# A finite-state morphological grammar of Hebrew

**Shlomo Yona**
Department of Computer Science
University of Haifa
31905 Haifa, Israel
shlomo@cs.haifa.ac.il

**Shuly Wintner**
Department of Computer Science
University of Haifa
31905 Haifa, Israel
shuly@cs.haifa.ac.il

## Abstract

Morphological analysis is a crucial component of several natural language processing tasks, especially for languages with a highly productive morphology, where stipulating a full lexicon of surface forms is not feasible. We describe HAMSAH (HAifa Morphological System for Analyzing Hebrew), a morphological processor for Modern Hebrew, based on finite-state linguistically motivated rules and a broad coverage lexicon. The set of rules comprehensively covers the morphological, morpho-phonological and orthographic phenomena that are observable in contemporary Hebrew texts. Reliance on finite-state technology facilitates the construction of a highly efficient, completely bidirectional system for analysis and generation. HAMSAH is currently the broadest-coverage and most accurate freely-available system for Hebrew.

## 1 Hebrew morphology: the challenge

Hebrew, like other Semitic languages, has a rich and complex morphology. The major word formation machinery is root-and-pattern, where roots are sequences of three (typically) or more consonants, called *radicals*, and patterns are sequences of vowels and, sometimes, also consonants, with "slots" into which the root's consonants are being inserted (interdigitation). Inflectional morphology is highly productive and consists mostly of suffixes, but sometimes of prefixes or circumfixes.

As an example of root-and-pattern morphology, consider the Hebrew[1] roots *g.d.l* and *r.e.m* and the patterns *hCCCh* and *CiCwC*, where the 'C's indicate the slots. When the roots combine with these patterns the resulting lexemes are *hgdlh*, *gidwl*, *hremh*, *riewm*, respectively. After the root combines with the pattern, some morpho-phonological alternations take place, which may be non-trivial: for example, the *htCCCwt* pattern triggers assimilation when the first consonant of the root is *t* or *d*: thus, *d.r.e+htCCCwt* yields *hdrewt*. The same pattern triggers metathesis when the first radical is *s* or *e*: *s.d.r+htCCCwt* yields *hstdrwt* rather than the expected *htsdrwt*. Frequently, root consonants such as *w* or *i* are altogether missing from the resulting form. Other *weak* paradigms include roots whose first radical is *n* and roots whose second and third radicals are identical. Thus, the roots *q.w.m*, *g.n.n*, *n.p.l* and *i.c.g*, when combining with the *hCCCh* pattern, yield the seemingly similar lexemes *hqmh*, *hgnh*, *hplh* and *hcgh*, respectively.

The combination of a root with a pattern produces a *base* (or a *lexeme*), which can then be inflected in various forms. Nouns, adjectives and numerals inflect for number (singular, plural and, in rare cases, also dual) and gender (masculine or feminine). In addition, all these three types of nominals have two phonologically distinct forms, known as the *absolute* and *construct* states. Unfortunately, in the standard orthography approximately half of the nomi-

[1] To facilitate readability we sometimes use a transliteration of Hebrew using ASCII characters:

| a | b | g | d | h | w | z | x | v | i | k |
|---|---|---|---|---|---|---|---|---|---|---|
| א | ב | ג | ד | ה | ו | ז | ח | ט | י | כ |
| l | m | n | s | y | p | c | q | r | e | t |
| ל | מ | נ | ס | ע | פ | צ | ק | ר | ש | ת |

nals appear to have identical forms in both states, a fact which substantially increases the ambiguity. In addition, nominals take pronominal suffixes which are interpreted as possessives. These inflect for number, gender and person: *spr+h→sprh* "her book", *spr+km→sprkm* "your book", etc. As expected, these processes involve certain morphological alternations, as in *mlkh+h→mlkth* "her queen", *mlkh+km→mlktkm* "your queen". Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect, which is traditionally analyzed as having the values past, present, future, imperative and infinite. Verbs can also take pronominal suffixes, which in this case are interpreted as direct objects, but such constructions are rare in contemporary Hebrew of the registers we are interested in.

These matters are complicated further due to two sources: first, the standard Hebrew orthography leaves most of the vowels unspecified. It does not explicate *[a]* and *[e]*, does not distinguish between *[o]* and *[u]* and leaves many of the *[i]* vowels unspecified. Furthermore, the single letter ו *w* is used both for the vowels *[o]* and *[u]* and for the consonant *[v]*, whereas י *i* is similarly used both for the vowel *[i]* and for the consonant *[y]*. On top of that, the script dictates that many particles, including four of the most frequent prepositions (*b* "in", *k* "as", *l* "to" and *m* "from"), the definite article *h* "the", the coordinating conjunction *w* "and" and some subordinating conjunctions (such as *e* "that" and *ke* "when"), all attach to the words which immediately follow them. Thus, a form such as *ebth* can be read as a lexeme (the verb "capture", third person singular feminine past), as *e+bth* "that+field", *e+b+th* "that+in+tea", *ebt+h* "her sitting" or even as *e+bt+h* "that her daughter". When a definite nominal is prefixed by one of the prepositions *b*, *k* or *l*, the definite article *h* is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite: *bth* can be read either as *b+th* "in tea" or as *b+h+th* "in the tea".

An added complexity stems from the fact that there exist two main standards for the Hebrew script: one in which vocalization diacritics, known as *niqqud* "dots", decorate the words, and another in which the dots are missing, and other characters represent some, but not all of the vowels. Most of the texts in Hebrew are of the latter kind; unfortunately, different authors use different conventions for the undotted script. Thus, the same word can be written in more than one way, sometimes even within the same document, again adding to the ambiguity.

In light of the above, morphological analysis of Hebrew forms is a non-trivial task. Observe that simply stipulating a list of surface forms is not a viable option, both because of the huge number of potential forms and because of the complete inability of such an approach to handle out-of-lexicon items; the number of such items in Hebrew is significantly larger than in European languages due to the combination of prefix particles with open-class words such as proper names. The solution must be a dedicated morphological analyzer, implementing the morphological and orthographic rules of the language.

Several morphological processors of Hebrew have been proposed, including works by Choueka (1980; 1990), Ornan and Kazatski (1986), Bentur et al. (1992) and Segal (1997); see a survey in Wintner (2004). Most of them are proprietary and hence cannot be fully evaluated. However, the main limitation of existing approaches is that they are ad-hoc: the rules that govern word formation and inflection are only implicit in such systems, usually intertwined with control structures and general code. This makes the maintenance of such systems difficult: corrections, modifications and extensions of the lexicon are nearly impossible. An additional drawback is that all existing systems can be used for analysis but not for generation. Finally, the efficiency of such systems depends on the quality of the code, and is sometimes sub-optimal.

## 2 Finite-state technology

*Finite-state technology* (Beesley and Karttunen, 2003) solves the three problems elegantly. It provides a language of extended regular expressions which can be used to define very natural linguistically motivated grammar rules. Such expressions can then be compiled into finite-state networks (automata and transducers), on which efficient algorithms can be applied to implement both analysis and generation. Using this methodology, a computational linguist can design rules which closely follow standard linguistic notation, and automatically ob-

tain a highly efficient morphological processor.

While the original Two-Level formulation (Koskenniemi, 1983) of finite-state technology for morphology was not particularly well suited to Semitic languages (Lavie et al., 1988), modifications of the Two-Level paradigm and more advanced finite-state implementations have been applied successfully to a variety of Semitic languages, including Ancient Akkadian (Kataja and Koskenniemi, 1988), Syriac (Kiraz, 2000) and Arabic. In a number of works, Beesley (1996; 1998; 2001) describes a finite-state morphological analyzer of Modern Standard Arabic which handles both inflectional and derivational morphology, including interdigitation. In the following section we focus on a particular finite-state toolbox which was successfully used for Arabic.

In this work we use XFST (Beesley and Karttunen, 2003), an extended regular expression language augmented by a sophisticated implementation of several finite-state algorithms, which can be used to compactly store and process very large-scale networks. XFST grammars define a binary relation (a *transduction*) on sets of strings: a grammar maps each member of a (possibly infinite) set of strings, known as the *surface*, or *lower* language, to a set of strings (the *lexical*, or *upper* language). The idea is that the surface language defines all and only the grammatical words in the language; and each grammatical word is associated with a set of lexical strings which constitutes its *analyses*. As an example, the surface string *ebth* may be associated by the grammar with the set of lexical strings, or analyses, depicted in figure 1.

XFST enables the definition of *variables*, whose values, or *denotations*, are sets of strings, or languages. Grammars can set and use those variables by applying a variety of *operators*. For example, the *concatenation* operator (unfortunately indicated by a space) can be used to concatenate two languages: the expression '`A B`' denotes the set of strings obtained by concatenating the strings in `A` with the strings in `B`. Similarly, the operator '`|`' denotes set union, '`&`' denotes intersection, '`~`' set complement, '`-`' set difference and '`*`' Kleene closure; '`$A`' denotes the set of strings containing at least one instance of a string from `A` as a substring. The empty string is denoted by '`0`' and '`?`' stands for any alphabet symbol. Square brackets are used for bracketing.

In addition to sets of strings, XFST enables the definition of binary relations over such sets. By default, every set is interpreted as the identity relation, whereby each string is mapped to itself. But relations can be explicitly defined using a variety of operators. The '`.x.`' operator denotes cross product: the expression '`A.x.B`' denotes the relation in which each string in `A` is mapped to each string in `B`. An extremely useful operation is composition: denoted by '`.o.`', it takes two *relations*, `A` and `B`, and produces a new relation of pairs $(a, c)$ such that there exists some $b$ that $(a, b)$ is a member of `A` and $(b, c)$ is a member of `B`.

Finally, XFST provides also several *replace* rules. Expressions of the form '`A->B || L _ R`' denote the relation obtained by replacing strings from `A` by strings from `B`, whenever the former occur in the context of strings from `L` on the left and `R` on the right. Each of the context markers can be replaced by the special symbol '`.#.`', indicating a word boundary. For example, the expression '`[h]->[t] || ? _ .#.`' replaces occurrences of '`h`' by '`t`' whenever the former occurs before the end of a word. Composing this example rule on an (identity) relation whose strings are various words results in replacing final `h` with final `t` in all the words, not affecting the other strings in the relation.

XFST supports diverse alphabets. In particular, it supports UTF-8 encoding, which we use for Hebrew (although subsequent examples use a transliteration to facilitate readability). Also, the alphabet can include *multi-character symbols*; in other words, one can define alphabet symbols which consist of several (print) characters, e.g., '`number`' or '`tense`'. This comes in handy when *tags* are defined, see below. Characters with special meaning (such as '`+`' or '`[`') can be escaped using the symbol '`%`'. For example, the symbol '`%+`' is a literal plus sign.

Programming in XFST is different from programming in high level languages. While XFST rules are very expressive, and enable a true implementation of some linguistic phenomena, it is frequently necessary to specify, within the rules, information that is used mainly for "book-keeping". Due to the limited memory of finite-state networks, such information is encoded in *tags*, which are multi-character symbols attached to strings. These tags

```
[+verb][+id]9430[+base]ebt[+root]ebt[+binyan]+Pa'al[+agr]+3p/F/Sg[+tense]+past
[+verb][+id]1541[+base]ebh[+root]ebh[+binyan]+Pa'al[+agr]+3p/F/Sg[+tense]+past
[+conj]e[+prep]b[+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+true
[+conj]e[+prep]b[+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+false
[+conj]e[+prep]b[+defArt][+noun][+id]19804[+base]th[+gender]+M[+number]+Sg[+construct]+false
[+conj]e[+noun][+id]19130[+base]bth[+gender]+F[+number]+Sg[+construct]+false
[+conj]e[+noun][+id]1379[+base]bt[+gender]+F[+number]+Sg[+construct]+false[+poss]+3p/F/Sg
[+noun][+id]17280[+base]ebt[+gender]+F[+number]+Sg[+construct]+false[+poss]+3p/F/Sg
```

Figure 1: The analyses of the surface string שבתה *ebth*

can be manipulated by the rules and thus propagate information among rules. For example, nouns are specified for *number*, and the number feature is expressed as a concatenation of the tag `number` with the multi-character symbol `+singular` or `+plural`. Rules which apply to plural nouns only can use this information: if `nouns` is an XFST variable denoting the set of all nouns, then the expression `$[number %+plural] .o. nouns` denotes only the plural nouns. Once all linguistic processing is complete, "book-keeping" tags are erased.

## 3 A morphological grammar of Hebrew

The importance of morphological analysis as a preliminary phase in a variety of natural language processing applications cannot be over-estimated. The lack of good morphological analysis and disambiguation systems for Hebrew is reported as one of the main bottlenecks of a Hebrew to English machine translation system (Lavie et al. (2004)). The contribution of our system is manyfold:

- HAMSAH is the broadest-coverage and most accurate publicly available morphological analyzer of Modern Hebrew. It is based on a lexicon of over 20,000 entries, which is constantly being updated and expanded, and its set of rules cover all the morphological, morphophonological and orthographic phenomena observed in contemporary Hebrew texts. Compared to Segal (1997), our rules are probably similar in coverage but our lexicon is significantly larger. HAMSAH also supports non-standard spellings which are excluded from the work of Segal (1997).
- The system is fully reversible: it can be used both for analysis and for generation.
- Due to the use of finite-state technology, the system is highly efficient. While the network has close to 2 million states and over 2 million arcs, its compiled size is approximately 4Mb and analysis is extremely fast (between 50 and 100 words per second).
- Morphological knowledge is expressed through linguistically motivated rules. To the best of our knowledge, this is the first formal grammar for the morphology of Modern Hebrew.

The system consists of two main components: a *lexicon* represented in Extensible Markup Language (XML), and a set of finite-state rules, implemented in XFST. The use of XML supports standardization, allows a format that is both human and machine readable, and supports interoperability with other applications. For compatibility with the rules, the lexicon is automatically converted to XFST by dedicated programs. We briefly describe the lexicon in section 3.1 and the rules in section 3.2.

### 3.1 The lexicon

The lexicon is a list of lexical entries, each with a *base* (citation) form and a unique *id*. The base form of nouns and adjectives is the absolute singular masculine, and for verbs it is the third person singular masculine, past tense. It is listed in dotted and undotted script as well as using a one-to-one Latin transliteration. Figure 2 depicts the lexical entry of the word *bli* "without". In subsequent examples we retain only the transliteration forms and suppress the Hebrew ones.

```
<item dotted="בְּלִי" id="4917"
   translit="bli" undotted="בלי">
  <conjunction type="coord"/>
</item>
```

Figure 2: The lexical entry of *bli* "without"

The lexicon specifies morpho-syntactic features (such as gender or number), which can later be used by parsers and other applications. It also lists several lexical proerties which are specifically targeted at morphological analysis. A typical example is the feminine suffix of adjectives, which can be one of *h*, *it* or *t*, and cannot be predicted from the base form. The lexicon lists information pertaining to non-default behavior with idiosyncratic entries.

Adjectives inflect regularly, with few exceptions. Their citation form is the absolute singular masculine, which is used to generate the feminine form, the masculine plural and the feminine plural. An additional dimension is status, which can be absolute or construct. Figure 3 lists the lexicon entry of the adjective *yilai* "supreme": its feminine form is obtained by adding the *t* suffix (hence `feminine="t"`). Other features are determined by default. This lexicon entry yields *yilai*, *yilait*, *yilaiim*, *yilaiwt* etc.

```
<item id="13852" translit="yilai">
 <adjective feminine="t" />
</item>
```

Figure 3: A lexicon item for *yilai* "supreme"

Similarly, the citation form of nouns is the absolute singular masculine form. Hebrew has grammatical gender, and the gender of nouns that denote animate entities coincides with their natural gender. The lexicon specifies the feminine suffix via the *feminine* attribute. Nouns regularly inflect for number, but some nouns have only a plural or only a singular form. The plural suffix (*im* for masculine, *wt* for feminine by default) is specified through the *plural* attribute. Figure 4 demonstrates a masculine noun with an irregular plural suffix, *wt*.

```
<item id="5044" translit="ewlxn">
 <noun gender="masculine"
       number="singular"
       plural="wt" /></item>
```

Figure 4: A lexicon item for the noun *ewlxn* "table"

Closed-class words are listed in the lexicon in a similar manner, where the specific category determines which attributes are associated with the citation form. For example, some adverbs inflect for person, number and gender (e.g., *lav* "slowly"), so this is indicated in the lexicon. The lexicon also specifies the person, number and gender of pronouns, the type of proper names (location, person, organization), etc. The lexical representation of verbs is more involved and is suppressed for lack of space.

Irregularities are expressed directly in the lexicon, in the form of additional or alternative lexical entries. This is facilitated through the use of three optional elements in lexicon items: *add*, *replace* and *remove*. For example, the noun *chriim* "noon" is also commonly spelled *chrim*, so the additional spelling is specified in the lexicon, along with the standard spelling, using *add*. As another example, consider Segolate nouns such as *bwqr* "morning". Its plural form is *bqrim* rather than the default *bwqrim*; such stem changing behavior is specified in the lexicon using *replace*. Finally, the verb *ykwl* "can" does not have imperative inflections, which are generated by default for all verbs. To prevent the default behavior, the superfluous forms are *remove*d.

The processing of irregular lexicon entries requires some explanation. Lexicon items containing *add*, *remove* and *replace* elements are included in the general lexicon without the *add*, *remove* and *replace* elements, which are listed in special lexicons. The general lexicon is used to build a basic morphological finite-state network. Additional networks are built using the same set of rules for the *add*, *remove* and *replace* lexicons. The final network is obtained by subtracting the *remove* network from the general one (using the set difference operator), adding the *add* network (using the set union operator), and finally applying *priority union* with the *replace* network. This final finite-state network contains only and all the valid inflected forms.

The lexicon is represented in XML, while the morphological analyzer is implemented in XFST, so the former has to be converted to the latter. In XFST, a lexical entry is a relation which holds between the surface form of the lemma and a set of lexical strings. As a surface lemma is processed by the rules, its associated lexical strings are manipulated to reflect the impact of inflectional morphology. The surface string of XFST lexical entries is the citation form specified in the XML lexicon. Figure 5

lists the XFST representation of the lexical entry of the word *bli*, whose XML representation was listed in figure 2.

```
[+negation][+id]21542[+undotted]
בלי[+translit]bli
```

Figure 5: The lexicon item of *bli* in XFST

### 3.2 Morphological and orthographic rules

In this section we discuss the set of rules which constitute the morphological grammar, i.e., the implementation of linguistic structures in XFST. The grammar includes hundreds of rules; we present a small sample, exemplifying the principles that govern the overall organization of the grammar. The linguistic information was collected from several sources (Barkali, 1962; Zdaqa, 1974; Alon, 1995; Cohen, 1996; Schwarzwald, 2001; Schwarzwald, 2002; Ornan, 2003).

The grammar consists of specific rules for every part of speech category, which are applied to the appropriate lexicons. For each category, a variable is defined whose denotation is the set of all lexical entries of that category. Combined with the category-specific rules, we obtain morphological grammars for every category (not including idiosyncrasies). These grammars are too verbose on the lexical side, as they contain all the information that was listed in the lexicon. Filters are therefore applied to the lexical side to remove the unneeded information.

Our rules support surface forms that are made of zero or more prefix particles, followed by a (possibly inflected) lexicon item. Figure 6 depicts the high-level organization of the grammar (recall from section 2 that '`.o.`' denotes composition). The variable `inflectedWord` denotes a union of all the possible inflections of the entire lexicon. Similarly, `prefixes` is the set of all the possible sequences of prefixes. When the two are concatenated, they yield a language of all possible surface forms, vastly over-generating. On the upper side of this language a prefix particle filter is composed, which enforces linguistically motivated constraints on the possible combinations of prefixes with words. On top of this another filter is composed, which handles "cosmetic" changes, such as removing "book-keeping" tags. A similar filter is applied to the the lower side of the network.

```
tagAffixesFilter
.o.
prefixesFilters
.o.
[ prefixes inflectedWord ]
.o.
removeTagsFilter
```

Figure 6: A high level view of the analyzer

As an example, consider the feminine singular form of adjectives, which is generated from the masculine singular by adding a suffix, either *h*, *it* or *t*. Some idiosyncratic forms have no masculine singular form, but do have a feminine singular form, for example *hrh* "pregnant". Therefore, as figure 7 shows, singular feminine adjectives are either extracted verbatim from the lexicon or generated from the singular masculine form by suffixation. The rule `[ %+feminine <- ? || %+gender _ ]` changes the gender attribute to `feminine` for the inflected feminine forms. This is a special form of a replace rule which replaces any symbol ('`?`') by the multi-character symbol '`+feminine`', in the context of occurring after '`+gender`'. The right context is empty, meaning *anything*.

```
define femininеSingularAdjective [
 [$[%+gender [%+feminine]]
  .o. adjective ] |
 [ %+feminine <- ? || %+gender _ ]
 .o. [ sufH | sufT | sufIT ]
];
```

Figure 7: Feminine adjectives

Figure 8 shows how the suffix *h* (the value of the variable `HE`) is used in the inflection. The default is not to add an additional *h* if the masculine adjective already terminates with it, as in *mwrh* "male teacher" → *mwrh* "female teacher". This means that exceptions to this default, such as *gbwh* "tall, m" → *gbwhh* "tall, f", are being improperly treated. Such forms are explicitly listed in the lexicon as idiosyncrasies (using the add/replace/remove mechanism), and will be corrected at a later stage. The suffixes *t*

and *it* are handled in a similar way.

```
define sufH [
 [ [ $[%+feminine %+h] .o.
     masculineSingularAdjective ]
   [ 0 .x. addedHE ] ]
 .o. [ addedHE -> 0 || HE _ .#. ]
 .o. [ addedHE -> HE ]
];
```

Figure 8: Adding the suffix *h*

Figure 9 shows how plural nouns with the *wt* suffix are processed. On the lower side some conditional alternations are performed before the suffix is added. The first alternation rule replaces *iih* with *ih* at the end of a word, ensuring that nouns wrttent with a spurious *i* such as *eniih* "second" are properly inflected as *eniwt* "seconds" rather than *eniiwt*. The second alternation rule removes final *t* to ensure that a singular noun such as *meait* "truck" is properly inflected to its plural form *meaiwt*. The third ensures that nouns ending in *wt* such as *smkwt* "authority" are properly inflected as *smkwiwt*. Of course, irregular nouns such as *xnit* "spear", whose plural is *xnitwt* rather than *xniwt*, are lexically specified and handled separately. Finally, a final *h* is removed by the fourth rule, and subsequently the plural suffix is concatenated.

```
define pluralWTNoun [
 [
  [ %+plural <- %+singular || %+number _ ]
  .o. $[%+number %+singular]
  .o. $[%+plural %+wt]
  .o. noun
  .o. [ YOD YOD HE -> YOD HE || _ .#. ]
  .o. [ ALEF YOD TAV -> ALEF YOD || _ .#. ]
  .o. [ VAV TAV -> VAV YOD || _ .#. ]
  .o. [ [HE|TAV] -> 0 || _ .#. ]
 ] [ 0 .x. [VAV TAV] ]
];
```

Figure 9: Plural nouns with *wt* suffix

The above rules only superficially demonstrate the capabilities of our grammar. The bulk of the grammar consists of rules for inflecting verbs, including a complete coverage of the weak paradigms. The grammar also contains rules which govern the possible combinations of prefix particles and the words they combine with.

## 4 Conclusion

We described a broad-coverage finite-state grammar of Modern Hebrew, consisting of two main components: a lexicon and a set of rules. The current underlying lexicon includes over 20,000 items. The average number of inflected forms for a lexicon item is 33 (not including prefix sequences). Due to the use of finite-state technology, the grammar can be used for generation or for analysis. It induces a very efficient morphological analyzer: in practice, over eighty words per second can be analyzed on a contemporary workstation.

For lack of space we cannot fully demonstrate the output of the analyzer; refer back to figure 1 for an example. HAMSAH is now used for a number of projects, including as a front end for a Hebrew to English machine translation system (Lavie et al., 2004). It is routinely tested on a variety of texts, and tokens with zero analyses are being inspected manually. A systematic evaluation of the quality of the analyzer is difficult due to the lack of available alternative resources. Nevertheless, we conducted a small-scale evaluation experiment by asking two annotators to review the output produced by the analyzer for a randomly chosen set of newspaper articles comprising of approximately 1000 word tokens. The following table summarizes the results of this experiment.

| | number | % |
|---|---|---|
| tokens | 959 | 100.00% |
| no analysis | 37 | 3.86% |
| no correct analysis | 41 | 4.28% |
| correct analysis produced | 881 | 91.86% |

The majority of the missing analyses are due to out-of-lexicon items, particularly proper names.

In addition to maintenance and expansion of the lexicon, we intend to extend this work in two main directions. First, we are interested in automatic methods for expanding the lexicon, especially for named entities. Second, we are currently working on a disambiguation module which will rank the analyses produced by the grammar according to context-dependent criteria. Existing works on part-of-speech tagging and morphological disambiguation in Hebrew (Segal, 1999; Adler, 2004; Bar-Haim, 2005) leave much room for further research. Incorporating state-of-the-art machine learning techniques

for morphological disambiguation to the output produced by the analyzer will generate an optimal system which is broad-coverage, effective and accurate.

## Acknowledgments

This work was funded by the Israeli Ministry of Science and Technology, under the auspices of the Knowledge Center for Processing Hebrew. We are grateful to Yael Cohen-Sygal, Shira Schwartz and Alon Itai for their help.

## References

Meni Adler. 2004. Word-based statistical language modeling: Two-dimensional approach. Thesis proposal, Ben Gurion University, Beer Sheva, April.

Emmanuel Alon. 1995. *Unvocalized Hebrew Writing: The Structure of Hebrew Words in Syntactic Context*. Ben-Gurion University of the Negev Press. In Hebrew.

Roy Bar-Haim. 2005. Part-of-speech tagging for Hebrew and other Semitic languages. Master's thesis, Computer Science Department, Technion, Haifa, Israel.

Shaul Barkali. 1962. *Lux HaP'alim HaShalem (The Complete Verbs Table)*. Reuven Mass, Jerusalem. In Hebrew.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. CSLI, Stanford.

Kenneth R. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of COLING-96, the 16th International Conference on Computational Linguistics*, Copenhagen.

Kenneth R. Beesley. 1998. Arabic morphology using only finite-state operations. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57, Montreal, Quebec, August. COLING-ACL'98.

Kenneth R. Beesley. 2001. Finite-state morphological analysis and generation of Arabic at Xerox Research: Status and plans in 2001. In *ACL Workshop on Arabic Language Processing: Status and Perspective*, pages 1–8, Toulouse, France, July.

Esther Bentur, Aviella Angel, Danit Segev, and Alon Lavie. 1992. Analysis and generation of the nouns inflection in Hebrew. In Uzzi Ornan, Gideon Arieli, and Edit Doron, editors, *Hebrew Computational Linguistics*, chapter 3, pages 36–38. Ministry of Science and Technology. In Hebrew.

Yaacov Choueka. 1980. Computerized full-text retrieval systems and research in the humanities: The Responsa project. *Computers and the Humanities*, 14:153–169.

Yaacov Choueka. 1990. MLIM - a system for full, exact, on-line grammatical analysis of Modern Hebrew. In Yehuda Eizenberg, editor, *Proceedings of the Annual Conference on Computers in Education*, page 63, Tel Aviv, April. In Hebrew.

Haim A. Cohen. 1996. klalei ha-ktiv xasar ha-niqqud. *leshonenu la&am*, special edition, May. In Hebrew.

Laura Kataja and Kimmo Koskenniemi. 1988. Finite-state description of Semitic morphology: A case study of Ancient Akkadian. In *COLING*, pages 313–315.

George Anton Kiraz. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, March.

Kimmo Koskenniemi. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. The Department of General Linguistics, University of Helsinki.

Alon Lavie, Alon Itai, Uzzi Ornan, and Mori Rimon. 1988. On the applicability of two-level morphology to the inflection of Hebrew verbs. In *Proceedings of the International Conference of the ALLC*, Jerusalem, Israel.

Alon Lavie, Shuly Wintner, Yaniv Eytani, Erik Peterson, and Katharina Probst. 2004. Rapid prototyping of a transfer-based Hebrew-to-English machine translation system. In *Proceedings of TMI-2004: The 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD, October.

Uzzi Ornan and Wadim Kazatski. 1986. Analysis and synthesis processes in Hebrew morphology. In *Proceedings of the 21 National Data Processing Conference*. In Hebrew.

Uzzi Ornan. 2003. *The Final Word*. University of Haifa Press, Haifa, Israel. In Hebrew.

Ora Schwarzwald. 2001. *Moden Hebrew*, volume 127 of *Languages of the World/Materials*. LINCOM EUROPA.

Ora Schwarzwald. 2002. *Studies in Hebrew Morphology*. The Open University of Israel.

Erel Segal. 1997. Morphological analyzer for unvocalized hebrew words. Unpublished work, available from `http://www.cs.technion.ac.il/~erelsgl/hmntx.zip`.

Erel Segal. 1999. Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Technion, Israel Institute of Technology, Haifa, October. In Hebrew.

Shuly Wintner. 2004. Hebrew computational linguistics: Past and future. *Artificial Intelligence Review*, 21(2):113–138.

Yizxaq Zdaqa. 1974. *Luxot HaPoal (The Verb Tables)*. Kiryath Sepher, Jerusalem. In Hebrew.