# SiMoN • Documentation

Simeon Herteis

as of February 5, 2015

# Contents

# Chapter 1

# Introduction

## 1.1 AnIta and SiMoN

This document describes the morphological parser SiMoN[1], an analyzer for the Sicilian language. It is inspired by and based on *AnIta*, a morphological analyzer for the Italian language created by Tamburini and Melandri (2012) that consists of a larger lexicon of Italian words. SiMoN currently comprises of the regular verbal morphology of Sicilian and while being available for usage as a standalone application its purpose is to act as extension to AnIta and provide Sicilian morphological paradigms for processing in parallel to Italian. SiMoN is part of the *TyDiaDy* research project. The subsequent chapter contains technical information for the setup of SiMoN. The third chapter describes the basic commands to analyze Sicilian words with SiMoN. Chapters 4 on page 11 and 5 on page 17 explain SiMoN's lexicon and rule set that form the morphological paradigms of Sicilian.

## 1.2 Relevant Terminology

A few terms specific to the field of computational linguistics will occur throughout this documentation. They will be explained below.

**parsing:** general term for analyzing and/or filtering text with specific rule sets

**parser:** application/collection of rules to analyze/filter a certain type of textual information

**POS:** **P**art **O**f **S**peech (tagging) - identifying word types (verb, noun, adverb,…) in a text

**lemma:** entry in a (linguistic) lexicon, providing the basic form of a word

**paradigm:** contains a 'list' of word forms belonging to the same word + in the case of SiMoN, a paradigm holds all inflectional forms of a lemma

[1] Acronym for **Si**cilian **Mo**rphology for **N**atural Language Processing

**FST:** **F**inite **S**tate **T**ransducer - mathematical algorithms from the theory of (finite state) automata; core technology the *two-level morphology* (see section 1.3) builds on

**HFST:** **H**elsinki **F**inite **S**tate **T**echnology - suite of applications that are based on FSTs, developedat the University of Helsinki

**XFST:** set of applications by the XEROX company implementing FSTs

## 1.3 The *two-level* principle

SiMoN (as AnIta) consists of two parts: A lexicon and a rule set. They are part of a practice of representing morphologies first introduced by Kimmo Koskenniemi (1983) that is known as *two-level morphology*. In short, the idea is to combine the morphological lexicon and morphological rules of a language into a single finite state automaton that can be implemented as computer program thanks to a specific formalism (here XFST) the two parts are written in. Such morphologies - as the name suggests - contain two levels, the *surface level* and the *internal level*. In the following paragraph these two components are explained in brief.

On the *internal level* grammatical information concerning the morphological and phonetic features is recorded in the lexicon on a word by word basis. In the case of SiMoN the lexicon contains various lemmas with basic word forms and their stems, each annotated with their word categories and other morphological features plus inflection class. These classes determine the associated inflection patterns, e.g. a conjugation pattern for verbs or a declination pattern for nouns. A conjugational paradigm of a verb can be build by appending each suffix from the pattern to its associated stem.
Since grammatical features should not be visible when generating inflected forms with the patterns an approach is required to remove them from the produced words while keeping their information intact for the generation process. In addition, many linguistic phenomena - especially those concerning phonology - can not easily be represented by the additive approach of a lexicon with inflectional patterns. This is what the second component, the rule set is used for. It contains various filters that both eliminate annotations from generated forms and manage phenomena outside the scope of the lexicon. Applying those rules to the lexicon on the *internal level* produces the cleaned up results on the *surface level*. See figure 1 on the following page for an illustration.

Another aspect of the *two-level* concept is the efficiency of the algorithm used for implementation. It allows executing the combination of patterns and application of rules in parallel and increases speed. Further details about the underlying concepts and their implementation can be found in the articles by E. Antworth (1991), L. Karttunen (2001) and the original work about *two-level* morphology (Koskenniemi, 1983).
SiMoN's lexicon and rules are constructed in a way that allows *bidirectional* access to the morphological information in order to both analyze and generate words with the lemmas available in the lexicon. The features of a lemma are displayed for linguistic analysis both when looking up or generating a word in the lexicon. The structure allowing such usage is explained in the dedicated chapters 4 on page 11 and 5 on page 17.

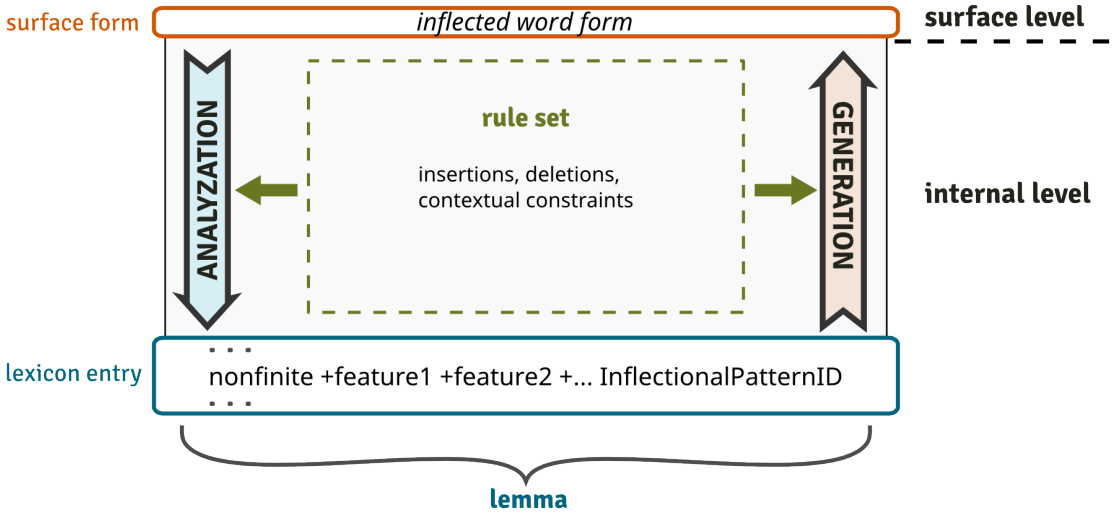


**Figure 1:** Two-Level representation

# Chapter 2

# Prepare & Compile SiMoN

## 2.1 Prerequesites

In order to use SiMoN for morphological analysis, the HFST-Suite is required.

HFST is available for MAC/UNIX and Windows platforms and can be installed according to the information provided at the project's wiki page. SiMoN is currently tested against hfst v3.8.1.

## 2.2 Using the GNU Makefile

In case you have recieved binaries of SiMoN, you can skip this chapter after installing the HFST software and take a look at the usage chapter. If you received the SiMoN source files, you can use the provided Makefile[1] for to auto-compile the binaries. Simply typing `make` in a terminal should get you set up.[2]

### 2.2.1 Make Options

The default behaviour is to compile the generator for Sicilian morphology only.

#### All In One

For convenience, there is an additional command that builds all parts of SiMoN and AnIta respectively and combines them into a single analyzer-generator pair that can parse Italian and Sicilian at the same time.

[1] The HFST-Package needs to be installed and its binaries must be present in the systems $PATH variable. Run `make check` to make sure dependencies are met.

[2] **This has been tested for Mac OSX and Linux only!** Even though GNU Make is available for Windows, users of that platform are advised to follow the instructions in the section *manual compilation*. For these the only requirement is the HFST software.

> **Note:** This needs AnIta's files to be present within SiMoN's directory.

```
make bundle
```

Given you have a copy of AnIta's sources and only want to compile those, you can override the default target by issuing the following command:

```
make target=italiano
```

#### Keep Intermediate Binaries

If you want to keep the intermediate binaries generated by conversion and combination of the ruleset and the lexicon, you can set the variable `keep_intermediates` to `yes`:

```
make keep-intermediates=yes
```

The compiled binaries can be cleaned with `make clean`.
**Caution!** This removes *any* hfst-binary created by executing the Makefile.

## 2.3 Manual Compilation

SiMoN consists of two files, a lexicon (*siciliano.lexc*) and a ruleset (*siciliano.twolc*) in XFST format. In order to convert these to HFST binaries for looking up and generating morphological paradigms, follow the steps below:

> **Note:** AnIta can be compiled in the same way. Replace *siciliano* with *italiano* in the steps 1. - 3.

#### Step 1: Convert XFST sources to HFST binaries

```
$ hfst-lexc siciliano.lexc -o siciliano.lexc.hfst
$ hfst-twolc siciliano.twolc -o siciliano.twolc.hfst
```

#### Step 2: Combine binaries to analyzer

```
$ hfst-compose-intersect siciliano.lexc.hfst \
  siciliano.twolc.hfst -o siciliano.analyzer.hfst
```

#### Step 3: Invert analyzer to function as generator

```
$ hfst-invert siciliano.analyzer.hfst -o siciliano.generator.hfst
```

### (Optional) Step 4: Unify SiMoN and AnIta

Merges the two parsers for parallel analysis of Sicilian and Italian.

```
$ hfst-union -v italiano.generator.hfst \
  siciliano.generator.hfst -o it-scn-analysator.hfst
```

# Chapter 3

# Using SiMoN

The binaries of SiMoN can be read with the various tools of the HFST-Suite to perform morphological analysis for the Sicilian/Italian language on a word basis. Below you can find some explanatory usage examples. A full list of available HFST tools can be found on the respective HFST wiki page. Be aware that not all of these tools are useful in the context of this project. For example, SiMoN supports no weighed arcs (needed for `hfst-guessify`).

## Operation of the generator and analyzer

SiMoN's lexicon and ruleset are implemented to allow a bidirectional output of grammatical information both when analyzing and generating words.

### Analyzing Words

The command `hfst-lookup` is used to look up words in hfst transducers. For analysis the *analyzer* file is loaded:

```
$ hfst-lookup it-scn.analyzer.hfst

> non
non l_non+ADV

> dormivo
dormivo l_dormire+V_FIN+IND+IMPERF+1+SING

> durmivu
durmivu l_durmiri+V_FIN+IND+IMPERF+1+SING
durmivu l_durmiri+V_FIN+IND+PAST+1+SING
```

#### 3.0.1 Generating Words

Loading the *generator* file with the lookup tool allows generating words:

```
$ hfst-lookup it-scn.generator.hfst

> l_durmiri+V_FIN+IND+FUT+1+SING
l_durmiri+V_FIN+IND+FUT+1+SING durmirò

> l_vuliri+V_FIN+COND+PRES+2+PLUR
l_vuliri+V_FIN+COND+PRES+2+PLUR vuliriavu
```

### Dump Surface Forms

The command `hfst-fst2strings` yields a list of all words that can be generated by the morphological parser. The output can also be limited to show only a given number of entries (counting from the top) as seen in the example below.

```
$ hfst-fst2strings -n 5 siciliano.generator.hfst
l_essiri+V_IRI+V_FIN+IND+IMPERF+2+SING:eri
l_essiri+V_IRI+V_FIN+IND+IMPERF+1+SING:era
l_essiri+V_IRI+V_FIN+IND+IMPERF+3+SING:era
l_essiri+V_IRI+V_FIN+IND+IMPERF+1+PLUR:eramu
l_essiri+V_IRI+V_FIN+IND+IMPERF+3+PLUR:eranu
l_essiri+V_IRI+V_FIN+IND+IMPERF+2+PLUR:eravu
l_essiri+V_IRI+V_FIN+IND+IMPERF+3+PLUR:erunu
l_essiri+V_IRI+V_NOFIN+INF+PRES:essiri
```

```
l_essiri+V_IRI+V_NOFIN+GER+PRES:essennu
l_essiri+V_IRI+V_FIN+IND+PAST+1+PLUR:fomu
```

# Chapter 4

# The lexicon

This chapter gives an overview of the general structure of SiMoN's lexicon and the different sub lexicons containing the regular and irregular verbs in particular. It closely follows the documentation of AnIta (Tamburini and Melandri, 2012).

## 4.1 Components

The Lexicon of SiMoN has a hierarchical tree structure, closely following that of AnIta. At present, SiMoN covers Sicilian morphology paradigms for regular verbs, according to the grammar descriptions of J. K. Bonner (2001).

There are three sections to the lexicon:

1. The so called *Multichar Symbols*
2. The root lexicon
3. The sub lexicons

### 4.1.1 The Root Lexicon

While the verbal inflection patterns are documented in their dedicated section on page 15 to 16, the actual Sicilian verbs are listed in the *Root* lexicon:

```
l_ammazzari:ammazz+V_ARI SufVerbAri ;
l_cantari:cant+V_ARI SufVerbAri ;
l_durmiri:durm+V_IRI SufVerbIri ;
```

These entries are called lemmas. Each lemma contains a verb in its nonfinite form[1]. Additionally, the lemma records the stem of the word (e.g. *cant* for *cantari*, to sing), its word

[1] prefixed with *l_* for easier discrimination from the generated inflected forms of that verb

class (verb, either ending in *ari* or *iri*) and the sub lexicon containing the inflectional suffixes of that word class (*SufVerbAri* and *SufVerbIri*)

The vocabulary of SiMoN is taken from Bonner (2001) and comprises of around 60 regular Sicilian verbs. Additionally a verb list automatically extracted from the Sicilian Wikipedia pages is included[2].

### 4.1.2 Multichar Symbols

The *multicharacter symbols* can be seen as tags used to indicate the respective *POS* type and other grammatical informations of an entry in the lexicon. Examples are `+V_ARI, +SING, +PLUR`. They denote a verb ending in the suffix *ari* and the number of a word (singular or plural). *multichar symbols* also act as variables in combination with the ruleset definitions. In the context of the lexicon however, they serve as annotation tags containing grammatical and morphological information.

#### Symbol Meanings

Below you find the *multichar symbols* present in SiMoN explained:

| symbol | meaning |
|---|---|
| **+V_ARI** | has suffix *ari* |
| **+V_IRI** | has suffix *iri* |
| **+V_FIN** | finite form |
| **+V_NOFIN** | nonfinite/gerund form |
| **+V_PP** | past participle form |

**Table 1:** *multichar symbols* for verb forms

| symbol | meaning |
|---|---|
| **+SING** | singular |
| **+PLUR** | plural |
| **+1:** | first person |
| **+2:** | second person |
| **+3:** | third person |

**Table 2:** *multichar symbols* for gender and number

[2]The verbs from Wikipedia have been picked according to a very basic pattern, matching regular inflectional suffixes. It is very likely that irregular forms are contained in this list as well.

| symbol | meaning |
|---|---|
| **+INF** | infinitve |
| **+IND** | indicative |
| **+SUBJ** | subjunctive |
| **+COND** | conditional |
| **+IMP** | imperative |

**Table 3:** *multichar symbols* for modes

| symbol | meaning |
|---|---|
| **+PART** | participle tense |
| **+GER** | gerund |
| **+PRES** | present tense |
| **+IMPERF** | imperfect tense |
| **+PAST** | past tense |
| **+FUT** | future tense |

**Table 4:** *multichar symbols* for tenses

| symbol | meaning |
|---|---|
| **+GLI** | stem with *glide* |
| **+VEL_A** | velar stem ending |
| **+VEL_Y** | velar stem ending |

**Table 5:** special *multichar symbols* (*phonetics*, see section 5 on page 19)

The complete manual to *multichar symbols* and their syntax conventions can be found in the HFST-Documentation.

### 4.1.3 Sub Lexicons

A *sub lexicon* or *continuation class* (CCl) contains the inflectional suffixes of a word. Those suffixes are provided as a list of word endings each annotated with grammatical information like *gender*, *number*, *tense* or *mode*, marked by its equivalent *multichar symbol* respectively.

Each suffix entry additionally contains a *multichar symbol* indicating to which word type it belongs. The sub lexicon for a verb is the corresponding conjugation pattern with the respective tenses of different modes for first, second and third person. Figure 2 provides a visual representation of the lexicons structure.

Combining the stem of a word from the lemmas with the appropriate CCl yields the inflectional paradigm of that word. The generated forms of that paradigm are called *surface forms*. A surface form contains no *multichar symbols*, all grammatical information is suppressed through the rule set. On the '*surface*' of the lexicon (the actual inflected verb), the internal grammatical representation is not visible. Figure 3 on the next page shows the surface form generation for 1st person plural of the Sicilian verb *to sleep* in conditional mode and present.

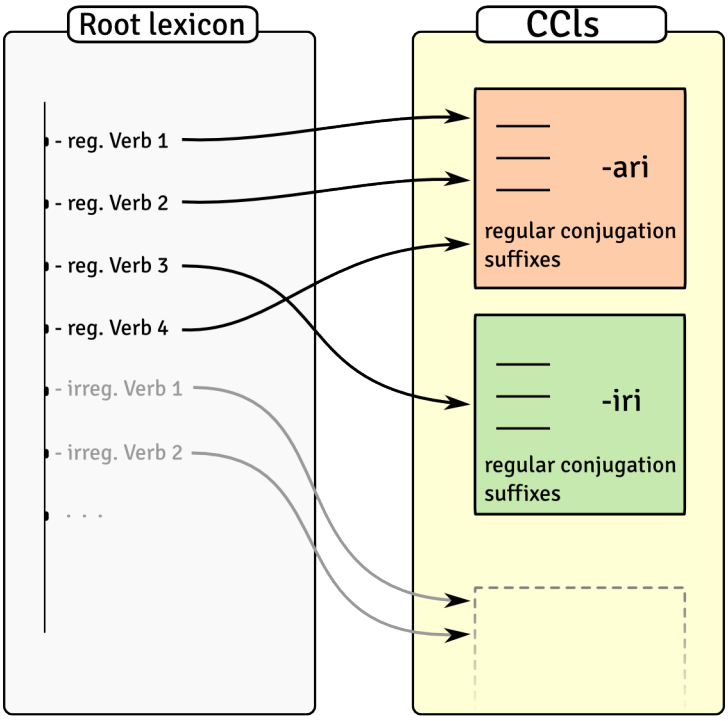


**Figure 2:** Lexicon Structure

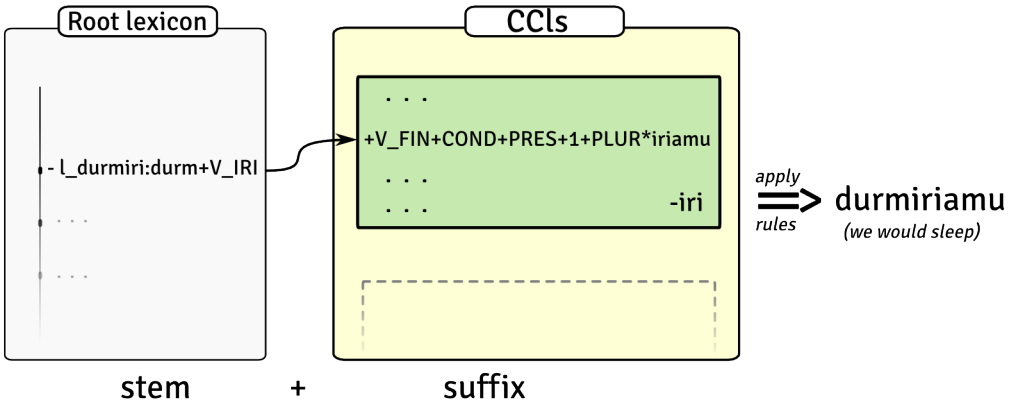


**Figure 3:** Generating a surface form

## 4.2 Regular Verb Paradigms

The special sub lexicons (or CCls) for Sicilian regular verb forms created for SiMoN are described in sections 4.2.1 through 4.2.5.

### 4.2.1 SufVerbAri

Contains the suffixes of regular verbs ending in **ari** for the tenses of the modes infinitive, indicative, subjunctive, conditional, and imperative as well as participle forms.

### 4.2.2 SufVerbIri

Same as *SufVerbAri* but for the verbal ending **iri**.

### 4.2.3 SiriEssiri

This is in fact a CCl of a frequent *irregular* auxiliary verb. Since it occurs in the two forms *siri* and *essiri* that both have the same suffixes, their lemmas both point to this special CCl to avoid writing the same paradigm in the root lexicon twice.

### 4.2.4 rscIri

A group of special CCls that only apply to regular verbs that have the suffix **iri** *and* their stem ending in *isc*. Words in this category basically have two (or more) stems each of which only occurs in a certain mode or tense of the conjugation paradigm. The suffixes stay the same while the different stems go with different sections of the paradigm that do not overlap. Currently, there is only one lemma in the lexicon to which this property applies to: the verb

*finiri*. It branches into three stem versions: `finisci~`, `finisc~` and `fin~`. A sample of its CCls can be found below.

```
l_finiri:finisci+V_IRI rscIri ;
l_finiri:finisc+V_IRI rscIri2 ;
l_finiri:fin+V_IRI rscIri3 ;

LEXICON rscIri

    +V_FIN+IND+PRES+1+SING:+IND+PRES+1+SING*u #;

    +V_FIN+IND+PRES+3+PLUR:+IND+PRES+3+PLUR*unu #;

LEXICON rscIri2

    +V_FIN+IND+PRES+2+SING:+IND+PRES+2+SING*i #;
    +V_FIN+IND+PRES+3+SING:+IND+PRES+3+SING*i #;

    +V_FIN+IND+PRES+3+PLUR:+IND+PRES+3+PLUR*inu #;
```

The stem variation `finisci~` only applies to the forms of *1st person singular* and *3rd person plural* while `finisc~` is used for *2nd* and *3rd person singular* plus *3rd person plural*. The CCl `rscIri3` contains all the remaining tenses and modes and is almost an exact copy of the regular CCl `SufVerbIri`.

### 4.2.5 Inter-root entries

The frequent irregular auxiliary verbs *aviri* (to have) and *fari* (to do/make) have their paradigms listed entirely in the root section since the irregularity prevents effective compartmentalization in CCls.

# Chapter 5

# The rule set

As mentioned before, SiMoN has a lexicon and a rule compound. While the former contains all the necessary lemmas and patterns to generate inflectional paradigms, the latter determines how the *multichar symbols* are treated and additionally takes care of accounting for particular morpho-phonetic phenomena like gemination in the generation process applying to certain surface forms. The *surface forms* introduced in the previous chapter are built from the lexicon with the use of these rules. For technical details regarding the implementation as FSA, see (Karttunen and Beesley, 2001).

## The rule set sections

The rule set of SiMoN is divided into specific parts, just as the lexicon. As of now SiMoN itself formulates no custom rules, the included rules file is a lightly adjusted copy of AnIta's rule set. The respective components are described below. A complete introduction to writing rule sets can be found in the official HFST documentation.

### Alphabet

The alphabet section contains a list of letters valid for the used language. In case of Sicilian this includes the letters *a* - *z* plus accented vowels like *à*, *â*, *è*, *é* and so on. All available *multichar symbols* are contained in the alphabet as well. They are listed as two-layer pairs that determine `0` as emission for each of them. An exemplary line from SiMoNs alphabet:

```
+V_ARI:0  +V_IRI:0 +V_FIN:0 +V_NOFIN:0 +V_PP:0
```

The line above filters all verbal features and eliminates the symbols `V_ARI`, `V_IRI`, etc. from the final surface form of a verb. All the other *mulitchar symbols* are included like this as well, resolving them all to the symbol `0`.

### Sets

The sets defined for SiMoN are bags of letters, one containing all vowels the other containing all consonants of the alphabet. These sets are used in the context rules and simplify accessing single letters by their type.

### Definitions

The definitions in the rule set could be described as categories for the *multichar symbols* that represent grammatical information. They allow grouping specific information together.

**A few examples:**

- the group `NonVerbs` contains all word classes but verbs

  `+NN: +NN_P: +ADJ: +ADJ +ADV +PRON`

- included in `Verbs`, all the available verbal annotations are included:

  `+V_ARI +V_IRI +V_FIN +V_NOFIN +V_PP`

- any tags labeling different modes for suffixes in the CCls are included in `Mode`:

  `+INF +IND +SUBJ +COND +IMP +PART +GER`

The remaining definitions for `Head`, `Conj`, `Tense`, `Gender`, `Number` and `Modif` are composed analogously. Referring to these groups within in context sensitive rules allows the construction of complex constraints.

### Context rules

Context rules check for specific conditions in the context of a character and execute certain actions if these conditions are met. The general form of these rules looks like this:

```
S:R =>,<=,<=>,\<= [LH] _ (OPT) [RH]
```

The `_` character signifies the center of context in which the specified replacement should be applied. For better readability the different parts are labeled with abbreviations:

**S:** symbol that is going to be replaced (eg. a vowel or a specific *multichar symbol*)

**R:** symbol replacing `S`

**LH:** left hand context (eg. a consonant), may also be a set of characters

**OPT:** optional context symbols, containing *multichar symbols* or sets of them (definition names)

**RH:** right hand context

The comma separated arrow characters signify the available context constraints:

**=>:** specifies contexts in which provided replacement is valid for given symbol
**<=:** constraints certain replacement to specific context
**<=>:** requires exact match for symbol and context
**\<=:** excludes symbol replacement from specified context

Let's take a look at one of the context rules in SiMoN (originally from AnIta):

```
"Insert h for c|g with velar = a - 1"
   +VEL_A:h <=> [c | g] _ (NonVerbs) (Gender) (Number) (Modif) *: [e | è | é] ;
```

The above rule is valid in the context of a lemma including the *multichar symbol* `+VEL_A` that is preceded by either `c` or `g` and has its stem followed by the vowels `e`, `è` or `è`. The definition names within the braces are optional. Depending on whether a verb or a noun is matched with this particular rule, the noun feature groups `NonVerbs` and `Modif` are skipped. Upon execution this rule replaces the `+VEL_A` with the letter `h` and ensures the pronounciation remains intact.
For example, the lemma entry for the adjective *comico*[1] (comical) for *feminine plural* (without the infinitive part) is: `comic+VEL_A+ADJ+FEMM+PLUR:*e`. Applying the rule produces the grammatically correct form *comic**h**e* instead of *comice*.

[1]Since the rule only applies to non verbs and SiMoN currently contains only verbs, the example is taken from AnIta.

# References

Evan L. Antworth. 1991. Introduction to two-level phonology. In *Notes on Linguistics. Summer Institute of Linguistics*, pages 4–18. SIL; SIL, Inc., May.

J. Kirk Bonner. 2001. *Introduction to Sicilian grammar*. Legas, Brooklyn, NY, editions.

Lauri Karttunen and Kenneth R. Beesley. 2001. A short history of two-level morphology. August. Presented at the ESSLLI-2001 Special Event titled "Twenty Years of Finite-State Morphology."

Kimmo Koskenniemi. 1983. *Two-level morphology: a general computational model for word-form recognition and production.*Publications (Helsingin yliopisto. Yleisen kielitieteen laitos). University of Helsinki, Dept. of General Linguistics, editions.

Fabio Tamburini and Matias Melandri. 2012. AnIta: a powerful morphological analyser for Italian. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Mariani. Joseph, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul; Turkey. European Language Resources Association (ELRA).