nodes represent individual verbal lexemes, and each of the nonterminal nodes in the tree defines default properties shared by the lexemes that it dominates.

Each of the nodes in a theory houses a set of *rules*. We represent the verb *d-b-r* 'speak' by a node:

```
Speak:
1    <root> = d b r
2    {vowel2 perfect 3 sg masc} = ē
3    <> = PIEL:<ANALYZE:<"<root>">>
```

The node, named `Speak`, has three rules, which we number for discussion purposes only. KATR syntax requires that a node be terminated by a single period (full stop), which we omit here. Our convention is to name the node for a lexeme by a capitalized English word (here `Speak`) representing its meaning.

Rule 1 says that a query asking for the root of this verb should produce a three-atom result containing `d`, `b`, and `r`. Our rules assemble Hebrew words in logical order, which appears in this document as left-to-right.[2]

Rule 2 accounts for an irregularity exhibited by this particular verb. The vowel following the second stem consonant in the third person singular masculine is usually *a* in the Ṗi'ēl, but this rule overrides that default.

Rule 3 says that all other queries are to be referred to the `PIEL` node, which we introduce below. Before referring the query, this rule appends to the query the result of analyzing the root for weak consonants, which we discuss later. In this case, the analysis yields an empty set of weak-consonant markers.

A *query* is a list of atoms, such as `<root>` or `<vowel2 perfect 3 sg masc>`, addressed to a node such as `Speak`. In our theory, the atoms in queries generally represent *morphological formatives* (such as `root`, `binyanprefix`, `cons1`, `vowel2`), *morpho-syntactic properties* (such as `perfect`, `sg`, `fem`), or *surface forms* (specific orthographic characters).

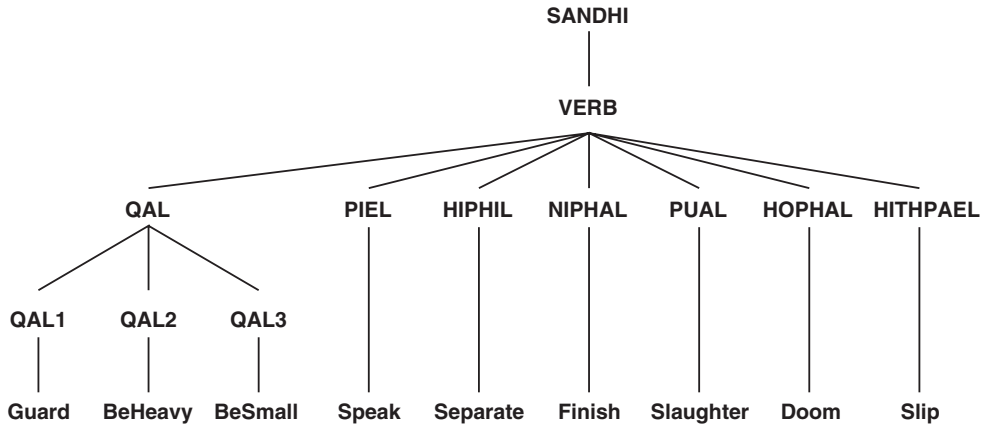

Fig. 1 A network of nodes for generating forms of verbs in seven binyanim

A query addressed to a given node is matched against all the rules housed at that node. A rule *matches* if all the atoms on its left-hand side match the atoms in the query. A rule can match even if its atoms do not exhaust the entire query. In the case of Speak, a query <root perfect> is matched by Rules 1 and 3, but not Rule 2 (which has no atom root).

Left-hand sides expressed with *path notation* (<pointed brackets>) only match if their atoms match an initial substring of the query. Left-hand sides expressed with *set notation* ({braces}) match if their atoms are all expressed, in whatever position, in the query. We usually use set notation for queries based on morphological formatives and morphosyntactic properties, where order is insignificant, but path notation for queries based on surface forms, where order is significant.

When several rules match, KATR picks the best match, that is, the one whose left-hand side 'uses up' the most of the query. This choice embodies Pāṇini's principle, which entails that if two rules are applicable, the more restrictive rule applies, to the exclusion of the more general rule. We sometimes speak of a rule's *Pāṇini precedence*, which is the cardinality of its left-hand side. If a node in a KATR theory houses two applicable rules with the same Pāṇini precedence, we consider that theory malformed.

In our case, Rule 3 of Speak applies only when Rule 1 does not apply, because Rule 1 is always a better match if it applies at all. Rule 3 is called a *default rule*, because it applies by default if no other rule applies. Default rules define a hierarchical relation among some of the nodes in a KATR theory; thus, in the tree structure depicted in Fig. 1, node X immediately dominates node Y iff Y houses a default rule that refers queries to X.

KATR generates output based on queries directed to nodes representing individual lexemes. Since these nodes, such as Speak, are not referred to by other nodes, they are called *leaves*, as opposed to nodes like PIEL, which are called *internal nodes*. The KATR theory itself indicates the list of queries to be addressed to all leaves. Here is the output that KATR generates for twenty-one queries directed to the Speak node.

```
binyan = piel
perfect,sg,3,masc = dibēr
perfect,sg,3,fem = dibərāh
perfect,sg,2,masc = dibártā
perfect,sg,2,fem = dibartí
perfect,sg,1,masc = dibártî
perfect,pl,3,masc = dibərû
perfect,pl,2,masc = dibarteM
perfect,pl,2,fem = dibarteN
perfect,pl,1,masc = dibárnû
imperfect,sg,3,masc = yədabēr
imperfect,sg,3,fem = tədabēr
imperfect,sg,2,masc = tədabēr
imperfect,sg,2,fem = tədabərî
imperfect,sg,1,masc = 'ădabēr
imperfect,pl,3,masc = yədabərû
imperfect,pl,3,fem = tədabērnāh
imperfect,pl,2,masc = tədabərû
imperfect,pl,2,fem = tədabērnāh
imperfect,pl,1,masc = nədabēr
participle,active = mədabēr
```

The rule for Speak illustrates two strategies upon which we build KATR theories.

First, a node representing a specific category (here, the Speak verb) may house a rule that overrides a rule housed at some more general node. Thus, Rule 2 at the Speak node specifies a particular vowel in a particular morphosyntactic context, overriding a default vowel for that context specified by a rule at the PIEL node. We call this strategy *overriding*.

Second, a node representing a specific category may provide information (here, the consonants of the verb's root) needed by but not provided by more general nodes (here, PIEL and the nodes to which it, in turn, refers). We call this strategy *provisioning*. As we see below, rules in the more general nodes refer to provisioned information by means of quoted paths.

We refer to the individual segments of a morphological formative by means of particular atoms:

- the atoms finiteconsonantprefix and finitevowelprefix refer to the segments of a binyan prefix;

- the atoms cons1, vowel1, cons2, vowel2, shortVowel2, and cons3 refer to the segments of a stem;
- the atom vowel3 refers to the vocalic segment of a verb suffix;
- the atom lengthener denotes the vowel separating the stem from the verb suffix.

4 The PIEL Node

We now turn to the PIEL node, to which the Speak node refers. We first show a simplified version without weak-consonant rules.

```
PIEL:
 1    <> = VERB
 2    {binyan} = piel
 3    {finitevowelprefix perfect} =
 4    {vowel1 imperfect} = a
 5    {vowel1 perfect} = i
 6    {cons2} = ROOT2:<"<root>"> ·
 7    {vowel2 imperfect} = ē
 8    {vowel2 perfect} = a
 9    {participleprefix} = m :
10    {partvowel1} = a
11    {partvowel2} = ē
12    {participle passive} = -
```

As with the Speak node, PIEL defers most queries to its parent, in this case the node called VERB, as Rule 1 indicates.

Rule 2 answers the binyan query. We only address this query to leaf nodes; the Speak node defers it to PIEL, which answers the query.

Rule 3 is an overriding rule. The VERB node specifies that the verb-prefix slot has an empty consonant and a shəwa vowel. Rule 3 blocks the appearance of that vowel in a _pi‘ēl verb's perfect forms; an empty right-hand side in a rule means that the result of a matching query is the empty string.

The VERB node makes no assumptions about the vowels following the first two stem consonants, because they are very binyan-specific. Rules 4, 5, 7, and 8 are provisioning rules that specify those vowels for the two tenses. For clarity, our convention is to place any atom referring to a morphological formative first in a set, followed by atoms referring to morphosyntactic properties in any order.

Rule 6 overrides a rule that VERB introduces, according to which the second stem consonant is the second root consonant. _Pi‘ēl verbs double their second consonant by applying a dagesh. The term on Rule 6's right-hand side is a node name (ROOT2), a colon, and new query to present to that node. The new query involves a quoted path, "<root>". KATR treats quoted paths in this context as queries to the node to which the original query was first addressed, that is, Speak. In our case, the right-hand side of this rule is equivalent to ROOT2:<d b r>, because of the first rule in the Speak node.

ROOT2 itself is one of a family of three nodes each of which isolates a particular consonant in a verb's triliteral root.

```
#vars $consonant:  ’ b g d h w v z ḥ ṭ y
 k K l m M n N s ‘ p P ṣ Ṣ q r ś š t .
ROOT1:
1    <$consonant#1> = $consonant#1
ROOT2:
1    <$consonant#1 $consonant#2>
     = $consonant#2
ROOT3:
1    <$consonant#1 $consonant#2
  $consonant#3> = $consonant#3
```

The declaration #vars $consonant identifies a particular class of atoms (that of consonant characters) as the domain of the variable $consonant. Each of the three ROOT nodes has a single rule that matches one, two, or three consonants of a root, selecting one of those consonants as its result.

These three nodes follow a third strategy for writing KATR theories: A node (such as ROOT1) may be invoked solely to provide information (here, a particular consonant in a verb's root) needed by other rules. We refer to this strategy as *lookup*. Lookup nodes do not participate in the hierarchical relationships defined by the network's default rules.

The PIEL node is a typical binyan node: it specifies binyan-sensitive components of a verb's morphology, primarily the vocalism of the stem.

We turn briefly to the HOPHAL node as a point of comparison.

```
HOPHAL:
1     <> = VERB
2     {binyan} = hophal
3     {finiteconsonantprefix perfect} = h
4     {finitevowelprefix} = o
5     {vowel1} = :
6     {vowel2} = aa
7     {participle active} = -
8     {participleprefix} = m o
```

In the *hof'al*, the vowels of the stem are independent of tense, leading to a simple description. The only unusual feature of this node is in Rule 6, where we introduce a morphophoneme aa where one might expect the vowel a. When we apply sandhi rules (described later), we usually convert aa to a, but before the combination ': (alef-shəwa), we convert aa to o. Alef is one of the weak consonants that forces vowel shifts; we cover weak consonants in detail later. We introduce several morphophonemes like aa as if they were surface-form components; we make sure that sandhi postprocessing converts them to legitimate forms. We call this strategy *morphophoneme introduction*.

5 The VERB Node

Queries addressed to Speak are generally deferred to its parent, PIEL, which then defers them further to VERB.

```
VERB:
 1    {analysis} = ANALYZE:<"<root>">
 2    {finiteconsonantprefix} =
 3    {finitevowelprefix} = :
 4    {cons1} = ROOT1:<"<root>">
 5    {cons2} = ROOT2:<"<root>">
 6    {cons3} = ROOT3:<"<root>">
 7    {shortVowel2} = :
 8    {vowel3} = :
 9    {lengthener} = :
10    {1 fem} = !
11    {3 perfect pl fem} = !
12    <> = SANDHI:<⌈ VERBPREFIX
        "<finiteconsonantprefix>"
        "<finitevowelprefix>"
        STEM VERBSUFFIX ⌉>
13    {participleprefix} =
14    {partvowel1} = "<vowel1 perfect>"
15    {partvowel2} = "<vowel2 perfect
        3 sg masc>"
16    {participle} = SANDHI:<⌈
        "<participleprefix>" "<cons1>"
        "<partvowel1>" "<cons2>"
        "<partvowel2>" "<cons3>" : ⌉>
```

Rule 1 computes the answer to the query analysis, which reports the presence of weak consonants. It uses the same method as the Speak node: It sends a root query to the starting point (here, Speak) and sends the result (here, d b r) as a query to ANALYZE, which we discuss in full when we discuss weak consonants.

Rules 2 and 3 establish the default values for the verb-prefix slot. Most binyanim have an empty verb prefix. Of the rest, most use a shəwa (:) as the vowel. We have chosen a compromise, having VERB specify an inconsistent default that contains a vowel but no consonant. Each binyan node must override one or the other of these two specifications.

Rules 4–6 of VERB establish defaults for the three stem consonants. Individual binyan nodes can override these defaults. In our case, <cons2> has been determined by Rule 6 at the PIEL node, but the other consonants have not. That is, if we pose the query Speak:<cons2>, the Speak node defers it to the PIEL node, which resolves it. But the query Speak:<cons3> is not resolved by PIEL; it is deferred to VERB, which resolves it now by means of Rule 6.

Rule 7 introduces a provisioning. In particular, it specifies shəwa as the value of the query shortVowel2; in this way, the STEM lookup node invoked in Rule 12 can define a verb's stem with reference to this provisioned value.

Rules 8 to 9 provision defaults that the VERBSUFFIX lookup node needs when Rule 12 invokes it. The atom vowel3 represents the vowel separating the stem from the verb suffix (in many, but not all, combinations of tense, gender, number, and person). The atom lengthener represents that separating vowel in the imperfect feminine plural second and third person.

Rules 10–11 are failure rules indicating that we do not wish to provide any result at all for certain queries; when these rules apply, the ! on the right-hand side causes the query to fail. In particular, both the feminine first person and the perfect feminine plural third person are always identical with their masculine counterparts, so we choose not to compute them. (Nevertheless, the Hebrew theory is built so that it computes the correct forms even if we remove these two rules.) We can also use ! to indicate that a particular paradigm is defective; we don't use this feature in our Hebrew theory.

In computing the contents of the four slots constituting a verb's form, Rule 12 evaluates the morphological-formative queries "<finitevowel prefix>" and "<finitevowelprefix>" (both of which are addressed to the original query node Speak by virtue of being quoted paths) and invokes VERBPREFIX, STEM, and VERBSUFFIX. The query addressed to these three nodes is the query originally addressed to Speak and subsequently passed by default rules to PIEL and now VERB. Rule 12 concatenates the results of these queries, surrounds the result with brackets (shown here as ⌈ and ⌉), and passes the entire collected contents to the SANDHI node, which makes euphonic and spelling adjustments. We discuss each of these other nodes below.

This rule demonstrates two more strategies for programming KATR theories, which we call *combining* (concatenating parts) and *postprocessing* (invoking a node to manipulate a surface form).

6 Nodes for Verb Slots

Verbs in the imperfect take personal prefixes.

```
VERBPREFIX:
1    <> =
2    {imperfect} = t
3    {imperfect 1 sg} = ’
4    {imperfect 1 pl} = n
5    {imperfect 3 masc} = y
```

We do not include the vowel following the prefix consonant as part of this node, but rather as part of VERB, because the prefix vowel, unlike the prefix consonant, is sensitive to binyan.

Rule 1 indicates that for all queries containing the morphosyntactic-property atom perfect, there is no verb prefix. This single rule concisely covers many cases, implicitly included because the morphosyntactic-property atoms pertaining to number, person, and gender are omitted. The other rules all apply to the imperfect tense. In the first and second person, the prefix is independent of gender; in the third person masculine, the prefix is independent of number.

Rule 2 covers the six imperfect tense situations, such as feminine third singular, not explicitly covered by more restrictive rules of Rules 3–5. This set of rules demonstrates the *specializing* strategy for building KATR theories: A rule may show an exception to a more general pattern introduced by another rule housed at the same node. (This strategy embodies, in effect, Pāṇini's principle.) Specializing can reduce the number of rules in a node, although perhaps with a loss of clarity. In our case, we could show nine restrictive rules for the imperfect, covering all combinations of person and gender. Instead, we show only three.

Suffixes appear in both tenses. In the VERBSUFFIX node, we include the vowel that separates the suffix from the stem, although this vowel may be sensitive to binyan.

```
VERBSUFFIX:
 1   {perfect 1 sg} = "<vowel3>" t î @
 2   {perfect 2 sg masc} = "<vowel3>"
      t ā @
 3   {perfect 2 sg fem} = "<vowel3>" t :
 4   {perfect 3 sg masc} = :
 5   {perfect 3 sg fem} = āh
 6   {perfect 1 pl} = "<vowel3>" n û @
 7   {perfect 2 pl masc} = "<vowel3>"
      t e m
 8   {perfect 2 pl fem} = "<vowel3>"
      t e n
 9   {perfect 3 pl} = û
10   {imperfect sg} = :
11   {imperfect 2 sg fem} = î
12   {imperfect 1 pl ++} = :
13   {imperfect pl masc} = û
14   {imperfect pl fem} = "<lengthener>"
      n āh @
```

Rules 1, 2, 6, and 14 include the @ character, which is a prosodic diacritic that indicates that the given syllable should not be accented. Hebrew words are generally accented on the ultima; we place @ on the ultima so that sandhi postprocessing can situate the accent on the penultima.

The left-hand side of Rule 12 includes the symbol ++. In the terminology of Stump, the ++ symbol identifies rules that apply in 'expanded mode' (Stump, 2001). This symbol overrides the usual Pāṇini precedence, marking this rule as preferable to any other matching rule. The situation arises for the query <imperfect pl 1 masc>, for instance. Both Rules 12 and 13 match this query, but we want Rule 12 to take precedence. Another way we could have represented this situation is by restricting Rule 13 to second or third person, either by explicitly indicating these morphosyntactic properties or by adding the negative morphosyntactic-property atom !1, which means 'not first person'. Still another way would be to assume that gender distinctions are inoperative in the first person, so that the query <imperfect pl 1 masc> (unlike <imperfect pl 1> and <imperfect pl masc>) would be ill-formed.

Those verb suffixes that begin with vowels are not separated from the stem by vowel3 (Rules 4, 5 and 9–13). Two of those suffixes are effectively zero (Rules 4 and 10); following the principle that every consonant must have a following vowel, we introduce a shəwa here. Spelling rules (part of sandhi postprocessing) often suppress this final shəwa.

Verb suffixes that begin with consonants generally use vowel3 as a separator (Rules 1–3 and 6–8). One situation (Rule 14) uses a potentially different separating vowel with subquery atom lengthener. Each of these subqueries is delimited by quote marks, addressing them to the start node (Speak in our case) and allowing them to be satisfied by the most specific node (lowest node in the tree) that has a matching rule. Node VERB provides shəwa as the default value for both these subquery atoms. The definitions of verb forms containing weak consonants occasionally override these defaults.

The STEM node assembles the components of a verb's stem.

```
STEM:
1    <> = "<cons1>" "<vowel1>"
     "<cons2>" <anyvowel2> "<cons3>"
2    <anyvowel2> = "<vowel2>"
3    {anyvowel2 perfect 3 sg fem}
      = "<shortVowel2 perfect>"
4    {anyvowel2 3 perfect pl}
      = "<shortVowel2 perfect>"
5    {anyvowel2 imperfect 2 sg fem}
      = "<shortVowel2 imperfect>"
6    {anyvowel2 imperfect !1 pl masc}
      = "<shortVowel2 imperfect>"
```

Rule 1 uses combining to assemble the stem components, alternating all the consonants and vowels. It omits the third vowel, which 'belongs' to the VERBSUFFIX node.

Most of these component subqueries are surrounded in quote marks, addressing them to the start node (Speak).

However, instead of <vowel2>, this rule queries <anyvowel2> without quote marks. The absence of quote marks addresses this query to the current node, that is, STEM; the remaining rules determine what vowel is appropriate.

Rule 2 indicates that in most cases, anyvowel2 is determined by addressing a vowel2 subquery to the start node. In verb forms whose endings are comprised only of a long vowel, the penultimate vowel is shortVowel2; usually shəwa (provisioned by the VERB node), but with a different, overriding value at the HIPHIL binyan node and in the definition of certain verb forms containing weak consonants.

7 Postprocessing

Many languages have rules of euphony. These rules are often called *sandhi* operations, based on a term from Sanskrit grammar. We use the node SANDHI to introduce rules for euphony, spelling, and placing explicit penultimate accent.

Sandhi rules often depend on classes of consonants and vowels. We begin by introducing the

necessary classes, starting with the $consonant class introduced earlier.

```
#vars $consonant:  ’  b g d h w z ḥ ṭ y
  k K l m M n N s ‘ f F ṣ Ṣ q r ś š t .
   #vars $finalConsonant:  K M N F Ṣ .
   #vars $hasFinalConsonant: k m n f ṣ.
   #vars $noFinalConsonant:  $consonant
    -$hasFinalConsonant-
       $finalConsonant .
   #vars $begedkefet:  b g d k f t .
#vars $dagesh:  · .
#vars $guttural:  ’  h ḥ ‘ .
#vars $vowel:  a aa aaa ā āh eh : û wō u
   i î ō e ē ŏ ă ĕ ō’ ā’ e’ ē’ ə o ey ēy.
   #vars $spelledVowel:  āh eh û wō î ō’
    ā’ e’ ē’ ey ēy.
   #vars $unSpelledVowel:
    $vowel-$spelledVowel.
   #vars $SN: ə .
   #vars $longVowel: ā û î ō ē ō’wō ē’
    ey ēy.
   #vars $soundAVowel:  a ā aa aaa .
   #vars $nonAVowel:$vowel-$soundAVowel.
   #vars $hatafVowel:  ŏ ă ĕ .
   #vars $unaccentableVowel: : ə ŏ ă ĕ .
   #vars $accentableVowel:
    $vowel-$unaccentableVowel.
#vars $accent:  ´ @ .
#vars $letter:  $vowel + $consonant
   + $accent + $dagesh.
#vars $noAccent: $letter -
   $accentableVowel - $accent .
```

Each class contains a subset of the Hebrew characters (here shown Romanized). Some vowels are represented by digraphs, such as those that involve a mater lectionis (consonant letter acting as part of a vowel sign); indeed, most members of the class spelledVowel are digraphs.

The first three classes are defined by enumeration. The fourth class, $noFinalConsonant, is defined in terms of previously defined classes; specifically, all consonants except those that have final forms, and the final forms themselves. We use combinators + and - to define other segmental classes as well. KATR does not let us mix the two combinators in one definition.

The SANDHI node is the longest and most complex of the entire Hebrew KATR theory. A query to this node is a complete surface-form string, delimited by ⌈ and ⌉. Within the delimiters, there is a strict alternation of ‘enlarged consonants’ and ‘enlarged vowels’, where an ‘enlarged consonant’ consists of a consonant and an optional dagesh, and an ‘enlarged vowel’ consists of a vowel and an optional accent mark. Unlike the other nodes so far, SANDHI uses path notation for its rules, because it needs to consider the surface-form components in order and in context with each other. We present the rules a few at a time.

```
SANDHI:
1    <$letter> = $letter <>
2    < ⌉ > =
3    < ⌈ > = <>
```

SANDHI works by inspecting the segment or segments near the front of the string, modifying them, possibly outputting a resulting form, and addressing the rest of the string, possibly modified, back to itself.

Rule 1 is a default that is often overridden by other rules; it outputs the first letter, passing the remainder of the string (represented by <> on the right-hand side) to the SANDHI node for further evaluation. Rule 2 removes the prosodic diacritic ⌉ at the end of the string. Its empty right-hand side indicates that processing is finished; no further nodes are addressed. Rule 3 removes an initial ⌈. These three rules, if they were the only ones in the SANDHI node, would have the effect of converting the initial query addressed to the node into a string comprised of just the letters between the initial ⌈ and final ⌉ symbols.

```
4    <$accentableVowel#1 $noAccent*
      $accentableVowel#2 @ +1> =
        <$accentableVowel#1´ $noAccent*
         $accentableVowel#2>
5    <@> = <>
6    <$accent $letter $unaccentableVowel
     ⌉> = <$letter $unaccentableVowel ⌉>
```

Rule 4 places accents in words that contain the prosodic diacritic @, which indicates ‘do not accent

this syllable.' The left-hand side matches paths that begin with an accentable vowel, followed by any number (zero or more, indicated by the Kleene star *) of letters that cannot be accented, followed by a second accentable vowel, followed by the @ mark. The rule removes the @ and places an accent mark after the first accentable vowel matched. It then addresses the entire result back to SANDHI for further processing. This rule requires a slightly higher Pāṇini precedence than its four-element left-hand side would give it; so we artificially add one to that precedence. Rule 5 removes the prosodic diacritic @ in the few cases where it is not removed by Rule 4.

Rule 6 is a spelling rule: An explicit accent mark is not shown on the last syllable, because this position ordinarily holds the accent.

```
 7    < ⌈ $consonant $dagesh? :/ə> =+= <>
 8    < : $consonant#1 $dagesh?
        :/ə $consonant#2> =+= <>
 9    <$longVowel $consonant :/ə +1>=+=<>
10    <$dagesh :/ə> =+= <>
11    <$consonant#1 $dagesh?
        :/ə $consonant#1> =+= <>
12    < ə/: $consonant : ⌉> =+= <>
```

These rules convert the resting shəwa (:) to a moving shəwa (ə). They all use a KATR shorthand. The =+= separator retains all the matched atoms of the left-hand side and places them inside the path on the right-hand side. Substitutions are marked with the / sign. So the rules replace the resting shəwa with a moving shəwa (or vice-versa) under various conditions and then address the result back to the SANDHI node. The ? marker on a variable (such as $dagesh?) means that this element is optional; if a matching atom is present in the query, it is matched, but if not, the rest of the match can still proceed. We indicate enlarged consonants by allowing for an optional dagesh.

The first five rules convert a resting shəwa to a moving shəwa, when (Rule 7) the shəwa is on the first consonant of a word, (Rule 8) the shəwa is the second in a row, (Rule 9) the shəwa follows an unaccented long vowel, (Rule 10) the shəwa is on a consonant that has a dagesh, and (Rule 11) the shəwa is on the first of two identical consonants. Finally, Rule 12 reverts a moving shəwa to a resting shəwa near the end of a word that ends with a shəwa.

```
13   <⌈ $letter#1 :/ i $letter#2 :> =+= <>
14   <⌈ ’ :/e $letter : > =+= <>
15   <⌈ ’ : y : > = <⌈ ’ î>
16   <⌈ $letter i y :> = <⌈ $letter î>
17   <⌈ $letter#1 :/i $letter#2 · > =+=<>
18   <⌈ ’ :/e $letter · > =+= <>
```

These rules modify a shəwa near the start of a word. Rule 13 replaces the first of two initial shəwas with the i vowel. But Rule 14 overrides that rule if the first consonant is an alef (ʼ); it inserts an e vowel. Both these rules have five components on the left-hand side, but KATR treats Rule 14 as more specific, because an explicit atom (like ’) is more specific than a variable (like $letter#1). Rule 15 is still more specific, inserting the vowel $\hat{i}$ if the first consonant is ’ and the second is *y*.

Rule 16 converts the pattern *iy :* to $\hat{i}$ near the start of a word. Rule 17 replaces shəwa with the vowel *i* in a word's initial syllable if the following consonant has a dagesh. Rule 18 overrides that rule, using the vowel *e* as a replacement if the first consonant is ʼ.

```
19    <$unaccentableVowel ⌉> =
20    <K $unaccentableVowel ⌉> = K :
21    <t · : ⌉> = t · :
22    < : $consonant : ⌉> = :
      FINAL:<$consonant> :
```

These three spelling rules pertain to shəwa at the end of a word. Generally (Rule 19), a final shəwa of any sort is suppressed, but not if it follows *K* (Rule 20) or *ṫ* (Rule 21) or if the previous vowel is also shəwa (Rule 22). This last rule replaces the last consonant by its final form by addressing it to the FINAL node, which we omit here.

```
23    <$hasFinalConsonant $dagesh?
        $unSpelledVowel? ⌉> =
        <FINAL:<$hasFinalConsonant>
        $dagesh? $unSpelledVowel? ⌉>
```

Rule 23 converts the last consonant of a word to its final form unless the final vowel is spelled out (with a mater lectionis).

```
24    <$accentableVowel $accent?
        $consonant · : ⎦> =
      <$accentableVowel $accent?
      $consonant : ⎦>
```

Rule 24 is another spelling rule: A dagesh in a word's last consonant is omitted unless it is preceded by an unaccentable vowel (typically a shəwa).

```
25  <$vowel $accent n : n>
      = $vowel $accent n · <>
26  <⎡ $consonant#1 $vowel n :
      $consonant#2> = <⎡ $consonant#1
      $vowel $consonant#2 · >
27  <⎡ $consonant#1 o n : $consonant#2 >
    = <⎡ $consonant#1 u  $consonant#2 ·>
```

The consonant *n* following an accented vowel and followed by a shəwa combines with a neighboring *n* to form *ṅ* (Rule 25). Near the start of a word, the consonant *n* can drop out entirely, placing a dagesh on the next consonant (Rule 26). More specifically, if the first vowel is *o*, it changes to *u* as the *n* drops out (Rule 27).

```
28  <⎡ $begedkefet> = <$begedkefet · >
29  < : $begedkefet > =: <$begedkefet · >
30  < : $begedkefet : ⎦ +1> =
    : FINAL:<$begedkefet> · :
```

The six letters *bgdkft*, classed as `begedkefet`, always acquire a dagesh at the start of a word (Rule 28) and after a resting shəwa (Rule 29), changing to final form, if necessary, if at the end of a word (Rule 30).

```
31  <aa> = <a>
32  <aa $guttural :> = <aa $guttural ă>
33  <aaa> = <a>
34  <aaa $accent? ’ :> = ā’ <$accent?>
35  <$guttural ə> = <$guttural ă>
36  <ē ’ ə ⎦> = ē’
37  <ā $guttural :> = ā $guttural <ŏ>
38  <o $guttural :> = ā $guttural <ŏ>
39  <i $guttural :> = a $guttural <ă>
40  <e $guttural :> = e $guttural <ĕ>
41  <$hatafVowel $consonant $dagesh? :>
      = <FULL:<$hatafVowel>
      $consonant $dagesh? :>
42  <i $guttural ·> = <ē $guttural>
43  <e $guttural ·> = <ē $guttural>
44  <i ’ :> = <ō’>
45  <e ´ ’ :> = <e’ ´>
46  <a $accent? ’  :> = ē’ <$accent?>
47  <ō $accent? ’ : ⎦> = ā’ <$accent? ⎦>
48  <’ e ’ :> = ’ ō <>
49  <i r ·> = <ē r >
50  <a r ·> = <ā r >
51  <u r ·> = <ō r >
52  <a ḥ : t : ⎦> = a ´ ḥ a t · :
53  <a ‘ : t : ⎦> = a ‘ a t · :
54  <ē $accent? ḥ :> = <a $accent? ḥ :>
```

Rules 31–54 deal with vowel shifts near guttural consonants. The morphophoneme `aa` is usually realized as the vowel *a* (Rule 31), but before a guttural, it is considered long, and it forces a ḥataf vowel a on the guttural (Rule 32). Similarly the morphophoneme `aaa` is usually realized as *a* (Rule 33), but it combines with an alef (’) to form the spelled vowel *ā′* (Rule 34).

Gutturals are not allowed to have a moving shəwa; that vowel always becomes *ă* (Rule 35) except on a final alef, in which case the vowel combines with the alef to form a spelled vowel (Rule 36).

A resting shəwa on a guttural becomes a ḥataf vowel, sometimes adjusting the previous vowel (Rules 37–40). However, a ḥataf vowel before a resting shəwa converts to its full form (Rule 41, invoking node `FULL`, which we omit).

Gutturals cannot accept a dagesh. Instead, the preceding vowel lengthens (Rules 42 and 43).

In several cases, an alef combines with the preceding vowel to form a spelled vowel, sometimes with a sound adjustment (Rules 44–47). In another case, two alefs in a row reduce to a single alef with a vowel adjustment (Rule 48).

Although the consonant *r* is not guttural (it may accept a moving shəwa), it shares the behavior of gutturals in refusing a dagesh, adjusting the previous vowel (Rules 49–51).

The guttural consonants ʿ and ḥ acquire a full vowel a instead of a in specific circumstances (Rules 52 and 53), and in one rare case, the ḥ forces the previous vowel to *a* (Rule 54).

```
55    <$nonAVowel $accent? ʿ : ⎤>
        = $nonAVowel ʿ a
56    <$nonAVowel $accent? ḥ : ⎤>
        = $nonAVowel  ḥ a
```

A furtive vowel *a* is required on final consonants ʿ and ḥ (Rules 55 and 56). Although this vowel is placed after the consonant in the standard orthography, it is actually pronounced before the consonant.

```
57  <⎡ $consonant i w :>
      = <⎡ $consonant w ō>
58  <⎡ $consonant a w :>
      = <⎡ $consonant w ō>
59  <⎡ $consonant o w :> = <⎡ $consonant û>
60  <⎡ $consonant : w ē> = <⎡ $consonant ē>
61  <⎡ w> = <⎡ y>
```

Rules 57–61 deal with cases where the first stem consonant is the weak letter *w*, which assimilates into the surrounding vowels in specific ways. We choose to let sandhi rules cover these cases, although we could instead use the more weak-letter specific techniques we discuss later.

8 Weak Consonants

Weak consonants appearing in a root often cause modifications of nearby vowels. Sometimes the weak consonant disappears entirely. Sometimes the effect of a weak consonant is predictable independently of the binyan, but often the weak consonant and the binyan interact in complex ways.

The following chart lists the categories of weak consonants that may appear as the first, second, or third consonant of a root.

```
y1   y in position 1
n1   n in position 1
w1   w in position 1
g1   ', h, ḥ, or ʿ in position 1
w2   w in position 2
y2   y in position 2
g2   ', h, ḥ, or ʿ in position 2
a3   ' in position 3
y3   y in position 3
g3   ḥ or ʿ in position 3
mg   second and third consonant
     identical
```

We considered several approaches for including weak consonants in the Hebrew KATR theory. The first is to make subcategories of each binyan, with a node, for instance, PIEL-G2. The advantage of this idea is that the ordinary PIEL node can still provide defaults, but the more specific PIEL-G2 node can provision them specifically for the g2 case. Unfortunately, a single word can carry several weak consonants, leading to an unwieldy set of nodes such as PIEL-G2-A3.

A second approach is to discard binyan as the primary classifier and to use weak-consonant category instead. However, there is much more commonality within all the verbs of a binyan than among all verbs with a particular weak-consonant category.

A third approach is to introduce morphophonemes for weak consonants and to let the sandhi postprocessing make the necessary changes. We gave up on this approach when the number of morphophonemes and the complexity of the resulting sandhi grew too large.

We settled on a fourth approach: to add morphophonological diacritics such as w2 to the morphosyntactic queries addressed to binyan nodes. For instance, instead of addressing <imperfect sg 2 fem> to the PIEL node for the root *k-w-m*, we address <imperfect sg 2 fem w2> to that node. All w2-specific rules in PIEL override the more general rules, typically because they have higher Pāṇini precedence.

We deduce the morphophonological diacritics appropriate for a given root by referring to the ANALYZE node.

```
ANALYZE:
 1    <y $letter#2 $letter#3>
        = y1 <nc $letter#2 $letter#3>
 2    <n $letter#2 $letter#3>
        = n1 <nc $letter#2 $letter#3>
```

```
 3    <w $letter#2 $letter#3>
       = w1 <nc $letter#2 $letter#3>
 4    <$guttural $letter#2 $letter#3>
       = g1 <nc $letter#2 $letter#3>
 5    <$letter#1 w $letter#3>
       = w2 y2 <$letter#1 nc $letter#3>
 6    <$letter#1 y $letter#3> =
       y2 <$letter#1 nc $letter#3>
 7    <$letter#1 $guttural $letter#3>
       = g2 <$letter#1 nc $letter#3>
 8    <$letter#1 $letter#2 ’ ++>
       = a3 <$letter#1 $letter#2 nc>
 9    <$letter#1 $letter#2 h ++>
       = y3 <$letter#1 $letter#2 nc>
10    <$letter#1 $letter#2 $guttural>
       = g3 <$letter#1 $letter#2 nc>
11    <$letter#1 $letter#2 $letter#2>
       = mg <$letter#1 $letter#2 nc>
12    <> =
```

Each of these rules detects one category of weak consonant, replacing the consonant with the atom nc (for neutral consonant) and then addressing the result back to the same node to find further categories. Rule 5 reports both w2 and y2, because these two categories share many peculiarities.

When we address a morphosyntactic query to a lexeme node like Speak (Rule 3, Section 3), the lexeme node first addresses a query built of the three root consonants to the node ANALYZE, which augments the set of morphosyntactic properties of the original query with a set of morphophonological diacritics. Then the lexeme node addresses the resulting query to a binyan node such as PIEL.

The full PIEL node, including all rules involving morphophonological diacritics, is as follows. We use selective indentation to assist the reader in grouping related rules.

```
PIEL:
 1   <> = VERB
 2   {binyan} = piel
 3     {binyan w2} = polel
 4     {binyan mg} = poel
 5   {finitevowelprefix perfect} =
 6   {vowel1 imperfect} = a
 7   {vowel1 perfect} = i
 8     {vowel1 w2 ++} = ô
 9     {vowel1 mg ++} = ô
10   {cons2} = ROOT2:<"<root>"> ·
11     {cons2 w2} = ROOT3:<"<root>">
12     {cons2 mg} = ROOT3:<"<root>">
13   {vowel2 imperfect} = ē
14     {vowel2 perfect !3 y3} = î
15     {vowel2 perfect 3 y3} = āh
16     {vowel2 perfect 3 masc sg mg} = ē
17     {vowel2 imperfect g3} = a
18     {vowel2 imperfect y3} = eh
19     {vowel2 imperfect pl fem y3} = ệ
20   {vowel2 perfect} = a
21     {vowel2 perfect sg 3 masc w2} = ē
22     {shortVowel2 w2} = ă
23     {shortVowel2 mg} = ă
24     {cons3 y3} =
25   {participleprefix} = m:
26   {partvowel1} = a
27     {partvowel1 w2 ++} = ô
28     {partvowel1 mg ++} = ô
29   {partvowel2} = ē
30     {partvowel2 y3} = eh
31   {participle passive} = -
```

The w2 and mg diacritics often have similar effects on ṗi‘ēl. They change the name of the binyan to ṗôlēl and ṗô‘ēl, respectively (Rules 3 and 4). They change the first vowel to *ô* in both tenses (Rules 8 and 9). They deduce a stem’s second consonant from the third rather than the second root consonant, without adding a dagesh (Rules 11 and 12). They change the short version of the second vowel from the default *:* to *ă* (Rules 22 and 23). They change the first vowel in the participle from *a* to *ô* (Rules 27 and 28). In addition, w2 verbs occasionally change the second vowel (Rule 21).

The y3 diacritic modifies the second vowel (Rules 14, 15, 18, 19, and 30). Rule 19 is quite specific. It actually holds only for the second and third person feminine, although this restriction is not mentioned; in the first person feminine, Rule 10 of VERB (Section 5) prevents our Rule 19 from being addressed at all. Rule 24 is surprising: the third consonant completely disappears. Dropping this consonant leads to the invalid situation in which two vowels appear together. One of those vowels

will be a shəwa; we use sandhi rules (not shown) to preserve only the other vowel.

Finally, g3 verbs have a special second vowel in the imperfect (Rule 17).

The PIEL node is quite typical of binyan nodes in the way morphophonological diacritics cause the defaults to be modified. Rules for exceptional situations outnumber rules for ordinary situations. Situations arise when an entire consonant and its following (or preceding) vowel are dropped.

In contrast, rules sensitive to morphophonological diacritics appear in practically no nodes other than binyan nodes such as PIEL. A rare exception is in the VERBSUFFIX node (Section 6), which has an additional rule

```
5.1    {perfect 3 sg fem y3} = t āh
```

to override Rule 5 in the y3 situation.

A few verbs engender multiple morphophonological diacritics. For example, the verb 'to know' has root *w-d-*', placing it in both the w1 and g3 categories. In many such situations, there is no conflict between the relevant rules. When there is a conflict, the g2 and g3 rules tend to dominate the others. Thus in the QAL1 node, we have these rules:

```
1    {vowel2 imperfect g2 ++} = a
2    {vowel2 imperfect g3 ++} = a
3    {vowel2 imperfect w1} = ē
```

Here, Rules 1 and 2 take precedence over Rule 3.

9 Validation

Our Hebrew KATR theory generates all the paradigms in complete agreement with Gesenius (Gesenius and Kautzsch, 1910), with the exception of infinitives, imperatives, jussive, and personal-object suffixes. We hope to add these last features to our theory. Early experiments show that we need to introduce long-distance sandhi rules to shorten vowels that occur far before the accented syllable when words have personal-object suffixes.

Once we achieved agreement with Gesenius, we turned to the Groves-Wheeler Westminster Hebrew Morphology (WHM) (Westminster Hebrew Institute, 2005), which is a computer-readable edition of the Hebrew Bible with morphological markup. After a few days' effort, we were able to modify our KATR theory to achieve a very high level of consistency with the first 1000 verbs in the Bible. We did not need to change the structure of the KATR theory at all; we only needed to tweak a few rules and introduce a small number of new rules for unusual cases.

Our test uses the following strategy.

- Parsing the WHM. We convert the codes used by WHM into leaf nodes in the KATR theory. A typical leaf node includes rules for <root> and the default <>. The WHM also leads to the initial query, such as <imperfect pl 1 masc>. We add the morphosyntactic marker pausal or connected if the cantillation mark indicates a pausal position or the word ends with a hyphen, respectively. Originally our KATR theory did not deal with pausal and connected forms, which modify both stress and vowel choice, but we added these markers in order to cover more of verbs coded in the WHM. Some verbs are irregular. We add overrides to the default rules in the leaf nodes in those cases, including the heteroclite *h l k̲* 'walk', verbs with an irregular imperfect singular 3 masculine form, cases where Gesenius gives two alternatives and the KATR theory generates the one not chosen in the Bible, distinctions between subclasses of the Qal that the WHM does not indicate, and a few other cases.
- Modifications to the WHM. The WHM uses final forms of letters in the third position of the lemma; we convert them to medial forms. The WHM considers a third lemma letter *y* to be *h*, and it considers a first-letter *w* to be a *y*; we convert such lemmata to our conventions. In addition, we modify the lemmata of about 14 verbs to indicate a strong *v* in the second letter (as in *g v* ' 'expire'), a weak *l* that elides (as in *l k ḥ* 'take'), and other changes.
- Comparing KATR with WHM. Unlike KATR, the WHM does not distinguish between the moving and resting shəwa, so we ignore all such differences. Sometimes correctness depends on modifying the KATR output to account for sentence-level sandhi, adding or removing an initial dagesh. We accept variant vowel spellings;

KATR generally outputs full spelling, but the Bible often uses defective spellings.

Of the first 1000 verbs in the WHM, KATR gives a perfect result for 878; a further forty-six are acceptable after sentence sandhi modifications, and twenty-two more use variant spellings. Only fifty-four are unaccountable. Of these, some are unexpected pausal forms on nonpausal accents (such as *ya'avodu* in Genesis 15:14), some are inexplicable forms (such as *yodon* in Genesis 6:3 and *yozmu* in Genesis 11:6; Gesenius notes that these forms are unusual), and some KATR simply computes incorrectly. Further effort would certainly reduce the number of errors.

In conducting this test, we modified our KATR theory in a few ways. We introduced new consonants to represent a strong *v* and *n*, which do not elide, and a weak *l*, which does. We improved the sandhi rules, particularly dealing with a shəwa near the start of a word and vowel shifts near guttural consonants. We added rules for pausal and connected forms of verbs, primarily one new rule in the `STEM` node. We added and adjusted existing overriding rules for situations where a root has multiple weak consonants.

10 Comparison to Finite-state Morphological Grammars

Yona and Wintner (2005) present a finite-state approach to Hebrew grammar that has similar goals to our work. They cover a larger subset of Hebrew, including nouns, pronounds, and adjectives in addition to verbs. However, they restrict their attention to consonants. We find that the most intricate parts of Hebrew morphology lie in the vowels. Our scheme of defaults with overrides allows us to easily handle binyan-specific vocalization and vowel shifts in the presence of weak consonants. Although finite-state morphology could theoretically accomplish the same actions (after all, the problem, though large, is finite), we expect the necessary rules would be quite complex and hard to program. Yona and Wintner do not report on tests of their scheme against actual texts, although they do claim that it is in heavy use in several follow-on projects.

11 Benefits

The Hebrew KATR theory provides a fairly clear picture of the morphology of Hebrew verbs. Different users might find different aspects of it attractive.

- A *linguist* can peruse the theory to gain an appreciation for the structure of Semitic verbs in general and Hebrew verbs in particular, with all exceptional cases clearly marked either by morphophonological diacritics or by rules of sandhi, which are segregated from all the other rules.
- A *teacher* of the language can use the theory as a foundation for organizing lessons in morphology.
- A *student* of the language can suggest verb roots and use the theory to generate all the appropriate forms, instead of locating the right paradigm in a book and substituting consonants.

Our KATR theory is able to handle heteroclite verbs (Stump, 2006) such as the following:

```
Approach:
1  <root ++> = n g ś
2  {perfect} = NIPHAL:<ANALYZE:<"<root>">>
3  <> = QAL1:<ANALYZE:<"<root>">>
Lead:
1  <root ++> = n ḥ y
2  <> = QAL1:<ANALYZE:<"<root>">>
3  {imperfect}=HIPHIL:<ANALYZE:<"<root>">>
```

12 DATR, KATR, and LATR

We discuss KATR and its relation to DATR extensively elsewhere (Finkel *et al.*, 2002); here we only summarize the differences. The DATR formalism is quite powerful; we have demonstrated that it is capable of emulating a Turing machine. The KATR enhancements are therefore aimed at usability, not theoretical power. The principal innovations of KATR are:

- Set notation. The left-hand sides of DATR rules may use only list notation. KATR allows set notation as well, which allows us to deal with

morphosyntactic properties in any order. In the VERBSUFFIX node (Section 6), Rule 14 identifies *nāh* as an exponent of number, gender, and tense, but not of person; Rule 9 identifies û as an exponent of person, number, and tense, but not of gender. Both rules are indifferent to the order in which properties of person, number, tense, and gender are listed in any matching query. DATR's restriction to lists complicates these rules, forcing us to include either a variable over properties of person (Rule 14) or a variable over properties of gender (Rule 9).

- Regular expressions. KATR allows limited regular expressions in lists in left-hand sides of rules; DATR has no such expressions. We use this facility in the SANDHI node, both for the Kleene star * (Rule 4, Section 7) and for the ? operator (Rule 7, Section 7). Regular expressions often help us represent nonlocal sandhi phenomena, such as the Sanskrit rule of *n*-retroflexion (Whitney, 1889).
- Nonsubtractive rules. DATR rules have a subtractive quality: The atoms of the query matched by the left-hand side are removed from the query used for subsequent evaluation in the right-hand side. The KATR =+= operator (used, for instance, in the Sandhi node, Rule 7, Section 7) allows us to represent rules that preserve the atoms matched by the left-hand side, substituting new atoms where necessary.
- Adjusting Pāṇini precedence. Competing rules sometimes have left-hand sides of the same length, but we prefer one of the rules. KATR lets us enhance the Pāṇini precedence of a rule by using ++ (as in VERBSUFFIX Rule 12, Section 6) or +1 (as in SANDHI Rule 4, Section 7).
- Syntax. KATR allows special characters to be used as atoms if escaped by the \ character. Variables can be computed instead of being enumerated; we use this facility in defining the $letter variable. KATR allows greater control over which nodes are to be displayed under default queries. The interactive KATR program has facilities for rapid testing and debugging of theories.

We have two implementations of KATR, one in Java™, and the other a Perl (Wall and Schwartz, 1990) translator that generates a Prolog program (Sterling and Shapiro, 1986). Both can compile the Hebrew theory and evaluate all the forms of a verb in about a second.

We turned to Prolog in the hope that the declarative nature of a Prolog program would let us parse a surface form as easily as we can generate one. In other words, the same set of rules should be usable in the reverse direction. Although some simple KATR theories do, in fact, generate reversible Prolog, we discovered that KATR does not have enough information to direct the reverse search in reasonable directions. A search that follows the nonproductive path of adding more and more y3 diacritics, for example, does not succeed in parsing a y3 verb.

We have begun to design and implement LATR, a successor to KATR with strong typing. The programmer specifies the query and result types of each node; these types provide the extra information needed to parse as well as generate.

Our KATR implementations, as well as our KATR theories for Hebrew and other languages, are available under the GNU General Public License.

13 Strategies for Building KATR Theories

We have been applying KATR to natural-language morphology for several years. In addition to Hebrew, we have built a complete morphology of Latin verbs and nouns, large parts of Sanskrit (and other related languages), and smaller studies of Bulgarian, Swahili, Georgian, Lingala, Spanish, Polish, and Turkish. KATR allows us to represent morphological rules for these languages with great elegance. It is especially well-suited to cases like Hebrew verbs, where a similar structure applies across the entire spectrum of words, and where that spectrum is partitioned into binyanim with distinguishable rules, but where euphony introduces standard vowel shifts based on accent and weak consonants.

As we have gained experience with KATR, we have noted encoding strategies that apply across

language families; we used each of these in our Hebrew verb specification.

- *Specializing.* A rule introduces a specific exception to a more general pattern specified by another rule housed at the same node.
- *Overriding.* A node representing a specific category answers a query that is usually answered (with different results) by a more general node to which queries are usually referred. The strategies of specializing and overriding exploit the nonmonotonicity inherent in KATR's semantics.
- *Provisioning.* A node representing a specific category provides information needed by more general nodes to which it refers queries. Rules in the more general nodes refer to provisioned information by means of quoted queries.
- *Lookup.* A node is invoked solely to provide information needed by rules at other nodes.
- *Morphophoneme introduction.* A morphophoneme is introduced as if it were a surface-form component; postprocessing converts it (and its surrounding surface forms) to true surface forms.
- *Combining.* A rule concatenates various morphological units by addressing queries to multiple nodes.
- *Postprocessing.* The result of combining morphological units is addressed to a node that makes local adjustments. These adjustments can account for euphony and other sandhi principles, or they can convert from one format to another. (We have a `ROMANIZE` node and a `REVERSE` node for generating various styles of output.)

Writing specifications in KATR is not easy. KATR is capable of representing elegant theories, but arriving at those theories requires considerable effort. Early choices color the structure of the resulting theory, and the author must often discard attempts and rethink how to represent the target morphology.

Our Hebrew theory rests on the hypothesis that an inflected word's morphological form is determined by a system of realization rules organized in a default inheritance hierarchy. Other researchers follow competing approaches to defining Hebrew verb inflection; one could, for example, assume that an inflected word's form is determined by a ranked system of violable constraints on morphological structure, as in Optimality Theory (Prince and Smolensky, 1993), or by a finite-state machine (Karttunen, 1993). The facts of Hebrew verb inflection are apparently compatible with any of these approaches. Even so, there are strong theoretical grounds for preferring our approach. (1) It provides a uniform, well-defined architecture for the representation of both morphological rules and lexical information. (2) It embodies the assumption that inflectional morphology is inferential and realizational, readily accommodating such phenomena as extended exponence and the frequent underdetermination of morpho-syntactic content by inflectional form. In this sense, it effectively excludes a morpheme-based conception of word structure, unlike both the optimality-theoretic and the finite-state approaches.

Our KATR theory for Hebrew covers a significant fragment of the entire inflectional morphology of Hebrew verbs. It does not cover the waw-consecutive forms, which arise in a straightforward manner through the addition of prefixes to forms that we do cover (perfect and imperfect). It also does not cover the infinitive, makor, or addition of personal pronouns as suffixes marking direct objects. These matters are mostly straightforward, but suffix addition involves vowel shortening based on distance from the accented syllable that is difficult to capture precisely. For the participial forms, it is straightforward to generalize beyond the singular masculine form.

Our larger goal is to host a library of KATR theories for various languages as a resource for linguists. Such a library will provide interested researchers with morphological descriptions that can be directly converted into actual word forms and will serve as a substitute, to some extent, for voluminous natural-language and table-based descriptions. In the case of endangered languages, it will act as a repository for linguistic data that may be essential for preservation.

Acknowledgements

We would like to thank Rabbi H. D. Uriel Smith for his insights into Hebrew morphology, particularly in his explanations of some of the more obscure sandhi rules. Lei Shen and Suresh Thesayi were instrumental in implementing our Java™ version of KATR. Nancy Snoke assisted in implementing our Perl/Prolog version.

We would like to thank the reviewers of the first version of this article for suggesting we consider the finite-state transducer approach and that we use the Westminster Hebrew Morphology for testing.

We especially thank Kirk E. Lowery, Director of The Westminster Hebrew Institute, for making the Groves-Wheeler Westminster Hebrew Morphology available for our testing.

This work was partially supported by the National Science Foundation under Grant 0097278 and by the University of Kentucky Center for Computational Science. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Beesley, K. and Karttunen, L. (2003). *Finite State Morphology*. Center for the Study of Language and Information, Stanford.

Beesley, K. R. (1996). *Arabic Finite-state Morphological Analysis and Generation*. *Proceedings of COLING-96*, Copenhagen.

Beesley, K. R. (1998). *Arabic Morphology using Only Finite-state Operations*. In Rosner, M. (ed.), *Proceedings of the Workshop on Computational Approaches to Semitic languages*, Montreal, pp. 50–57.

Bolozky, S. (1996). *501 Hebrew Verbs*, Barron's Educational Series.

Corbett, G. G. and Fraser, N. M. (1993). Network Morphology: A DATR account of Russian nominal inflection. *Journal of Linguistics*, **29**: 113–142.

Daniels, P. T. (1993). The Unicode Consortium: The Unicode standard. *Language: Journal of the Linguistic Society of America*, **69**(1): 225–225.

Evans, R. and Gazdar, G. (1989). *Inference in DATR*, *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*. Manchester, pp. 66–71.

Finkel, R., Shen, L., Stump, G. and Thesayi, S. (2002). KATR: A Set-based Extension of DATR, *Technical Report 346-02*, University of Kentucky Department of Computer Science, Lexington, KY. ftp://ftp.cs.uky.edu/cs/techreports/ 346-02.pdf.

Fraser, N. M. and Corbett, G. G. (1995). Gender, Animacy, and Declensional Class Assignment: A Unified Account for Russian. In Booij, G. and van Marle, J. (eds), *Yearbook of Morphology 1994*. Kluwer, Dordrecht, pp. 123–150.

Fraser, N. M. and Corbett, G. G. (1997). Defaults in Arapesh. *Lingua*, **103**: 25–57.

Gesenius, W. and Kautzsch, E. (1910). *Gesenius' Hebrew Grammar*, Cowley, A. E. (ed.), Oxford University Press.

Karttunen, L. (1993). Finite-state Constraints. In Goldsmith, J. (ed.), *The Last Phonological Rule*. University of Chicago Press, Chicago.

Kiparsky, P. (1973). "Elsewhere" in Phonology. In Anderson, S. and Kiparsky, P. (eds), *A Festschrift for Morris Halle*, pp. 93–106.

Koskenniemi, K. (1990). Finite-state Parsing and Disambiguation. In Karlgren, H. (ed.), *Papers Presented for the 13th International Conference on Computational Linguistics*. Vol. 2, Helsinki, pp. 229–323.

Lambdin, T. O. (1971). *Introduction to Biblical Hebrew*. Charles Scribner's Sons.

Pelt, M. V. V. and Pratico, G. D. (2001). *Basics of Biblical Hebrew Grammar*. Zondervan.

Prince, A. S. and Smolensky, P. (1993). Optimality Theory: Constraint Interaction in Generative Grammar, *Technical Report RuCCs Technical Report #2*, Rutgers University Center for Cognitive Science, Piscataway, NJ.

Sterling, L. and Shapiro, E. (1986). *The Art of Prolog*. MIT Press, Cambridge, MA.

Stump, G. T. (2001). *Inflectional Morphology*, Cambridge University Press, Cambridge, England.

Stump, G. T. (2006). Heteroclisis and Paradigm Linkage. *Language*, **82**: 279–322.

Wall, L. and Schwartz, R. L. (1990). *Programming Perl*. O Reilly and Associates.

Westminster Hebrew Institute (2005). Groves-Wheeler Westminster Hebrew Morphology, Version 4.4.

Whitney, D. W. (1889). *Sanskrit Grammar*. 2nd edn, Harvard University Press, Cambridge, MA, pp. 64–66.

Yona, S. and Wintner, S. (2005). A *Finite-state Morphological Grammar of Hebrew*, *Proceedings of the ACL-2005 Workshop on Computational Approaches to Semitic Languages*, Ann Arbor, MI, pp. 9–16. http: //yeda.cs.technion.ac.il/~yona/morphgram.pdf.

Notes

1 We use the term 'weak' to refer both to consonants that tend to drop out of surface forms, such as *w* and *y*, as well as to the gutturals ʾ, *h*, *ḥ*, and ʿ, which place restrictions on nearby vowels.

2 We accomplish visual reversal for orthographic Hebrew output by rules in a `REVERSE` node (not shown in this paper) or a postprocessing step through a program that presents bidirectional text correctly.