# Integrating Classical Sindarin Morphology and Syntax using XFST and XLE

## Machine Language Processing WS2007/2008

Thomas Zink, Andreas Engl, and Wolfgang Miller

University of Konstanz
{*thomas.zink*, *andreas.engl*, *wolfgang.miller*} @uni-konstanz.de

**Abstract.** Sindarin is a constructed language invented by the author and linguist J.R.R. Tolkien. It is the language spoken by the Elves. Though the number of complete sindarin sentences is rather limited[1] enough information is available to derive syntactic rules and build a Sindarin grammar. This project aims at modelling the basics of Classical Sindarin grammar in LFG using Xerox XLE system. In addition it interfaces to Xerox XFST to lookup morphological information which has already been implemented in another project. For more information about Classical Sindarin morphology see the project documentation[2]. The project is mainly based on information provided by David Salo's "A Gateway to Sindarin"[1].

## 1 System Overview

Figure 1 shows an overview of the system implementation. It consists of three components which are further discussed in more detail.

### 1.1 Tokenizer

The tokenizer takes textual input and generates tokens that can be further processed and analysed by the morphological analyser. It performs the following tasks:

- all characters are cast to lower-case.
- accents are removed.
- binding articles (like *i-*) are split from nouns.
- numbers are identified and isolated
- special and punctuation characters characters are isolated.
- identified tokens are separated by a token boundary TB.

Tokenization removes ambiguities and simplifies the creation and maintenance of lexicon files for the morphological analyser and the **xle** grammar. Though **xle** provides a basic tokenizer that is suitable for english and german grammars it is not sufficient for parsing Sindarin. The main reason is the existence of articles that bind directly to the corresponding noun. They have to be split and

*Konstanzer Online-Publikations-System (KOPS)*
*URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-175956*

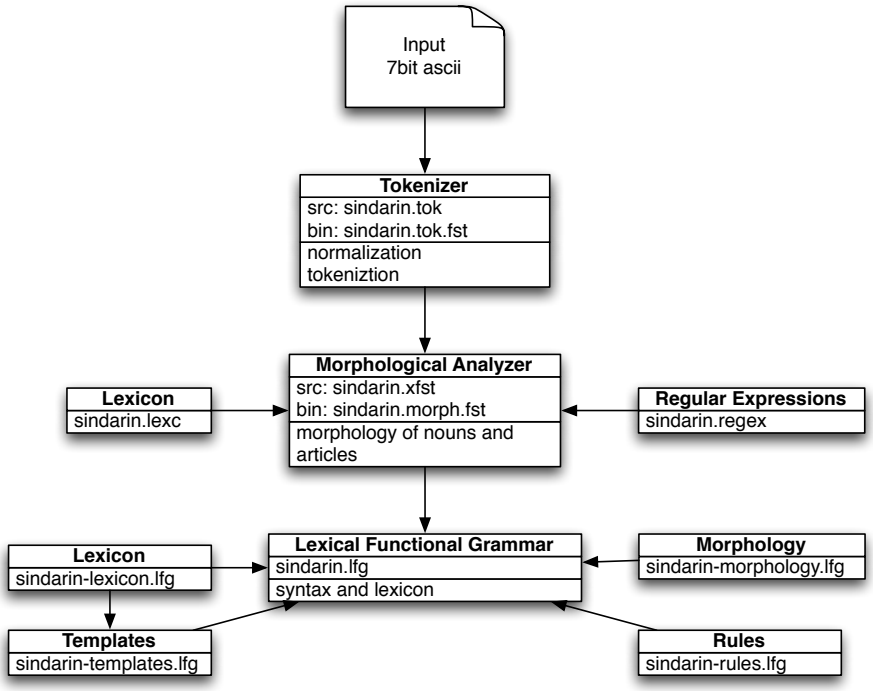


**Fig. 1.** System Overview

provided as a distinct token for analysis. Otherwise, all possible permutations of binding articles and nouns would have to be present in the **xle** and **lexc** lexicons. Another reason are the numerous accentuations present in Sindarin. Table 1 shows the Sindarin vowels and their accentuations. Accentuations undergo different morphological changes, sometimes to another accent (like long to stressed) sometimes disappearing completely. Again to prevent the **xle** lexicon to degenerate the accentuations are completely removed and normalized to their unaccented vowel. Since the version of **xfst** used for this project cannot deal with unicode characters (and even the unicode character set would not suffice to represent all Sindarin sounds) the whole system is designed to process only 7bit ascii encoded files. Using other encodings can lead to erroneous results.

The tokenizer can be tested using an input file or the **echo** command. During

```
# echo "i-tˆol acharn." | tokenize sindarin.tok.fst
i-@tol@@acharn@.@@
```

Fig. 2. Testing the tokenizer

system development it turned out, that **xle** expects the @ sign as token boundary. So the tokenizer outputs @ as token boundary character.

| | | | | | | |
|---|---|---|---|---|---|---|
| Short | a | e | i | o | u | y |
| Long | 'a | 'e | 'i | 'o | 'u | 'y |
| Stressed | ^a | ^e | ^i | ^o | ^u | ^y |

**Table 1.** Sindarin Accentuation

### 1.2 Morphological Analyzer

The morphological analyzer takes tokenized input and returns the stem or lemma of the token and morphological tags that bear information about the part-of-speech and morphological changes of the stem. It is implemented in **xfst** and is based on an implementation done in a former project [2]. See the project's documentation for further information on sindarin morphology and the implementation in **xfst**. However, significant changes had to be made to successfully integrate the morphological analyzer into **xle**. For a description of the morphological analyzer see section 2. The morphological analyzer is split into two components, a lexicon written in **lexc** and regular expressions defining the morphological rules. It only includes morphology for sindarin nouns and determiners. Figure 3 shows an example lookup and output of the morphological analyser using the **xfst** interface.

```
xfst[1]: up edhel
edhel+Indef+Noun+Sg
edhel+Def+Noun+Sg
edhel+Def+Noun+Gen
```

Fig. 3. Example output of the morphological analyser.

### 1.3 Lexical Functional Grammar

Lexical Functional Grammar, or short LFG, is a theory of grammar that provides rules and structures to describe human language. The system used in this project to implement a basic Sindarin grammar is xerox' **xle**. It is possible to use only **xle** to define morphology, syntax and semantics. However, **xle** provides an interface to use finite state networks implemented in **xfst** to use as tokenizers and morphological analyzers. This strongly reduces development and maintenance time, since each morphological form would require a unique entry

in the **xle** lexicon. If an external morphological analyzer is used only the stems plus part-of-speech tags need to be present. Since the morphological analyzer only handles nouns and determiners only noun phrases interface to **xfst**. See section 4 for further information on how the integration is realized and section 3 for a description of sindarin grammar.

## 2 Sindarin Morphology

The implementation used is based on the work done in a former project. However, the original implementation was not suitable for integration in **xle** for various reasons. First, the system features a combined morphology of articles and nouns. A seperate lookup of articles or even noun forms affected by articles is not possible. This drastically affects the size of the **xle** lexicon which would have to list every combination of nouns and articles. Second, the sindarin writing system called Tengwar is phonologic, that is every sound has it's own symbol. Due to the lack of enough characters in the ascii character set some sounds had to be represented using a set of two and in rare occasions even three characters. To closely resemble Tengwar, the Sindarin writing system, which is phonologic, these sets have been implemented as multicharacter symbols, resulting in a single state in the state machine. As it turns out, this raises problems in evaluating the morphology, since all multicharacter symbols are automatically returned as tags by **lookup**. Thus, the correct stem cannot be retrived. Figure 4 shows an example of a lookup in the original morphological analyser. Clearly this would fail to generate a correct input for **xle**. For these reasons the Sindarin morphological

```
# echo "i meraid" |tokenize sindarin.tok.fst |lookup -flags mb sindarin.old.fst

9 input symbols processed.
i i +?

meraid meraid +?
```

Fig. 4. Lookup in the original Sindarin morphological analyser

analyser had to be completely rewritten. The new system renounces the use of multicharacter symbols to represent Sindarin sounds. It allows the lookup of articles independent of nouns and their affected forms. Figure 5 shows the same lookup used earlier in the new implemented morphological analyser. As can be seen **lookup** now outputs a pair of stems and morphological tags for all input tokens. This output is suitable for further processing in **xle**.

### 2.1 Morphological TAGS

Table 2 shows the available tags used in the morphological analyser. The tags are

```
# echo "i meraid" |tokenize sindarin.tok.fst |lookup -flags mu sindarin.morph.fst

9 input symbols processed.
i i +PlCDet

meraid barad +Def+cNoun+Pl
```

Fig. 5. Lookup in the new implementation of the Sindarin morphological analyser

| TAG | POS | meaning | example |
|---|---|---|---|
| +SgDet | determiner | regular singular determiner | i-+SgDet |
| +SgIDet | determiner | co-occurring with Nouns starting with an i sound | ir+SgIDet |
| +PlDet | determiner | regular plural determiner | in+PlDet |
| +PlCDet | determiner | co-occurring with Nouns starting with a consonant | i+PlCDet |
| +GenDet | determiner | genitive determiner | en+GenDet |
| +Indef | nouns | noun is in an indefinit state, ie bears no article | amon+Indef+Noun+Sg |
| +Def | nouns | noun is in a definit state, ie bears an article | amon+Def+Noun+Sg |
| +Sg | nouns | singular person | amon+Def+Noun+Sg |
| +Pl | nouns | plural person | amon+Def+Noun+Pl |
| +Gen | nouns | genitive case | edhel+Def+Noun+Gen |
| +Noun | nouns | regular noun | amon+Indef+Noun+Sg |
| +cNoun | nouns | noun starting with a consonant, affect plural determiner | barad+Def+cNoun+Sg |
| +iNoun | nouns | noun starting with an i sound, affect singular determiner | ˆidh+Def+iNoun+Sg |
| +pNoun | nouns | plural noun, i.e. the base form is plural | nˆel+Indef+pNoun+Pl |
| +Ath | nouns | plural ath-suffix | hwˆin+Indef+cNoun+Ath |
| +Hoth | nouns | plural hoth-suffix | glam+Indef+pNoun+Hoth |

**Table 2.** Tags with their part-of-speech and meaning

set to ensure an unambiguous legal combination of determiners and nouns. Some noun forms can be identical in different cases while allowing only the presence or absence of a specific article. Figure 6 shows an example. As can be seen, the

```
# echo "i-edhel" |tokenize sindarin.tok.fst |lookup -flags mu sindarin.morph.fst

8 input symbols processed.
i-i-+SgDet

edhel edhel +Def+Noun+Sg
edhel edhel +Def+Noun+Gen
edhel edhel +Indef+Noun+Sg
```

Fig. 6. The significance of tags for morphological analysis

form *edhel* can occur in three cases. However, together with the singular article *i-* there is only one legal combination which is the first in the example. Figure 7 shows what happens if the hyphen is replaced by a space character. Again, the

```
# echo "i edhel" |tokenize sindarin.tok.fst |lookup -flags mu sindarin.morph.fst

8 input symbols processed.
i i +PlCDet

edhel edhel +Def+Noun+Sg
edhel edhel +Def+Noun+Gen
edhel edhel +Indef+Noun+Sg
```

Fig. 7. The significance of tags for morphological analysis

token *edhel* can occur in three cases. However, the plural article *i* is tagged with *+PlCDet* meaning it can only co-occur with nouns starting with a consonant that are tagged with *cNoun*. The noun phrase *i edhel* thus is illegal and would be rejected by the grammar. These conditions can be modeled in **xle** and a careful choice of tags and condition reduces the development and maintenance effort.

## 3 Sindarin Grammar

Sindarin is a consistently *head-left* driven language. That is, in any phrase the *head* will be on the left and their modifying complements will follow on the right. The following sections describe phrases in more detail.

### 3.1 Noun Phrases

A simple noun phrase consists only of a single noun, pronoun or name. A noun phrase can also consist of a simple noun phrase followed by other noun phrases. Our implementation specifies a noun phrase base (NPB) which can be connected with conjuncts (NPC) without restrictions.

```
NP --> { NPB PN | PN } (NPC)*.

Example: Im narvi.
```

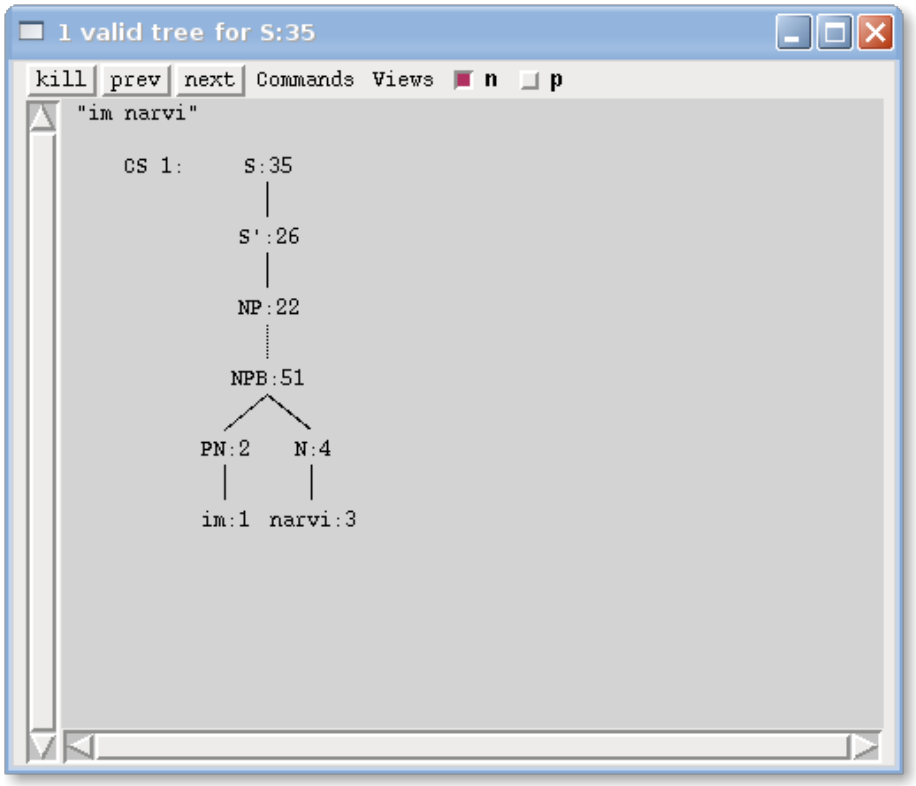


**Fig. 8.** sentence: Im narvi

In more complex examples noun phrases can be substituted by a pronoun or followed by a suffix (NPS).

### 3.2 Noun and Prepositional Phrase

A noun phrase can consist of a noun head followed by a prepositional phrase that modifies the noun.

```
S --> NP (PP).

Example: Celebrimbor o Eregion.
```

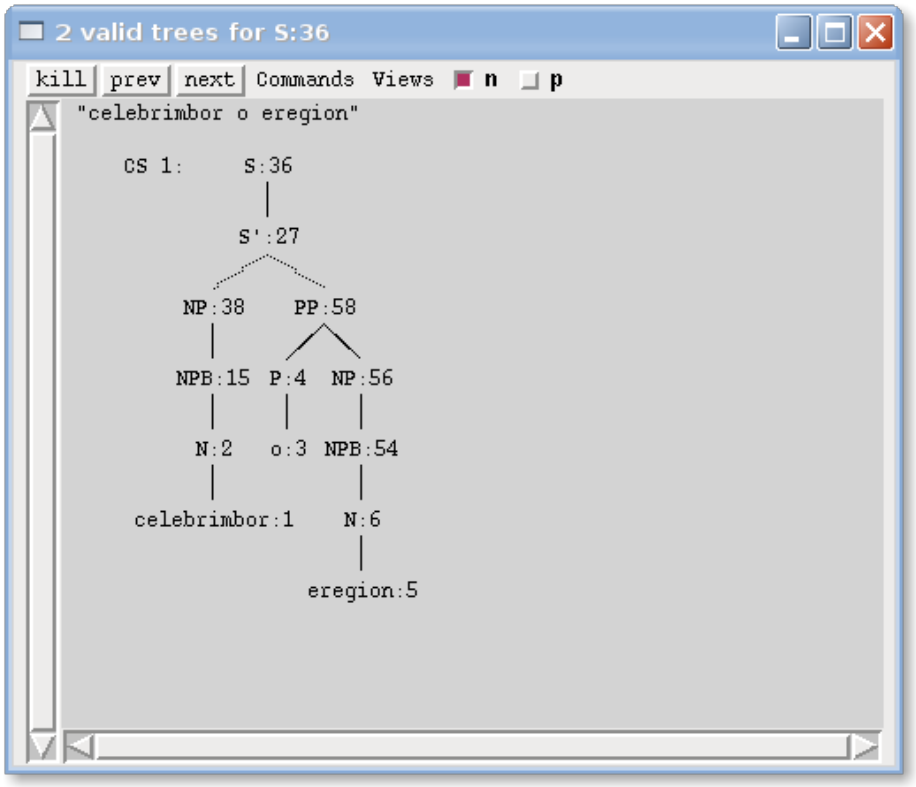


**Fig. 9.** sentence: Celebrimbor o Eregion

### 3.3 Determiner Phrases

A determiner phrase consists of various forms of the article 'i' followed by a noun phrase. We incorporated the rule into our Noun-Phrase-Base definition because the article can be repeated when using conjuncts.

```
NPB --> D N.
```

#### Singular Determiner Phrases

```
Example: i-annon.
```

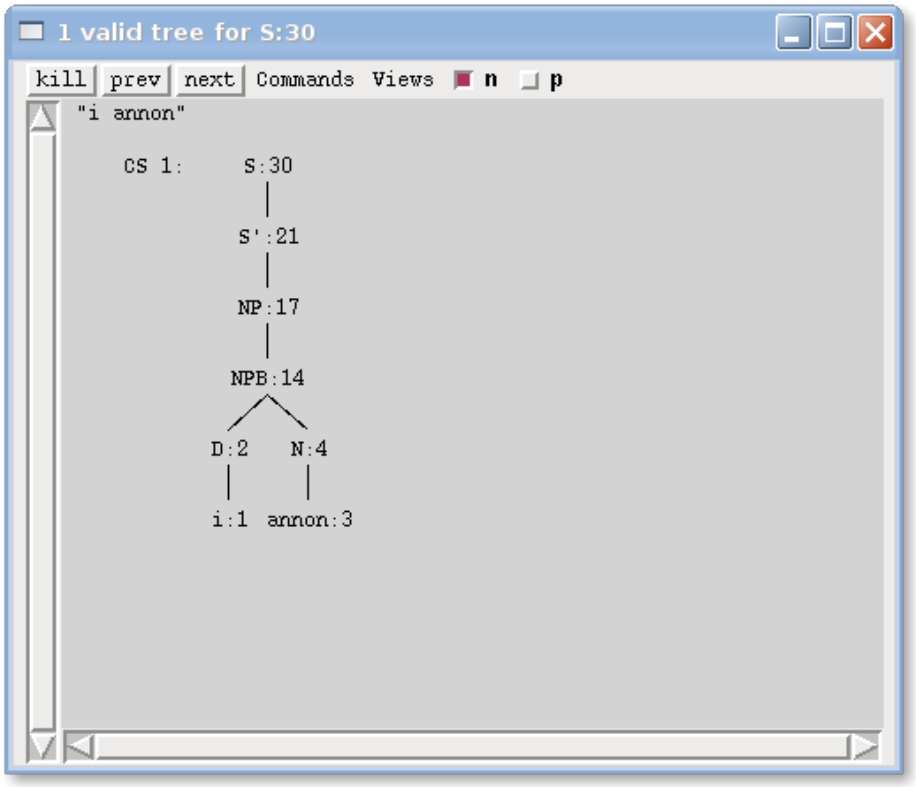


**Fig. 10.** sentence: i-annon

### Plural Determiner Phrases

```
Example: i thiw.
```

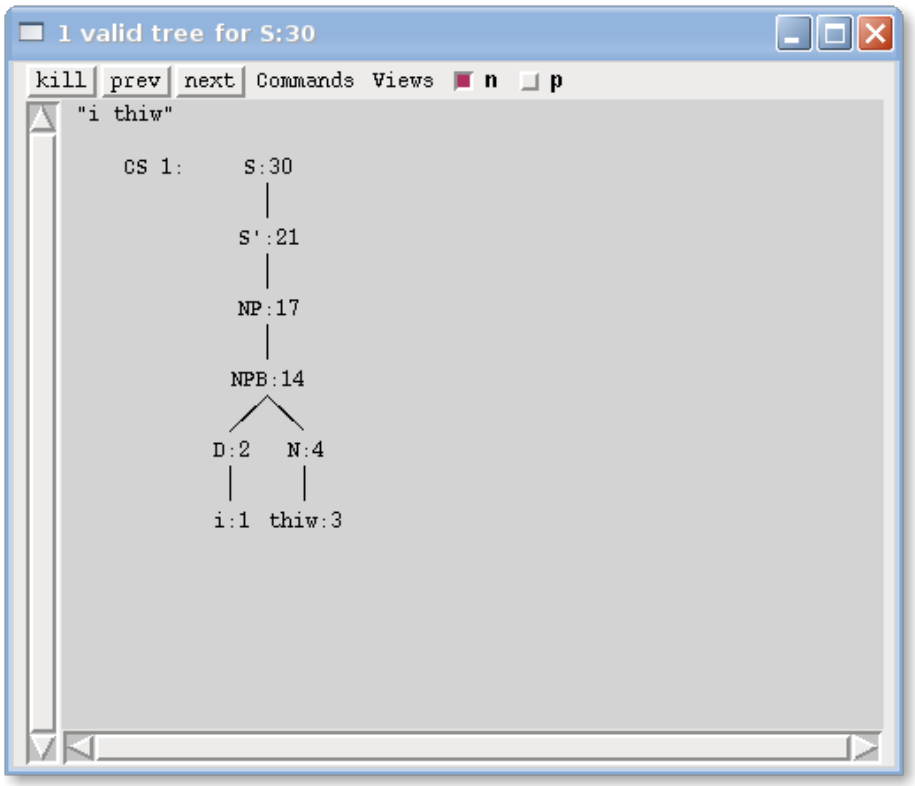


**Fig. 11.** sentence: i thiw

### 3.4 Adjective Phrases

An adjective phrase may consist of an adjective or an adjective combined with other modifiers. It may include adverbs or prepositional phrases, which either precede or follow the adjective. Our implementation is limited to the case where an adjective is preceded by simple modifiers like a degree, as seen in the PAP rule.

```
AP --> "Adjective Phrase"
          (PAP)* ADJ.

PAP --> "Prefix for Adjective Phrases"
           { DEG | NUM NP }.

Example: leben tail brand.
```

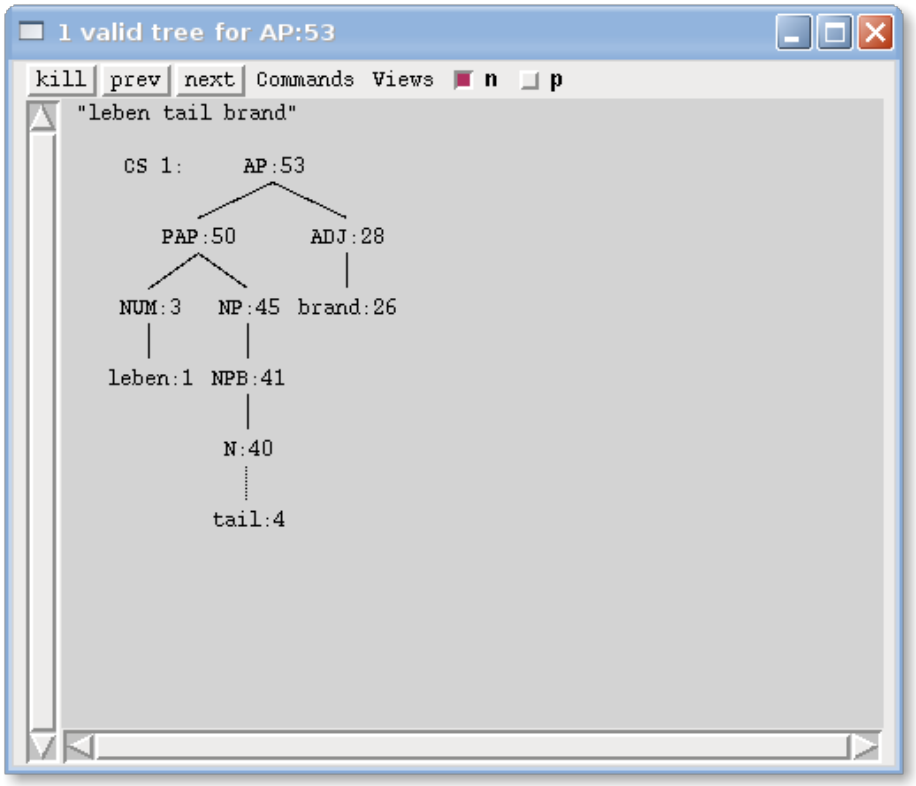


**Fig. 12.** sentence: leben tail brand

### 3.5 Prepositional Phrases

A prepositional phrase consists of preposition followed by a pronoun, a determiner phrase, or a noun phrase.

**Preposition and Pronoun** Preposition merge with a following pronoun.

```
Example: enni.
```

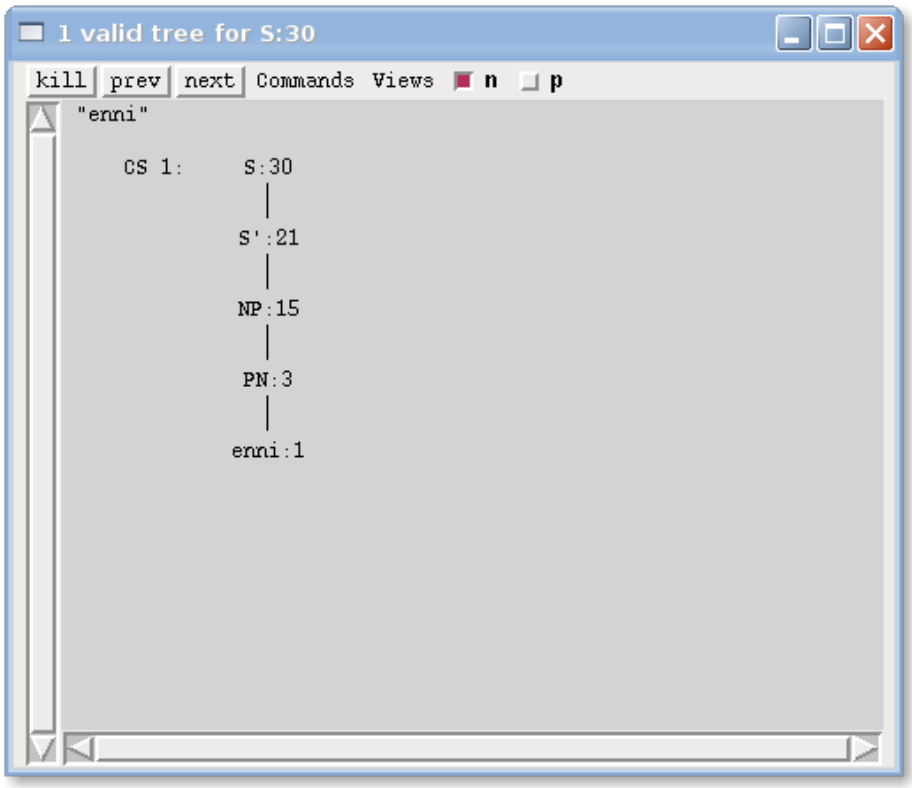


**Fig. 13.** sentence: enni.

**Preposition and Determiner Phrases** These prepositional phrases consist of a preposition followed by a determiner phrase.

```
Example: ni Pheriannath.
```

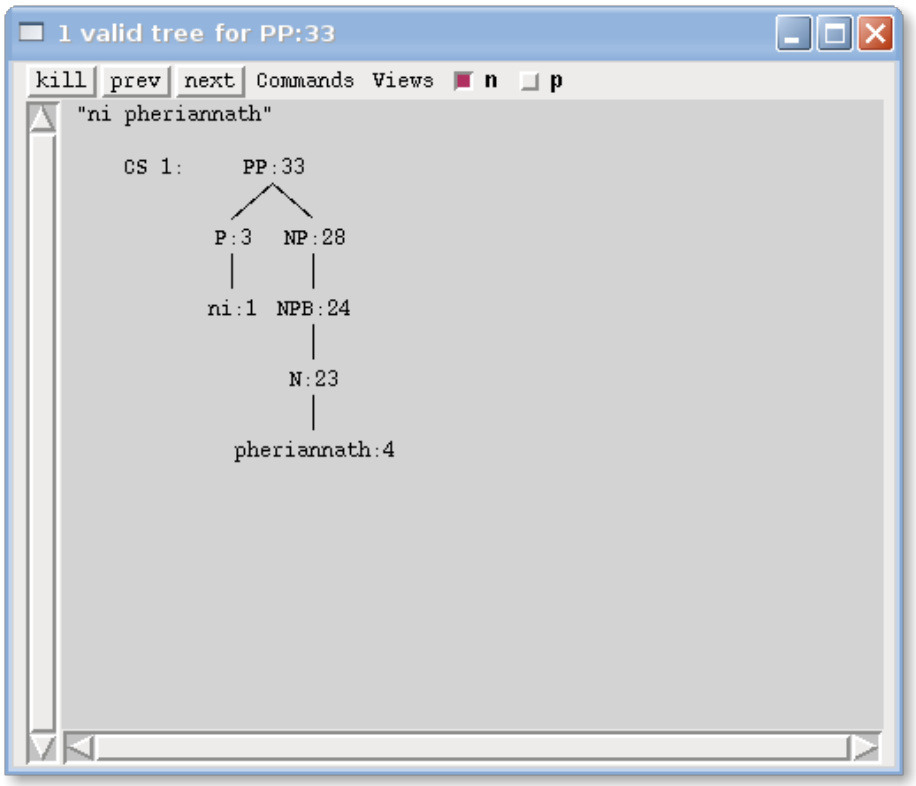


**Fig. 14.** sentence: ni Pheriannath.

**Noun-and-Prepositional-Phrase Sentences** A sentence can consist of a noun phrase and a prepositional phrase. In that case the prepositional phrase does not form part of the noun phrase, but functions adverbially to the verb 'to be' which itself is unexpressed.

```
S --> NP PP.

    Examples: - gurth an glamhoth.
              - aglar ni pheriannath.
```

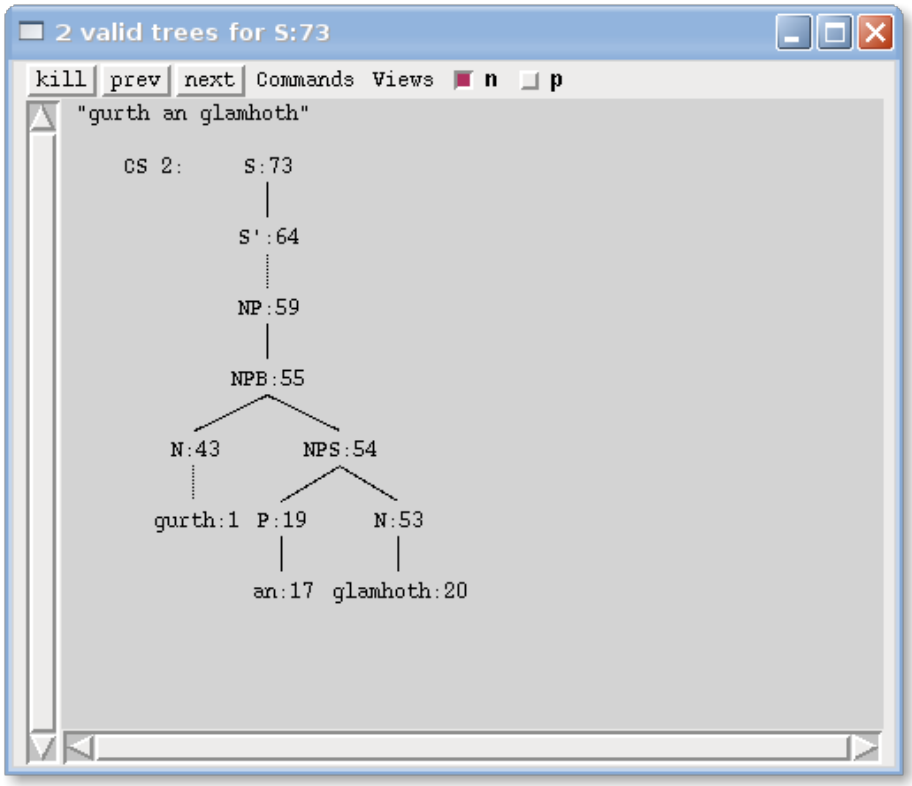


**Fig. 15.** sentence: gurth an glamhoth.

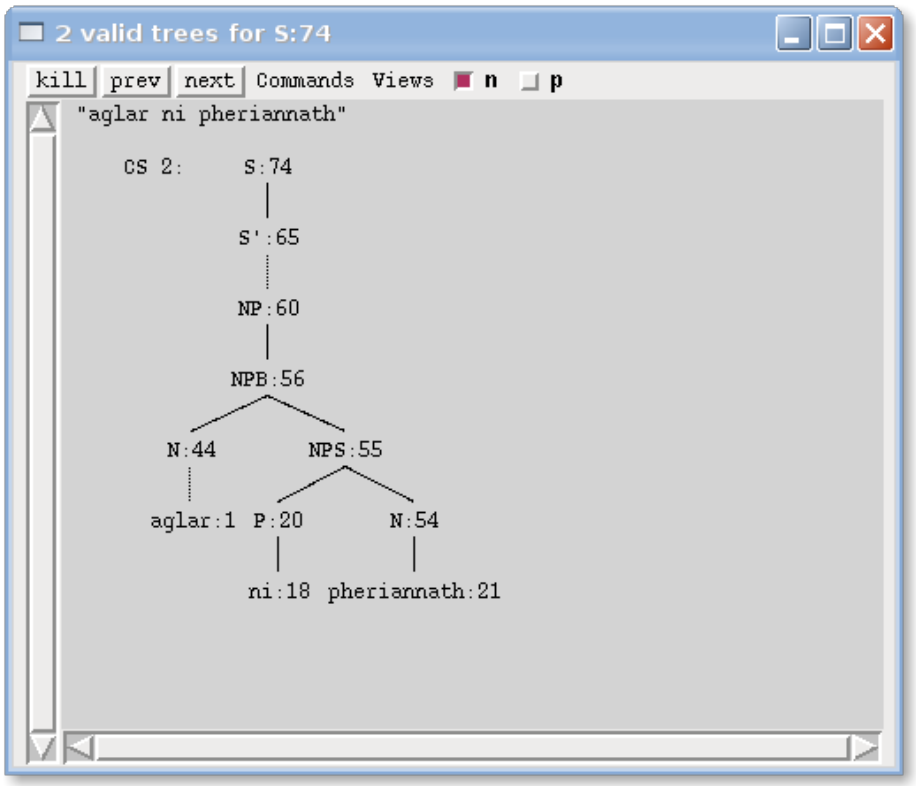


**Fig. 16.** sentence: aglar ni pheriannath.

### 3.6 Adjectival Sentences

A adjectival sentence can consist of a single adjective phrase or an adjective phrase preceding a noun phrase. In the following example, the noun phrase is preceded by a predicative adjective phrase.

```
Example: leben tail brand i-annon.
```

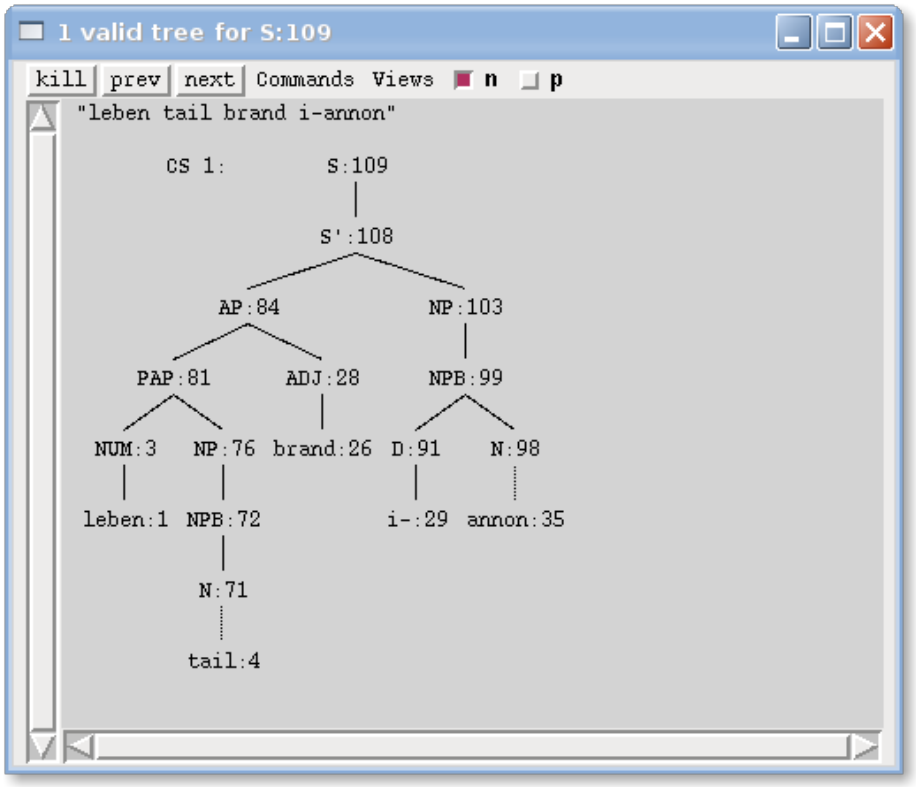


**Fig. 17.** sentence: leben tail brand i-annon.

### 3.7 Verbal Sentences

The basic word order of a verbal sentence is Verb-Subject(-Object) or VS(O). There is, however, a so-called 'topic position' external to the basic VS(O) order which holds a topic or emphasized phrase. Whenever a subject or object appears in this topic position, it falls out of the normal word order. Therefore it is difficult to derive a correct tree in respect of the subject and object position. Our grammar rule implementation makes no explicit attempts to overcome that problem.

#### Intransitive: VS

`Example: tol acharn.`

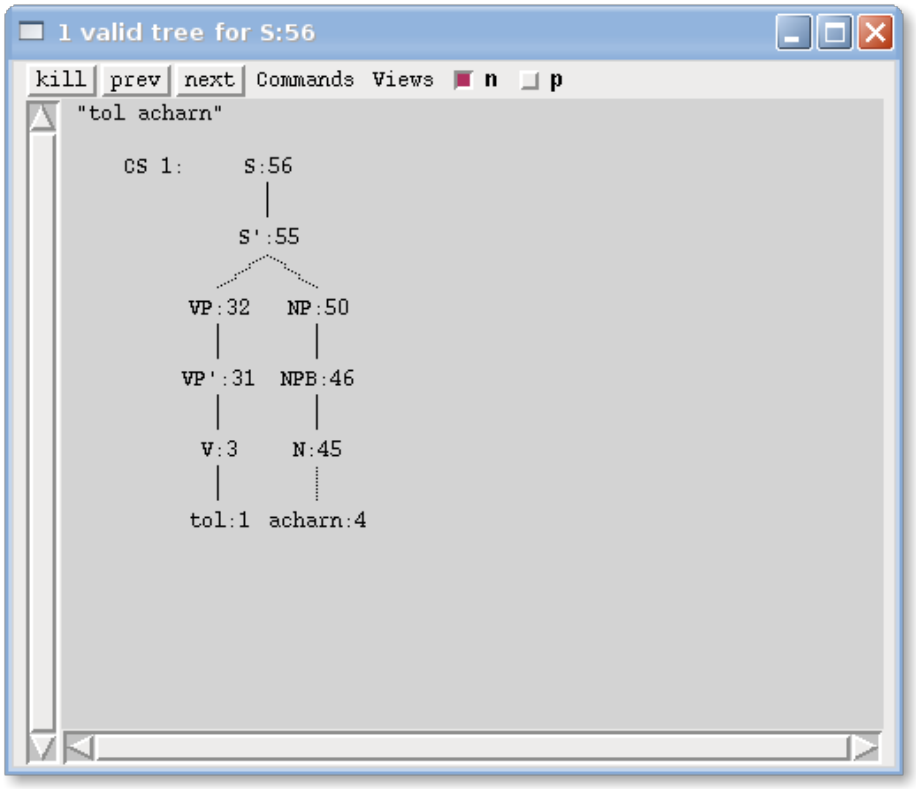


**Fig. 18.** sentence: tol acharn.

**Transitive: OVS** Sentences with pronominal object become OVS. The OVS structure shares its implementation with OV. To overcome the problem of the potentially missing subject (which may be omitted in Sindarin) we use a primitive term as denoted by 'e' in the following rule.

```
S' --> "OV(S)-type sentences"
   NP:(^OBJ)=!; VP { NP:(^SUBJ)=! | e:(^SUBJ PRED)='PRO' }.
```

Example: le linnon im Tinuviel.

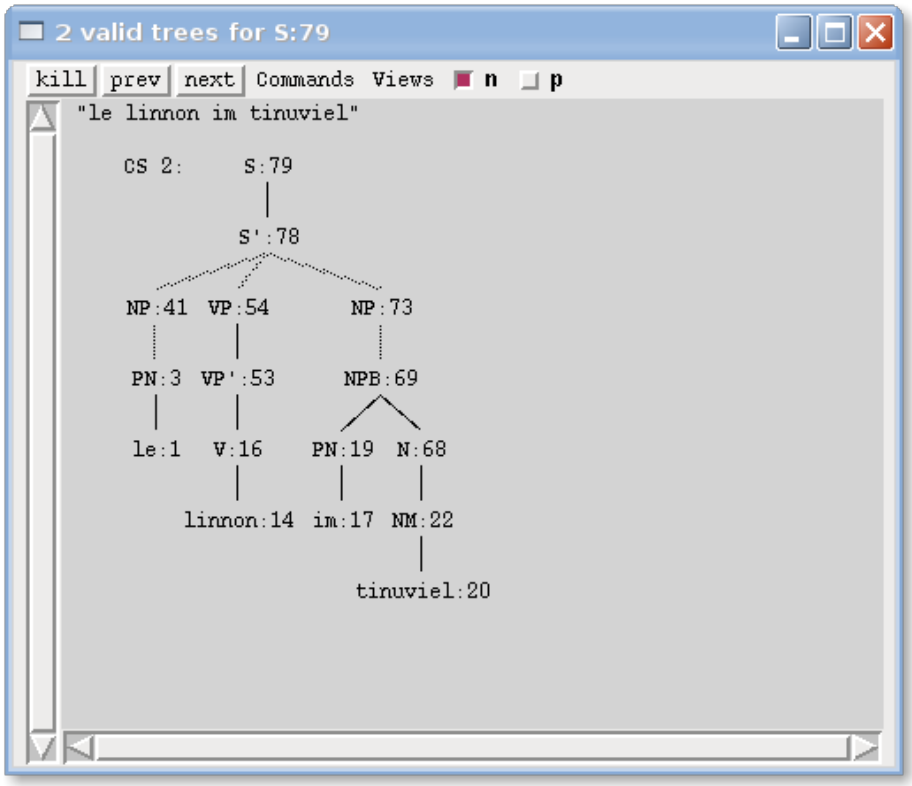


**Fig. 19.** sentence: le linnon im Tinuviel.

**Sentences With Omitted Subject Pronouns** Sentences with a first or second person pronoun subject may omit the pronoun, creating a V(O) structure.

*Transitive VO*

`Example: lasto beth lammen.`

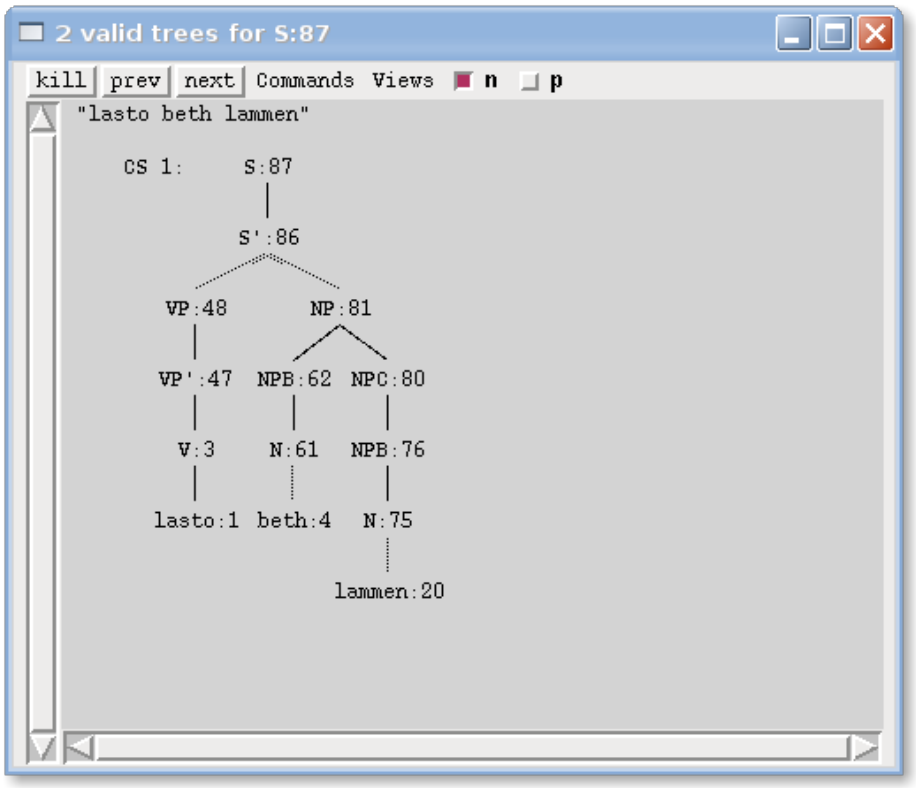


**Fig. 20.** sentence: lasto beth lammen.

*Intransitive V*

```
Example: edro.
```

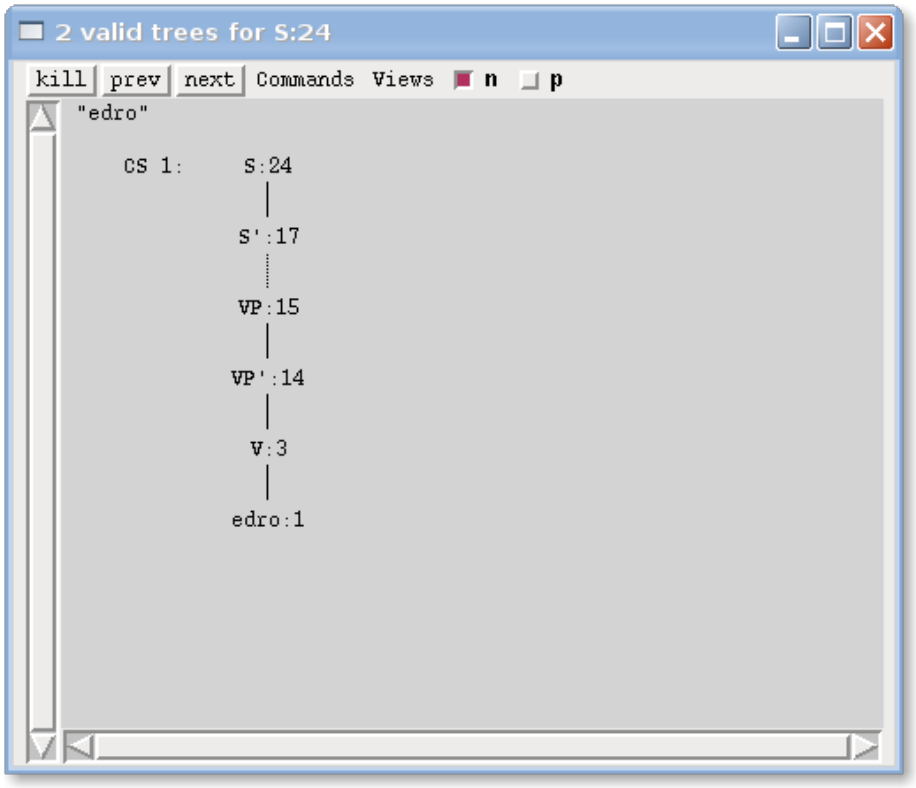


**Fig. 21.** sentence: edro.

*Transitive OV*

```
Example: le linnathon.
```

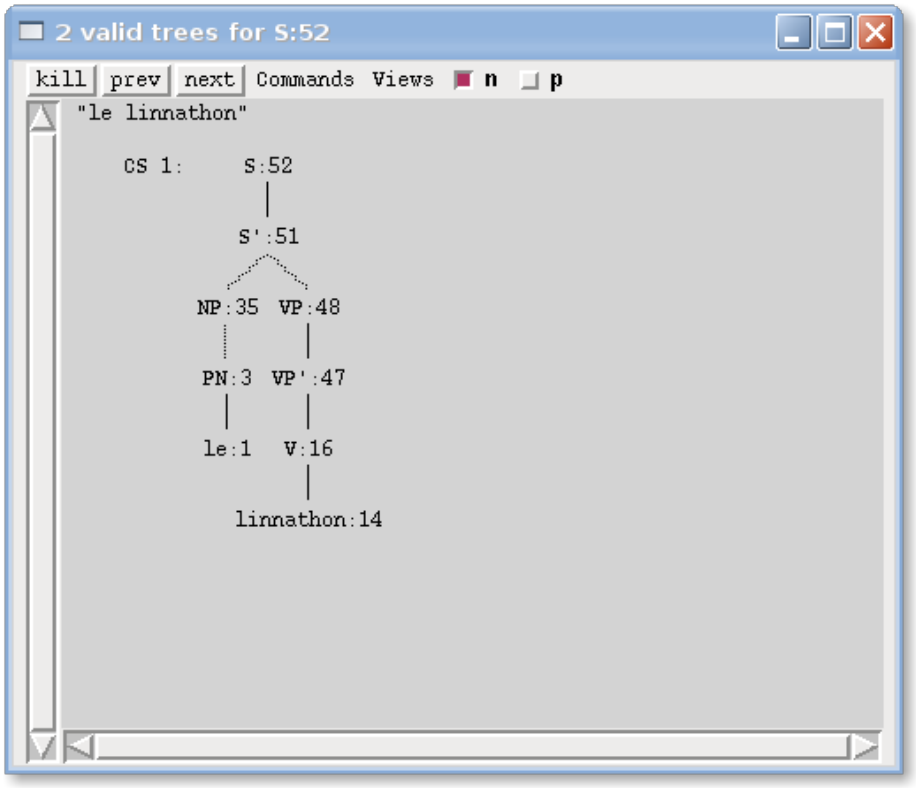


**Fig. 22.** sentence: le linnathon.

*Transitive SVO*

```
Example: Celebrimbor o Eregion teithant i thiw hin.
```

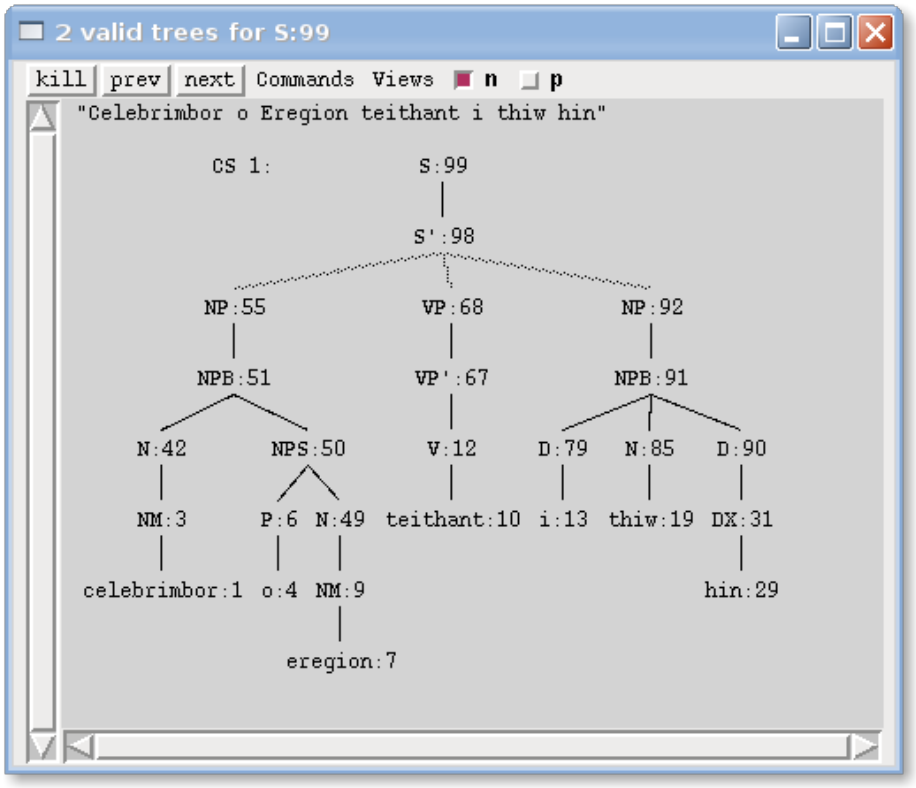


**Fig. 23.** sentence: Celebrimbor o Eregion teithant i thiw hin.

*Intransitive SV*

```
Example: guren bed enni.
```

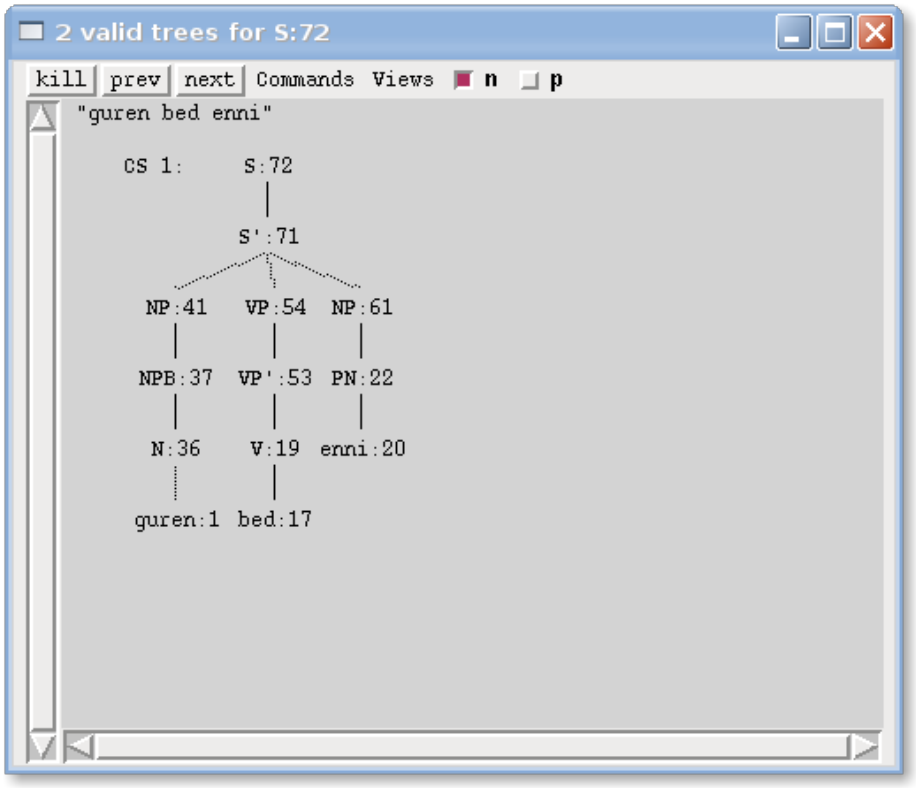


**Fig. 24.** sentence: guren bed enni.

*Transitive SOV*

```
Example: im Narvi hain echant.
```

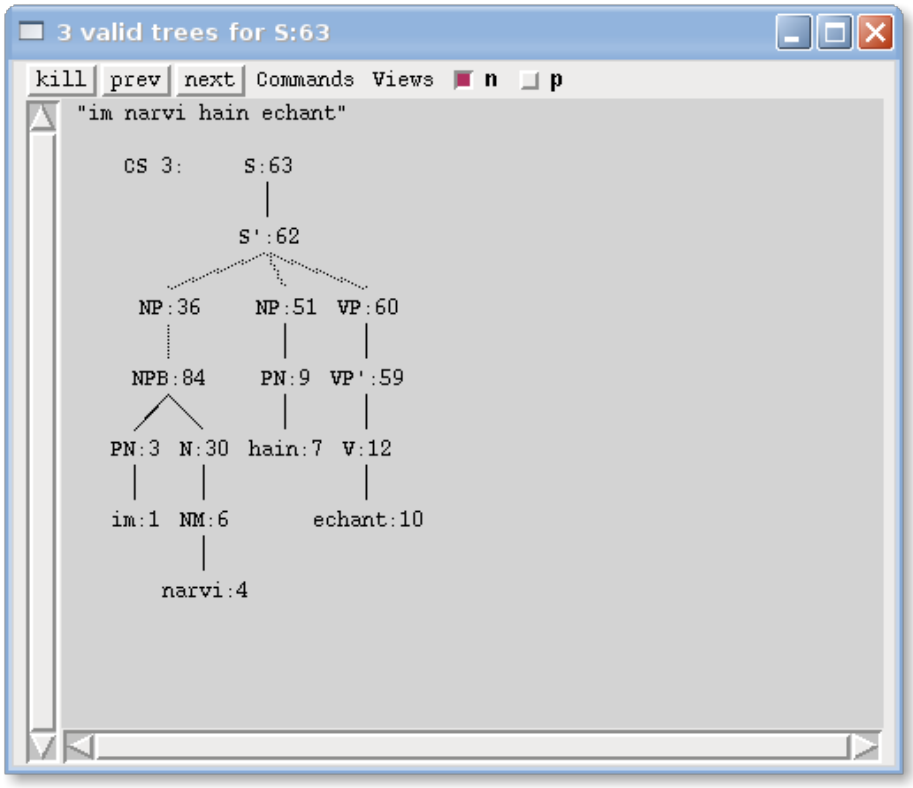


**Fig. 25.** sentence: im Narvi hain echant.

### 3.8 Adverbials

Adverbs or adverbial prepositional phrases cannot be inserted inside a noun phrase, determiner phrase or prepositional phrase. Their normal position is after the verb, but they can also appear at the beginning or end of a sentence. Our implementation wraps the root rule "S –¿ X" to allow for Adverbial placement at the beginning and end of a sentence, as well as conjuncts in between. The wrapper for the Verb-Phrase-Rule (VP), in turn, simply allows an optional adverbial placement.

```
S --> (ADV)* S' (CONJ; S')* (ADV)*.
VP --> VP' (ADV)*.
```

**Ambiguous Adverbials.** If an adverbial is after the verb at the end of a sentence it can be immediately followed by another one.

```
Example: guren bed enni.
```

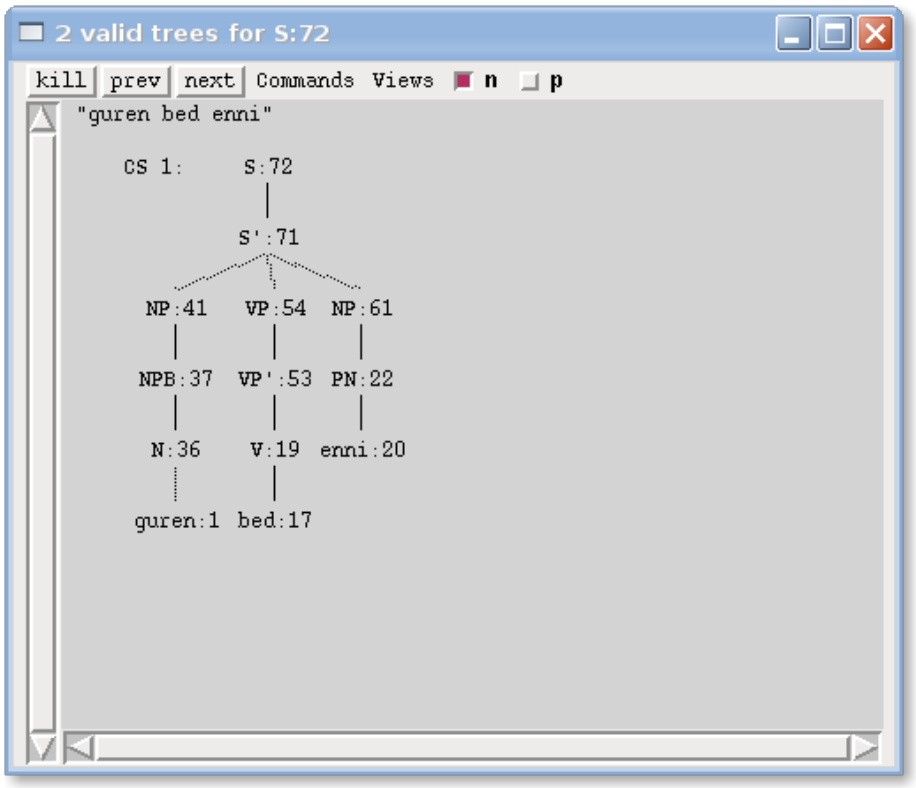


**Fig. 26.** sentence: guren bed enni.

# 4 Integrating XFST and XLE

To successfully integrate **xfst** finite state machines into **xle** a few prerequisits have to be met. The process is as follows. Upon parsing input in **xle** the input string is first run through the tokenizer to produce tokens suitable for lookup in the grammar. The tokens are then looked up in morphological analyzer to retrieve the stem and morphological tags. Then the stem and tags are checked in the **xle** lexicon to generate the f-structures and c-structures and apply the lexical rules. This requires that the tokenizer seperates tokens with an @ sign as token boundary and that the morphology provides lookup of the tokens generated by the tokenizer. It is also important to mention that **xle** cannot handle flag diacritics. Therefore all flags in the morphology network must be eliminated using the *eliminate flag* command. The surface strings must not contain multi-character symbols as they are automatically returned as being part of the lexical string. This is awkward behavior that is unfortunately not documented anywhere. Also the part-of-speech tags returned by the morphology should be unambiguous and benefit the construction of lexical rules and feature constraints. If the tags are awkwardly chosen designing the grammar can become complicated and tedious. Three components are needed to combine *xfst* and *xle*.

- Morphology Configuration
- Lexical Entries
- Sublexical Rules

They are discussed in the next section in more detail.

## 4.1 Morphology Configuration

To inform **xle** of the presence of a tokenizer and morphological analyzer the "MORPHOLOGY" option in the "CONFIG" section must be set. In addition a new "MORPHOLOGY" section must be introduced that states the path and files to use for tokenization and morphological analysis. Figure 27 shows the configuration used for this project. **xle** provides commands to test the correct

```
TOY   SINDARIN   MORPHOLOGY (1.0)

TOKENIZE:
./sindarin.tok.fst

ANALYZE:
./sindarin.morph.fst

----
```

Fig. 27. xle Morphology configuration

function of the tokenizer and analyzer. Figures 28 and 29 show the commands and output for testing the successfull integration of the tokenizer and analyzer. As can be seen the command *tokens* returns the tokens and token boundaries

```
% tokens "i-edhel"

analyzing {i-edhel}

i- @ edhel @
```

Fig. 28. Testing the tokenizer in xle

```
% morphemes {i-edhel}

analyzing {i-edhel}

i- {"+Token"|"+SgDet"}
edhel
  { "+Def"  "+Noun"  {"+Sg"|"+Gen"}
   |"+Indef"  "+Noun"  "+Sg"
   |"+Token"}
```

Fig. 29. Testing the morphological analyzer in xle

while the command *morphemes* first uses the tokenizer and then checks the tokens in the morphology network to return the stems and tags.

### 4.2 Lexical Entries

All lexical entries in **xle** have the same structure. They are composed of four parts.

1. **word.** it equals the stem or tag that is either found in the morphology or a string that should be recognized upon input
2. **category.** a part-of-speech identifier used to construct sentences using the lexical rules.
3. **morphcode.** states whether the word is found in the morphology or not. For the former the code is *XLE*, for the second use ***.
4. **schemata.** the schematas for the word, ie the features that are set or the templates that are called.

Figure 30 shows an example lexical entry for the Sindarin noun "edhel". In

```
edhel N XLE (^PRED)=%stem.
```

Fig. 30. Lexical entry for the noun edhel

addition to the word, all the tags that can co-occur with the stems must have lexical entries themselves. The tags will set or require specific features to form a legal phrase. Table 3 shows the categories for all possible tags and their schemata. For the sake of simplicity and to provide a lower level of documentation we set

| Tag | Category | Schemata |
|---|---|---|
| +Indef | N_DEF | (^DEF)=- |
| +Def | N_DEF | (^DEF)=+ |
| +Noun | N_SFX | (^NTYPE)=noun |
| +pNoun | N_SFX | (^NTYPE)=pnoun |
| +iNoun | N_SFX | (^NTYPE)=inoun |
| +cNoun | N_SFX | (^NTYPE)=cnoun |
| +Sg | N_NUM | (^NUM)=sg |
| +Pl | N_NUM | (^NUM)=pl |
| +Gen | N_NUM | (^CASE)=gen |
| +Ath | N_NUM | (^NUM)=pl |
| +Hoth | N_NUM | (^NUM)=pl |
| +SgDet | D_SFX | (^DEF)=+ (^NUM)=sg (^NTYPE)~=c inoun |
| +SgIDet | D_SFX | (^DEF)=+ (^NUM)=sg (^NTYPE)=c inoun |
| +PlDet | D_SFX | (^DEF)=+ (^NUM)=pl (^NTYPE)~=c cnoun |
| +PlCDet | D_SFX | (^DEF)=+ (^NUM)=pl (^NTYPE)=c cnoun |
| +GenDet | D_SFX | (^DEF)=+ (^NUM)=gen |

**Table 3.** Tags, Categories and their schemata

aside the usage of templates. The actual implementation, however, makes heavy use of this feature.

### 4.3 Sublexical Rules

For **xle** to recognize the tags as being part of their corresponding words it needs *sublexical rules* that are composed of the categories ordered in the order of appearance in the morphology. *Sublexical rules* are formed by taking all the categories of the words and tags and adding a suffix *_BASE*. Table 31 shows a basic ruleset including the *sublexical rules* needed for nouns and determiners. At this stage, the grammar already recognizes sentences composed of a single noun or a determiner followed by a noun. This is perfectly legal in sindarin. Figure 32 shows the syntax tree for the phrase *i-edhel*. In order to show the morphemes the command *Show Morphemes* has to be chosen from the context menu inside the tree view. The actual implementation has of course a much more complex set

```
S -->NP (PERIOD).
NP  -->(D) { N | N NP }.
N  -->N_BASE (N_NORM_BASE) N_DEF_BASE N_SFX_BASE N_NUM_BASE.
D  -->D_BASE D_SFX_BASE D_NUM_BASE.
```

Fig. 31. Basic ruleset including sublexical rules

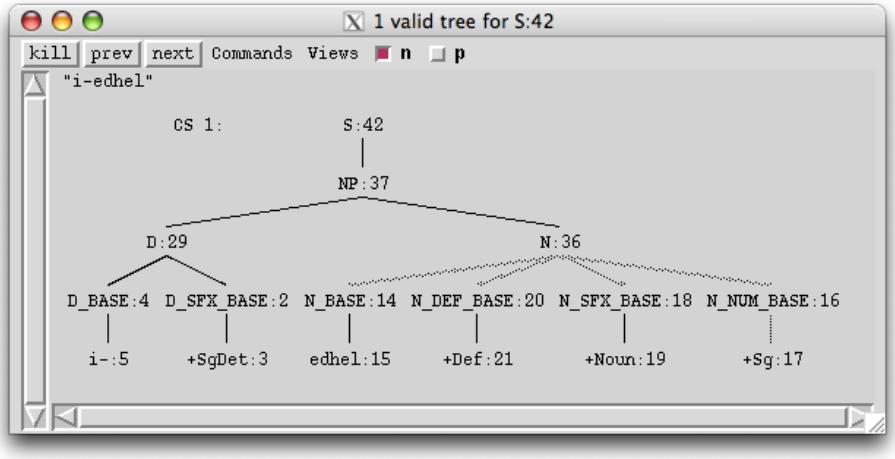


**Fig. 32.** syntax tree of the noun phrase 'i-edhel' after integration of the morphology and grammar

of rules. It recognizes almost all different kinds of phrases present in Sindarin. Figure 33 shows a more complex example.

## 5 Implementation

The file *testcases.html* contains a selection of sentences representing a majority of phrases known to exist in Classical Sindarin. Every sentence is annotated with english translation, the category and a state of implementation in the project. See section 5.2 for the content of this file. Apart from a few of these sentences all should result in at least one possible solution. In addition to these test sentences the system also recognizes all nouns that have been implemented in the Sindarin Morphology project. See the files *sindarin.lexc* and *sindarin-lexicon.lfg* for a complete list of all recognized words.

### 5.1 Known Bugs

There are some minor bugs present in the implementation which could not be tracked down due to limited linguistic and practical knowledge.

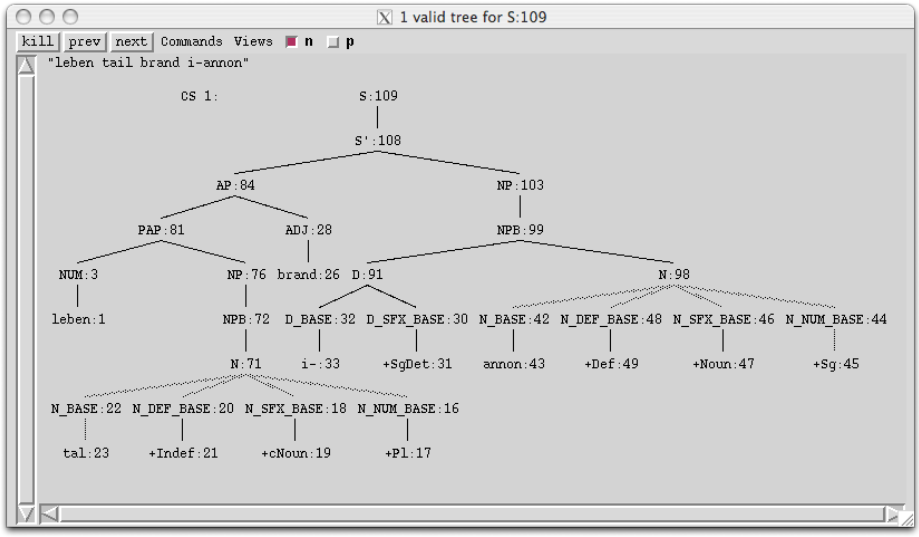


**Fig. 33.** syntax tree of test sentence 4

One major problem that arised during implementation is based on ambiguity of some words even in the same category. Especially the article *i* is problematic. Due to the morphology *i* is the shortened form of the plural article *in* if it precedes a noun starting with a consonant. However, *i* can also be an article meaning something like "which that, who". All efforts to distinguish these two cases lead to no results. The grammar could not distinguish which schema of the article is correct in the context. This is the reason why the test sentences five and six do not lead to a solution.

Ambiguity is also the reason why most of the test sentences lead to more than one solution and syntax tree.

Another problem arises with verb topicalization. The phrase *Tiro nin*, that means something like *watch over me*, can not be parsed though looking simple. The problem is the word *nin* that can either be the definite plural form of the noun *nen*, a pronoun meaning *me* or a preposition. Of these only the noun could appear, since the verb *tiro* can not be succeeded by a single pronoun or preposition. The sentence structures verb - preposition or verb - pronoun are not legal in Sindarin. No solution could be found to create a legal sentence by topicalization of an object. The same phrase is also found in sentence 17.

Other cases that do not work are vocatives (sentences 16 and 17) which have been generally disregarded.

## 5.2 Testcases



Tôl acharn. 1
'Vengeance comes.'
Category: Normal VS(O)-type sentence
Intransitive: VS
One of the two trees will (wrongly) assume a VO sentence structure. We found no way, however, to distinguish VS from VO type sentences syntactically.
Boundary cases:
- Acharn tôl.

Gurth an glamhoth. 2
'(Let there be) death to the orcs.'
Category: Noun-and-prepositional-phrase sentences
No comments.
Boundary cases:
- Gurth glamhoth.
- An glamhoth.

Aglar 'ni pheriannath. 3
'(Let there be) glory to the halflings.'
Category: Noun-and-prepositional-phrase sentences
No comments.
Boundary cases:
- Aglar pheriannath.
- 'Ni pheriannath.

**Leben tail brand** i annon. 4
'**Five feet high** (is) the door.'
Category: Adjectival sentence
Conflicting heads.
Boundary cases: Ii
- Tail brand i annon.
- Leben tail i annon.
- Leben tail brand.

*Caro* den **i innas lín**. 5
'*May* one *do* **your will**.' (= may your will be done).
Category: Normal VS(O)-type sentence
Transitive: VSO
Discarded due to vocabulary problems.
Boundary cases: None

*Tolo* i arnad lín. 6
'*May* **your kingdom** *come*.'
Category: Normal VS(O)-type sentence
Intransitive: VS
Discarded due to vocabulary problems.
Boundary cases: None

**Le** *linnon* im Tinúviel. 7
'I Nightingale, *sing* to **you**.'
Category: Normal VS(O)-type sentence
Transitive: OVS
Problem with OVS vs. SVO order. See #1.
Boundary cases:
- Le linnon.
- Le linnon Tunúviel.

*Lasto* **beth lammen**. 8
'*Listen* to **the word of my tongue**.'
Category: Sentence with omitted subject pronouns
Transitive: VO
No comments.
Boundary cases: None

*Edro*. 9
'*Open!*.'
Category: Sentence with omitted subject pronouns
Intransitive: V
No comments.
Boundary cases: None

**Le** *Linnathon*. 10
'I *will sing* to **you**.'
Category: Sentence with omitted subject pronouns
Transitive: OV
The parser will (in one of the two trees) say that 'le' is the subject, while it's in fact the object. It's difficult, however, to distinguish those cases since the ommited 'I' could be at the exact same position (see SV, case #12). Problem is analog to #1.
Boundary cases: None

Celebrimbor o Eregion *teithant* **i thiw hin**. 11
'Celebrimbor of Eregion *wrote* **these letters**.'
Category: Sentence with omitted subject pronouns
Transitive: SVO
No comments.
Boundary cases: None

Guren *bêd* enni. 12
'My heart *speaks* for me.'
Category: Sentence with omitted subject pronouns
Intransitive: SV
No comments.
Boundary cases: None

Im Narvi **hain** *echant*. 13
'I, Narvi, *made* **them**.'

Category: Sentence with omitted subject pronouns
Transitive: SOV

There's still a problem with I and Narvi being seperated (and becoming subject/object respectively).

Boundary cases: None

*Tiro* **nin**. 14
'*Watch over* **me**.'

Category: Verb topicalization

Discarded because
a) we found no example for Object Topicalization
b) the rule is rarely used in Sindarin

Boundary cases: None

Pedo mellon a minno. 15
'Say "friend" and enter' (or 'Let a friend speak and enter').

Category: Conjunct sentence

No comments.

Boundary cases: None

Annon edhellen, *edro* hi ammen. 16
'O Elvish gate, *open* now for us.'

Category: Vocatives (beginning of sentence)

Vocatives were generelly discarded.

Boundary cases: None

A *tiro* nin, Fanuilos. 17
'O *watch over* me, Ever-white.'

Category: Vocatives (end of sentence)

See #16.

Boundary cases: None

*Diheno* ammen i úgenh vín. 18
'*Forgive* to us our misdeeds.'

Category: Adverbials (after verb)

VS vs. VO order, see #1.

Boundary cases: None

*Edro* hi ammen. 19
'*Open* now for us.'

Category: Adverbials (ambiguous)

No comments.

Boundary cases: None

*Cuio* i pheriain anann. 20
'*May* the halflings *live* for a long time.'

Category: Adverbials (end of sentence)

Discarded due to vocabulary problems.

Boundary cases: None

Si loth a galadh *lasto*. 21
'Now flower and tree, *listen*.'

Category: Adverbials (beginning of sentence)

SV vs. OV order problem, see #1.

Boundary cases: None

## 6 Discussion

With **xle** and **xfst** Xerox provides a powerful framework for modeling languages. With transfer rules even translation between source and target languages are possible. Integrating the two systems is not difficult. However, a few things that are not well documented have to be considered. During development a lot of different sources had to be consulted to gather all the needed information (see the bibliography for references). Problems arouse especially with the token boundary character and **xfst**'s flag diacritics and multicharacter symbols. Another problem was, that the morphological analyzer designed in a previous project was not suitable for integration into **xle** due to a combined representation of articles and nouns and the usage of multicharacter symbols. It is crucial for successful and useful integration that each component is designed with the whole system in mind. If one of the components is developed as a stand-alone application later integration can proof to be be highly tedious or even impossible without rewriting major parts.

Sindarin has never been completely documented by it's creator Tolkien. All that is known about the syntax and morphology of this language was derived by professional and hobbyist linguists all over the world. Though the grammar of Sindarin can be considered complete, the vocabulary is limited to a few thousand words. This projects covers only few of them. The morphology recognizes only singular and plural determiners and nouns. Since Sindarin morphology is much more complex a lot has been disregarded due to complexity. The lfg model of Sindarin, however, is nearly complete with some minor problems. If the morphology was extended to represent the whole Sindarin morphology it should be easy to build a complete represenetation of Classical Sindarin (and even other Sindarin dialects) based on this project by making only minor changes to the grammar.

## References

1. Salo, D.: A Gateway to Sindarin. The University of Utah Press, Salt Lake City 2004.
2. Zink, T., Engl A.: Sindarin Morphology. Finite-State Morphology SS 2007, University of Konstanz.
3. Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, Paula Newman: XLE Documentation. http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html .
4. Ron Kaplan and Tracy King: Grammar Engineering. http://www2.parc.com/isl/members/thking/ling239e/.
5. Kenneth R. Beesley and Lauri Karttunen: Finite-State Morphology. CSLI Publications, 2003.