2-22-2010

Lexical Analysis

Andrew R. Hippisley
University of Kentucky, andrew.hippisley@uky.edu

Lexical Analysis

3

Lexical Analysis

Andrew Hippisley
University of Kentucky

3.1 Introduction

Words are the building blocks of natural language texts. As a proportion of a text's words are morphologically complex, it makes sense for text-oriented applications to register a word's structure. This chapter is about the techniques and mechanism for performing text analysis at the level of the word, *lexical analysis*. A word can be thought of in two ways, either as a string in running text, for example, the verb *delivers*; or as a more abstract object that is the cover term for a set of strings. So the verb DELIVER names the set {*delivers, deliver, delivering, delivered*}. A basic task of lexical analysis is to relate *morphological variants* to their *lemma* that lies in a lemma dictionary bundled up with its invariant semantic and syntactic information. Lemmatization is used in different ways depending on the task of the natural language processing (NLP) system. In machine translation (MT), the lexical semantics of word strings can be accessed via the lemma dictionary. In transfer models, it can be used as part of the source language linguistic analysis to yield the morphosyntactic representation of strings that can occupy certain positions in syntactic trees, the result of syntactic analyses. This requires that lemmas are furnished not only with semantic but also with morphosyntactic information. So *delivers* is referenced by the item DELIVER + {3rd, Sg, Present}. In what follows we will see how the mapping between *deliver* and DELIVER, and the substring *s* and {3rd, Sg, Present} can be elegantly handled using finite state transducers (FSTs).

We can think of the mapping of string to lemma as only one side of lexical analysis, the parsing side. The other side is mapping from the lemma to a string, morphological generation. Staying with our MT example, once we have marphosyntactically analyzed a string in the source language, we can then use the resulting information to generate the equivalent morphologically complex string in the target language. Translation at this level amounts to accessing the morphological rule of the target language that

introduces the particular set of features found from the source language parse. In information retrieval (IR), parsing and generation serve different purposes. For the automatic creation of a list of key terms, it makes sense to notionally collapse morphological variants under one lemma. This is achieved in practice during *stemming*, a text preprocessing operation where morphologically complex strings are identified, decomposed into invariant stem (= lemma's canonical form) and affixes, and the affixes are then deleted. The result is texts as search objects that consist of stems only so that they can be searched via a lemma list. Morphological generation also plays a role in IR, not at the preprocessing stage but as part of query matching. Given that a lemma has invariant semantics, finding an occurrence of one of its morphological variants satisfies the semantic demands of a search. In languages with rich morphology it is more economical to use rules to generate the search terms than list them. Moreover, since morphology is used to create new words through derivation, a text that uses a newly coined word would not be missed if the string was one of many outputs of a productive morphological rule operating over a given lemma. Spelling dictionaries also make use of morphological generation for the same reason, to account for both listed and 'potential' words. Yet another application of lexical analysis is text preprocessing for syntactic analysis where parsing a string into morphosyntactic categories and subcategories furnishes the string with POS tags for the input of a syntactic parse. Finally tokenization, the segmentation of strings into word forms, is an important preprocessing task required for languages without word boundaries such as Chinese since a morphological parse of the strings reveals morphological boundaries, including words boundaries.

It is important from the start to lay out three main issues that any lexical analysis has to confront in some way. First, as we have shown, lexical analysis may be used for generation or parsing. Ideally, the mechanism used for parsing should be available for generation, so that a system has the flexibility to go both ways. Most lexical analysis is performed using FSTs, as we will see. One of the reasons is that FSTs provide a trivial means from flipping from parsing (analysis) to generation. Any alternative to FST lexical analysis should at least demonstrate it has this same flexibility. Two further issues concern the linguistic objects of lexical analysis, morphologically complex words. The notion that they are structures consisting of an invariant stem encoding the meaning and syntactic category of a word, joined together with an affix that encodes grammatical properties such as number, person, tense, etc is actually quite idealistic. For some languages, this approach takes you a long way, for example, Kazakh, Finnish, and Turkish. But it needs refinement for the languages more often associated with large NLP applications such as English, French, German, and Russian.

One of the reasons that this is a somewhat idealized view of morphology is that morphosyntactic properties do not have to be associated with an affix. Compare, for example, the string *looked* which is analyzed as LOOK+{Past} with *sang*, also a simple past. How do you get from the string *sang* to the lemma SING+{Past, Simple}? There is no affix but instead an alternation in the canonical stem's vowel. A related problem is that the affix may be associated with more than one property set: *looked* may correspond to either LOOK+{Past, Simple} or LOOK+{Past, Participle}. How do you know which *looked* you have encountered? The second problem is that in the context of a particular affix, the stem is not guaranteed to be invariant, in other words equivalent to the canonical stem. Again not straying beyond English, the string associated with the lemma FLY+{Noun, Plural} is not **flys* but *flies*. At some level the parser needs to know that *flie* is part of the FLY lemma, not some as yet unrecorded FLIE lemma; moreover this variant form of the stem is constrained to a particular context, combination with the suffix *–s*. A further complication is changes to the canonical affix. If we propose that *–s* is the (orthographic) plural affix in English we have to account for the occasions when it appears in a text as *–es*, for example, in *foxes*.

In what follows we will see how lexical analysis models factor in the way a language assigns structure to words. Morphologists recognize three main approaches to word structure, first discussed in detail in Hockett (1958) but also in many recent textbooks, for example, Booij (2007: 116–117). All three approaches find their way into the assumptions that underlie a given model. An item and arrangement approach (I&A) views analysis as computing the information conveyed by a word's stem morpheme with that of its affix morpheme. Finite state morphology (FSM) incorporates this view using FSTs. This works well for the 'ideal' situation outlined above: *looked* is a stem plus a suffix, and information that the word

conveys is simply a matter of computing the information conveyed by both morphemes. Item and process approaches (I&P) account for the kind of stem and affix variation that can happen inside a complex word, for example, *sing* becomes *sang* when it is past tense, and a vowel is added to the suffix *–s* when attached *fox*. The emphasis is on possible phonological processes that are associated with affixation (or other morphological operations), what is known as *morphonology*. Finally, in word and paradigm approaches (W&P), a lemma is associated with a table, or paradigm, that associates a morphological variant of the lemma with a morphosyntactic property set. So *looked* occupies the cell in the paradigm that contains the pairing of LOOK with {Past, Simple}. And by the same token *sang* occupies the equivalent cell in the SING paradigm. Meaning is derived from the definition of the cell, not the meaning of stem plus meaning of suffix, hence no special status is given to affixes.

FSTs have been used to handle *morphonology*, expressed as spelling variation in a text, and *morphotactics*, how stems and affixes combine, and how the meaning behind the combination can be computed. We begin with FSTs for morphonology, the historic starting point for FSM. This leaves us clear to look at lexical analysis as morphology proper. We divide this into two main parts, the model that assumes the I&A approach using FSTs (Section 3.3) and the alternative W&P model (Section 3.5). Section 3.4 is a brief overview of the types of 'difficult' morphology that the paradigm-based approaches are designed to handle but which FSM using the I&A approach can negotiate with some success too.

3.2 Finite State Morphonology

Phonology plays an important role in morphological analysis, as affixation is the addition of phonological segments to a stem. This is phonology as *exponent* of some property set. But there is another 'exponent-less' way in which phonology is involved, a kind of phonology of morpheme boundaries. This area of linguistics is known as *morphophonology* or *morphonology*: "the area of linguistics that deals with the relations and interaction of morphology with phonology" (Aronoff and Fudeman 2005: 240). Morpheme boundary phonology may or may not be reflected in the orthography. For example, in Russian word final voiced obstruents become voiceless—but they are spelled as if they stay as they are, unvoiced. A good example of morphonology in English is plural affixation. The plural affix can be pronounced in three different ways, depending on the stem it attaches to: as /z/ in *flags*, as /əz/ in *glasses* and as /s/ in *cats*. But only the /əz/ alternation is consequential because it shows up as a variant of orthographic *–s*. Note that text to speech processing has to pay closer attention to morphonology since it has to handle the two different pronunciations of orthographic *–s*, and for the Russian situation it has to handle the word final devoicing rule. For lexical analysis to cope with morphonological alternations, the system has to provide a means of mapping the 'basic' form with its orthographic variant. As the variation is (largely) determined by context, the mapping can be rule governed. For example, the suffix *–s* you get in a plural word shows up as *–es* (the mapping) when the stem it attaches to ends in a *–s-* (specification of the environment). As we saw in the previous section, stems can also have variants. For *flie* we need a way of mapping it to basic *fly*, and a statement that we do this every time we see this string with a *–s* suffix. Note that this is an example of orthographic variation with no phonological correlate (*flie* and *fly* are pronounced the same).

The favored model for handling morphonology in the orthography, or morphology-based orthographic spelling variation, is a specific type of finite state machine known as a finite state transducer (FST). It is assumed that the reader is familiar with finite state automata. Imagine a finite state transition network (FSTN) which takes two tapes as input, and transitions are licensed not by arcs notated with a single symbol but a pair of symbols. The regular language that the machine represents is the *relation* between the language that draws from one set of symbols and the language that draws from the set of symbols it is paired with. An FST that defines the relation between underlying *glass^s* (where ^ marks a morpheme boundary) and surface *glasses* is given in Figure 3.1.

Transition from one state to another is licensed by a specific correspondence between symbols belonging to two tapes. Underlying, more abstract representations are conventionally the upper tape. The colon

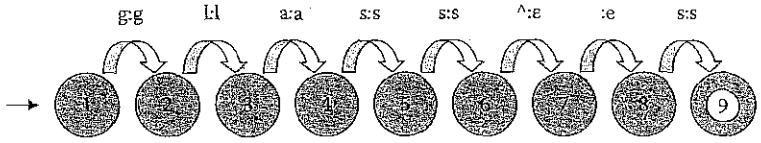

FIGURE 3.1 A spelling rule FST for *glasses*.

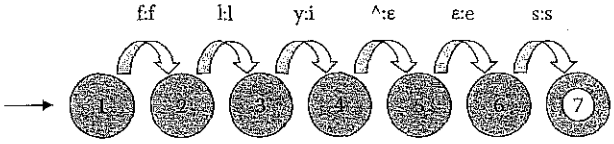

FIGURE 3.2 A spelling rule FST for *flies*.

between symbols labeling the arcs declares the correspondence. The analysis of the surface string into its canonical morphemes is simply reading the lower language symbol and printing its upper language correspondent. And generation is the inverse. Morpheme boundaries do not have surface representations; they are deleted in generation by allowing the alphabet of the lower language to include the empty string symbol ε. This correspondence of ^ to ε labels the transition from State 6 to State 7. The string *glasses* is an example of *insertion*-based orthographic variation where a character has to be inserted between stem and suffix. Since ε can also belong to the upper language its correspondence with lower language *e* provides for the insertion (State 7 to State 8). In a similar vein, an FST can encode the relation between underlying *fly*^*s* and surface *flies* (Figure 3.2). This is a combination of *substitution* based variation (the symbol *y* is substituted by *i*) and insertion based variation, if we treat the presence of *e* as the same as in the *glasses* example. Variation takes place both in the stem and in the suffix.

A practical demonstration of an FST treatment of English orthographic variation, i.e., spelling rules helps to show these points. To do this we will use the lexical knowledge representation language DATR (Evans and Gazdar 1996). DATR notation for FSTs is a good expository choice since its syntax is particularly transparent, and has been shown to define FSTs in an economical way (Evans and Gazdar 1996: 191–193).* And as we will be using DATR anyway when we discuss an alternative to finite state–based lexical analysis, it makes sense to keep with the same notation throughout. But the reader should note that there are alternative FST notations for lexical analysis, for example, in Koskenniemi (1983), Beesley and Karttunen (2003), and Sproat (1997). DATR expresses the value for some attribute, or a set of attributes, as an association of the value with a path at some node. A DATR definition is given in (3.1).

(3.1)
```
State_n:
    <g> == g.
```

Basically, (3.1) says that at a particular node, `State_n`, the attribute `<g>` has the value `g`. Nodes are in (initial) upper case, attribute sets are paths of one or more atoms delimited by angle brackets. We could think of (3.1) as a trivial single state FST that takes an input string `g`, represented by an attribute path, and generates the output string `g`, represented by the value. DATR values do not need to be explicitly stated, they can be *inherited* via another path. And values do not need to be simple; they can be a combination of atom(s) plus inheriting path(s). Imagine you are building a transducer to transliterate words in the Cyrillic alphabet into their Roman alphabet equivalents. For example, you want the FST to capture the proper name `Саша` transliterated as `Sasha`. So we would have `<С> == S, <а> == a`. For `ш` we need two glyphs and to get them we could associate `<ш>` with the complex value `s<h>`, and somewhere else provide the equation `<h> == h`. So `<ш> == s <h>` implies `<ш> == s h`. We will

* Gibbon (1987) is an FST account of tone morphonology in Tem and Baule, African languages spoken in Togo and the Ivory Coast. In a later demonstration, Gibbon showed how DATR could be used to considerably reduce the number of states needed to describe the same problem (Gibbon 1989).

see the importance of including a path as part of the value to get (3.1) to look more like a more serious transducer that maps *glass^s* to *glasses*. This is given in (3.2).

(3.2)
```
Glasses_FST:
    <g> == g <>
    <l> == l <>
    <a> == a <>
    <s> == s <>
    <s> == s <>
    <^> == e <>
    <s> == s <>
    <> == .
```

The input string is the path <g l a s s ^ s>. The path <g> that we see in the second line of (3.2) is in fact the *leading subpath* of this path. The leading subpath expresses the first symbol of the input string. It is associated with the atom g, a symbol of the output string. So far, we have modeled a transition from the initial state to another state, and transduced g to g. Further transitions are by means of the <> and this needs careful explanation. In DATR, any extensions of a subpath on the left of an equation are automatically transferred to a path on the right of the equation. So the extensions of <g> are transferred into the path <> as <l a s s ^ s>. This path then needs to be evaluated by linking it to a path on the left hand side. The path <l> in the third line is suitable because it is the leading subpath of this new path. As we can see the value associated with <l> is the atom l, so another symbol has been consumed on the input string and a corresponding symbol printed onto the output string. The extensions of <l a s s ^ s> fill the path <> on the right side. To take stock: at this point the evaluation of <g l a s s ^ s> is g l together with the extended path <a s s ^ s>. As we continue down the equation list the leading subpaths are always the next attribute atom in the input string path, and this path is given the equivalent value atom. But something more interesting happens when we get to the point where the leading subpath is <^>. Here a nonequivalent value atom is given, the atom e. This of course expresses the *e* insertion that is the essence of the spelling rule. The deletion of the ^ is represented very straightforwardly as saying nothing about it, i.e., no transduction. Finally, the equation at the bottom of the list functions to associate any subpath not already specified, expressed as <>, with a null value. Suppose we represent input strings with an end of word boundary #, so we have the lexical entry <g l a s s ^ e s #>. Through the course of the evaluation <#> will ultimately be treated as a leading subpath. As this path is not explicitly stated anywhere else at the node, it is implied by <>. So <> == is interpreted as the automatic deletion of any substring for which there is no explicit mapping statement. This equation also expresses the morphologically simple input string <g l a s s #> mapping to g l a s s. The theorem, expressing input and output string correspondences licensed by the FST in (3.2) is given in (3.3).

(3.3)
```
<g l a s s #> = g l a s s
<g l a s s ^ s #> = g l a s s e s
```

The FST in (3.2) is very useful for a single word in the English language but says nothing about other words, such *class:classes*, *mass:masses*, or *fox:foxes*. Nor does it provide for 'regular' plurals such as *cat:cats*. FSTs are set up to manage the regular situation as well as problems that are general to entire classes. To do this, symbols can be replaced by *symbol classes*. (3.4) replaces (3.3) by using a symbol class represented by the expression $abc, an abbreviatory variable ranging over the 26 lower case alphabetic characters used in English orthography (see Evans and Gazdar 1996: 192–193 for the DATR FST on which this is based).

(3.4)
```
Glasses&Classes:
    <$abc> == $abc <>
    <^> == e <>
    <> == .
```

For an input string consisting of a stem composed of alphabetic symbols, the first equation takes this string represented as a path and associates whatever character denotes its leading subpath with the equivalent character as an atomic value. Equivalence is due to the fact that `$abc` is a bound variable. If the string is the path `<g l a s s ^ s #>` then the leading subpath `<g>` is associated with the atomic value g; by the same token for the string `<c a t ^ s>` the subpath `<c>` would be associated with c. The extension of this path fills `<>` on the right hand side as in (3.2), and just in case the new leading subpath belongs to `$abc`, it will be evaluated as `<$abc> == &abc <>`. This represents a self-loop, a transition whose source and destination state is the same. In case we hit a morpheme boundary, i.e., we get to the point where the leading subpath is `<^>`, then as in (3.2) the value given is e. Whatever appears after the morpheme boundary is the new leading subpath. And since it is `<s>`, the plural affix, it belongs to `$abc` so through `<$abc> == $abc <>` will map to s. As before, the # will be deleted through `<> ==` since this symbol does not belong to `$abc`.

Whereas (3.2) undergeneralizes, we now have an FST in (3.4) which overgeneralizes. If the input is `<c a t ^ s>` the output will be incorrect `c a t e s`. A morphonological rule or spelling rule (the orthographic counterpart) has to say not only (a) what changes from one level representation to another, and (b) where in the string the change takes place but also (c) under what circumstances. The *context* of e insertion is an s followed by the symbol sequence ^s. But if we want to widen the context so that *foxes* is included in the rule, then the rule needs to specify e insertion when not just s but also x is followed by ^ s. We can think of this as a class of contexts, a subset of the stem symbol class above. Figure 3.3 is a graphical representation of a transition labeled by the symbol class of all stem characters, and another transition labeled by the class of just those symbols providing the left context for the spelling rule.

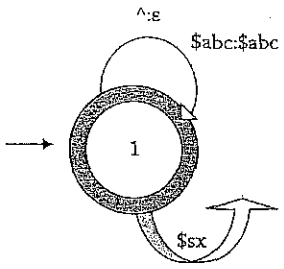

FIGURE 3.3 An FST with symbol classes.

Our (final)FST for the *–es* spelling rule in English is given in (3.5) with its theorem in (3.6). The context of e insertion is expressed by the variable `$sx` (left context) followed by the morpheme boundary symbol (right context). As 'regular' input strings such as `<c a t ^ s>`, `<d o g ^ s>`, `<c h a i r ^ s>` do not have pre-morpheme boundary s or x they avoid the path leading to e insertion.

(3.5)
```
Es_Spelling_Rule:
    <$abc> == $abc <>
    <$sx ^> == $sx e <>
    <^> == <>
    <> == .
```

(3.6)
```
<g l a s s #> = g l a s s.
<g l a s s ^ s #> = g l a s s e s.
<f o x #> = f o x.
<f o x ^ s> = f o x e s.
<c a t #> = c a t.
<c a t ^ s #> = c a t s.
```

A final comment on the spelling rule FST is in order. In (3.5) how do we ensure that the subpath `<x>` for `<f o x ^ s>` will not be evaluated by the first equation since x belongs to `$abc` as well as `$sx`? In other words, how do we 'look ahead' to see if the next symbol on the input string is a ^ ? In DATR, look ahead is captured by the 'longest path wins' principle so that any extension of a subpath takes precedence over the subpath. As `<x ^>` is an extension of `<x>`, the `<x ^>` path 'wins' and gets evaluated, i.e., it overrides the shorter path and its value. We look more closely at this principle when we use DATR to represent default inheritance hierarchies in Section 5.

3.2.1 Closing Remarks on Finite State Morphonology

Morphonological alternations at first glance seem marginal to word structure, or morphology 'proper.' In the previous discussion, we have barely mentioned a morphosyntactic feature. But their importance in lexical analysis should not be overlooked. On the one hand, text processors have to somehow handle orthographic variation. And on the other, it was earlier attempts at computationally modeling theoretical accounts of phonological and morphonological variations that suggested FSTs were the most efficient means of doing this. Kaplan and Kay in the 1980s, belatedly published as Kaplan and Kay (1994), demonstrated that the (morph)phonological rules for English proposed by Chomsky and Halle (1968) could be modeled in FSTs. Their work was taken up by Koskenniemi (1983) who used FSTs for the morphonology of Finnish, which went beyond proof of concept and was used in a large-scale text-processing application. Indeed Koskenniemi's Two-Level Morphology model is the real starting point for finite state–based analysis. Its motivation was to map the underlying lexical (= lemma) representation to the surface representation without the need to consult an intermediary level. Indeed, intermediary levels can be handled by cascading FSTs so that the output of FST1 is the input of FST2, and the output of FST2 is in the input of FST3. But then the ordering becomes crucial for getting the facts right. Koskenniemi had the FSTs operate in parallel. An FST requires a particular context that could be an underlying or surface symbol (class) and specifies a particular mapping between underlying and surface strings. It thus acts as a constraint on the mapping of underlying and surface representations, and the specific environment of this mapping. All FSTs simultaneously scan both underlying and surface strings. A mapping is accepted by all the FSTs that do not specify a constraint. For it to work the underlying and surface strings have to be equal length, so the mapping is one to one. One rule maps underlying y to surface i provided that a surface e comes next; so the context is the surface string. The other is sensitive to the underlying string where it ensures a surface e appears whenever y precedes the morpheme boundary, shown in (3.6).

(3.6)

f	l	y	0	^	s	
f	l	i	e	0	s	

Koskenniemi's model launched FST-based morphology because, as Karttunen (2007: 457) observes, it was "the first practical model in the history of computational linguistics for analysis of morphologically complex languages." Despite its title, the framework was essentially for morphonology rather than morphology proper, as noted in an early review (Gazdar 1985: 599). Nonetheless, FST morphonology paved the way for FST morphology proper which we now discuss.

3.3 Finite State Morphology

In the previous section we showed how lexical analysis has to account for surface variation of a canonical string. But the canonical string with morpheme boundaries is itself the lower string of its associated lemma. For example, *fox^s* has the higher-level representation as the (annotated) lemma *fox+noun^plural*. FSTs are used to translate between these two levels to model what we could think of as morphology 'proper.' To briefly highlight the issues in FSM let us consider an example from Turkish with a morphosyntactic translation, or interlinear gloss, as well as a standard translation.

(3.7) gör-mü-yor-du-k
see-NEG-PROGR-PAST-1PL
'We weren't seeing' (Mel'čuk 2006: 299)

In Turkish, the morphological components of a word are neatly divisible into stem and contiguous affixes where each affix is an exponent of a particular morphosyntactic property. Lexical analysis treats the interlinear gloss (second line) as the lemma and maps it onto a morphologically decomposed string. The

language of the upper, or lexical, language contains symbols for morphosyntactic features. The ordering of the morphemes is important: Negation precedes Aspect which precedes Tense which in turn precedes Subject Agreement information. For a correct mapping, the FST must encode morpheme ordering, or *morphotactics*. This is classic I&A morphological analysis. As in the previous section, we can demonstrate with an FST for English notated in DATR (3.8). English does not match Turkish for richness in inflectional morphology but does better in derivational morphology. The lexical entries for the derivational family *industry, industrial, industrialize, industrialization* are given in (3.8b).

(3.8)
```
DERIVATION:
    <$abc> == $abc <>
    <+noun> == Noun_Stem:<>.
Noun_Stem:
    <> == \#
    <^ +adj> == ^ a l Adj_Stem:<>.
Adj_Stem:
    <> == \#
    <^ +vb> == ^ i z e Verb_Stem:<>.
Verb_Stem:
    <> == \#
    <^ +noun> == ^ a t i on <>.
```

(3.8b)
```
<i n d u s t r y +noun>
<i n d u s t r y +noun ^ +adj>
<i n d u s t r y +noun ^ +adj ^ +vb>
<i n d u s t r y +noun ^ +adj ^ +vb ^ +noun>.
```

The FST maps the lemma lexical entries in (3.8b) to their corresponding (intermediary) forms, the noun *industry#*, the adjective *industry^al#*, the verb *industry^al^ize#*, and the noun *industry^al^ize^ation#*. As in the morphonological demonstration in the previous section, the trivial alphabetical mapping is performed through a variable expressing a symbol class and path extensions for arc transitioning. The first difference is a path with a morphosyntactic feature as its attribute, <+noun> showing that in this FST we have lemmas and features as input. We see that this feature licenses the transition to another set of states gathered round the node `Noun_Stem`. In FSM, lemmas are classified according to features such as POS to enable appropriate affix selection, and hence capture the morphotactics of the language. Three nodes representing three stem classes are associated with the three affixes *–al, -ize, -ation*. For `^ a l` to be a possible affix value the evaluation must be at the `Noun_Stem` node. Once the affix is assigned, further evaluation must be continued at a specified node, here `Adj_Stem`. This is because the continuation to *–al* affixation is severely restricted in English. We can think of the specified 'continuation' node as representing a *continuation class*, a list of just those affixes that can come after *–al*. In this way, a lemma is guided through the network, outputting an affix and being shepherded to the subnetwork where the next affix will be available. So (3.8) accounts for *industy^al^ize^ation#* but fails for **industry^ation^al^ize#* or **industy-ize-al-ation#*. It also accounts for *industry#* and *industy^al^ize#* by means of the equation <> == \ # at each continuation class node. Note that # is a reserved symbol in DATR, hence the need for escape \.

Let us quickly step through the FST to see how it does the mapping `<i n d u s t r y +noun ^ +adj ^ +vb> = i n d u s t r y ^ a l ^ i z e #`. The first path at `DERIVATION` maps the entire stem of the lemma to its surface form, in the manner described for the spelling rule FST. After this, the leading subpath is <+noun>; the path extensions are passed over to the node `Noun_Stem`. The first line at `Noun_Stem` covers the morphologically simple `<i n d u s t r y +noun>`. For this string, there is no further path to extend, i.e., no morphological boundaries, and transduction amounts to appending to the output string the word boundary symbol. Similar provision is made at all

nodes just in case the derivation stops there. If, however, the lemma is annotated as being morphologically complex, and specifically as representing adjectival derivation, <^ +adj>, the output is a morpheme boundary plus *-al* affix (second line at the Adj_Stem node). At this point the path can be extended as <^ +vb> in the case of derivative *industrialize* or *industrialization*, or not in the case of *industrial*. With no extensions, evaluation will be through <> == \# yielding i n d u s t r y ^ a l #. Otherwise an extension with leading subpath <^ + vb> outputs suffix ^ i z e and is then passed onto the node Verb_Stem for further evaluation. As there is no new subpath the end of word boundary is appended to the output string value. But if the input path happened to extend this path any further evaluation would have to be at Verb_Stem, e.g., adding the affix *-ation*.

3.3.1 Disjunctive Affixes, Inflectional Classes, and Exceptionality

Affix continuation classes are important for getting the morphotactics right but they also allow for more than one affix to be associated with the same morphosyntactic feature set. This is very common in inflectionally rich languages such as Russian, French, Spanish, and German. To illustrate, consider the paradigm of the Russian word *karta* 'map.' I am giving the forms in their transliterated versions for expository reasons, so it should be understood that *karta* is the transliteration of карта. Note the suffix used for the genitive plural –Ø. This denotes a 'zero affix,' i.e., the word is just the stem *kart* (or карт) in a genitive plural context.

(3.9) Karta

	Singular	Plural
Nominative	*kart-a*	*kart-y*
Accusative	*kart-u*	*kart-y*
Genitive	*kart-y*	*kart-Ø*
Dative	*kart-e*	*kart-am*
Instrumental	*kart-oj*	*kart-ami*
Locative	*kart-e*	*kart-ax*

The FST in (3.10) maps lexical entries such as <k a r t +noun ^ sg nom> to its corresponding surface form k a r t ^ a #.

(3.10)
```
RUSSIAN:
    <$abc> == $abc <>
    <+noun> == Noun_Stem:<>.
Noun_Stem:
    <> == \#
    <^ sg nom> == ^ a <>
    <^ sg acc> == ^ u <>
    <^ sg gen> == ^ y <>
    <^ sg dat> == ^ e <>
    <^ sg inst> == ^ o j <>
    <^ sg loc> == ^ e <>
    <^ pl nom> == ^ y <>
    <^ pl acc> == ^ y <>
    <^ pl gen> == ^ 0 <>
    <^ pl dat> == ^ am <>
    <^ pl inst> == ^ a m i <>
    <^ pl loc> == ^ a x <>.
```

This FST accounts for *any* Russian noun. But this makes it too powerful as not all nouns share the inflectional pattern of *karta*. For example, *zakon* 'law' has a different way of forming the genitive

TABLE 3.1 Russian Inflectional Classes

	I Zakon	II Karta	III Rukopis'	IV Boloto
		Singular		
Nom	*zakon-ø*	*kart-a*	*rukopis'-ø*	*bolot-o*
Acc	*zakon-ø*	*kart-u*	*rukopis'-ø*	*bolot-o*
Gen	*zakon-a*	*kart-y*	*rukopis-i*	*bolot-a*
Dat	*zakon-u*	*kart-e*	*rukopis-i*	*bolot-u*
Inst	*zakon-om*	*kart-oj*	*rukopis-ju*	*bolot-om*
Loc	*zakon-e*	*kart-e*	*rukopis-i*	*bolot-e*
		Plural		
Nom	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Acc	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Gen	*zakon-ov*	*kart-ø*	*rukopis-ej*	*bolot-ø*
Dat	*zakon-am*	*kart-am*	*rukopis-jam*	*bolot-am*
Inst	*zakon-ami*	*kart-ami*	*rukopis-jami*	*bolot-ami*
Loc	*zakon-ax*	*kart-ax*	*rukopis-jax*	*bolot-ax*

singular: it affixes *−a* to the stem and not *−y* (*zakon-a*). And *bolot-o* 'swamp' differs in its nominative singular. Finally *rukopis'* 'manuscript' has a distinct dative singular *rukopisi*. Because of these and other distinctions, Russian can be thought of as having four major inflectional patterns, or *inflectional classes*, that are shown in Table 3.1.

To handle situations where there is a *choice* of affix corresponding to a given morphosyntactic property set, an FST encodes subclasses of stems belonging to the same POS class. (3.11) is a modified version of the FST in (3.10) that incorporates inflectional classes as sets of disjunctive affixes. For reasons of space, only two classes are represented. Sample lexical entries are given in (3.12).

(3.11)
```
RUSSIAN_2:
    <$abc> == $abc <>
    <+noun> == Noun_Stem:<>.
Noun_Stem:
    < 1 > == Stem_1:<>
    < 2 > == Stem_2:<>
    < 3 > == Stem_3:<>
    < 4 > == Stem_4:<>.
Stem_1:
    <> == \#
    <^ sg nom> == ^ 0 <>
    <^ sg acc> == ^ 0 <>
    <^ sg gen> == ^ a <>
    <^ sg dat> == ^ u <>
    <^ sg inst> == ^ o m <>
    <^ sg loc> == ^ e <>
    <^ pl nom> == ^ y <>
    <^ pl acc> == ^ y <>
    <^ pl gen> == ^ o v <>
    <^ pl dat> == ^ a m <>
    <^ pl inst> == ^ a m i <>
    <^ pl loc> == ^ a x <>.
```

```
Stem_2:
    <> == \#
    <^ sg nom> == ^ a <>
    <^ sg acc> == ^ u <>
    <^ sg gen> == ^ y <>
    <^ sg dat> == ^ e <>
    <^ sg inst> == ^ o j <>
    <^ sg loc> == ^ e <>
    <^ pl nom> == ^ y <>
    <^ pl acc> == ^ y <>
    <^ pl gen> == ^ 0 <>
    <^ pl dat> == ^ am <>
    <^ pl inst> == ^ a m i <>
    <^ pl loc> == ^ a x <>.
```

(3.12)
```
<k a r t +noun 2 ^ sg nom>
<k a r t +noun 2 ^ sg acc>
<z a k o n +noun 1 ^ sg nom>
<z a k o n +noun 1 ^ sg acc>
```

What is different about the partial lexicon in (3.12) is that stems are annotated for stem class (1, 2, 3, 4) as well as POS. The node `Noun_Stem` assigns stems to appropriate stem class nodes for affixation. Each of the four stem class nodes maps a given morphosyntactic feature sequence to an affix. In this way, separate affixes that map to a single feature set do not compete as they are distributed across the stem class nodes. Even English has something like inflectional classes. There are several ways of forming a past participle: suffix *–ed* as in 'have looked,' suffix *–en* as in 'have given,' and no affix (-Ø) as in 'have put.' An English verb FST would encode this arrangement as subclasses of stems, as in the more elaborate Russian example.

Classifying stems also allows for a fairly straightforward treatment of exceptionality. For example, the Class I noun *soldat* is exceptional in that its genitive plural is not **soldat-ov* as predicted by its pattern, but *soldat-Ø* 'soldier.' This is the genitive plural you expect for a Class 2 noun (see Table 3.1). To handle this we annotate *soldat* lexical entries as Class 1 for all forms *except* the genitive plural, where it is annotated as Class 2. This is shown in (3.13) where a small subset of the lemmas are given.

(3.13)
```
<s o l d a t +noun 1 ^ sg nom>
<s o l d a t +noun 1 ^ sg gen>
<s o l d a t +noun 1 ^ pl nom>
<s o l d a t +noun 2 ^ pl gen>
```

Another type of exception is represented by *pal'to* 'overcoat.' What is exceptional about this item is that it does not combine with any inflectional affixes, i.e., it is an indeclinable noun. There are a number of such items in Russian. An FST assigns them their own class and maps all lexical representations to the same affixless surface form, as shown in (3.14).

(3.14)
```
Stem_5:
    <> == \#.
```

Our last type of exception is what is called a *pluralia tantum* word, such as *scissors* in English, or *trousers*, where there is no morphological singular form. The Russian for 'trousers' is also pluralia tantum: *brjuk-i*. We provide a node in the FST that carries any input string singular features to a node labeled for input plural features. This is shown in (3.15) as the path `<^ sg>` inheriting from another path at another node, i.e., `<^ pl>` at `Stem_2`. This is because *brjuki* shares plural affixes with other Class 2 nouns.

(3.15)
```
Stem_6:
    <> == \#
    <^ sg> == Stem_2:<^ pl>.
```

FSTs for lexical analysis are based on I&A style morphology that assumes a straightforward mapping of feature to affix. Affix rivalry of the kind exemplified by Russian upsets this mapping since more than one affix is available for one feature. But by incorporating stem classes and continuation affix classes they can handle such cases and thereby operate over languages with inflectional classes. The subclassification of stems also provides FSM with a way of incorporating exceptional morphological behavior.

3.3.2 Further Remarks on Finite State Lexical Analysis

FSTs can be combined in various ways to encode larger fragments of a language's word structure grammar. Through *union* an FST for, say Russian nouns, can be joined with another FST for Russian verbs. We have already mentioned that FSTs can be cascaded such that the output of FST1 is the input to FST2. This operation is known as *composition*. The FSTs for morphology proper take lemmas as input and give morphologically decomposed string as output. These strings are then the input of morphonological/spelling rules FSTs that are sensitive to morpheme boundaries, i.e., where the symbols ^ and # define contexts for a given rule as we saw with the English plural spelling rule. So, for example, the lemma <k a r t + noun ^ sg nom> maps on to an intermediate level of representation k a r t ^ a #. Another transducer takes this as input path k a r t ^ a # and maps it onto the surface form k a r t a, stripping away the morpheme boundaries and performing any other (morphonological) adjustments. Intermediate levels of representation are dispensed with altogether if the series of transducers is *composed* into a single transducer, as detailed in Roark and Sproat (2007) where <x, y> representing the upper and lower tape of transducer 1 and <y, z> the upper and lower tapes of transducer 2 are composed into a single transducer T as <x, z>, i.e., intermediary <y, z> is implied. As we saw in the previous section, Two-Level Morphology does not compose the morphonological FSTs but intersects them. There is no intermediary level of representation because the FSTs operate orthogonally to a simple finite state automaton representing lexical entries in their lexical (lemma) forms.

Finite state approaches have dominated lexical analysis from Koskenniemi's (1983) implementation of a substantial fragment of Finnish morphology. In the morphology chapters of computational linguistics textbooks the finite state approach takes centre stage, for example, Dale et al. (2000), Mitkov (2004), and Jurafsky and Martin (2007), where it takes center stage for two chapters. In Roark and Sproat (2007) computational morphology *is* FSM. From our demonstration it is not hard to see why this is the case. Implementations of FSTNs are relatively straightforward and extremely efficient, and FSTs provide the simultaneous modeling of morphological generation and analysis. They also have an impressive track record in large-scale multilingual projects, such as the Multext Project (Armstrong 1996) for corpus analysis of many languages including Czech, Bulgarian, Swedish, Slovenian, and Swahili. More recent two-level systems include Ui Dhonnchadha et al. (2003) for Irish, Pretorius and Bosch (2003) for Zulu, and Yona and Wintner (2007) for Hebrew. Finite state morphological environments have been created for users to build their own models, for example, Sproat's (1997) *lex tools* and more recently Beesley and Karttunen's (2003) *xerox finites state tools*. The interested reader should consult the accompanying Web site for this chapter for further details of these environments, as well as DATR style FSTs.

3.4 "Difficult" Morphology and Lexical Analysis

So far we have seen lexical analysis as morphological analysis where there are two assumptions being made about morphologically complex word: (1) one morphosyntactic feature set, such as 'Singular Nominative,' maps onto one *exponent*, for example, a suffix or a prefix; and (2) the exponent itself is identifiable as a

sequence of symbols lying contiguous to the stem, either on its left (as a prefix) or its right (as a suffix). But in many languages neither (1) nor (2) necessarily hold. As NLP systems are increasingly multilingual, it becomes more and more important to explore the challenges other languages pose for finite state models, which are ideally suited to handle data that conform to assumptions (1) and (2). In this section, we look at various sets of examples that do not conform to a neat I&A analysis. There are sometimes ways around these difficult cases, as we saw with the Russian case where stem classes were used to handle multiple affixes being associated with a single feature. But our discussion will lead to an alternative to I&A analysis that finite state models entail. As we will see in Section 3.5, the alternative W&P approach appears to offer a much more natural account of word structure when it includes the difficult cases.

3.4.1 Isomorphism Problems

It turns out that few languages have a morphological system that can be described as one feature (or feature set) expressed as one morpheme, the exponent of that feature. In other words, *isomorphism* turns out not to be the common situation. In Section 3.3, I carefully chose Turkish to illustrate FSTs for morphology proper because Turkish seems to be isomorphic, a property of agglutinative languages. At the same time derivational morphology tends to be more isomorphic than inflection, hence an English derivational example. But even agglutinative languages can display non-isomorphic behavior in their inflection (3.16) is the past tense set of forms for the verb 'to be' (Haspelmath 2002: 33).

(3.16)

ol-i-n	'I was'
ol-i-t	'you (singular) were'
ol-i	'he/she was'
ol-i-mme	'we were'
ol-i-tte	'you (plural) were'
ol-i-vat	'they were'

A lexical entry for 'I was' would be <o l +verb ^ past ^ 1stSg> mapping to o l ^ i ^ n #. Similarly for 'we were,' <o l +verb ^ past ^ 1stSPL> maps to o l ^ i ^ mme #. But what about 'he/she was'? In this case there is no exponent for the feature set '3rd Person Singular' to map onto; we have lost isomorphism. But in a sense we had already lost it since for all forms in (3.16) we are really mapping a *combination* of features to a single exponent: a Number feature (plural) and a Person feature (1st) map to the single exponent *-mme*, etc. Of course the way out is to use a symbol on the lexical string that describes a feature combination. This is what we did with the Russian examples in Section 3.3.2 to avoid the difficulty, and it is implicit in the Finnish data above. But back to the problem we started with: where there is a 'missing' element on the surface string we can use a Ø a 'zero affix,' for the upper string feature symbol to map to. Indeed, in Tables 3.1 and 3.2 I used Ø to represent the morphological complexity of some Russian word-forms. So the Russian FST maps the lemma <z a k o n +noun ^ sg ^ nom> onto z a k o n ^ Ø #. To get rid of the Ø, a morphonological FST can use empty transitions in the same way it does to delete morpheme boundaries. Variations of this problem can be met with variations of the solution. The French adverb 'slowly' is *lentement* where *lent-* is the stem and *-ment* expresses 'adverb.' This leaves the middle *-e-* without anything to map onto. The mapping is one upper string symbol to two lower string symbols. The solution is to squeeze in a zero feature, or 'empty morpheme' between the stem and the 'adverb' feature: <l e n t ^ Ø ^ adverb>. The converse, two features with a single exponent, is collapsing the two features into a feature set, as we discussed. The alternative is to place zeros on the lower string: <o l +verb Past ^ 1stPerson ^ Singular> maps to <o l ^ i ^ m m e ^ Ø #. But then the choice of what feature has the zero affix is arbitrary.

Finally, we can think of the competition of exponents for a particular feature as a similar kind of isomorphism problem: one feature, several exponents. In Section 3.3.1, we showed how Russian has inflectional classes, where nouns are grouped according to the choice of suffix they make for a given

TABLE 3.2 Russian Inflectional Classes

	I Zakon	II Karta	III Rukopis'	IV Boloto
		Singular		
Nom	*zakon-ø*	*kart-a*	*rukopis'-ø*	*bolot-o*
Acc	*zakon-ø*	*kart-u*	*rukopis'-ø*	*bolot-o*
Gen	*zakon-a*	*kart-y*	*rukopis-i*	*bolot-a*
Dat	*zakon-u*	*kart-e*	*rukopis-i*	*bolot-u*
Inst	*zakon-om*	*kart-oj*	*rukopis-ju*	*bolot-om*
Loc	*zakon-e*	*kart-e*	*rukopis-i*	*bolot-e*
		Plural		
Nom	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Acc	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Gen	*zakon-ov*	*kart-ø*	*rukopis-ej*	*bolot-ø*
Dat	*zakon-am*	*kart-am*	*rukopis-jam*	*bolot-am*
Inst	*zakon-ami*	*kart-ami*	*rukopis-jami*	*bolot-ami*
Loc	*zakon-ax*	*kart-ax*	*rukopis-jax*	*bolot-ax*

feature. To handle this we used affix classes together with stem classes such that different stem classes were licensed to navigate over different parts of the network where the appropriate affixes are stored.

3.4.2 Contiguity Problems

As well as isomorphism, I&A approaches rely on contiguity: the exponent of a feature should be found at the left or right edge of the stem of the lower string, mirroring the position of the feature relative to the stem on the upper lexical string. But one does not need to look far to find examples of 'noncontiguous' morphology. In Section 3.1, I discussed the potential problem of *sang*, past tense of *sing*. This is an example of a feature mapping onto an exponent, which is really the change we make to the stem's vowel. How can an FST map the correct lower string to the upper string `<s i n g ^ past>`? To account for the feature *not* mapping onto an affix, it can use the Ø as we did (extensively) with the isomorphism problems above. This then leaves the vowel alternation as the exponent of the feature. One way is to target the stem vowel so that lexical `i` maps onto surface `a`; and allow navigation through the subnetwork just for those items that behave this way (*sing/sang, ring/rang, spin/span*). This is represented in (3.17). Lexical entries for regular *cook* and irregular *sing* and *ring* are given in (3.18).

(3.17)
```
Ablaut_FST:
    <$abc> == $abc <>
    <s i n g ^ past> == Past_Stem
    <r i n g ^ past> == Past_Stem
    <^ present> == ^ 0 <>
    <^ past> == ^ e d <>
    <> == \#.
Past_Stem:
    <$abc> == $abc <>
    <$vow> == a <>
    <^ past> == ^ 0 <>
    <> == \#.
```

(3.18)
```
<s i n g ^ present>  <s i n g ^ past>
<r i n g ^ present>  <r i n g ^ past>
<c o o k ^ present>  <c o o k ^ past>
```

The first node provides for the trivial stem string mappings required by regular non-ablauting verbs. The fourth equation expresses lexical 'present' mapping to Ø (where no account is taken of –s affixation for '3rd person singular'). The fifth equation handles regular past formation in *–ed*, as in *cooked*. But just in case the path to evaluate is `<s i n g ^ past>` or `<r i n g ^ past>`, an extra node is provided for the evaluation, the node `Past_Stem`. At this node, the symbol class for all stem characters is used to perform the normal upper to lower stem mapping, as in previous DATR examples. The difference is that there is a new symbol class expressed as the variable $vow, ranging over vowels that alternate with *a* in the past. The path equation `<$vow> == a <>` takes the vowel in the stem and maps it to a, the vowel alternation that is the exponent of 'past.' Any other character belonging to the input (stem) string maps onto itself on the lower string (through `<$abc> == $abc <>`). Finally `<^ past> == ^ 0 <>` represents the fact that for this class of verbs the zero affix is used as exponent of 'past.' The theorem is given in (3.19).

(3.19)
```
<s i n g ^ present> = s i n g ^ 0 #.
<s i n g ^ past> = s a n g ^ 0 #.
<r i n g ^ present> = r i n g ^ 0 #.
<r i n g ^ past> = r a n g ^ 0 #.
<c o o k ^ present> = c o o k ^ 0 #.
<c o o k ^ past> = c o o k ^ e d #.
```

Another example of noncontiguous morphology is *infixation*, where an affix attaches not to the right or left edges but within the stem itself. (3.20) is an infixation example from Tagolog, a language spoken in the Philippines with nearly sixteen million speakers worldwide (*Ethnologue*). The data are from Mel'čuk (2006: 300).

(3.20)

	Sulat 'write'	Patay 'kill'
ACTIVE	*s-um-ulat*	*p-um-atay*
PASSIVE	*s-in-ulat*	*p-in-atay*

A way of handling infixation with FSTs is to add an intermediary level of representation, as outlined in Roark and Sproat (2007: 29–31, 39–40). A first transducer maps the location of the affixation to an 'anchor' symbol within the stem. A second transducer operates over intermediate representations and maps the anchor with either `^ in ^` or `^ um ^` depending on the voice feature (Active or Passive). Note that features need to be preserved from the lexical to intermediate levels. This is because the only difference between the two string types is the anchor; and as it is an affix it needs boundaries, and as it is an infix the boundaries need to be on both left and right. This is important in case there are any morphonological consequences to the infixation. This approach informs the DATR transducers in (3.21) and (3.23).

(3.21)
```
Itermediate_FST:
    <$features> == $features <>
    <$abc> == $abc <>
    <^ $abc $vow> == ^ $abc & $vow <>
    <> == .
```

(3.22)
```
<active ^ s u l a t>  <passive ^ s u l a t>
<active ^ p a t a y>  <passive ^ p a t a y>
```

As feature annotations are needed in the output string, we need a symbol class for them, expressed as the variable `$features`; then identity mapping is treated as in the previous FSTs. The second equation is the (by now) familiar trivial stem character mapping. Note that lexical entries come with features to the left of the canonical stem (3.22). The third equation in (3.21) represents the way the FST targets the first character followed by the first vowel of a stem. The right hand side expresses the transduction: the boundary symbol

is preserved, as is the first letter of the stem and the first vowel of the stem. But at the same time a symbol & is inserted between them. This is our anchor. The output of the transducer is therefore: `active ^ s & u l a t`, and `passive ^ s & u l a t`. These strings are then the input to a second transducer in (3.23) by recasting them as paths.

(3.23)
```
Infixation_FST:
    <active ^> == Active:<>
    <passive ^> == Passive:<>.
Active:
    <$abc> == $abc <>
    <& > == ^ um ^ <>
    <> == \#.
Passive:
    <$abc> == $abc <>
    <& > == ^ in ^ <>
    <> == \#.
```

Here the feature symbols are used to decode the anchor &. Paths beginning `<active ^>` are evaluated at the `Active` node, and paths beginning `<passive ^>` at the `Passive` node. The first line at each node maps stem letters to themselves in the normal way. The second line maps & to a boundary delimited `um` at the Active node and `in` at the Passive node. End of word boundaries are inserted when there is no more path to extend, as previously. The theorem is given in (3.24).

(3.24)
```
<active ^ s & u l a t> = s ^ um ^ u l a t #
<passive ^ s & u l a t> = s ^ in ^ u l a t #
```

Our final example of noncontiguous morphology is where the root is interrupted, and at the same time so is the exponent: "where affixes interrupt roots and are interrupted by elements of roots themselves." (Mel'čuk 2006: 300). Mel'čuk uses the term 'transfixation' but in the FST literature this type is usually called 'root and template,' 'root and pattern,' or simply 'non-concatenative' morphology. In Arabic, the root for 'draw' is the consonant template, or skeleton, *r-s-m*. The exponent is the flesh. For 'active perfect,' the exponent is the vowel series *-a-a-*, for 'passive perfect,' there is another exponent *-u-i-*. So *rasam(a)* 'he has drawn,' and *rusim(a)* 'it has been drawn' (where the bracketed *-a* is the more normal suffix exponent): the root is interrupter and interrupted, as is the exponent. Another way of thinking about situations of this sort is to have different *tiers* of structural information. Booij (2007: 37) gives a Modern Hebrew example. The root of the verb grow is the consonant series *g-d-l*. The root occupies one tier. The formal distinction between the word 'grow' *gadal* and its nominalization 'growth' *gdila* is expressed by a pattern of consonants and vowels occupying separate tiers. This is represented in (3.25).

(3.25)

gadal					
Tier 1		a		a	
Tier 2	C	V	C	V	C
Tier 3	g		d		l

gdila					
Tier 1			i		a
Tier 2	C	C	V	C	V
Tier 3	g	d		l	

There are therefore three separate pieces of surface information: the root (consonants) as tier 3 information, the exponent (vowels) as tier 1, and the instruction for how they interleave as tier 2. How is this modeled as finite state machine? A method reported in Kiraz (2000) for Arabic and widely discussed in the FSM literature is to have an n- tape FST where $n > 1$. Each tier has representation on one of

several lower tapes. Another tape is also provided for concomitant prefixation or suffixation. Rather ingeniously, a noncontiguous problem is turned into a contiguous one so that it can receive a contiguous solution. Roark and Sproat (2007) propose a family of transducers for different CV patterns (where V is specified) and the morphosyntactic information they express. The union of these transducers is composed with a transducer that maps the root consonants to the Cs. The intermediate level involving the pattern disappears.

3.5 Paradigm-Based Lexical Analysis

The various morphological forms of Russian *karta* in (3.9) were presented in such a way that we could associate form and meaning difference among the word-forms by consulting the cell in the table, the place where case and number information intersect with word-form. The presentation of a lexeme's word-forms as a *paradigm* provides an alternative way of capturing word structure that does not rely on either isomorphism or contiguity. For this reason, the W&P approach has been adopted by the main stream of morphological theorists with the view that "paradigms are essential to the very definition of a language's inflectional system" (Stewart and Stump 2007: 386). A suitable representation language that has been used extensively for paradigm-based morphology is the lexical knowledge representation language DATR, which up until now we have used to demonstrate finite state models. In this section, we will outline some of the advantages of paradigm-based morphology. To do this we will need to slightly extend our discussion of the DATR formalism to incorporate the idea of *inheritance* and *defaults*.

3.5.1 Paradigmatic Relations and Generalization

The FSM demonstrations above have been used to capture properties not only about single lexical items but whole classes of items. In this way, lexical analysis goes beyond simple listing and attempts generalizations. It may be helpful at this point to summarize how generalizations have been captured. Using symbol classes, FSTs can assign stems and affixes to categories, and encode operations over these categories. In this way, they capture classes of environments and changes for morphonological rules, and morpheme orderings that hold for classes of items, as well as selections when there is a choice of affixes for a given feature. But there are other generalizations that are properties of paradigmatic organization itself, what we could think of as *paradigmatic relations*. To illustrate let us look again at the Russian inflectional class paradigms introduced earlier in Table 3.1, and presented again here as Table 3.3.

TABLE 3.3 Russian Inflectional Classes

	I Zakon	II Karta	III Rukopis'	IV Boloto
		Singular		
Nom	*zakon-ø*	*kart-a*	*rukopis'-ø*	*bolot-o*
Acc	*zakon-ø*	*kart-u*	*rukopis'-ø*	*bolot-o*
Gen	*zakon-a*	*kart-y*	*rukopis-i*	*bolot-a*
Dat	*zakon-u*	*kart-e*	*rukopis-i*	*bolot-u*
Inst	*zakon-om*	*kart-oj*	*rukopis-ju*	*bolot-om*
Loc	*zakon-e*	*kart-e*	*rukopis-i*	*bolot-e*
		Plural		
Nom	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Acc	*zakon-y*	*kart-y*	*rukopis-i*	*bolot-a*
Gen	*zakon-ov*	*kart-ø*	*rukopis-ej*	*bolot-ø*
Dat	*zakon-am*	*kart-am*	*rukopis-jam*	*bolot-am*
Inst	*zakon-ami*	*kart-ami*	*rukopis-jami*	*bolot-ami*
Loc	*zakon-ax*	*kart-ax*	*rukopis-jax*	*bolot-ax*

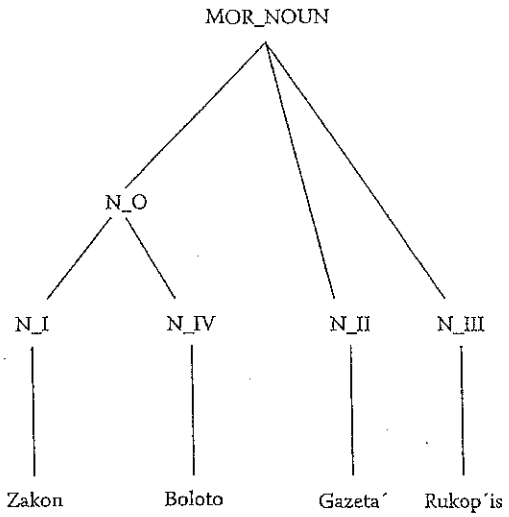

FIGURE 3.4 Russian nouns classes as an inheritance hierarchy (based on Corbett, C.G. and Fraser, N.M., Network Morphology: A DATR account of Russian inflectional morphology. In Katamba, F.X. (ed), pp. 364–396. 2003).

One might expect that each class would have a unique set of forms to set them apart from the other class. So looking horizontally across a particular cell pausing at, say, the intersection of Instrumental and Singular, there would be four different values, i.e., four different ways of forming a Russian singular instrumental noun. Rather surprisingly, this does not happen here: for Class 1 the suffix *–om* is used, for Class II *–oj*, for Class III *–ju*, but Class IV does the same as Class I. Even more surprising is that there is not a single cell where a four-way distinction is made. Another expectation is that within a class, each cell would be different from the other, so, for example, forming a nominative singular is different from a nominative plural. While there is a tendency for *vertical* distinctions across cells, it is only a tendency. So for Class II, dative singular is in *–e*, but so is locative singular. In fact, in the world's languages it is very rare to see fully horizontal and fully vertical distinctions. Recent work by Corbett explores what he calls 'canonical inflectional classes' and shows that the departures from the canon are the norm, so canonicity does not correlate with frequency (Corbett 2009). Paradigm-based lexical analysis takes the departures from the canon as the starting point. It then attempts to capture departures by treating them as *horizontal relations* and *vertical relations*, as well as a combination of the two. So an identical instrumental singular for Classes I and IV is a relation between these classes at the level of this cell. And in Class II there is a relationship between the dative and locative singular. To capture these and other paradigmatic relations in Russian, the inflectional classes in Table 3.2 can be given an alternative presentation as a hierarchy of nodes down which are inherited instructions for forming morphologically complex words (Figure 3.4).

Horizontal relations are expressed as inheritance of two daughter nodes from a mother node. The node `N_O` stores the fact about the instrumental singular, and both Classes I and IV, represented as `N_I` and `N_IV`, inherit it. They also inherit genitive singular, dative singular, and locative singular. This captures the insight of Russian linguists that these two classes are really versions of each other, for example, Timberlake (2004: 132–141) labels them 1a and 1b. Consulting Table 3.2, we see that all classes in the plural form the dative, instrumental, and locative in the same way. The way these are formed should therefore be stated at the root node `MOR_NOUN` and from there inherited by all classes. In the hierarchy, leaf nodes are the lexical entries themselves, each a daughter of an appropriate inflectional class node. The DATR representation of the lexical entry `Karta` and the node form that it inherits is given in (3.25). The ellipsis '. . . .' indicates here and elsewhere that the node contains more equations, and is not part of the DATR language.

(3.25)
```
Karta:
    <mor> == N_II
    <stem> == kart.
N_II:
    <mor sg nom> == "<stem>" ^a
    <mor sg acc> == "<stem>" ^u
    <mor sg dat> == "<stem>" ^e
    <mor sg loc> == "<stem>" ^e
....
```

The first equation expresses that the Karta node inherits path value equations from the N_II node. Recall that in DATR a subpath implies any extension of itself. So the path <mor> implies any path that is an extension of <mor> including <mor sg nom>, <mor sg acc>, <mor sg dat>, <mor sg loc>, etc. These are all paths that are specified with values at N_II. So the first equation in (3.25) is equivalent to the equations in (3.26). But we want these facts to be inherited by Karta rather than being explicitly stated at Karta to capture the fact that they are shared by other (Class II) nouns.

(3.26)
```
Karta:
    <mor sg nom> == "<stem>" ^a
    <mor sg acc> == "<stem>" ^u
    <mor sg dat> == "<stem>" ^e
    <mor sg loc> == "<stem>" ^e
    ....
```

The value of these paths at N_II is complex: the concatenation of the value of another path and an exponent. The value of the other path is the string expressing a lexical entry's stem. In fact, this represents how paradigm-based approaches model word structure: the formal *realization* of a set of morphosyntactic features by a rule operating over stems. The value of "<stem>" depends on what lexical entry is the object of the morphological query. If it is Karta, then it is the value for the path <stem> at the node Karta (3.25). The quoted path notation means that inheritance is set to the context of the initial query, here the node Karta, and is not altered even if the evaluation of <stem> is possible locally. So we could imagine a node (3.27) similar to (3.25) N_II but adding an eccentric local fact about itself, namely that Class II nouns always have *zakon* as their stem, no matter what their real stem is.

(3.27)
```
N_II:
    <mor sg nom> == "<stem>" ^a
    <stem> == zakon
    ...
```

Nonetheless, the value of <mor sg nom> will not involve altering the initial context of <stem> to the local context. By quoting <stem> the value will be kart^a and not zakon^a. There are equally occasions where local inheritance of paths is precisely what is needed. Equation 3.25 fails to capture the vertical relation that for Class II nouns the dative singular and the locative singular are the same. Local inheritance expresses this relation, shown in (3.28).

(3.28)
```
N_II:
    <mor sg dat> == "<stem>" ^e
    <mor sg loc> == <mor sg dat>
    ....
```

The path <mor sg loc> locally inherits from <mor sg dat>, so that both paths have the value kart^e where Karta is the query lexical entry.

Horizontal relations are captured by hierarchical relations. (3.29) is the partial hierarchy involving MOR_NOUN, N_O, N_I and N_IV.

(3.29)
```
MOR_NOUN:
    <mor pl loc> == "<stem>" ax
    <mor sg dat> == "<stem>" e
    ...
  N_O:
    <> == MOR_NOUN
    <mor sg gen> == "<stem>" a
    <mor sg dat> == "<stem>" u
    <mor sg inst > == "<stem>" om
    <mor sg loc > == "<stem>" e.
N_IV:
  <> == N_O
  <mor sg nom > == "<stem>" o
  <mor pl nom > == "<stem>" a
  ...
N_I:
  <> == N_O
  <mor sg nom > == "<stem>"
  <mor pl nom > == "<stem>" y
  ...
```

Facts about cells that are shared by inflectional classes are stated as path: value equations gathered at MOR_NOUN. These include instructions for forming the locative singular and the locative plural. Facts that are shared only by Classes I and IV are stored at an intermediary node N_O: the genitive, dative, instrumental, and locative singular. Nodes for Classes I and IV inherit from this node, and via this node they inherit the wider generalizations stated at the hierarchy's root node. The passing of information down the hierarchy is through the empty path <>, which is the leading subpath of every path that is not defined at the local node. So at N_I the empty path implies <mor gen sg> at its mother node N_0 but *not* <mor sg nom> because this path is already defined at N_I.

From Figure 3.4, we can observe a special type of relation that is at once vertical and horizontal. In the plural, the accusative is the same as the nominative in Class I, but this same vertical relation extends to all the other classes: they all have an accusative/nominative identity in the plural. To capture this we store the vertical relation at the root node so that all inflectional class nodes can inherit it (3.30).

(3.30)
```
MOR_NOUN:
    <mor pl acc > == "<mor pl nom >"
    ...
```

It needs to be clear from (3.30) that what is being hierarchically inherited is not an exponent of a feature set but a way of getting the exponent. The quoted path at the right hand side expresses that the nominative plural value depends on the global context, i.e., what noun is being queried: if it is a Class I noun it will be stem plus *–y*, and if it is a Class IV noun it will be stem plus *–a*. These will therefore be the respective values for Class I and Class IV accusative singular forms. In other words, the horizontal relation being captured is the vertical relation that the accusative and nominative plural for a given class is the same, although in principle the value itself may be different for each class.

3.5.2 The Role of Defaults

Interestingly the identity of accusative with nominative is not restricted to the plural. From Figure 3.4 we see that for Classes I, III, and IV the singular accusative is identical to the nominative but Class II has separate exponents. If inheritance from MOR_NOUN is interpreted as default inheritance then (3.31) captures a strong tendency for the accusative and nominative to be the same. The bound variable $num ranges over sg and pl atoms.

(3.31)
```
MOR_NOUN
    <mor $num acc > == "<mor $num nom>"
    ...
```

This vertical relation does not, however, extend to Class II which has a unique accusative singular exponent. Class II inherits facts from MOR_NOUN, as any other class does. But it needs to override the fact about the accusative singular (3.32).

(3.32)
```
N_II:
    <> == MOR_NOUN
    <mor sg nom> == "<stem>" a
    <mor sg acc> == "<stem>" u
    ...
```

A path implies any extension of itself, but the value associated with the implied path is by default and can be overridden by an explicitly stated extended path. In (3.32) the empty path implies all of its extensions and their values held at MOR_NOUN, including <mor sg acc> == <mor sg nom>. But because <mor sg acc> is stated locally at N_II, the explicitly stated local evaluation overrides the (implied) inherited one. In similar vein we can capture the locative singular in *–e* as a generalization over all classes *except* Class III (see Table 3.2) by stating <mor sg loc> == "<stem>" e at the root node, but also stating <mor sg loc> == "<stem>" i at the node N_III.

Defaults also allow for a straightforward account of exceptional or semi-regular lexical entries. In Section 3.3.1, I gave several examples from Russian to show how exceptionality could be captured in FSM. Let us briefly consider their treatment in a paradigm-based approach. Recall that the item *soldat* 'soldier' is a regular Class 1 noun in every respect except for the way it forms its genitive plural. Instead of **soldatov* it is simply *soldat*. As in the FSM account, it can be assigned Class II status just for its genitive plural by overriding the default for its class, and inheriting the default from another class.

(3.33)
```
Soldat:
    <mor> == N_I
    <stem> == soldat
    <mor pl gen> == N_II.
```

The other example was a pluralia tantum noun, *brjuki* 'trousers.' Here the singular had plural morphology. This is simply a matter of overriding the inheritance of <mor sg> paths from N_II with the equation <mor sg> == <mor pl>. Of course <mor pl> and its extensions will be inherited from the mother node in the same way it is for all other Class II nouns.

(3.34)
```
Brjuki:
    <mor> == N_II
    <stem> == brjuk
    <mor sg> == <mor pl>.
```

Finally, indeclinable *pal'to* can be specified as inheriting from a fifth class that generalizes over all indeclinables. It is not *pal'to* that does the overriding but the class itself. All extensions of <mor> are overridden and assigned the value of the lexical entry's stem alone, with no exponent.

(3.35)
```
N_V:
    <> == MOR_NOUN
    <mor> == "<stem>".
```

3.5.3 Paradigm-Based Accounts of Difficult Morphology

In Section 3.4, we described 'difficult' morphology as instances where isomorphism, one feature (set) mapping to one exponent, breaks down. The Russian demonstration of the paradigm-based approach has run into this almost without noticing. We have already seen how more than one feature can map onto a single exponent. First, an exponent is a value associated with an attribute path that defines *sets* of features: Number and Case in our running example of Russian. Second, the whole idea behind vertical relations is that two different feature sets can map onto a single exponent: the *–e* suffix maps onto two feature sets for Class II nouns, the singular dative and the singular locative, similarly the accusative and nominative mapping onto a single exponent for all but Class II. This is handled by setting one attribute path to inherit from another attribute path. In paradigm-based morphology, this is known as a *rule of referral* (Stump 1993, 2001: 52–57). Then the reverse problem of more than exponent realizing a singe feature (set) is handled through the stipulation of inflectional classes, so that affixes for the same feature are not in competition. Finally, a feature that does not map onto any exponent we treat as *zero exponence* rather than a zero affix. For example, the genitive plural of a Russian Class II noun is the stem plus nothing. The attribute path is associated with the query lexical entry's stem, and nothing more.

(3.36)
```
N_II
    <> == MOR_NOUN
    <mor pl gen> == "<stem>"
    ...
```

In paradigm-based accounts, a stem of a word in a given cell of a particular set of features is in contrast with stems of the same word in different cells. Morphological structure is then the computation of what makes the stem contrastive together with the feature content of the cell. For this reason, affixation is not given any special status: contrast could be through ablaut, stress shift, or zero exponence as shown above since it also can mark a contrast: by having no exponence, the genitive plural is opposed to all other cells since they do have an exponent. Noncontiguous morphology does not pose a problem for paradigm-based approaches, as Stump (2001) demonstrates. The exponent a rule introduces does not have to target stem edges (affixation) but can target any part of the stem: for example, *sing* has the past tense form *sang*. Cahill and Gazdar (1999) handle ablaut morphological operations of this kind in German plural nouns by defining stems as syllable sequences where the syllable itself has internal structure: an onset consonant, vowel, and a coda consonant. The target of the rule is definable as stem-internal, the syllable vowel. With an inheritance hierarchy, ablaut operations are captured as semi-regular by drawing together items with the similar ablaut patterns under a node that houses the non-concatenative rule: which part of the syllable is modified, and how it is modified. Many nouns undergo ablaut and simultaneously use regular suffixation, for example, *Mann* 'man' has plural *Männer*. The hierarchical arrangement allows for a less regular ablaut together with inheritance of the more regular suffixation rule, *-er* added to the right edge of stem, i.e., after the coda of rightmost syllable. Default inheritance coupled with realization rules simultaneously captures multiple exponence, semi-regularity, and nonlinear morphology.

In more recent work, Cahill (2007) used the same 'syllable-based morphology' approach for root-and-pattern morphological description that we discussed in Section 3.4 with reference to Hebrew. The claim is that with inheritance hierarchies, defaults and the syllable as a unit of description, Arabic verb morphology lies more or less within the same problem (and solution) space as German (and English) ablaut cases. A morphologically complex verb such as *kutib* has the default structural description: `<mor word> == "<agr prefix>" "<tense prefix>" "<ph root form>" "<agr suffix>"`. Each quoted path expresses the value of an exponent

inferable from the organization of the network. These exponents can appear as prefixes and suffixes, which are by default null. But an exponent can be a systematic internal modification of the root: "<ph root form>". Just as for German, the root (or stem) is syllable defined, and more specifically as a series of consonants, e.g., *k t b* 'write.' In syllable terms, these are (by default) the onset of the first (logical order) syllable and the onset and coda of the second. The vowels that occupy nucleus positions of the two syllables can vary and these variations are exponents of tense and mood. So one possible perfect passive form is *kutib* that contrasts with perfect passive *katab*. There are many such root alternations and associated morphosyntactic property sets. To capture similarities and differences, they are organized into default inheritance hierarchies.

This is necessarily an oversimplified characterization of the complexity that Cahill's account captures, but the important point is that defaults and W&P morphology can combine to give elegant accounts of noncontiguous morphology. In Soudi et al. (2007), the collection of computational Arabic morphology papers where Cahill's work appears, it is significant that most of the symbolic accounts are lexeme-based and make use of inheritance; two are DATR theories. Finkel and Stump (2007) is another root-and-pattern account, this time of Hebrew. It also is W&P morphology with default inheritance hierarchies. Its implementation is in KATR, a DATR with 'enhancements,' including a mechanism for expressing morphosyntactic features as unordered members of sets as well as ordered lists to better generalize the Hebrew data. Kiraz (2008) and Sproat (2007) note that Arabic computational morphology has been neglected; this is because "root-and-pattern morphology defies a straightforward implementation in terms of morpheme concatenation" (Sproat 2007: vii). Cahill (2007), and Finkel and Stump (2007) offer an alternative W&P approach, which suggests that perhaps Arabic is not really "specifically engineered to maximize the difficulties for automatic processing."

3.5.4 Further Remarks on Paradigm-Based Approaches

Many computational models of paradigm-based morphological analysis are represented in the DATR lexical knowledge representation language. These include analyses of major languages, for example, Russian in Corbett and Fraser (2003), the paper which the W&P demonstration in this chapter is based on, and more recently in Brown and Hippisley (forthcoming); also Spanish (Moreno-Sandoval and Goñi-Menoyo 2002), Arabic as we have seen (Cahill 2007), German (Cahill and Gazder 1999, Kilbury 2001), Polish (Brown 1998), as well as lesser known languages, for example, Dalabon (Evans et al. 2000), Mayali (Evans et al. 2002), Dhaasanac (Baerman et al. 2005: Chapter 5), and Arapesh (Fraser and Corbett 1997). The paradigm-based theory Network Morphology (Corbett and Fraser 2003, Brown and Hippisley forthcoming) is formalized in DATR. The most well articulated paradigm-based theory is Paradigm Function Morphology (Stump 2001, 2006), and DATR has also been used to represent Paradigm Function Morphology descriptions (Gazdar 1992). DATR's mechanism for encoding default inference is central to Network Morphology and Paradigm Function Morphology. Defaults are used in other theoretical work on the lexicon as part of the overall system of language. For example, HPSG (Pollard and Sag 1994, Sag et al. 2003), uses inheritance hierarchies to capture shared information; inheritance *by default* is used for specifically lexical descriptions in some HPSG descriptions, for example, Krieger and Nerbonne (1992) and more recently Bonami and Boyé (2006).*

We close this section with a comment about DATR. Throughout our discussion of FSM and the paradigm-based alternative, we have used DATR as the demonstration language. But it is important to note that DATR theories are 'one way' as they start with the lexical representation and generate the

* HPSG has been ambivalent over the incorporation of defaults for lexical information but Bonami and Boyé (2006) are quite clear about its importance:

> In the absence of an explicit alternative, we take it that the use of defaults is the only known way to model regularity in an HPSG implementation of the stem space.

The latest HPSG textbook appears to endorse defaults for the lexicon (Sag et al. 2003: 229–236).

surface representation; they do not start with the surface representation and parse out the lexical string. In and of itself, this restriction has not hampered a discussion of the major lexical analysis issues, which is the motivation for using DATR in this chapter but it is important to bear in mind this aspect of DATR. There are ways round this practical restriction. For example, a DATR theory could be compiled into a database so that its theorem is presented as a surface string to lexical string look up table. Earlier work has experimented with extending DATR implementations to provide inference in reverse. Langer (1994) proposes an implementation of DATR that allows for 'reverse queries,' or inference operating in reverse. Standardly, the query is a specific node/path/value combination, for example, `Karta:<mor nom sg>`. The value is what is inferred by this combination. By treating a DATR description as being analogous to a context free phrase structure grammar (CF-PSG), with left hand sides as nonterminal symbols, which rewrite as right hand sides that include nonterminal and terminal symbols, reverse query can be tackled as a CF-PSG bottom-up parsing problem. You start with the value string (`kart^a`) and discover how it has been inferred (`Karta:<mor nom sg>`). Finally, a DATR description could be used to generate the pairing of lexical string to surface string, and these pairings could then be the input to an FST inducer to perform analysis. For example, Gildea and Jurafsky (1995, 1996) use an augmented version of Oncina et al.'s (1993) OSTIA algorithm (Onward Subsequential Transducer Inference Algorithm) to induce phonological rule FSTs. More recently, Carlson (2005) reports on promising results of an experiment inducing FSTs from paradigms of Finnish inflectional morphology in Prolog.

3.6 Concluding Remarks

If the building blocks of natural language texts are words, then words are important units of information, and language-based applications should include some mechanism for registering their structural properties. Finite state techniques have long been used to provide such a mechanism because of their computational efficiency, and their invertibility: they can be used both to generate morphologically complex forms from underlying representations, and parse morphologically complex forms into underlying representations. The linguistic capital of the FSM is an I&A model of word structure. Though many languages, including English, contain morphologically complex objects that resist an I&A analysis, FSM handles these situations by recasting these problems as I&A problems: isomorphism is retained trough empty transitions, collecting features into symbol sets, and introducing stem classes and affix classes. For nonlinear morphology, infixation, and root-and-template, the problem is recast as a linear one and addressed accordingly. FSM can capture morphological generalization, exceptionality, and the organization of complex words into inflectional classes. An alternative to FSM is an approach based on paradigmatic organization where word structure is computed by the stem's place in a cell in a paradigm, the unique clustering of morphologically meaningful elements, and some phonologically contrasting element, not necessarily an affix, and feasibly nothing. W&P approaches appear to offer a better account of what I have called difficult morphology. They also get at the heart of morphological generalization.

Both approaches to lexical analysis are strongly rule based. This puts lexical analysis at odds with most fields of NLP, including computational syntax where statistics plays an increasingly important role. But Roark and Sproat (2007: 116–117) observe that hand-written rules can take you a long way in a morphologically complex language; at the same time ambiguity does not play such a major role in morphological analysis as it does in statistical analysis where there can be very many candidate parse trees for a single surface sentence. That is not to say that ambiguity is not a problem in lexical analysis: given the prominence we have attached to inflectional classes in this chapter, it is not hard to find situations where a surface string has more than one structure. This is the case for all vertical relations, discussed in the previous section. In Russian, the string *karte* can be parsed as a dative or locative, for example. A worse case is *rukopisi*. Consulting Table 3.2 you can see that this form could be a genitive, dative, or locative singular; or nominative or accusative plural. Roark and Sproat go on to note that resolving these ambiguities requires too broad a context for probability information to be meaningful. That is not to say

.nat morphological analysis does not benefit from statistical methods. Roark and Sproat give as examples Goldsmith (2001), a method for inducing from corpora morphonological alternations of a given lemma (Goldsmith 2001), and Yarowski and Wicentowski (2001), who use pairings of morphological variants to induce morphological analyzers.

It should be noted that the important place of lexical/morphological analysis in a text-based NLP system is not without question. In IR, there is a view that symbolic, rule-based models are difficult to accommodate within a strongly statistically oriented system, and the symbolic statistical disconnect is hard to resolve. Furthermore, there is evidence that morphological analysis does not greatly improve performance: indeed stemming can lower precision rates (Tzoukerman et al. 2003). A rather strong position is taken in Church (2005), which amounts to leaving out morphological analysis altogether. His paper catalogues the IR community's repeated disappointments with morphological analysis packages, primarily due to the fact that morphological relatedness does not always equate with semantic relatedness: for example, *awful* has nothing to do with *awe*. He concludes that simple listing is preferable to attempting lexical analysis. One response is that inflectional morphologically complex words are more transparent than derivational, so less likely to be semantically disconnected from related forms. Another is that Church's focus is English which is morphologically poor anyway, whereas with other major languages such as Russian, Arabic, and Spanish listing may not be complete, and will certainly be highly redundant.

Lexical analysis is in fact increasingly important as NLP reaches beyond English to other languages, many of which have rich morphological systems. The main lexical analysis model is FSM that has a good track record for large-scale systems, English as well as multilingual. The paradigm-based model is favored by linguists as an elegant way of describing morphologically complex objects. Languages like DATR can provide a computable lexical knowledge representation of paradigm-based theories. Communities working within the two frameworks can benefit from one another. Kay (2004) observes that early language-based systems were deliberately based on scientific principles, i.e., linguistic theory. At the same time, giving theoretical claims some computational robustness led to advances in linguistic theory. The fact that many DATR theories choose to implement the morphonological variation component of their descriptions as FSTs shows the intrinsic value of FSM to morphology as a whole. The fact that there is a growing awareness of the paradigm-based model within the FSM community, for example, Roark and Sproat (2007) and Karttunen (2003) have implementations of Paradigm Function Morphology accounts in FSTs, may lead to an increasing awareness of the role paradigm relations play in morphological analysis, which may lead to enhancements in conventional FST lexical analysis. Langer (1994) gives two measures of adequacy of lexical representation: declarative expressiveness and accessing strategy. While a DATR formalized W&P theory delivers on the first, FSM by virtue of generation and parsing scores well on the second. Lexical analysis can only benefit with high scores in both.

Acknowledgments

I am extremely grateful to Roger Evans and Gerald Gazdar for their excellent comments on an earlier draft of the chapter which I have tried to take full advantage of. Any errors I take full responsibility for. I would also like to thank Nitin Indurkhya for his commitment to this project and his gentle encouragement throughout the process.

References

Armstrong, S. 1996. Multext: Multilingual text tools and corpora. In: Feldweg, H. and Hinrichs, W. (eds). *Lexikon und Text*. Tübingen, Germany: Niemeyer.

Aronoff, M. and Fudeman, K. 2005. *What is Morphology*? Oxford, U.K.: Blackwell.

Arpe, A. et al. (eds). 2005. *Inquiries into Words, Constraints and Contexts (Festschrift in Honour of Kimmo Koskenniemi on his 60th Birthday)*. Saarijärvi, Finland: Gummerus.

Baerman, M., Brown, D., and Corbett, G.G. 2005. *The Syntax-Morphology Interface: A Study of Syncretism*. Cambridge, U.K.: Cambridge University Press.

Beesley, K. and Karttunen, L. 2003. *Finite State Morphology*. Stanford, CA: CSLI.

Bonami, O. and Boyé, G. 2006. Deriving inflectional irregularity. In: Müller, S. (ed). *Proceedings of the HPSG06 Conference*. Stanford, CA: CSLI.

Booij, G. 2007. *The Grammar of Words*. Oxford, U.K.: Oxford University Press.

Brown, D. and Hippisley, A. Under review. *Default Morphology*. Cambridge: Cambridge University Press.

Brown, D. 1998. Defining 'subgender': Virile and devirilized nouns in Polish. *Lingua* 104 (3–4). 187–233.

Cahill, L. 2007. A syllable-based account of Arabic morphology. In: Soudi, A. (eds). pp. 45–66.

Cahill, L. and Gazdar, G. 1999. German noun inflection. *Journal of Linguistics* 35 (1). 1–42.

Carlson, L. 2005. Inducing a morphological transducer from inflectional paradigms. In: Arpe et al. (eds). pp. 18–24.

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

Church, K.W. 2005. The DDI approach to morphology. In Arpe et al. (eds). pp. 25–34.

Corbett, G.G. 2009. Canonical inflection classes. In: Montermini, F., Boyé, G. and Tseng, J. (eds). *Selected Proceedings of the Sixth Decembrettes: Morphology in Bordeaux*, Somerville, MA: Cascadilla Proceedings Project. www.lingref.com, document #2231, pp. 1–11.

Corbett, G.G. and Fraser, N.M. 2003. Network morphology: A DATR account of Russian inflectional morphology. In: Katamba, F.X. (ed). pp. 364–396.

Dale, R., Moisl, H., and Somers, H. (eds). 2000. *Handbook of Natural Language Processing*. New York: Marcel Dekker.

Evans, N., Brown, D., and Corbett, G.G. 2000. Dalabon pronominal prefixes and the typology of syncretism: A network morphology analysis. In: Booij, G. and van Marle, J. (eds). *Yearbook of Morphology 2000*. Dordrecht, the Netherlands: Kluwer. pp. 187–231.

Evans, N., Brown, D., and Corbett, G.G. 2002. The semantics of gender in Mayali: Partially parallel systems and formal implementation. *Language* 78 (1). 111–155.

Evans, R. and Gazdar, G. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics* 22. 167–216.

Finkel, R. and Stump, G. 2007. A default inheritance hierarchy for computing Hebrew verb morphology. *Literary and Linguistics Computing* 22 (2). 117–136.

Fraser, N. and Corbett, G.G. 1997. Defaults in Arapesh. *Lingua* 103 (1). 25–57.

Gazdar, G. 1985. Review article: Finite State Morphology. *Linguistics* 23. 597–607.

Gazdar, G. 1992. Paradigm-function morphology in DATR. In: Cahill, L. and Coates, R. (eds). *Sussex Papers in General and Computational Linguistics*. Brighton: University of Sussex, CSRP 239 pp. 43–53.

Gibbon, D. 1987. Finite state processing of tone systems. *Proceedings of the Third Conference, European ACL*, Morristown, NJ: ACL. pp. 291–297.

Gibbon, D. 1989. 'tones.dtr'. Located at ftp://ftp.informatics.sussex.ac.uk/pub/nlp/DATR/dtrfiles tones.dtr

Gildea, D. and Jurafsky, D. 1995. Automatic induction of finite state transducers for single phonological rules. *ACL* 33. 95–102.

Gildea, D. and Jurafsky, D. 1996. Learning bias and phonological rule induction. *Computational Linguistic* 22 (4). 497–530.

Goldsmith, J. 2001. Unsupervised acquisition of the morphology of a natural language. *Computational Linguistics* 27 (2). 153–198.

Haspelmath, M. 2002. *Understanding Morphology*. Oxford, U.K.: Oxford University Press.

Hockett, C. 1958. Two models of grammatical description. In: Joos, M. (ed). *Readings in Linguistic* Chicago, IL: University of Chicago Press.

Jurafsky, D. and Martin, J.H. 2007. *Speech and Language Processing.* Upper Saddle River, NJ: Pearson/Prentice Hall.

Kaplan, R.M. and Kay, M. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20. 331–378.

Karttunen, L. 2003. Computing with realizational morphology. In: Gelbukh, A. (ed). *CICLing 2003 Lecture Notes in Computer Science 2588.* Berlin, Germany: Springer-Verlag. pp. 205–216.

Karttunen, L. 2007. Word play. *Computational Linguistics* 33 (4). 443–467.

Katamba, F.X. (ed). 2003. *Morphology: Critical Concepts in Linguistics, VI: Morphology: Its Place in the Wider Context.* London, U.K.: Routledge.

Kay, M. 2004. Introduction to Mitkov (ed.). xvii–xx.

Kiraz, G.A. 2000. Multitiered nonlinear morphology using multitape finite automata. *Computational Linguistics* 26 (1). 77–105.

Kiraz, G. 2008. Book review of *Arabic Computational Morphology: Knowledge-Based and Empirical Methods. Computational Linguistics* 34 (3). 459–462.

Koskenniemi, K. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production.* Publication 11, Department of General Linguistics, Helsinki, Finland: University of Helsinki.

Krieger, H.U. and Nerbonne, J. 1992. Feature-based inheritance networks for computational lexicons. In: Briscoe, E., de Paiva, V., and Copestake, A. (eds). *Inheritance, Defaults and the Lexicon.* Cambridge, U.K.: Cambridge University Press. pp. 90–136.

Kilbury, J. 2001. German noun inflection revisited. *Journal of Linguistics* 37 (2). 339–353.

Langer, H. 1994. Reverse queries in DATR. *COLING-94.* Morristown, NJ: ACL. pp. 1089–1095.

Mel'čuk. I. 2006. *Aspects of the Theory of Morphology.* Trends in Linguistics 146. Berlin/New York: Mouton de Gruyter.

Mitkov, R. (ed). 2004. *The Oxford Handbook of Computational Linguistics.* Oxford, U.K.: Oxford University Press.

Moreno-Sandoval, A. and Goñi-Menoyo, J.M. 2002. Spanish inflectional morphology in DATR. *Journal of Logic, Language and Information.* 11 (1). 79–105.

Oncina, J., Garcia, P., and Vidal, P. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15. 448–458.

Pollard, C. and Sag, I.A. 1994. *Head-Driven Phrase Structure Grammar.* Chicago, IL: University of Chicago Press.

Pretorius, L. and Bosch, S.E. 2003. Finite-sate morphology: An analyzer for Zulu. *Machine Translation* 18 (3). 195–216.

Roark, B. and Sproat, R. 2007. *Computational Approaches to Syntax.* Oxford, U.K.: Oxford University Press.

Sag, I., Wasow, T., and Bender, E.M. (eds). 2003. *Syntactic Theory: A Formal Introduction.* Stanford, CA: CSLI.

Soudi, A. et al. (eds). 2007. *Arabic Computational Morphology: Knowledge-Based and Empirical Methods.* Dordrecht: Springer.

Sproat, R. 1997. *Multilingual Text to Speech Synthesis: The Bell Labs Approach.* Dordrecht, the Netherlands: Kluwer.

Sproat, R. 2000. Lexical analysis. In Dale, R. et al. (eds). pp. 37–58.

Sproat, R. 2007. Preface. In: Soudi, A. et al (eds). *Arabic Computational Morphology: Knowledge-Based and Empirical Methods.* Dordrecht, the Netherlands: Springer. pp. vii–viii.

Stewart, T. and Stump, G. 2007. Paradigm function morphology and the morphology-syntax interface. In: Ramchand, G. and Reiss, C. (eds). *Linguistic Interfaces.* Oxford, U.K.: Oxford University Press. pp. 383–421.

Stump, G. 2001. *Inflectional Morphology.* Cambridge, U.K.: Cambridge University Press.

Stump, G. 1993. On rules of referral. *Language* 69. 449–79.

Stump, G. 2006. Heteroclisis and paradigm linkage. *Language* 82, 279–322.
Timberlake, A.A. 2004. *A Reference Grammar of Russian*. Cambridge, U.K.: Cambridge University Press.
Tzoukermann, E., Klavans, J., and Strzalkowski, T. 2003. Information retrieval. In: Mitkov (ed.) 529–544.
Ui Dhonnchadha, E., Nic Phaidin, C., and van Genabith, J. 2003. Design, implementation and evaluation of an inflectional morphology finite-state transducer for Irish. *Machine Translation* 18 (3). 173–193.
Yarowski, D. and Wicentowski, R. 2001. Minimally supervised morphological analysis by multimodal alignment. *Proceedings of the 38th ACL*. Morristown, NJ: ACL. pp. 207–216.
Yona, S. and Wintner, S. 2007. A finite-state morphological grammar of Hebrew. *Natural Language Engineering* 14. 173–190.