Redefinition Of Turkish Morphology Using Flag Diacritics

Muhammet Şahin **Umut Sulubacak** **Gülşen Eryiğit**
Department of Computer Engineering
Istanbul Technical University
Istanbul, 34469, Turkey
{muhammetsahin, sulubacak, gulsen.cebiroglu}@itu.edu.tr

Abstract

This paper primarily discusses how to model Turkish morphotactics using flag diacritics. We present a two-level Turkish morphological analyzer based on a lexicon of word lemmata with over 49321 entries, as well as an auxiliary unknown word analyzer. Our main analyzer demonstrates the use of flag diacritics for Turkish, which is to date not a well-researched approach for the language. Turkish is an agglutinative language with many exceptions to phonetic and morphological rules, and flag diacritics are useful in handling these exceptions. Our unknown word analyzer operates without an extra lexicon, using affix stripping to find word lemmata by recursively removing affixes. We use the described methodology to find all possible lemmata which are not in our lexicon.

Keywords: Turkish, Morphology, Flag Diacritics

1 Introduction

Morphological analyzers are generally rule-based systems implemented using finite state transducers. The development of a morphological analyzer requires time and effort, and its performance depends on resources such as the lexicon and the phonetic modelling. Morphological analyzers are useful in alleviating lexicon word deficiencies and data sparseness problems often encountered in various natural language processing systems.

The two level description of the morphology of Turkish has been first described by Oflazer [1]. Since then, Oflazer's analyzer has evolved and improved with many extensions in the light of comments and critiques from the community. Although there have been many other studies of morphological analysis in the literature such as Hankamer [2], Eryiğit [3], Sak et al. [4], Zemberek [5], and more recently Çöltekin [6], Oflazer's work is still considered the state-of-the-art with its high coverage on most Turkish surface word forms. Furthermore, Oflazer's framework is compatible with the universal PoS-tag scheme [7] and has a significant history of corpora annotated using its tagset unlike the other frameworks. Regardless, with the growing interest in alternative corpus annotation projects and the high amount of web data to be parsed, we see that the analyzer still needs to be extended in order to successfully handle raw data. This paper presents our first effort towards a better morphological analyzer, and offers a redescription of the contemporary model of Turkish morphology in accordance with the advancements of the last twenty years.

Table 1. An example Turkish predicate nominative corresponding to an English sentence.

Lexical Form	Surface Form	English Meaning
gel+mA+Hyor+lAr	*gelmiyorlar*	"They are not coming."

2 Methodology

Our initial two level Turkish morphological analyzer uses flag diacritics, which are an extension of the Xerox finite state transducer (FST) implementation [8]. The FST supports feature setting unification that is able to reduce and speed up transducers, constrain the co-occurrence of morpheme pairs or the occurrence of single morphemes with certain words. Flag diacritics are also useful for marking lemmata for idiosyncratic morphological behavior that is feature-based rather than phonological. Feature setting and unification functions of flag diacritics are executed at runtime and used to determine whether a path is possible within a network. A network containing flag diacritics block many illegal paths that would otherwise cause over-generation, especially for a derivationally productive language such as Turkish.

Since Turkish is an agglutinative language, even single words can be often suffixed to correspond to whole sentences in English, as seen in Table 1. In the given figure, the '+' character denotes morpheme boundaries. There is a large variety of such inflectional suffixes, as well as derivational suffixes that can convert noun stems

Table 2. Case study of an incorrect but morphotactically valid analysis.

pat+lH+cA+Hn	+Noun+A3sg+Pnon+Nom ˆ*DB*+Adj+Withˆ*DB*+Adj+AsIf ˆ*DB*+Noun+Zero+A3sg+P2sg+Nom
pat`@U.CASE.var@`+*lH*`@U.CASE.yok@`+*cA*+*n*	

to verb or adjective stems, or verb stems to noun or adjective stems.

The main objective of using flag diacritics is to add a bit of memory to the finite state machine during the generation and analysis steps at runtime. Without this, a state transition on the FST would depend only on the current state and the input symbol, and there would be no constraint on which transition would be made next. Flag diacritics have also been used in other languages, such as Indonesian [9], Arabic [10] and Persian [11].

Our lexicon consists of 49321 word lemmata arranged under 14 parts of speech, namely *Noun*, *Verb*, *Pronoun*, *Adjective*, *Technical*, *Duplication*, *Verb*, *Postposition*, *Postposition-PCDat*, *Question*, *Determiner*, *Number*, *Number-Ordinal* and *Interjection*. Our non-trivial finite state automata for nouns, verbs and adjectives differ from the predecessor models in many respects, and are illustrated with drawings in this paper (see the appendical Figures 1, 2, 3 and 4).

An analysis of the word *patlıcan* ("eggplant") is given in Table 1. The given analysis would be a legal path derived from the root *pat* ("boom"), but the path should be blocked through the unification of the conflicting flags `@U.CASE.var@` and `@U.CASE.yok@`. Such conflicting flags are more likely to co-occur in productive derivations, and this fact moderates overgeneration during the analysis.

3 Flag Diacritics

The concatenation of certain affixes may not make sense according to morphotactic idiosyncrasies of a language, which may be impossible or impractical to attain by blocking state transitions in a finite state machine. The most important point of using flag diacritics is to disallow such illegal paths, in addition to scaling down surface form generation and speeding up the analysis. Since there are idiosyncratic exceptions to most morphotactic rules in Turkish, flag diacritics are very convenient.

Flag diacritics are multi-character symbols which are appended to lexical analyses of words, and the co-occurrence of some flags mark the lexical analysis as being invalid. The types of flag diacritics are Unification, Positive Setting, Negative Setting, Require Test, Disallow Test, and Clear Feature. This paper explains only the Unification, Require Test and Negative Setting types.

Table 3. Unification flag examples.

`@U.CASE.ABL@`
`@U.CASE.DAT@`
`@U.GEN.SIN@`
`@U.GEN.PL@`
`@U.num.sing@`
`@U.num.plural@`

Table 4. Examples of some verbs which allow the reciprocal suffix.

Root Form	English Meaning	Reciprocal Form	English Meaning
döv	"to beat"	*döv+Hş*	"to fight"
gül	"to laugh"	*gül+Hş*	"to laugh together"
böl	"to split"	*böl+Hş*	"to share"

Unification Test: The most commonly used and simplest flag diacritic is the Unification Test, with the template `@U.feature.value@`, as in the examples in Table 3. In the template, U stands for Unification and the arguments are replaced by arbitrary values.

The most straightforward way of denoting incompatible morphemes for U-type flag diacritics is to add flags to both morphemes with the same feature, but with different values. For instance, U-type flag diacritics are used especially for verb morphotactics, because most verb roots do not take reflexive or reciprocal suffixes, with some exceptions such as the reciprocal stems given in Table 4. To block the transition that would be made with the reciprocal suffix, the flag `@U.Hş.yok@` is appended to the verb root as shown in Table 5, which conflicts with the flag `@U.Hş.var@` in the reciprocal affix *+Hş*. Verbs without the `@U.Hş.yok@` flag are allowed to take the reciprocal suffix by default.

Negative Setting: The Negative Setting is the direct complement of the Unification Test. While the Unification Test restricts subsequent features to take a certain value in order to be valid, the Negative Setting requires them to take other values. A flag diacritic like `@N.feature.value@` functions by setting the value of the feature to the complement of the given parameter.

Table 5. Blocking and allowing the reciprocal suffix via unification flags.

Legal Paths	Illegal Paths
döv+Hş`@U.Hş.var@`	*yol+la*`@U.Hş.yok@`+*Hş*`@U.Hş.var@`
gül+Hş`@U.Hş.var@`	*sal+la*`@U.Hş.yok@`+*Hş*`@U.Hş.var@`
it+Hş`@U.Hş.var@`	*hava+lan*`@U.Hş.yok@`+*Hş*`@U.Hş.var@`

Table 6. Negative setting flag example.

gel`@N.Caus.present@`	*+DHr*`@U.Caus.present@`
"come"	+Causative

Table 7. Require flag examples.

+Hr`@U.Case.var@`
+Ar`@U.Case.var@`
+mHş`@U.Case.var@`
+cAsHnA`@R.Case@`

In the example given in Table 6, the verb stem *gel* takes the `@N.Caus.present@` flag, which means that the stem *gel* can take all verb morphotactics except for the suffix *+DHr*. Therefore, in a lexicon where all flags exclusively take Boolean values, the Unification Test and the Negative Setting are the dual of each other, providing the same functionality.

Require Test: The Require Test is a requirement condition for a specific feature, optionally with a specific value. If the flag is provided without a value specification, it only requires the feature to be present among the preceding unification flags, ignoring their values. If a value is also provided, the flag also requires the feature to be set to the specified value, i.e. the test performed by the flag `@R.feature.value@` succeeds if and only if there exists a previous feature `@U.feature.value@`, and differs from the Unification Test in that it is not compatible with the cases where the feature does not previously occur.

For example, the suffix *+cAsHnA* ("as though") is considered grammatical only if it follows one of the tense suffixes *+mHş*, *+Hr* or *+Ar*, and this behavior is modeled with the Require Test as shown in Table 7. As can be seen from the example, the Require Test is used not only for stems, but also for suffix sequences.

Disallow Test: Like the Require Test, the Disallow Test is a requirement condition for a specific feature, but with a value different from the optionally given parameter. Just as the Negative Setting is the complement of the Unification Test, the Disallow Test is the complement of the Require Test. As such, when the value parameter is omitted, the Disallow Test is only compatible with preceding flags with the given feature and a neutral value.

The test is used specifically for adjective morphotactics, since the adjective FST is frequently connected to the noun FST with zero input, and yet, some noun affixes are not convenient for nouns derived from adjective roots and must be disallowed. To handle this, the zero input symbol gets the `@U.Adjective.X@` unification flag, whereas the affixes to be disallowed within the noun FST get the disallow flag `@D.Adjective.X@`, as shown in Table 8.

Table 8. Disallow flag example.

`0@U.Adjective.X@:0@U.Adjective.X@`
+lAn`@D.Adjective.X@`

Table 9. Analyses demonstrating the affixation order of derivational and inflectional suffixes.

Döv	*+Hn*	*+mA*	*+yAcAk*	*+yDH*	*+Hm*
"beat"	+Reflexive	+Negative	+Future	+Past	+P1sg
Patla	*+t*	*+Hl*	*+yAcAk*	*+lAr*	
"explode"	+Causative	+Passive	+Future	+P3pl	

4 Adjective Morphotactics

Our finite state machine for adjective word stems are based on nominal morphotactics. However, some suffixes are exclusively added to adjectives.

The finite state automaton that models the affixation of adjective stems is shown in Figure 3. As can be seen in the figure, certain derivational suffixes or copula markers cause a part-of-speech shift in the adjective stem and convert it to a verb or noun stem. Although some adjective stems can be connected to nominal roots by zero input and used as noun stems, they are disallowed from taking certain nominal suffixes as a rule. This behavior is also modeled using the Disallow flag.

4.1 Verb Morphotactics

Verb stems follow an affixation pattern in which they optionally take reflexive, reciprocal, causative and/or passive derivational suffixes, followed by the optional polarity suffix and the mandatory tense and person suffixes as exemplified in Table 9. The tense is denoted by at least one aorist, progressive, perfect, future or narrative tense suffix or an imperative, necessitative, optative or conditional mood suffix, optionally followed by a second suffix for compound tenses. Due to this ordinal hierarchy, verb morphotactics are significantly more complicated than those for the other parts of speech. Flag diacritics are also often used for verb stems in addition to adjusting state transitions to implement some of these ordinal constraints.

Furthermore, there are many irregularities among verb stems that dictate which derivational suffixes may be appended to which stems, and in which combinations. These irregularities may

Table 10. Examples of various derivations of a single word resulting in shifts in part of speech.

Analysis	English Meaning
Oda+DA	"in the room" *(Noun)*
Oda+DA+yken	"while *[he is]* in the room" *(Adverb)*
Oda+DA+ymHş+CAsHnA	"as though *[he were]* in the room" *(Adverb)*
Oda+DA+ysA	"if *[he is]* in the room" *(Adverb)*
Oda+DA+ymHş	"*[he]* was in the room" *(Verb)*
Oda+DA+yHm	"*[I]* am in the room" *(Verb)*

Table 11. Examples of grammatically plural stems without an overt plural suffix.

ahali, "crowd"	Noun+A3pl+Pnon+Nom
enkaz, "debris"	Noun+A3pl+Pnon+Nom

be handled by grouping similarly behaving verb stems together and partitioning the lexicon so that verb stems of certain groups are only allowed to take certain combinations of suffixes. However, the lexicon partitioning method is too costly for modeling less common idiosyncrasies, at which point flag diacritics become a more practical approach again.

4.2 Nominal Morphotactics

There are two main groups of suffixes that nominal structures may take, both of which are optional. The first part consists of the possessive, plural and case suffixes, whereas the second part is for derivational suffixes that possibly convert the noun stem to an adverb or verb stem, as seen in the example in Table 10.

Lexicon partitioning is used for noun stems as well, in order to formalize three exceptional groups of nouns that take on respectively dative, plural and possessive meanings as stems, without the need for the dative, plural and possessive suffixes. Samples from the first two groups are shown in Tables 11 and 12.

The third group of noun stems is covered by a specific compound noun formation, which is made up of two nouns in a possessive relation fused together. Such compound nouns orthographically occur as a single word, but retain their possessive meanings, and are subject to a different possessive affixation paradigm. Samples from this group are shown in Table 13.

The fourth group of noun stems is the default group, with 3rd person plural agreement and in the nominative case by default. Samples from

Table 12. Examples of grammatically dative stems without an overt dative suffix.

içeri, "inward"	Noun+A3sg+Pnon+Dat
dışarı, "outward"	Noun+A3sg+Pnon+Dat
aşağı, "downward"	Noun+A3sg+Pnon+Dat
yukarı, "upward"	Noun+A3sg+Pnon+Dat

Table 13. Possessive suffix ambiguity caused by an idiomatic usage.

buzdolabı+Noun+A3sg+Pnon+Nom	*buzdolabı* "refrigerator"
buzdolabı+Noun+A3sg+P3sg+Nom	*buzdolabı* "his refrigerator"

Table 14. Default agreement, possession and case attributes in Turkish noun stems.

masa	Noun+A3sg+Pnon+Nom
sandalye	Noun+A3sg+Pnon+Nom
cam	Noun+A3sg+Pnon+Nom

this group are shown in Table 14.

4.3 Proper Noun Morphotactics

Proper noun morphotactics are basically the same as noun morphotactics. However, Turkish requires inflectional suffixes added to proper noun stems (except for the plural suffix *–lAr*) to be separated from the stem by an apostrophe character. Derivational affixes (as well as the plural suffix) are not subject to this rule. To model this orthographical phenomenon, we include duplicates of the relevant inflectional suffixes with apostrophes in the surface form in our lexicon, as shown in Table 15.

4.4 Pronoun Morphotactics

Pronouns, like adjectives and proper nouns, are a part of the nominal category. However, they have some differences from the noun-nominal root in affixation, and their analyses may return different tags. Furthermore, such differences also exist between the subdivisions of pronouns, such as personal (e.g. *ben*, "I"), reflexive (e.g. *kendi* "himself"), reciprocal (e.g. *birbiri*, "each other"), demonstrative (e.g. *ora*, "there") and interrogative pronouns (e.g. *kim*, "who"). The subtle differences between the different kinds of pronouns could not be defined via flag diacritics, therefore they have been modeled on separate FSTs for all five kinds of pro-nouns.

5 Analyzers

We have embedded our morphotactic transition scheme and 37 custom phonological rules into Xerox FST [8] and implemented a morphological analyzer for Turkish. To complement the transducer, we have developed a lexicon which includes flag diacritics. Our lexicon has

Table 15. Apostrophe usage in the affixation of proper nouns.

İstanbul+'DA	İstanbul'da
Ankara+'yH	Ankara'yı
Ali+'DAn	Ali'den

Table 16. The lexicon modification employed by the unknown word analyzer.

ˆUnknownNoun	noun-root;
masa	noun-root;
kitap	noun-root;

49321 additional lexemes and assigns morphotactic flags to all relevant words as described in the previous sections.

Our final analyzer can produce analyses only for morphologically valid words whose lemmata are contained in the lexicon. Our tests on the METU-Sabancı Turkish Treebank [12] show that our analyzer is able to correctly analyze 99.8% of all the words in the corpus. Regardless, as with all such morphological analyzers, our system is constrained by the size of our lexicon. Building up a lexicon that covers all possible roots and stems in the language is a task that requires years of effort. To circumvent this problem, we have also implemented a complementary unknown word analyzer, which uses an affix stripping to find all phonologically possible stems for a given word.

The unknown word analyzer is essentially an extension of the main analyzer, which makes use of wildcard entries that are able to morph into any phonologically valid Turkish stem, typeset as seen in Table 16. As such, the analyzer derives the input word from a lexicon stem if possible, or backs off to the unknown stem expression if the lexicon did not contain a valid stem.

6 Conclusion

Building upon previous studies on morphological analysis, we have developed a basis for a more inclusive analyzer and presented our preliminary work. Our analyzer is primarily based on Oflazer's two level Turkish morphological analyzer, which is the current state-of-the-art for Turkish. In comparing our results with Oflazer's output on the Turkish Treebank, we see that our analyzer is on par with Oflazer's analyzer in terms of the number of generated possible outputs. The analysis deficiencies are, for the most part, lexicon-based. For future work, we aim to fine-tune our analyzer in accordance with comments and criticism we have collected during years of annotation effort.

References

[1] Kemal Oflazer. Two-level description of turkish morphology. *Literary and linguistic computing*, 9(2):137–148, 1994.

[2] Jorge Hankamer. Finite state morphology and left-to-right morphology. In *West Coast Conference on Formal Linguistics, Bildiri Kitab*, 1986.

[3] Gülşen Eryiğit and Eşref Adalı. An affix stripping morphological analyzer for turkish. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria*, pages 299–304, 2004.

[4] Haşim Sak, Tunga Güngör, and Murat Saraçlar. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *GoTAL 2008*, volume 5221 of *LNCS*, pages 417–427. Springer, 2008.

[5] Ahmet Afşın Akın and Mehmet Dündar Akın. Zemberek, an open source nlp framework for turkic languages. *Structure*, 2007.

[6] Çağrı Çöltekin. A freely available morphological analyzer for turkish. In *LREC*, 2010.

[7] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. 2011.

[8] Kenneth R Beesley and Lauri Karttunen. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*, 2003.

[9] Femphy Pisceldo, Rahmad Mahendra, Ruli Manurung, and I Wayan Arka. A two-level morphological analyser for the indonesian language. In *Australasian Language Technology Association Workshop 2008*, volume 6, pages 142–150, 2008.

[10] Mohammed Attia. An ambiguity-controlled morphological analyzer for modern standard arabic modelling finite state networks. In *Challenges of Arabic for NLP/MT Conference, The British Computer Society, London, UK*. Citeseer, 2006.

[11] Karine Megerdoomian. Extending a persian morphological analyzer to blogs. In *Proceedings of the Second Workshop on Persian Language and Computers*, 2006.

[12] Bilge Say, Deniz Zeyrek, Kemal Oflazer, and Umut Özge. Development of a corpus and a treebank for present-day written turkish. In *Proceedings of the eleventh international conference of Turkish linguistics*, pages 183–192, 2002.

Figure 1. The finite state transducer for verbs.

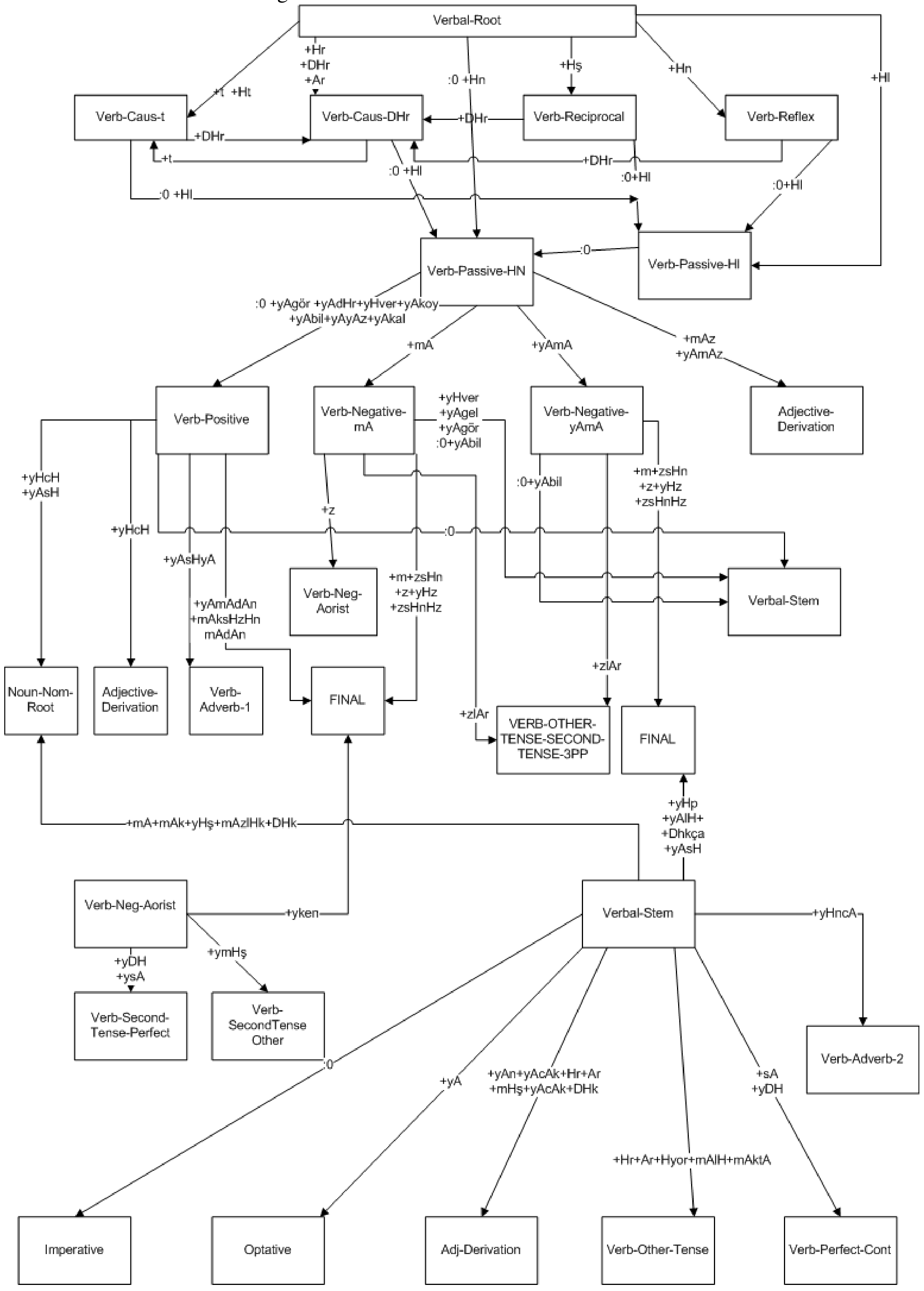


Figure 2. The finite state transducer for verbs.

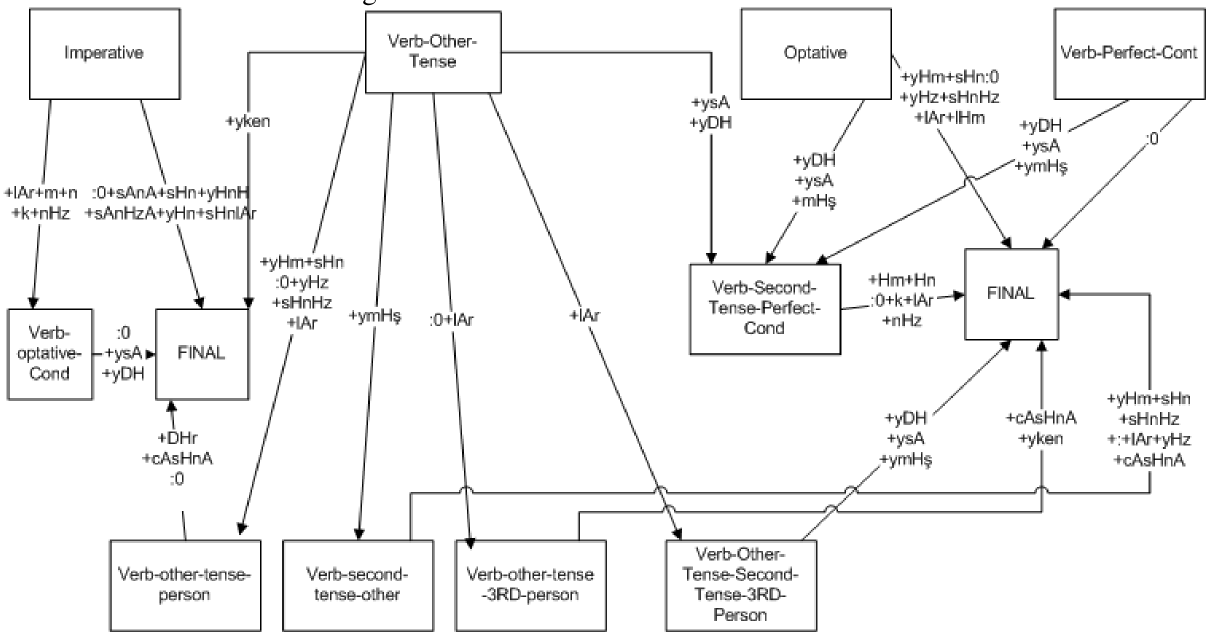


Figure 3. The finite state transducer for adjectives.

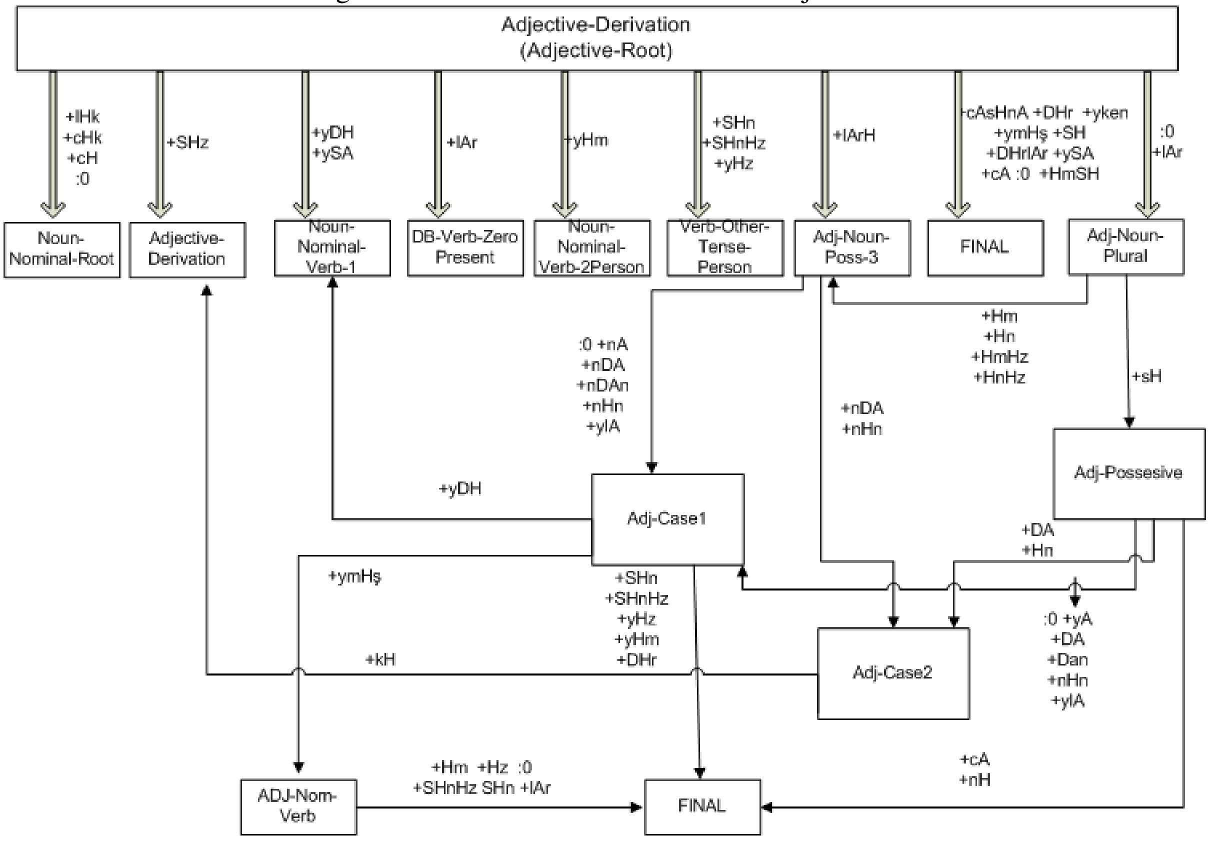
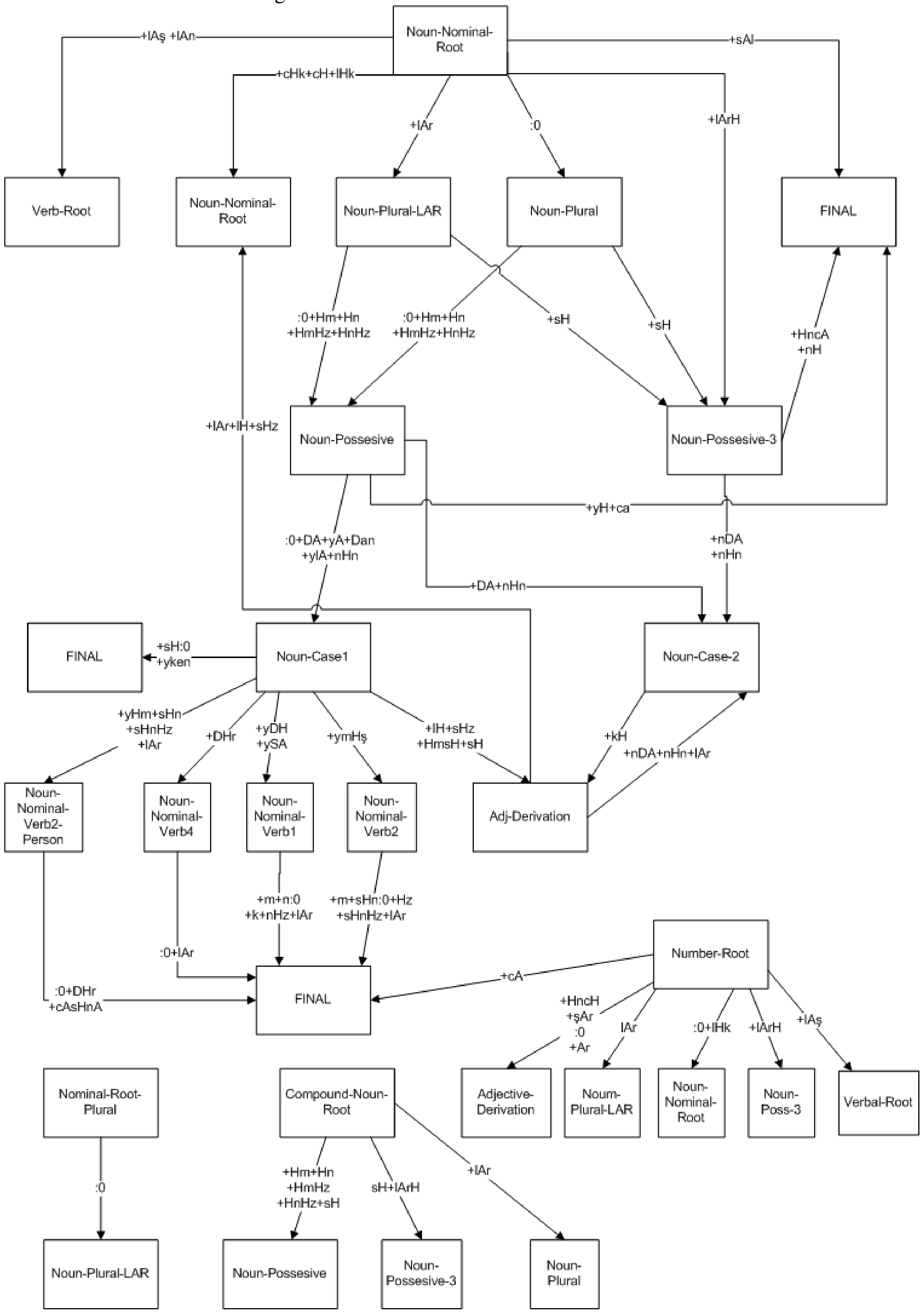


Figure 4. The finite state transducer for nouns.



Noun-Nominal-Root
+lAş +lAn
+sAl
+cHk+cH+lHk
+lAr
:0
+lArH
Verb-Root
Noun-Nominal-Root
Noun-Plural-LAR
Noun-Plural
FINAL
:0+Hm+Hn +HmHz+HnHz
:0+Hm+Hn +HmHz+HnHz
+sH
+sH
+HncA +nH
+lAr+lH+sHz
Noun-Possesive
Noun-Possesive-3
+yH+ca
:0+DA+yA+Dan +ylA+nHn
+nDA +nHn
+DA+nHn
FINAL
+sH:0 +yken
Noun-Case1
Noun-Case-2
+yHm+sHn +sHnHz +lAr
+DHr
+yDH +ySA
+ymHş
+lH+sHz +HmsH+sH
+kH
+nDA+nHn+lAr
Noun-Nominal-Verb2-Person
Noun-Nominal-Verb4
Noun-Nominal-Verb1
Noun-Nominal-Verb2
Adj-Derivation
+m+n:0 +k+nHz+lAr
+m+sHn:0+Hz +sHnHz+lAr
:0+lAr
Number-Root
+cA
:0+DHr +cAsHnA
FINAL
+HncH +şAr :0 +Ar
lAr
:0+lHk
+lArH
+lAş
Nominal-Root-Plural
Compound-Noun-Root
Adjective-Derivation
Noum-Plural-LAR
Noun-Nominal-Root
Noun-Poss-3
Verbal-Root
:0
+Hm+Hn +HmHz +HnHz+sH
sH+lArH
+lAr
Noun-Plural-LAR
Noun-Possesive
Noun-Possesive-3
Noun-Plural