*Natural Language Engineering* **1** (1): 000–000. Printed in the United Kingdom

# *A finite-state morphological grammar of Hebrew*

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

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

( *Received 9 May 2006*)

**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. This paper describes 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.

## 1 Introduction

Morphological analysis is a crucial component of most natural language processing (NLP) systems. Whether the goal of an application is information retrieval, question answering or machine translation, NLP applications must be aware of word structure. For some languages and for some applications, simply stipulating a list of surface forms is a viable option; this is not the case for languages with complex morphology, in particular Hebrew, 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 many European languages due to the combination of prefix particles with open-class words such as proper names. An alternative solution would be a dedicated morphological analyzer, implementing the morphological and orthographic rules of the language.

Existing morphological analyzers of Hebrew are limited (see section 1.2). 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). In this work we present 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 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 covers all the morphological, morpho-phonological and orthographic phenomena observed in contemporary Hebrew texts.
- 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. Its compiled size is approximately 4Mb and analysis speed is 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 remainder of this section illustrates the main aspects of Hebrew morphology and briefly reviews finite-state technology, and in particular the XFST system that was used for this work. Section 2 describes the lexicon and section 3 the morphological grammar. The system is evaluated in section 4 and we conclude with directions for future work.

### *1.1 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 inserted. 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

[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 |

All these characters can be considered consonants; vowels are usually not explicit in the script.

*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 nominals 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.

## *1.2 Related work*

Several morphological processors of Hebrew have been proposed (Choueka, 1980; Choueka, 1990; Ornan and Kazatski, 1986; Bentur *et al.*, 1992; Segal, 1997); see a survey in (Wintner, 2004). Most of them are proprietary and hence cannot be fully evaluated. Compared to the only publicly available system (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). See section 4 for a comparison of the two systems.

The main limitation of most 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.

*Finite-state technology* (Kaplan and Kay, 1994; Roche and Schabes, 1997) 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), to 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 obtain 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. We conclude that finite-state technology *is* indeed suitable for encoding Hebrew morphology.

## *1.3 Finite-state technology*

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 שבתה *ebth* may be associated by the grammar with the set of lexical strings, or analyses, depicted in figure 1.

```
[+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
```

Fig. 1. The analyses of the surface string שבתה *ebth*

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 (implied by specifying two expressions in a sequence) 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 also provides 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 `L` and `R` can be replaced by the special symbol ‘`.#.`’, indicating a word boundary;[2] each context marker can be omitted when the context is unconstrained. 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.

[2] The usage of boundary symbols is actually more liberal: they can occur at the far left side of any left context and at the far right side of any right context.

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 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 can be erased by a replace rule which maps them to `0`.

## 2 The lexicon

HAMSAH consists of two main components: a *lexicon*, represented in Extensible Markup Language (XML), and a set of finite-state rules (section 3), 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 a dedicated program.

The lexicon (Itai *et al.*, 2006) 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. The choice of the base form is natural: first, these forms are the traditional citation forms used in standard dictionaries; more importantly, these also happen to be the 'zero' inflected forms. For example, the feminine and plural forms of adjectives can all be obtained from the singular masculine (citation) form by concatenation. The base form 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.

In addition to the citation form and a unique ID, each lexicon item is specified for a major part of speech (POS) category, and sometimes for a subcategory. The choice of categories (and features) is inspired by existing standards (Ide *et al.*, 2003) but, following (Wynne, 2005), is consistent with traditional descriptions of Hebrew which are common in dictionaries and textbooks. Depending on the category, the

```
<item dotted="בְלִי" id="4917" translit="bli" undotted="בלי">
  <negation/>
</item>
```

Fig. 2. The lexical entry of *bli* "without"

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 properties which are specifically targeted at morphological analysis, such as non-default or idiosyncratic behavior.

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. Adjectives inflect regularly, with few exceptions. Their citation form is the absolute singular masculine, which is used to generate the entire paradigm. 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, such as the plural suffix, are determined by default. This lexicon entry is used by the morphological analyzer to generate the entire paradigm of the lexeme, including the feminine singular *yilait*, the masculine plural *yilaiim*, the feminine plural *yilaiwt* etc.

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

Fig. 3. The lexical entry of *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>
```

Fig. 4. The lexical entry of 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. For verbs, the lexicon stores two main pieces of information: a root and an *inflection pattern* (IP). The latter is a combination of the traditional *binyan* with some information about peculiarities of the inflectional paradigm of verbs in this *binyan*. Such information is required because of some arbitrariness in the way verbs inflect, even in the regular patterns. For example, the second person singular masculine future form of the roots *p.s.l* and *e.k.b* in the first *binyan* (*pa'al*) is *tipswl* and *tiekb*, respectively. Note the additional '*w*' in the first form which is missing in the second: both roots are regular, and such information must be encoded in the lexicon to indicate the different inflected forms. Inflection patterns are simply numbers, each referring to a paradigm of inflection. It is instructive to think of inflection patterns as collapsing together several of the 'rows' in traditional verb paradigm tables such as (Barkali, 1962) or (Zdaqa, 1974). When the lexicon is converted from XML to XFST (see below), information on root and IP is combined to generate the various inflection bases, which are later used by the grammar to generate the complete inflection paradigm. An abbreviated lexical entry of a verb is exemplified in figure 5.

```
<item id="7278" script="formal" translit="psq">
  <verb inflectionPattern="1" root="psq"/>
</item>
```

Fig. 5. The lexical entry of the verb *psq* "stop"

Irregularities and variation can be expressed directly in the lexicon, in the form of additional or alternative lexical entries. This is facilitated through the use of three optional XML 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[3] such as *cwrk* "need". Its plural form is *crkim* rather than the default *cwrkim*; such stem changing behavior is specified in the lexicon using *replace*. Finally, it is sometimes required to prevent part of the inflectional paradigm of words from being generated. For example, the verb *nhnh* "enjoy" 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

[3] Most Hebrew words have ultimate stress; Segolate nouns are a family of words with penultimate stress.

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 6 lists the XFST representation of the lexical entry of the word *bli*, whose XML representation was listed in figure 2. The lexical entry of the adjective *yilai* “supreme”, whose XML representation was listed in figure 3, has the XFST representation depicted in figure 7. Note that default values (e.g., for the plural suffix) are expanded in the XFST representation.

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

Fig. 6. The lexicon entry of *bli* in XFST

```
[+adjective][+id]13852[+undotted] עילאי[+translit]yilai
[+gender]+masculine[+number]+singular[+feminine]+t[+plural]+im
```

Fig. 7. The lexicon entry of *yilai* in XFST

The conversion of XML to XFST is more involved for verbs than for other categories. Recall that the lexicon specifies for each verb a root and an inflection pattern. Out of this information, the conversion script generates five inflection bases, one for each tense. This is illustrated in figure 8. Note in particular that this takes care of *interdigitation*, the morphological process of intertwining the root consonants with the pattern. Given these bases, verb inflection is purely concatenative.

```
[+verb][+id]7278[+translit]psq[+root]psq[+binyan]%+Pa’al[+inflectionPattern]1
.x.
[+ip1][+participle]pswq[+past]psq[+present]pwsq[+future]pswq[+imperative]pswq
```

Fig. 8. The lexicon entry of the verb *psq* in XFST

The lexicon was initially populated with a small number of words (about 200) used for development of the morphological analyzer. Then, about 3000 nouns and adjectives were automatically acquired from the HSpell lexicon (Har’El and Kenigsberg, 2004). The base forms of HSpell were taken directly, whereas features reflecting the morphological behavior of those forms were converted from the internal representation of HSpell to the XML format discussed above. Over 3500 verbs were added by typing in the roots and inflection patterns of (Zdaqa, 1974). Remaining entries were added manually by a lexicographer using a graphical user interface specifically developed for this purpose. This process is still ongoing, although we currently focus mainly on proper names. Over 13,000 of the entries in the lexicon are dotted (vocalized), and we continue to add dotted forms to the remaining entries. Table 1 lists the number of words in the lexicon by main part of speech.

| | | | |
|---|---|---|---|
| Noun | 10,294 | Preposition | 101 |
| Verb | 4,479 | Conjunction | 67 |
| Proper name | 4,192 | Pronoun | 62 |
| Adjective | 1,637 | Interjection | 40 |
| Adverb | 358 | Interrogative | 9 |
| Quantifier | 134 | Negation | 61 |
| Total: | | 21,379 | |

Table 1. *Size of the lexicon by main part of speech*

## 3 Morphological and orthographic rules

This section discusses the set of rules which constitute the morphological grammar, i.e., the implementation of linguistic structures in XFST. The grammar includes hundreds of rules; only a small sample is presented, 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, a morphological grammar for every category (not including idiosyncrasies) is obtained. These grammars are too verbose on the lexical side, as they contain all the information that was listed in the lexicon, as well as "book-keeping" tags. Filters are therefore applied to the lexical side to remove the unneeded information.

The rules support surface forms that are made of zero or more prefix particles, followed by a (possibly inflected) lexicon item. Figure 9 depicts the high-level organization of the grammar (recall from section 1.3 that '`.o.`' denotes composition whereas concatenation is denoted by a space). 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 (inflected) stems. On top of this another filter, `tagAffixesFilter`, is composed, which handles "cosmetic" changes, such as removing "book-keeping" tags. A similar filter (`removeTagsFilter`) is applied to the the lower side of the network.

As mentioned above, the variable `inflectedWord` denotes the entire language of (possibly inflected) lexical items, each paired with its analysis. This is obtained by

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

Fig. 9. A high level view of the analyzer

a union of the various category-specific networks. We now take a closer look into some of these networks. For simplicity, we begin with adjectives, whose inflectional paradigm is relatively regular.

First, the variable `msAdj` is defined, by selecting from the entire adjective lexicon only those items whose `gender` is either `masculine` or `masculine and feminine`, and whose `number` is `singular`, as in figure 10.

```
define msAdj [
  $[%+gender [%+masculine| %+masculine% and% feminine]]
  .o.
  $[%+number %+singular]
  .o.
  adjective
];
```

Fig. 10. Masculine singular adjectives

The variable msAdj is now the basis for generating the entire inflectional paradigm of adjectives. The generation of plural forms is exemplified in figure 11. On top of the `msAdj` network, a replace rule changes the value of `number` from `singular` to `plural`; on the lower side of the network (corresponding to surface strings), the letter *h* (the value of the variable `HE`) is elided when it is word final (i.e., preceding the end of word position, denoted by `.#.`); then, the plural suffix *im* (`YOD FINALMEM`) is concatenated to the lower side.

```
define pluralIMAdjective [
  [ [ %+plural <- %+singular || %+number _ ]
    .o.
    msAdj
    .o.
    [ HE -> 0 || _ .#. ]
  ] [0 .x.  [ YOD FINALMEM ]]
];
```

Fig. 11. Masculine plural adjectives

In a similar way, the feminine singular form of adjectives is generated from the masculine singular by adding a suffix, either *h*, *it* or *t* (figure 12). A replace rule which changes the `gender` to `feminine` is composed on top of the union of three networks, each dealing with one of the three possible feminine suffixes of adjectives. Consider the network `sufT`: it first selects, from all the masculine singular adjectives, those whose `feminine` attribute is `t`; and then concatenates *t* (`TAV`) to the end of the surface string. Similarly, the network `sufH` adds an *h* suffix but takes care to elide a final *h* if the masculine form happens to end in it. It should now be easy to see how the entire adjective paradigm, including feminine plural, construct state and forms with attached pronominal suffixes are generated from the basic forms. The rules are varied, but they are a true implementation of the morphological phenomena.

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

define sufT [
  [ $[%+feminine %+t] .o. msAdj ] [ 0 .x. [TAV] ]
];

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

Fig. 12. Feminine singular adjectives

The same holds for nouns, which are demonstrated using one of the more complicated cases, namely the *wt* plural suffix (figure 13). This example should best be read from the center: the plural inflection of nouns is based on the singular form, hence the use of the variable `noun`, denoting the entire noun lexicon; on top of it, a filter is applied which selects only those nouns whose `number` is `singular`; and on top of that, another filter selecting those singular nouns whose `plural` attribute is `wt`. Finally, a replace rule changes the value of the `number` attribute from `singular` to `plural`.

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 written with a spurious *i* such as *eniih* "second" are properly inflected as *eniwt* "seconds" rather than *eniiwt*. The second alternation rule removes a final *t* to ensure that a singular noun such as *meait* "truck" is properly construed as *meaiwt* (rather than *meaitwt*). The third ensures that nouns ending in *wt* such as *smkwt* "authority" are properly inflected as *smkwiwt* (rather than *smkwtwt*). Finally, a final *h* or *t* is removed by the fourth rule, and subsequently the plural suffix is concatenated.

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

Fig. 13. Plural nouns with *wt* suffix

The most complicated part of the grammar is the one that handles verbs, of course. Recall that the XFST representation of verbs has a dedicated inflection base for each tense (figure 8). In addition, the inflection pattern (IP) of the verb is specified. The IP defines which rules are applied to the verb. Since the inflection base is given, the task of the rules is relatively simple.

As an example, consider the simple case of the regular verb *psq* "stop", whose past tense inflection base is *psq*. To generate the first person singular form, the rule of figure 14 is used. The variable `pastBase` denotes the past tense verbs; on top of it, a replace rule sets the value of the `pgn` (person/gender/number) attribute to `1p/MF/Sg`; and on the lower side, the suffix *ti* (`TAV YOD`) is concatenated. Applying this rule to *psq* yields *psqti*.

```
define ip1Past1pMFSg [
 [
  [ %+1p%/MF%/Sg <- ? || %+pgn _ ]
  .o.
  pastBase
  ] [ 0 .x. [ TAV YOD ] ]
];
```

Fig. 14. First person singular past, verbs of IP 1

Consider now a verb of some weak paradigm, e.g., *rch* "want". Its IP is 13, which triggers a different rule for the same combination of person, number, gender and tense. This rule is depicted in figure 15. Note how the final *h* of the base is changed to *i* before the suffix is added, yielding *rciti* (rather than *rchti*) as the first person singular past form. Other weak roots are handled in the same manner, through dedicated inflection rules for each IP.

Finally, the grammar also includes rules which govern the possible combinations of prefix particles and the words they combine with. While there are only a handful

```
define ip13Past1pMFSg [
 [
  [ %+1p%/MF%/Sg <- ? || %+pgn _ ]
  .o.
  pastBase
  .o.
  [HE -> YOD || _ .#.]
  .o.
  [TAV -> 0 || _ .#.]
  ] [ 0 .x. [ TAV YOD ] ]
];
```

Fig. 15. First person singular past, verbs of IP 13

of such particles, they combine with each other in intricate ways. The rules that govern such combinations, as well as the constraints on the combination of prefixes with stems, are basically syntactic, since most of the prefixes are syntactically independent words. Figure 16 depicts the grammar fragment which determines the valid sequences of prefixes (where parentheses denote optionality). It handles the coordinating conjunction *w* (`conVAV`), the subordinating conjunctions *e, ke, me, lke*, the prepositions *b, l, k* and *m*, the definitie article *h* and the adverbial *k*.

```
define prefixalParticles [
 [
  (conVAV)
  (subConOrRelSHIN)
  (tempSubConKAFSHIN|tempSubConMEMSHIN|tempSubConLAMEDKAFSHIN)
  ([[prepBET|prepLAMED|prepKAF] ([definiteArticleTag:0])] |
    [prepMEM (defArtHE)] |
    [defArtHE])
  (adverbKAF)
 ]
];
```

Fig. 16. A grammar of prefix sequences

## 4 Evaluation

The analyzer is routinely tested on a variety of texts, and tokens with zero analyses are inspected manually. A systematic evaluation of the quality of the analyzer is difficult due to several reasons. First, there exist very few annotated corpora of Hebrew which can be used for evaluation (Wintner, 2004; Wintner and Yona, 2003); and the few that exist were used for the development of the grammar (as well as for the development of other analyzers, notably (Segal, 1997)). Second, HAMSAH is the first morphological analyzer of Hebrew whose output is standardized (Wintner and Yona, 2003); the format of other analyzers is different, and an automatic comparison of the output of two systems is problematic. Finally, since the grammar produces *all* the analyses of any given form, it is very difficult for a human annotator to detect omissions (under-generation): sometimes the analyses produced by the system are very plausible, whereas a rare, non-prominent analysis is missing. We therefore refrain from providing common evaluation measures such as precision and recall.

In light of the difficulties, we conducted a small-scale evaluation experiment to test the coverage and accuracy of the output of the analyzer and to compare it to the previous state-of-the-art (Segal, 1997). We selected five documents from four sections of the Hebrew daily newspaper HaAretz[4] (two from News, and one each from Economy, Sports and Leisure); we then selected the first 200 tokens of each of the documents, and analyzed them using HAMSAH and using the analyzer of Segal (1997). This yielded a small corpus of 1000 tokens, 658 types. Two human annotators manually inspected the outputs and indicated missing as well as spurious analyses in each of the systems. Differences between the annotators were consolidated so that a consensus was reached.

Both of the systems are capable of producing a proper name analysis for out-of-lexicon items; in most cases, this analysis is correct. Both systems also attempt to separate potential prefixes from forms which obtain no analysis, and to produce more than one proper name analysis. For example, the token *windzwr* "Windsor" is not included in the lexicon of either system; both therefore produce two analyses for this token, the proper name *windzwr* and the coordinating conjunction followed by a proper name: *w-indzwr*. When the proper name reading is correct, it is counted as a successful analysis; spurious proper name readings (as in the case of *windzwr*) are counted as spurious analyses.

The results of this evaluation are summarized in table 2. We report, for each of the systems, the number of missing analyses (here, a single token with $k$ missing analyses contributes $k$ to the sum); the number of tokens with missing analyses (here, a single token with $k$ missing analyses contributes 1 to the count); the number of spurious analyses; and the number of tokens with spurious analyses.

| | HAMSAH | Segal (1997) |
|---|---|---|
| Number of missing analyses | 119 | 401 |
| Tokens with missing analyses | 81 | 226 |
| Number of spurious analyses | 56 | 135 |
| Tokens with missing analyses | 45 | 97 |

Table 2. *Evaluation experiments on 1000 tokens, 658 types*

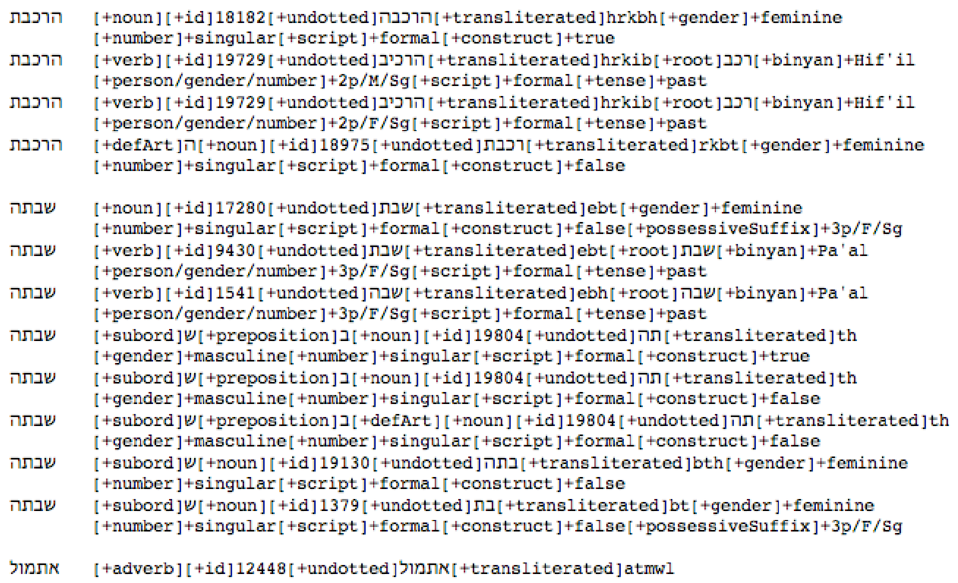
As the data show, approximately 12% of the tokens are problematic (have missing or spurious analyses). A careful error analysis reveals that the majority of the missing analyses can be attributed to missing lexical entries, mostly proper names;

[4] HaAretz, `http://www.haaretz.co.il`, April 21st, 2006.

the remaining omissions are due to a small number of rules which can be easily corrected. As for spurious analyses, most of those are also due to lexical omissions: when the analyzer cannot produce an analysis for a given form, a proper name is output, which in many cases is a wrong analysis (and is counted as a spurious one). The rules themselves do not over-generate.

## 5 Conclusion

We described HAMSAH, 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. A web-based interface to the system is available for demonstrating the capabilities of the analyzer, and it has been used extensively since it came on-line (an average of 1114 hits per month in the first three months of 2006). To exemplify the output of the system, figure 17 depicts the analyses obtained for the words הרכבת שבתה אתמול. HAMSAH is now used for a number of projects, including a front end for a cross-lingual information retrieval system and a Hebrew to English machine translation system (Lavie *et al.*, 2004).



```
הרכבת   [+noun][+id]18182[+undotted]הרכבה[+transliterated]hrkbh[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+true
הרכבת   [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
        [+person/gender/number]+2p/M/Sg[+script]+formal[+tense]+past
הרכבת   [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
        [+person/gender/number]+2p/F/Sg[+script]+formal[+tense]+past
הרכבת   [+defArt]ה[+noun][+id]18975[+undotted]רכבת[+transliterated]rkbt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false

שבתה    [+noun][+id]17280[+undotted]שבת[+transliterated]ebt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg
שבתה    [+verb][+id]9430[+undotted]שבת[+transliterated]ebt[+root]שבת[+binyan]+Pa'al
        [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה    [+verb][+id]1541[+undotted]שבה[+transliterated]ebh[+root]שבה[+binyan]+Pa'al
        [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה    [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+true
שבתה    [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+preposition]ב[+defArt][+noun][+id]19804[+undotted]תה[+transliterated]th
        [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+noun][+id]19130[+undotted]בתה[+transliterated]bth[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false
שבתה    [+subord]ש[+noun][+id]1379[+undotted]בת[+transliterated]bt[+gender]+feminine
        [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg

אתמול   [+adverb][+id]12448[+undotted]אתמול[+transliterated]atmwl
```

Fig. 17. The analyses of הרכבת שבתה אתמול

In addition to maintenance and expansion of the lexicon, we intend to extend this work in three main directions. First, we are interested in automatic methods for expanding the lexicon, especially for named entities. Currently, the analyzer is capable of stripping potential prefixes off surface forms, thereby proposing more than one option for a proper name analysis of out-of-lexicon words. We believe that such proposals can be the basis of a lexicon expansion procedure: proper names are not inflected, but they do combine with prefixes. It should therefore be possible to use un-annotated corpora for verifying whether a proposed form is indeed a proper name. For example, consider the form *windzwr* discussed in the previous section. The analyzer currently produces two possible proper name readings for it, namely *windzwr* and *indzwr*. Searching a Hebrew corpus is likely to reveal forms such as *bwindzwr* "in Windsor" or *lwindzwr* "to Windsor", but not *bindzwr* or *lindzwr*. This indicates that *windzwr* is a more likely analysis than *w+indzwr*.

Second, we would like to improve the handling of multi-word tokens. These are lexical units which consist of more than one token, and recent studies in several languages reveal that they are highly common in all types of texts. In Hebrew, such tokens interact with the peculiarities of the orthography and the complexity of the morphology and pose a real challenge for lexical representation and computational analysis. We hope to address these issues in the future.

Finally, 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; Bar-Haim *et al.*, 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. Special thanks are due to Danny Shacham for developing the annotation tool. We are also indebted to the anonymous reviewers for useful comments. All remaining errors are, of course, our own.

## References

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

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

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 39–46, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

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

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

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

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

Kenneth R. Beesley. 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 1998. COLING-ACL'98.

Kenneth R. Beesley. 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 2001.

Esther Bentur, Aviella Angel, Danit Segev, and Alon Lavie. 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, 1992. In Hebrew.

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

Yaacov Choueka. 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 1990. In Hebrew.

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

Nadav Har'El and Dan Kenigsberg. Hspell: a free Hebrew speller. Available from `http://www.ivrix.org.il/projects/spell-checker/`, 2004.

Nancy Ide, Laurent Romary, and Eric de la Clergerie. International standard for a linguistic annotation framework. In *SEALTS '03: Proceedings of the HLT-NAACL 2003 workshop on Software engineering and architecture of language technology systems*, pages 25–30, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Alon Itai, Shuly Wintner, and Shlomo Yona. A computational lexicon of contemporary Hebrew. In *Proceedings of The fifth international conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy, May 2006.

Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, September 1994.

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

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

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

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

Alon Lavie, Shuly Wintner, Yaniv Eytani, Erik Peterson, and Katharina Probst. 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 2004.

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

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

Emmanuel Roche and Yves Schabes, editors. *Finite-State Language Processing*. Language, Speech and Communication. MIT Press, Cambridge, MA, 1997.

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

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

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

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

Shuly Wintner and Shlomo Yona. Resources for processing Hebrew. In *Proceedings of the MT-Summit IX workshop on Machine Translation for Semitic Languages*, pages 53–60, New Orleans, September 2003.

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

Martin Wynne, editor. *Developing Linguistic Corpora: a Guide to Good Practice*. Oxbow Books, Oxford, 2005. Available online from `http://ahds.ac.uk/linguistic-corpora/`.

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