Paradigmatic Treatment of Arabic Morphology

Martine Smets
Cognitive and Computing Sciences
University of Sussex, Brighton BN1 9HQ, UK
martines@cogs.susx.ac.uk

Abstract

This paper[1] presents a language to express morphological processes, concatenative or non-concatenative. The language allows the definition of two kinds of paradigms: helping paradigms, which define partial morphological forms, and main paradigms, which combine two or more helping paradigms to define fully specified forms.

1 Introduction

This paper presents a language to express morphological processes and shows how that language applies straightforwardly to Arabic morphology[2]. This approach to morphology is directly compatible with unification-based frameworks in that lexical entries are feature structures, and morphological processes are related to unification of feature structures.

This work departs from two-level morphology, which has been at the center of computational morphology since the implementation of URKIMMO (Koskenniemi 1983) and was applied to Arabic morphology first by Kay (1987), then by Beesley (1989, 1996), Kiraz (1994, 1996a) and to Syriac morphology by Kiraz (1996b). Two-level morphology is characterized by an emphasis on phonological (or orthographic) rules, and has a rudimentary treatment of morphology itself.

The language presented in this paper proposes a declarative approach to morphology[3] in which morphological processes are represented as paradigms of relations between feature structures and orthographic/phonological forms. This approach is inspired from work done by Calder (1989) who uses paradigms to organize morphological information, and string equations to handle string operations.

2 Organization of the Lexicon

The language described here supports an organization of the lexicon into three components: feature component, morphological component and lexical entries.

- The feature component is a hierarchy of types related by monotonic multiple inheritance.
- The morphological component is a set of paradigms which relate feature structures of the feature component to inflected forms.
- The lexicon of "stems" is a collection of roots which are related to inflected forms by the morphological component. As is customary in accounts of Arabic, roots are ordered lists of consonants. They are associated with relevant idiosyncratic information (mainly semantic, for example the particular meaning associated with each root).

3 Description of the Language

3.1 The Feature Component

The feature component consists of a hierarchy of grammatical objects constrained by relevant features, similar to the type hierarchy of Head-driven Phrase Structure Grammar (Pollard and Sag 1994). This allows to capture generalities common to classes of words and avoid redundancies in the lexicon of roots. For example, verbs are constrained to be of category *verb*,

[1]The research was supported by a grant from the Research Board of the University of Illinois at Urbana-Champaign.

[2]I would like to thank two anonymous reviewers for their helpful comments.

[3]Only inflectional morphology is discussed in this paper; but the approach can be extended to derivational morphology.

$$(2)\ \left[\begin{array}{l} \textit{H-P mel-perf}^{5} \\ \left[\text{PHON } [(\text{C},\text{V}_1{}^*)^*,\text{V}_2,\text{C}^*]\right] \ \& \left[\begin{array}{l} \textit{perf active}\text{: } \text{V}_1 = \text{a V}_2 = \text{a;} \\ \textit{perf passive} \sqcap \text{not}(\textit{IX})\text{: } \text{V}_1 = \text{u V}_2 = \text{i;} \\ \textit{perf passive} \sqcap \textit{IX}\text{: } \bot \end{array}\right] \end{array}\right]$$

and to belong to one of ten forms or binyanim.[4]

The first binyan carries the meaning of the root and is unmarked morphologically, while other binyanim bring modification to the meaning and to the pattern of vowels and consonants (though the modification in meaning is not systematic and not always predictable: see Mc Carthy 1981).

Verbs are also characterized by different vowel melodies which vary in function of aspect and voice: a verb can be in the perfective, imperfective or participle aspect, and in the active or passive voice.

Additionally, verbs carry affixes which vary in function of person, number and gender. All such information which is predictable is organized in the feature component.
Thus, for example, the class *verb* is organized along two dimensions, *template* and *melody*; the class *template* has ten subclasses, one for each of the binyanim; the class *melody* is organized along two dimensions, aspect and voice, which have themselves subclasses. Classes of the hierarchy define appropriate constraints in a monotonic fashion.

Lexical items inherit predictable information either directly (appartenance to a specific class is stated in the lexical entry) or through the morphological component (the lexical entry specifies to which paradigm a lexeme belongs).

3.2 Morphological Component

The morphological component is a set of paradigms[6]. There are two kinds of paradigms: *main paradigms*, which specify fully inflected forms, and *helping paradigms*, which introduce partially specified forms. The latter introduce information specific to particular morphological relations, and this information is used by main paradigms to define fully inflected word forms.

The fact that helping paradigms can express information independently of a particular word form allows that information to be inherited in distinct environments, and to combine with information from other helping paradigms. Helping paradigms can then be seen as capturing generalizations about morphological relations, and main paradigms as expressing specific morphological forms.

As for the interface between syntax and morphology, each entry of a helping paradigm associates relevant features to the appropriate phonological realization through references to types of the sort hierarchy: the name or label of each entry of a helping paradigms directly refers to class(es) in the type hierarchy.

3.2.1 Helping Paradigms

Helping paradigms are sets of pairs $\langle F, S\rangle$, where S is a string specification and F a feature structure associated with the string specification. The string specification is a partial description of string which may contain variables (Calder 1989). The feature structure is inherited from types in the feature component. Thus, each helping paradigm is a subset of the Cartesian product of the set of feature structures consistent with the type hierarchy, and the set of strings of the language.

Syntax of Helping Paradigms. Helping paradigms have the concrete syntax in (1):

```
    H-P Name
(1) PHON String of characters &
    {Type labels: variable instantiation}
```

[4]Mc Carthy (1981) distinguishes 15 forms, but for the purposes of this paper, I will consider only the first ten ones, which are the productive forms.

[5]Actually, there are three different melodies for the perfective aspect of verbs of the first form (*a a*, *a i* and *a u*), but only one of the melodies is given here, as an illustration of helping paradigms.

[6]The notion of paradigm defined here is closely related to the traditional paradigm: paradigms are sets of words related to the leading form which is the conventional representation of the lexeme; words are considered as wholes, and affixes do not have any meaning independently of a root (Matthews 1991).

$$
(3)\quad \begin{bmatrix} \textit{H-P mel-imperf} & \\ [\mathrm{PHON}\ [V_1, (C,V_2{}^*)^*, V_3,C]]\ \& & \begin{bmatrix} \textit{imperf active} \sqcap \textit{I}\text{: } V_1 = a\ V_2 = nil\ V_3 = i; \\ \textit{imperf active} \sqcap \{\textit{II,III,IV}\}\text{: } V_1 = u\ V_2 = a\ V_3 = i; \\ \textit{imperf active} \sqcap \{\textit{V, VI}\}\text{: } V_1 = a\ V_2 = a\ V_3 = a; \\ \textit{imperf active} \sqcap \{\textit{VII, VIII,IX,X}\}\text{: } V_1 = a\ V_2 = a\ V_3 = i; \\ \textit{imperf passive} \sqcap \textit{I}\text{: } V_1 = u\ V_2 = nil\ V_3 = a; \\ \textit{imperf passive} \sqcap \textit{II,III ... VIII,X}\text{: } V_1 = u\ V_2 = a\ V_3 = a; \\ \textit{imperf passive} \sqcap \textit{IX}\text{: } \perp \end{bmatrix} \end{bmatrix}
$$

The name of helping paradigms is followed by the phonology,[7] a partial description of a string.[8] The curly brackets denote a set of paradigm entries, and *Type Labels* refers to types in the feature component.

Examples of Helping Paradigms. Examples of helping paradigms are given in (2) and (3): these are two of the melody paradigms. As said earlier, aspect and voice in Arabic are expressed through the choice of a particular melody: *jalas* is the perfective active of the verb *to sit down/*, while *ajlis* is the imperfective active of the same verb. The root consonants are similar, but the melody differentiates perfective from imperfective. This is expressed by defining melody paradigms which specify which melody is associated with which features; the melody is intertwined with an abstract root in the paradigm.

Capital letters in the phonology stand for variables: *C* is any consonant, *V* any vowel;[9] lower case letters are constants.

The symbol $\sqcap$ means unification, while the symbol & following the phonology means that the phonology feature is distributively associated with each of the entries (there are as many specializations of the phonology as there are entries).

The value *nil* means that this position in the string is empty; during unification, a variable unifying with *nil* is removed from the pattern.

[7] *Phonology* means the concrete representation of a string, the orthography or the phonology.

[8] In the helping paradigms which define vowel melody, the phonologies are actually regular expressions (where * means as usual zero or more occurences of a character).

[9] If two variables of the same type bear the same index, it means that they stand for the same constant.

Finally, a paradigm entry specified as $\perp$ is undefined.

Variables of the phonology are instantiated in the paradigm entries, according to syntactic features: the first entry of the paradigm called *mel-imperf* specifies the value of the vowels in the pattern $[V_1,(C,V_2{}^*)^*,V_3,C]$, for the imperfective active of verbs belonging to the first form.

The label preceding the colon, *active* $\sqcap$ *I*, refers to names of classes in the feature component, so that the constraints defined in these classes are inherited by the partial form specified by this paradigm entry.

Sometimes, the label contains a disjunction: for example in the paradigm *mel-imperf*, the label of the second entry is *imperf active* $\sqcap$ $\{II,III,IV\}$; the disjunction $\{II,III,IV\}$ means that the melody characterizing this paradigm entry is associated with any of these derived forms (II, III or IV) in the imperfective active.

The helping paradigms in (2) and (3) only partially specify the phonology and features of Arabic verbs: the melody has to be correctly intertwined with a particular pattern of consonants[10]. The pattern for verbs in the perfective aspect is defined in helping paradigm (4).

The template of triliteral verbs in the perfective aspect is given in the *PHON* attribute of the paradigm *perf-temp*: $[P,C_1,X,C_2,Y,C_3]$. The value of the consonants C_1, C_2 and C_3 is determined by particular lexical entries, while entries of the paradigm specify what kind of ma-

[10] Verbs are also inflected for person, number and gender. This can be readily expressed by defining more helping paradigms.

$$(4)\ \left[\begin{array}{ll} \textit{H-P perf-temp} & \\ \left[\text{PHON } [\text{P}, \text{C}_1, \text{X}, \text{C}_2, \text{Y}, \text{C}_3]\right] \ \& & \left[\begin{array}{l} \textit{I}\text{: P = nil X = } \text{V}_1 \text{ Y = } \text{V}_2; \\ \textit{II}\text{: P = nil X = } [\text{V}_1,\text{C}_2] \text{ Y = } \text{V}_2; \\ \textit{III}\text{: P = nil X = } [\text{V}_1,\text{V}_1] \text{ Y = } \text{V}_2; \\ \textit{IV}\text{: P = } [\text{h},\text{V}_1] \text{ X = nil Y = } \text{V}_2; \\ \textit{V}\text{: P = } [\text{t},\text{V}_1] \text{ X = } [\text{V}_1,\text{C}_2] \text{ Y = } \text{V}_2; \\ \textit{VI}\text{: P = } [\text{t},\text{V}_1] \text{ X = } [\text{V}_1,\text{V}_1] \text{ Y = } \text{V}_2; \\ \textit{VII}\text{: P = n X = } \text{V}_1 \text{ Y = } \text{V}_2; \\ \textit{VIII}\text{: P = nil X = } [\text{t},\text{V}_1] \text{ Y = } \text{V}_2; \\ \textit{IX}\text{: P = nil X = nil Y = } [\text{V},_1,\text{C}_3,\text{V}_2]; \\ \textit{IV}\text{: P = } [\text{s},\text{t},\text{V}_1] \text{ X = nil Y = } \text{V}_2; \end{array}\right] \end{array}\right]$$

terial comes in between those root consonants: for form *V*, the prefix is made of the consonant *t* and a vowel (whose value is determined by a melody paradigm); after the first consonant, there is a vowel (the same as the one of the prefix), and the second consonant is duplicated; finally, the third consonant is preceded by a vowel (which can be different from the first one). Each entry partially specifies the general pattern, and associates the resulting string with a label corresponding to a class of the feature component.

Again, the resulting entries are partial descriptions of strings associated with partial syntactic/semantic characterisations, and these descriptions of strings have to be combined with other partial descriptions of strings in order to yield fully specified verbs. This is the task of main paradigms. Before turning to main paradigms, a last characteristic of helping paradigms needs to be presented.

Inheritance. Helping paradigms can be related by monotonic inheritance. Similarities between paradigms can thereby be captured, and the encoding be more concise. For example, the template of verbs in the imperfective differs only slightly from the template of verbs in the perfective: a vowel is prefixed to the latter. This can be expressed very concisely with inheritance (5). The paradigm *imperf-temp* inherits all the information specified in the entries of the paradigm *perf-temp*.

$$(5)\ \left[\begin{array}{l} \textit{H-P imperf-temp: H-P perf-temp} \\ \left[\text{PHON } [\text{V}, \text{P}, \text{C}_1, \text{X}, \text{C}_2, \text{Y}, \text{C}_3]\right] \end{array}\right]$$

Inheritance also provides an elegant way of expressing subregularities: for example, if a class of verbs is almost completely regular, except for a few entries, the entries which are common to the regular and the irregular paradigms are expressed in one paradigm; both the regular and irregular paradigms inherit these entries, and specify only entries which are not common to both paradigms.

3.2.2 Main Paradigms

Main paradigms relate lexemes (lexical entries, so in the case of Arabic, consonantal roots) to fully specified forms, through reference to helping paradigms: which helping paradigms are combined together is defined by each main paradigm.

The operation which combines helping paradigms together is an extension of Cartesian product, denoted by the symbol *(X)*.

Helping paradigms are sets of pairs $\langle PHON, FS \rangle$ (pairs of phonology and feature structure); when two helping paradigms are combined together, each pair of the first paradigm is unified with each pair of the second helping paradigm: the PHON attributes are combined together, and the feature structures are combined together, resulting in a new pair $\langle PHON, FS \rangle$ if unification succeeds. If the values of some feature are not compatible, unification fails, and that particular pair is not returned[11].

[11]The way paradigm entries combine is reminiscent of Kay (1984) who mentions unification of patterns (which constrain the order in which syntactic constituents must appear) alongside unification of other types of features.

M-P verb-3 $([C_1,C_2,C_3],\mathrm{FS})$:

(7) FS ⊓ [*H-P mel-perf* (x) *H-P perf-temp*(PHON $[P,C_1,X,C_2,Y,C_3]$)];
FS ⊓ [*H-P mel-imperf* (x) *H-P imperf-temp* (PHON $[V,P,C_1,X,C_2,Y,C_3]$)];

Syntax of Main Paradigms. Main paradigms have the syntax in (6):

(6) M-P Name(Phon,FS): FS ⊓ {(H-P_1 (x) H-P_2 (x) ...), (H-P_3 (x) H-P_4 (x)...), ...}

In (6), *Name* is the name of the paradigm, and its argument is a lexical entry, which has two components, a phonology and a feature structure. The feature structure of the lexical entry is distributively unified with the elements resulting from the combination of the helping paradigms.
Thus, if an entry of a main paradigm refers to two helping paradigms, H-P_1 and H-P_2, each of the entries of H-P_1 is unified with each of the entries of H-P_2:

H-P_1 = $\{(F_1, P_1), (F_2, P_2), ..., (F_k, P_k)\}$.
H-P_2 = $\{(F'_1, P'_1), (F'_2, P'_2), ..., (F'_k, P'_k)\}$.
H-P_1 (x) H-P_2 = $\{(F_1, P_1), ..., (F_k, P_k)\}$ (x) $\{(F'_1, P'_1), ..., (F'_j, P'_j)\}$ =
$\{(F_1 \sqcap F'_1, P_1 \sqcap P'_1), ..., (F_k \sqcap F'_j, P_k \sqcap P'_j)\}$

The result is a new paradigm, a set of pairs $\langle PHON, FS\rangle$, but this time, the phonology does not contain any more variables, and the feature structure of the lexical entry is unified with the second member of each pair.

To illustrate, the main paradigm in (7) combines the helping paradigms defined in (2), (3), (4) and (5). The first entry of (7) combines the *perf-temp* helping paradigm with the *mel-perf* helping paradigm so that each entry of the first helping paradigm tries to unify with each entry of the second paradigm. For example, the third entry of the *perf-temp* paradigm unifies with the first entry of the *mel-perf* paradigm:

PHON $[[C_1,V_1,V_1,C_2,V_2,C_3] \sqcap [(C,a^*)^*,a,C^*]]$
& [III ⊓ *perf active* ⊓ {*II,III,IV*}]

The *PHON* values are unified by *string unification* (Calder 1989), the feature structures by standard unification of features. The form III unifies with the disjunction {*II,III,IV*} because *III* belongs to the disjunction. The result of unification is given in (8),with the class names expanded in feature structures (in HPSG style).

$$(8)\quad \begin{bmatrix} \mathrm{PHON} & [C_1,a,a,C_2,a,C_3] \\ \mathrm{HEAD} & \begin{bmatrix} \textit{verb} & \\ \mathrm{ASPECT} & \mathrm{imperfect} \\ \mathrm{VOICE} & \mathrm{active} \\ \mathrm{FORM} & \mathrm{III} \end{bmatrix} \\ \mathrm{CONT} & \ldots \end{bmatrix}$$

This is not a complete characterization of an Arabic verb of the third form in the imperfective active; it is just an illustration of how partial descriptions get combined to yield complete descriptions.

String Unification. String unification is the unification of two partial descriptions of strings which represent the same object. As shown by Siekmann (Calder 1989), string unification is decidable and has a finite number of substitutions if repeated variables are only permitted on one side of the equation. An example of string equation with the variable assignments resulting from string unification is given in (9).

```
(9) jVVlWs = RaaSaT
    R/j V/a S/l W/a T/s
```

String unification is a powerful tool which allows other kinds of morphological relations to be specified, besides concatenation: words are defined as lists of elements, which can be partial strings or individual characters. Some elements of the list are variables, which can be located in any position of the list representing the word, and get specified through unification with a complementary list. The ability to isolate particular positions in the word, together with the ability to define in helping paradigms specific substrings, allows the description of any type of morphological relation.

3.3 The Lexicon of Roots

Lexical entries are tuples $\langle P, F, M\rangle$ where P is the phonology (a consonantal root, as is customary in Arabic dictionaries), F the feature structure and M, the morphological information. The morphological information is the name of the main paradigm(s) to which the entry belongs.

The feature inventory of lexical entries is reduced, and contains only features that cannot be inferred from morphological processes relevant for the entries (typically, semantic information).

4 Comparison with other Frameworks

4.1 Calder's Paradigmatic Morphology

The approach described in this paper is directly inspired by Calder's paradigmatic approach (1989); however, there are major differences between the two approaches.

First, lexical rules constitute the interface between syntax and morphology, in Calder's approach, while there are no lexical rules in the present framework: lexical entries are not specified for features predictable from the morphology, and these are inherited from the feature component through specification of types in paradigm entries.

Second, there is no definition of partial specification of strings, in Calder (1989), while the distinction between helping and main paradigms is one of the main features of the present approach: it allows to define morphemes independently of any specific root, and thereby to capture generalizations about morphemes. Thus, in the case of Arabic, both the melody and the pattern of consonants are defined independently of a root in helping paradigms, while in Calder's approach, one has to be arbitrarily chosen as basic (as belonging to the root), and regularities about that morpheme are lost.

4.2 Two-level Morphology

The most successful approach to morphology in computational linguistics has been the two-level approach started by Koskenniemi (1983). Although particularly suited to concatenative morphology, this approach has been extended to other types of morphologies, and more precisely to templatic morphology first by Kay (1987), then by Beesley (1991,1996) and Kiraz (1994, 1996) among others. But even though two-level systems are successful in practice, they suffer from a number of shortcomings: they are not really morphological systems, but phonological systems, as they focus on the phonology (or orthography) of words, and not on the morphology; morphotactics is usually implemented using continuation lexicons[12] (which implies that bound morphemes are lexical entries like stems, a disputable assumption); it assumes a linear view of phonology, which has been shown since the late seventies to be inadequate for the description of natural language phonology (Goldsmith 1990). Also, since this approach to morphology is particularly suited to concatenative morphology, new systems have to be built in order to account for non-concatenative languages (with the exception, maybe, of Beesley's system which can reuse the runtime code used for other languages; but compilation necessitates new code specific to Arabic).

I will not try to argue that the framework I advocate will produce better results in terms of speed and efficiency: the advantages of finite state techniques in that respect have been shown countless times. But there are other aspects to consider, besides efficiency: type of linguistic analysis allowed by the framework; whether different types of morphological processes can be accounted for; interface with other components of an NLP system (such as a syntactic parser), since morphological analysis is usually not an end in itself.

The language presented in this paper is declarative: paradigms relate partial strings to relevant features, and richer descriptions of strings derive from the combination of two or more helping paradigms. The ability to define partial strings in isolation, and then to combine several partial descriptions of strings through unification entails that this framework is not biased towards a specific type of morphological process.

Also, this approach to morphology is similar to and therefore can readily interface with a syntactic theory like HPSG, which is used widely in computational linguistics.

Finally, this approach does not restrict the grammar writer to the framework of linear phonology, and allows for the testing of a certain type of morphological theories, which claim that bound morphemes do not exist as lexicon

[12]There are a number of exceptions, Bear (1986), Ritchie et al. (1992), Kiraz (1996).

entries, but exist only in morphological relations (and more particularly the Word and Paradigm approaches (Matthew 1991, Anderson 1992, Bybee 1985 and many others)).

4.3 DATR

There have been other approaches to morphology than the two-level approach in computational linguistics; Gazdar and Evans have developed DATR (1996), a knowledge representation language, which has been used to write fragments of the morphology of a number of languages, among them Arabic (Gibbons 1990, Cahill 1990). But as is the case with other approaches, DATR is more suited to express concatenative processes than other types of morphological processes: Cahill shows that DATR in itself cannot account for non linear morphology, and she creates a new module, MOLLUSC, to use in conjunction with DATR, whose purpose is to account for the phonology of morphological processes.

5 Future Work

This language has not been implemented yet, and implementation is the next goal. The operations that the language supports are well defined and well known: they are monotonic inheritance, unification of feature structures and string unification. Inheritance (monotonic and non-monotonic) is a feature of object-oriented programming languages, and both kinds of unification have been extensively used, unification of feature structures in the NLP community, string unification in the field of automatic theorem proving (Calder 1989); thus we can hope that implementation will be straightforward and results acceptable in terms of efficiency.

References

Anderson, S. 1992. *A-Morphous Morphology*, Camnbridge University Press.

Bear, J. 1986. "A morphological recognizer with syntactic and phonological rules", *Proceedings of the 7th International Conference on Computational Linguistics*, pages 272-276.

Beesley, K.R. 1991. "Computer analysis of Arabic morphology: a two-level approach with detours", Comrie, B. and Eid, M., eds., *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*, pages 155-172. Benjamins, Amsterdam.

Beesley, K.R. 1996. "Arabic finite-state morphological analysis and generation", *Proceedings of the 17th International Conference on Computational Linguistics*, vol.1, pages 89-94.

Bybee, J. 1985. *Morphology: a Study of the Relation between Meaning and Form*. Benjamins, Amsterdam.

Cahill, L.J. 1991. *Syllable-based Morphology for Natural Language Processing* . Cognitive Science Research Paper 181, University of Sussex.

Calder, J. 1989. "Paradigmatic morphology", *Proceedings of the 4th Conference of the EACL*, pages 106-111, University of Manchester.

Evans, R., and Gazdar, G. 1996. "DATR: a language for lexical knowledge representation", *Computational Linguistics* , vol.22, 2, pages 167-216.

Goldsmith, J. 1991. *Autosegmental and Metrical Phonology* , Blackwell, London.

Kay, M. 1984. "Functional unification grammar: a formalism for machine translation", *Proceedings of the 10th International Conference on Computational Linguistics*, pages 75-78.

Kay, M. 1987. "Nonconcatenative finite-state morphology", *Proceedings of the 3rd Conference of the EACL*, pages 2-10.

Kiraz, G. 1994. "Multi-tape two-level morphology: a case study in Semitic non-linear morphology", *Proceedings of the 15th International Conference on Computational Linguistics*, vol.1, pages 180-186.

Kiraz, G. 1996a. *Computational Approach to Non-Linear Morphology*. PhD thesis, University of Cambridge.

Kiraz, G. 1996b. "SemHe: a generalised two-level system", *Proceedings of the 34th Annual Meeting of the ACL*, pages 159-166, Santa Cruz, CA.

Koskenniemi, K. (1983). *Two-level Morphology*. PhD thesis, University of Helsinki.

McCarthy, J. 1981. "A prosodic theory of non-concatenative morphology", *Linguistic Inquiry*, 12, vol.3, pages 373-418. Cambridge University Press, Cambridge.

Matthews, P.H. 1991. *Morphology*. Cambridge University Press, Cambridge.

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

Ritchie, G., Black, A., Russell, G. and Pulman, S. 1992. *Computational Morphology: Practical Mechanisms for the English Lexicon*. MIT Press, Cambridge (MA).

Siekmann, J.H. 1975. *String-unification*, part 1. Ms, Essex University.