Generating Determiners and Quantifiers in Hebrew

Yael Dahan Netzer and Michael Elhadad
Ben Gurion University
Department of Mathematics and Computer Science, Beer Sheva, 84105, Israel
(yaeln|elhadad)@cs.bgu.ac.il

Abstract

This paper presents the part of HUGG, a generation grammar for Hebrew, that deals with determiners and quantifiers. Our main goal is to determine which set of features must be present in the input to the generation grammar to control the generation of complex determiners and quantifiers.

Hebrew determiners are characterized by two properties: (1) definiteness is marked in several places in the NP and it interacts with compound constructs; (2) the order of appearance of determiners and quantifiers within a complex NP is flexible but still restricted.

HUGG is developed with the goal to design an input specification language for syntactic realization as close as possible for English and Hebrew, to allow easy development of bilingual generation applications. We show in this paper how the original set of features controlling the generation of determiners in SURGE, a large generation grammar for English, has been enriched to account for the specificities of Hebrew determiners and how, in the process, SURGE has been modified and improved.

We present a set of functional features – organized around the categories of amount, partitive and identity – and show how these features determine a variety of syntactic constructs.

1 Introduction

A text generation system generally includes two main components: content determination (decide what to say) and surface realization (decide how to say it). Recent research has led to the development of reusable surface realization systems which encapsulate knowledge of the syntax for specific languages. Most of this research has been conducted for English, and large generation grammars for English are now available [4], [9].

Multi-lingual generation now appears as the "next frontier" in generation research: it is often easier to develop applications to generate text in different languages than it is to translate text. When developing a multi-lingual application, the engineering strategy is to develop a single content determination module and to plug it – more or less unchanged – into several surface realization modules for different languages – each encapsulating knowledge of a different language.

The main problem to address in this context is to determine which interface is appropriate between the content determination module and the linguistic modules. We are investigating this problem for the case of bilingual generation English-Hebrew. We start from an existing surface realization system for English – SURGE and have developed a generation grammar for Hebrew called HUGG with the goal that the input specification language for both grammars be as close as possible.

This paper presents the part of HUGG that deals with determiners and quantifiers. Our main goal is to determine which set of features must be present in the input to the generation grammar to control the generation of complex determiners and quantifiers. Because the syntactic structure of English and Hebrew determiners is very different, we use functional-semantic features in the input. We present a set of such features and evaluate (1) how the features can account for specific syntactic constructs in Hebrew and (2) how the same features can be used in English.

Hebrew determiners are characterized by two properties: (1) definiteness is marked in several places in the NP and it interacts with compound constructs; (2) the order of appearance of determiners and quantifiers within a complex NP is flexible but still restricted. We focus on these two issues in the rest of the paper.

1.1 Definiteness

As a polydefinite language, Hebrew marks definiteness in several places in the NP, unlike the definite/indefinite articles of English. Because of the complex nature of definiteness marking in Hebrew, we have found that the marking of definiteness as realized in SURGE does not suffice.

Non-Definite	**Definite**	**Definite with smixut**
(d1) *sefer meyuHad*	(d2) *ha-sefer ha-meyuHad*	(d3) *sefer ha-madaw ha-meyuHad*
book special	*the-book the-special*	*book the-science the-special*
a special book	*the special book*	*the special science book*

Examples d1/d2 show how the definiteness marking (ha-) appears on both the head and the modifier. Example d3 shows that when a noun-compound construction (called smixut) is used, then the definiteness marking disappears on the head. The head, however, remains semantically definite. Hebrew demonstratives can also be marked with the definiteness ha-:

Unmarked	*sfarym Eleh*	*books those*
Marked	*ha-sfarym ha-Eleh*	*the-books the-those*

The two forms are definite in their meaning (demonstrative implies definite), but the additional definite marking is still possible in Hebrew. The two forms differ in their pragmatic usage (the unmarked form tends to refer to previous text and the marked form to deixis proper [11] p.117)

The distinction between semantic definiteness and marked definiteness is reflected as well for the indefinite, as shown in the following examples:

Non-Definite - unmarked	**Non-Definite - marked**	**Non-definite selective**
sefer	*Eyzeh sefer*	*sefer EHad*
book	*which book*	*book one*
a book	*a book (undetermined)*	*a book, some book*

From these examples, it appears that a single binary feature `definite`, as used in SURGE, cannot account for the syntactic complexity of the definiteness. We had, therefore, to extend the input specification for definiteness.

1.2 Order of constituents

Unlike the "determiner sequence" defined in [8], [12], Hebrew determiners and quantifiers can be positioned before and after the head of the NP[1]:

[1] Determiners in English can occur after the head in the very restricted cases where a pronoun is modified by a "total" meaning – *e.g., we both, them all.*

(o1) *col ha-yeladym*	(o3) *harbe yeladym*	(o5) ** harbe ha-yeladym*
all the-children	*many children*	** many the-children*
All the children	*Many children*	

(o2) *hayeladym culam*	(o4) *yeladym rabym*	(o6) *ha-yeladym ha-rabym*
the-children all-them	*children numerous*	*the-children the-numerous*
All the children	*Many children*	*The many children*

Determiners and quantifiers positioned after the head behave more like adjectives: they agree in number, gender and definiteness with the head. Determiners with the same meaning can appear before or after the head, as shown in examples o1/o2 and o3/o4 above. In some cases, only one of the two options is available, as shown in examples o5/o6: because the determiner for *many* appearing before the head is lexically marked as indefinite, it cannot be used for the NP *the many children*.

In addition to the choice before/after head, when several determiners and quantifiers co-occur, they must be ordered in the right sequence. The distinction between determiners (D) and quantifiers (Q) has been proposed [3] to explain, among other facts, the order generally used:

col me'ot ha-mafginym	** me'ot col ha-mafginym*
all hundreds the-demonstrators	** hundreds all the-demonstrators*
All of the hundreds of demonstrators	** Hundreds of all of the demonstrators*

"Determiners" (like *col* / *all*) generally appear before "quantifiers" (like *me'ot* / *hundreds*). By this definition, determiners are those words which, among other characteristics, can enter in recursive constructs. In this case, there are no restrictions on the order of occurrence of the determiners:

col S'ar ha-mafginym	*S'ar col ha-mafginym*
all the-rest-of the-demonstrators	*the-rest-of all the-demonstrators*
All the rest of the demonstrators	*The rest of all the demonstrators*

The different orders, naturally, have different meanings.

There are, however, cases where the order between determiners can lead to syntactically unacceptable NPs:

col ot-am ha-yeladym	** ot-am col ha-yeladym*
all those-them the-children	** those-them all children*
all those children	** those all children*

The structural distinction Determiner/Quantifier proposed in [3] does not help to explain this restriction. The problem with the last example is related to the functional incompatibility of the two determiners. We present a functional account of these restrictions, based on a set of features derived from those used in SURGE.

In the rest of the paper, we first present previous work in the description of Hebrew determiners, and present the treatment of the determiner sequence in English for generation as implemented in SURGE. We then focus on the two issues illustrated above – definiteness and order of constituents – and present our treatment of the Hebrew determiner construction using a set of functional features.

We derive from this analysis an input specification for noun phrases for a generation grammar. This input specification extends the set of features used in SURGE and remains compatible with the English grammar, while providing a wide coverage of the Hebrew constructions.

2 Previous Work

Descriptive studies of the syntax of the Hebrew noun phrase do not generally distinguish between different functions of determiners but focus rather on the syntactic structure of the NP.

Glinert [7] adopts a functional perspective, more appropriate to the needs of a generation system, and identifies a general pattern of the NP that we use as a basis:

`[partitive determiner amount head classifiers describers post-det/quant qualifiers]`

In this pattern, the slots can be filled by any word that fits the function: partitive expressions identify a subset of the set referred to by the rest of the NP; amount expressions identify the quantity or number, exact or inexact, of the thing referred to; determiners identify the thing referred to more or less precisely or specifically. The general picture of the input specification is that of a reference set from which a subset is identified by different complementary means. The function of each word realizing this input specification determines where it appears in the syntactic structure.

Yizhar and Doron [3] [13] distinguish between two sets of determiners, that they call D and Q quantifiers. The distinction is based on syntactic features, such as location, ability to be modified, ability to participate in partitive structures, requirement to agree in number and gender with the head. This distinction is used to explain cooccurrence restrictions, the order of appearance of D vs Q quantifiers and the recursive structure of D determiners: D determiners can be layered on top of other D determiners. A single Q quantifier can occur in an NP and it remains attached closest to the head.

As discussed in the introduction, the distinction D/Q is not sufficient to explain some restrictions on the cooccurrence of several Ds in the same NP, and does not help to predict the restrictions on occurrence of determiners in different definiteness and partitive contexts.

In [1] we have, therefore, refined the D/Q classification using functional criteria: we map the Q quantifiers to the "amount" category defined by Glinert, and the D set is split into the partitive and determiner categories – each with a different function. Of these, only partitive are recursive.

Another controversial issue within Hebrew studies is whether determiners are types of nouns [2] [6]. This is an important issue to explain in a uniform manner why constructs like smixut (noun compounding) are used for certain determiners and why certain determiners determine the number and gender of the whole NP. We have implemented some of these constraints by defining lexical features for determiners which determine this behavior.

2.1 Determiner Sequence in Generation

Our goal is to define a set of functional features to include in the input to a generator. These features must provide enough information to construct a wide variety of determiner constructions both in Hebrew and in English. For the comparison with English, we based our work on the SURGE generation grammar [5].

The determiner sequence in SURGE is based on [12] p.513 and [8] and has the following pattern:

`[pre-det (of) det deictic2 ordinal cardinal quantifier (describer classifier head)]`.

Where *pre-det* can be any one of *all, both, half,* multipliers and fractions, *det* is a deictic determiner, *deictic2* is an adjective from a restricted class that expresses the anaphoric status of the thing referred to (*e.g., above, same, different*); and *quantifier* expresses the amount or quantity of the thing referred to.

In [4], we gathered a minimal set of 24 features to define the determiner sequence of the NP in English. Complex systems in SURGE implement the cooccurence restrictions among these features, definiteness and partitive constructs.

Feature	Example
Definite yes/no	*The/a book*
Distance far/near/none	*This/That/The book*
Selective yes/no	*Some/∅ children*
Total +/-/none	*All/None/∅ of the children*
Cardinal	*The three children*
Fraction	*One-third of the children*
Multiplier	*Twice his weight*

Feature	Example
degree +	*The many cars*
degree -	*A little butter*
degree none	*Some butter*
comparative yes	*More cars*
superlative yes	*The most cars*
evaluative yes	*Too many cars*
orientation -	Few cars

When studying the Hebrew determiner system, we found that this set of features had to be extended and refined. As a consequence, both the implementations of SURGE and of HUGG have been made compatible with the same set of features, which we motivate in the rest of the paper.

3 Definiteness

Definiteness in NP has two aspects: it corresponds to a complex semantic property of the thing referred to by the phrase and it is syntactically marked in different ways. We will not discuss how a noun phrase is defined as semantically definite (a development on this issue is provided in [1] Chap.5), and we assume that the decision that a referent is definite is taken by the content determination module and that the semantic definite feature is given in the input to the generation grammar. We keep the feature `definite yes` for this purpose.

In a Hebrew definite NP, the definite marker (ha-) is agglutinated to the head and to most modifiers (adjectives and quantifiers). Not all subconstituents of the NP are marked, however: in a noun-compound construction the head loses the marking and the compounded noun keeps it. When demonstratives are used, a marked and an unmarked form can be used. In addition, pronouns and proper nouns are intrinsically definite and do not receive the ha- marking.

To explain the distribution of the ha- marker, we have added the feature `mark-definite` which determines whether an article (definite or indefinite) is to be used for each subconstituent of the NP.

By default, the mark-definite of a subconstituent is equal to the definite of its mother constituent. However, certain constructions and certain lexical items block this propagation and can force a `mark-definite no` (for example, proper nouns have a lexical feature of `mark-definite no` and the noun-compounding construction enforces a mark-definite no on the head).

The following example illustrates how the feature mark-definite contrasts among the two distinct phrases:

Mark-definite no: (anaphoric)
nawar cfar Tov lev ze
boy village good heart this
This good-hearted country boy

Mark-definite yes: (deictic)
nawar ha-cfar Tov ha-lev ha-ze
boy the-village good the-heart the-this

$$\left[\begin{array}{ll} \text{cat} & \text{common} \\ \text{lex} & \text{“nawar”} \\ \text{definite} & \text{yes} \\ \text{distance} & \text{near} \\ \text{classifier} & \left[\begin{array}{ll} \text{lex} & \text{“cfar”} \end{array}\right] \\ \text{describer} & \left[\begin{array}{ll} \text{lex} & \text{“Tov”} \\ \text{classifier} & \left[\begin{array}{ll} \text{cat} & \text{common} \\ \text{lex} & \text{“lev”} \end{array}\right] \end{array}\right] \\ \text{mark-definite} & \text{yes } \textbf{no} \end{array}\right]$$

In this example, *nawar/boy* is not marked with ha- because it enters in a compound construction with *cfar*. Similarly *Tov/good* is not marked because it is an adjective compound of *lev/heart*.

While there is agreement as to the behavior of the marking of definiteness, lqThe marking of indefinite nouns is quite controversial. Some researchers claim that definite is not a binary but that an additional unmarked value is necessary (cf [2] for a review). We mainly adopt Ornan's view [10].

In this view, an indefinite NP is marked with a null article by default, but the indefinite can also be marked. In this case, we use the feature `(mark-definite indef)`. This feature explains the contrast between:

eyze delet	*some door*	`mark-definite indef`
delet	*door*	`mark-definite no`
delet aHat	*door one*	`selective yes`

We treat *delet aHat* as marked functionally as a selective phrase, following Halliday's feature as used in SURGE. In our analysis, `selective yes` implies `mark-definite indef`. Of the two marked indefinite articles, *eyze* is non selective while *eHad* is.

The following example, taken from an example by Winograd, illustrates the behavior of the `selective yes` feature:
Examining the cabinet, we noticed that a door was marred
The Hebrew for it will be:
? cSe-badaknu Et ha-Aron, rainu S-delet Svura
while-checked-us OM the-closet, saw-us that-door broken

cSebadaknu Et ha-Aron, rainu S-delet AHat *Svura*
while-checked-us OM the-closet, saw-us that-door one *broken*

In this context, the existence of a set of 2 or 4 doors is known pragmatically (world knowledge about closets). The function of the indefinite article is then to select one door out of this known set. In that case, the unmarked indefinite is quite inappropriate, because it would refer to a door that cannot be related to the closet.

In summary, the complex marking of definiteness in Hebrew has led us to extend the single semantic feature `definite yes/no` used in SURGE, with the additional feature `mark-definite yes/no/indef`.

4 Determining the Order

We have mostly investigated the order of determiners and quantifiers in Hebrew that appear before the head of the NP. We have found that semantics plays a main role in determining the relative order of determiners. This follows Glinert ([7] Chap.8 and 9), who distinguishes among the different functions fulfilled by determiners, and predicts the ordering accordingly, as discussed in Sect.2.

The primary functional distinction is between *quantifiers* – that express amount or portion – and *determiners* – that express identity. Accordingly, the word *col/all* for example, has two senses: one as a quantifier (with the *all* or collective meaning) and one as a determiner (with the *every* or distributive meaning).

Quantifiers are classified along two main dimensions: exact/inexact (*e.g., three vs. many*) and portion/amount. The portion/amount system has the most effect on the structure of the NP, as it can be realized either lexically by certain quantifiers, or by using an explicit partitive syntactic structure *X m-Y/X of Y*.

We define amount quantifiers (cardinals, quantifiers like *harbeh/many*, *yoter/more*) as quantifiers that express quantity without explicit relation to the "total" cardinality of the reference set[2]

[2]Although amount quantifiers are always semantically related to some reference set, this relation is presented

In contrast, partitive quantifiers express a quantity that gets its meaning only in relation to the cardinality of a reference set. For example, *rov/most* refers to at least half of the reference set. Similarly, *col/all* refers to all of the reference set, and is also partitive.

The distinction among partitive quantifiers, amount quantifiers and determiners is important as it predicts the order of the words in the NP. The standard order is:
`[partitive determiner amount head]`

We also note that only partitives can enter into recursive structures.

Accordingly, our input specification language enforces the constraint that only a single amount and a single identification feature can be present simultaneously. The realization grammar also uses the knowledge of which word realizes which function to determine the ordering. This is illustrated in the following example, where the standard order is obtained for the combination *partitive amount head*:

col waseret ha-mafginym
all ten the-demonstrators
All the ten demonstrators

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{total} & + \\ \text{ref-set} & \begin{bmatrix} \text{lex} & \text{“mafgyn”} \\ \text{definite} & \text{yes} \\ \text{cardinal} & \begin{bmatrix} \text{value} & 10 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that whenever a partitive quantifier is desired, the input specification must include a ref-set construct. This enforces the constraint that partitives yield recursive constructs, as shown in the following example:

waseret me-col ha-mafginym
ten of-all the-demonstrators
Ten of all the demonstrators

$$\begin{bmatrix} \text{cat} & \text{np} \\ \text{cardinal} & \begin{bmatrix} \text{value} & 10 \end{bmatrix} \\ \text{ref-set} & \begin{bmatrix} \text{total} & + \\ \text{ref-set} & \begin{bmatrix} \text{lex} & \text{“mafgyn”} \\ \text{definite} & \text{yes} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The decision to build an explicitly partitive construction *X m-Y/X of Y* is left to the realization grammar and can be quite complex. If a quantifier can be found that has the lexical property of being marked as partitive, then a non-explicitly partitive construction can be used, as in the first example above *all the ten demonstrators*. Note that in this case, in contrast to English, in Hebrew, the explicitly partitive construction is **not** possible ** col me-waseret ha-mafginym*. In contrast, if a feature modifying the ref-set is to realized by a non-lexically partitive quantifier, then an explicitly partitive construction must be used.

When the "portion" part is modified with adjectives, then an explicitly partitive construction must be used:
ha-rov ha-gadol mi-beyn ha-yeladym
the-most the-big of-from the-children
The vast majority of the children

In that case, the portion functions as the logical head of the NP with the realization of the reference set as a PP – which affects agreement with the verb.

In summary, we propose a refined classification of the determiners/quantifiers in Hebrew into amount, partitive and determiner. This functional classification determines the relative order of the determiners within the NP and also determines which recursive structures can be constructed. Finally, it is important to take into account the lexical properties of specific determiners which can require or forbid the use of explicitly partitive constructions. Our input specification language encompasses all of the features found in SURGE but organizes them into a new framework which enforces these constraints (with the explicit specification of the ref-set in the input).

implicitly for amounts: *3 children* refers to 3 children out of some reference set, but the reference set is deliberately left implicit.

5 Conclusion

We have presented in this paper observations on the syntax of the Hebrew NP and deduced from its properties an input specification language for complex NPs to a generation grammar.

The input specification language extends the one used in SURGE for English to provide enough information to enforce the constraints observed in Hebrew. We have specifically extended the treatment of definiteness (and indefiniteness) with the additional feature mark-definite and the treatment of partitives with a systematic classification of quantifiers as partitives, amounts and determiners.

The resulting formalism provides a good basis for a bilingual generation grammar, and is being used to that end in practical projects.

References

[1] Yael Dahan-Netzer. Hugg - unification-based grammar for the generation of hebrew noun phrases. Master's thesis, Ben Gurion University, Beer Sheva Israel, 1997. (In Hebrew).

[2] G. Danon. The syntax of determiners in hebrew. Master's thesis, Tel Aviv University, May 1996.

[3] Edit Doron. The NP structure. In E. Doron U. Ornan and A. Ariely, editors, *Hebrew Computational Linguistics*. Ministery of Science, 1991. In Hebrew.

[4] Michael Elhadad. *Using argumentation to control lexical choice: a unification-based implementation*. PhD thesis, Computer Science Department, Columbia University, 1992.

[5] Michael Elhadad. Lexical choice for complex noun phrases: Structure, modifiers and determiners. *Machine Translation*, 11:159–184, 1996.

[6] L. Glinert. Shadow noun phrases in spoken hebrew. *Hebrew Computational Linguistics*, (13), 1978. In Hebrew.

[7] L. Glinert. *The Grammar of Modern Hebrew*. Cambridge University, 1989.

[8] Michael A. K. Halliday. *An Introduction to Functional Grammar*. Edward Arnold, London, second edition, 1994.

[9] William C. Mann. An overview of the Penman text generation system. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 261–265, Washington, DC, August 22-26, 1983. Also appears as USC/Information Sciences Institute Tech Report RR-83-114.

[10] Uzi Ornan. *The Nominal Phrase in Modern Hebrew*. PhD thesis, Hebrew University, Jerusalem, 1964. (in Hebrew).

[11] H. Rosen. *Contemporary Hebrew*. Mouton, 1977.

[12] Terry Winograd. *Language as a Cognitive Process: Syntax*, volume I. Addison-Wesley, Reading, MA, 1983.

[13] D. Yzhar. Computational grammar for noun phrases in hebrew. Master's thesis, Hebrew University, Jerusalem, 1993. In Hebrew.