Rune

A computer program for interpretation of rune stones

Sigurd, Bengt; Dahl, Johan

1997

Document Version:
Publisher's PDF, also known as Version of record

Lund University, Dept. of Linguistics
Working Papers 46 (1997), 217–231

Rune: A computer program for interpretation of rune stones

Bengt Sigurd and Johan Dahl

Introduction and abstract

The language on Swedish rune stones written in the 16 character alphabet (*futhark*) constitutes a very special fragment of the Swedish of its time (about 800-1100). Because of the ritual character of the inscriptions the language is rather standardized. The greatest variation is in proper names. As has long been noted the typical formula is (in English translation): *N raised this stone after M his P*, where *N* and *M* are personal names and *P* is a kinship term. Additional sentences may state where the person died, e.g. *He fell in Greece.* Depending on the success of the new religion the formula *God help his soul* is sometimes also added (for safety). There are about 3000 rune stones with text of this type.

The purpose of this paper is to describe a computer program which is able to interpret such typical rune stone texts and translate them into Modern Swedish. In a first step the runes are transliterated into Latin letters. The second step is the parsing of the transliterated text resulting in a functional-semantic representation showing the analysis in terms of subject, predicate, objects and adverbials (functional roles) in addition to word meanings according to Swetra grammar. In a third step this functional representation is then used as an intermediate language (interlingua) in an automatic translation into Modern Swedish. The program may also be run in the other direction translating Modern Swedish into runes. Some stones are offered for demonstration.

The program Rune is based on the grammar developed in the project Swetra (Swetra Grammar; see Sigurd 1994). It has been implemented in the programming language LPAProlog and can be run on any Apple computer which supports this Prolog. It can be transported to other Prolog variants and other computers. The program can be demonstrated on demand at the Department of Linguistics, Lund University. The description in this paper will

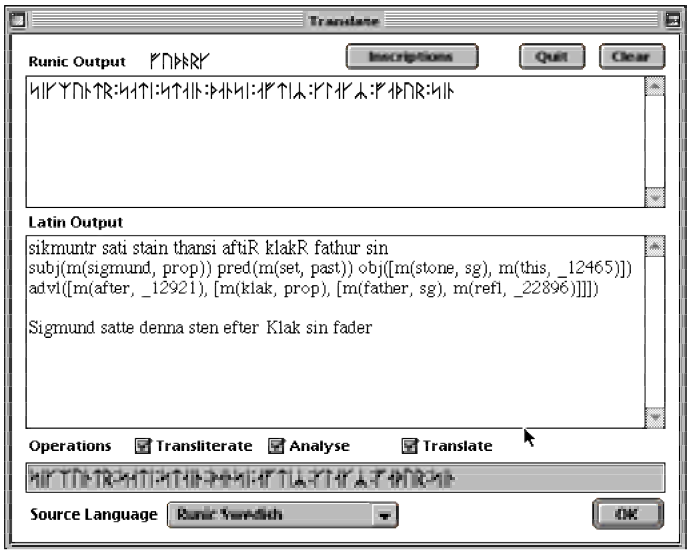

Figure 1. The main window of Rune showing transliteration, analysis and translation to Modern Swedish of the Dagstorp inscription.

be given with a view to linguistic readers rather than computer science readers. Some of the grammatical problems are discussed. Knowledge in the formalisms of generative grammar and the programming language Prolog is an advantage for the reader.

The program Rune may be used as an aid in the interpretation of rune stones or as a pedagogical tool for those who want to learn the runes or the language of the rune stones. There is a convenient interface with several windows and menus offering a choice of input language and a choice of operations: transliteration, grammatical analysis, translation (see figure 1).

The rune stones

As the Scandinavian nations were not established in these old days around 1000 and the Vikings spoke more or less the same language (tongue) the language described here could have been called Runic Norse, but as most of the stones discussed belong to what is nowadays called Sweden the term Runic Swedish has been preferred. It should be clear, of course, that the Vikings did not only speak in the style used on the rune stones, but the rune inscriptions are almost the only evidence we have and are therefore an important key to Old Swedish. The rune inscriptions written in the 24 character alphabet are generally older and more difficult to interpret. They are not treated in this paper.

Scholars have taken an interest in runes since the 17th century (Bureus), but the modern scientific approach started at the end of the 19th century

when the problems of the origin of the runes were discussed in the light of new historical-comparative methods by Wimmer, Bugge and von Friesen. The most elaborate documentation of Swedish rune inscriptions is the series *Sveriges runinskrifter* published by the Swedish Royal Academy of Letters (Kungl. Vitterhets Historie och Antikvitets Akademien) for each region separately since the beginning of the 20th century. It includes detailed discussions of all linguistic, archeological and historical matters. Similar series were started in Denmark and Norway. The rune stones of Skåne are treated in the Danish works.

A shorter survey including the most interesting stones is *Runinskrifter i Sverige* (1963, 1976) by the famous rune specialist Sven B. F. Jansson. Lars Rask's *Runläseboken* (1990) is a pedagogical presentation of runes and rune inscriptions including lessons in rune reading. The stones mentioned by Rask have been taken as a representative set and some of them are included in the demonstration menu (se print-outs).

Writing

The runic language treated here is the language written by the 16 character runes. The set of runes allowed in the system Rune are presented in a special window including word space : (colon). Pointed ('stung') runes for e.g. *e* distinguishing it from *i, p* distinct from *b,* are generally not accepted by the transliteration decoder. The Prolog program cannot mix runic writing with Latin writing in the same window. The grammar rules operate on transliterated texts.

The grammar will generate and analyze Runic Swedish spelled with transliterated runes. There are transliteration rules to decode the runes. The runes are written using the special runic font called *Bryggen*.

Table 1 presents the runes, the transliteration used and the phonological equivalents generally assumed. As is well known, several phonological distinctions, e.g. consonant voice and minor differences in vowel quality are disregarded by the 16-rune alphabet. The values of some runes are dubious, but this is of little importance for our project. We will not dwell on the writing problems in any detail and give no rune variants.

Only these runes are used in the program and when writing the input text these letters must be picked from a table in a special window. If other forms are met on a stone – and there are many variants – they must be identified with and rendered by one of these in order to be processed by our program.

Table 1. The runes and their transliteration.

Runes	*transliteration*	*phonetic values*
F	f	f, v
U	u	u, v(w), o, y, ö
›	th (thorn)	T (voiceless), dh (voiced)
ᛆ	a(n)	(nasal) a/o
r	r	r
k	k	k, g
h	h	h
n	n	n
i	i	i, j, e
a	a	a, ä, e
ß	s	s
T	t	t, d
b	b	p, b
m	m	m
l	l	l
y	R	R (palatal *r* < *z*)

Note that the thorn rune › is transliterated with *th* and R is used for the ʒ rune.

Nasals *(n)* were not generally marked (latent) in the writing resulting in e.g. *sikmutR* for *sikmuntR (SigmundR)*, *buati* for *buanti (buandi)*, *iklat* for *England*. The length of sounds was generally not marked in runic writing (e.g. by double letters).

The spelling of words was, of course, not standardized as in our modern national languages with a long orthographic tradition. In order to cover some of the variation some alternative spellings have been accepted in some lexical rules, e.g. *stain/stin* ‘stone’, *thansi/thasi* ‘this’, *raisthi/risthi, raisti, risti* ‘raised’, *satu/sautu* ‘set’, *thaiR/thaR* ‘they’, *aftiR/iftiR/uftiR/aft.* Further variants can easily be added to the grammar to increase its coverage.

The Prolog program interprets capital letters as variables and proper names can therefore not be spelled with initial capitals (unless surrounded by ' ').

The program and the grammar

The program presented is an experimental variant allowing variable input and operations. The input language may be in 16 character runes, which may then be transliterated by one operation. But one may alternatively input a rune text in Latin letters. Another operation is grammatical analysis of the transliterated text, which results in a functional representation with word meanings. A third operation is translation resulting in Modern Swedish. It is also possible to input Modern Swedish, analyze it to get the functional representation and translate it into a runic text. There are two grammars and lexicons in the system. They are bidirectional and can be used both for analysis and generation.

The interface (see figure 1) includes one window for Latin text, one window for runic text. There are menu windows for the choice of input language and operations. A window with a tablet of 16 runes and space (:) is available when the input language is set at Runic Swedish. A set of inscriptions is available for demonstration.

The syntactic patterns and the lexicon used are based on selected examples presented in surveys of rune inscriptions (see references). The program has ad hoc ways of handling some idiosyncracies of the rune inscriptions. A lacking object will be interpreted as ‘stone’ in sentences equivalent to *Ulf raised after Asmund*. A lacking subject will be interpreted as a pronoun (*he* or *they* depending on the agreement) in sentences equivalent to *died in London*, *fell in the east.* If required to interpret a rune inscription with unknown words the program will offer a solution where unknown words in nominal positions will be interpreted as proper names.

The grammar rules

The sentence is the basic unit of the grammar. Word space is marked by a colon (:), one of the markers used on the stones. The grammar described here does not generate nor analyze coordinated sentences. How this can be done can be seen in Sigurd 1994. But our grammar covers coordinated noun phrases, which are quite frequent on rune stones (e.g. *Tuke auk Biurn risthu stain thansi)*.

The typical runic sentence includes an *np* which has the functional role of subject (e.g. *Tuki;* in the nominative), a finite verb (e.g. *risthi* ‘raised’; number sg), an *np* which has the functional role of object (*stain thansi* ‘stone this’; acccusative case) and a prepositional phrase which plays the functional role of adverbial (*aftiR Tuma*; accusative case).

The first sentence rule below states (when used in analysis) that if we find a noun phrase with the semantic value *N1* in the nominative (*agr(nom,Nu,_)*) followed by a finite verb *V* agreeing with the preceding noun phrase as to number (*agr(_,Nu,_)*) and a following *np N2* in the accusative case (*agr(acc,_,_)*) and possibly an adverbial phrase this will get the functional analysis shown last to the left of the arrow *(subj(N1),pred(V),obj(N2),advl(A1))*. The surface case marking has disappeared from this representation and only the semantic values of the words are accounted for. The order of the functional roles within the square brackets is arbitrary, but standardized in this way in Swetra grammar.

Following Swetra grammar the mode (declarative, question, imperative, optative) of the sentence is shown in the first slot after the top Prolog predicate *s(entence)* and the semantic value of the topicalized (first) constituent of the sentence shown in the second slot (in this case the subject, *N1*). The values of the constituents represented as capital letters are derived by the later phrase rules.

If read in a generative (synthetic) way the rule states that a *d(eclarative)* sentence with topicalized subject and the functional representation found within square brackets in the third slot can consist of an *np* in the nominative agreeing with the following finite verb as to number, the finite verb and an *np* in the accusative, followed by an adverbial phrase.

```
s(d,N1,[subj(N1),pred(V),obj(N2),advl(A1)]) -->
      np(agr(nom,Nu,_),N1),
      vfin(agr(_,Nu,_),V),np(agr(acc,_,_),N2),advp(A1).
```

Note how agreement is handled. A value cannot be unified with a distinct other value, e.g. *nom* not with *acc*, but a value can be unified with an identical value and an unspecified value: _. The subject noun must be a nominative form which is shown in the agreement complex *agr(nom,Nu,_)*. The number value of the subject (*sg* or *pl*) is given by the variable *Nu* to be used to control verb agreement. There are no requirements for a certain gender in the agreement between subject and verb. If *Nu* has the value *sg* in the subject, it must have the same value (or no value _, as for modal verbs) in the verb agreement slot; if the value of *Nu* is *pl* in the subject the verb must have this value to (or no value). This takes care of the variation between e.g. *risti (sg)* and *ristu (pl)*, *fil (sg)* and *filu (pl)*.

The good thing with this formalism (Definite Clause Grammar, DCG) is that it can be run directly as a program in Prolog provided certain phrase and lexical rules are also implemented. If we write to the program:

```
s(M,T,F,[suin,risti,stain,thansi,aftiR,ulf],[]),
```

the program will ‘solve’ the variables M, T, F giving,

```
M=d, (The mode: declarative)
T= m(suin,prop), (The topic)
F=[subj(m(suin,prop),pred(m(raise,past)),obj([m(stone,sg),m(this,_)],
      advl([m(after,_),m(ulf,prop))])].
```

The (transliterated) rune sentence to be analyzed is placed in the fourth slot as a list with commas between the words. If, on the other hand, we enter a functional representation, the Prolog program will solve for the variable in the fourth slot and generate a grammatical runic sentence as illustrated by the following call:

```
s(d,T,[subj(m(tuki,prop)),pred(m(raise,past)),
obj([m(stone,pl),m(these,_)]),advl([]))],S,[])
```

This time the program will solve S and deliver the output:

```
S=[tuki,risti,staina,thisi].
```

The adverbial was set at [] (empty list).

The program Rune includes an equivalent Modern Swedish grammar, which uses Swedish categories (prefixed with *m*) but the same functional representations and representations of word meanings. This enables the program to translate between Runic and Modern Swedish by writing:

```
s(M,T,F,Runic,[]),ms(M,T,F,ModSwed,[]).
```

Several sentence rules are needed. The following is the equivalent Modern Swedish sentence rule.

```
ms(d,N1,[subj(N1),pred(V),obj(N2),advl(A1)]) -->
      mnp(agr(nom,_,_),N1),mvfin(agr(_,_,_),V),
      mnp(agr(acc,_,_),N2),madvp(A1).
```

The cases nominative and accusative are needed only for pronouns.

Noun phrases and modifiers

The following are some rules of noun phrases, which have been presumed in the rules above. They specify different types of noun phrases, including coordinated noun phrases which may be complex. The individual lexical items are presented later. The rules show the importance of agreement in Runic

Swedish. As mentioned the agreement values in the noun phrases are accounted for by the complex variable *Agr*.

The first rule shows the case where the noun phrase consists of a noun only. The second shows how a pronoun may be an *np*.

```
np(Agr,N) --> n(Agr,N). % a noun only, e.g. Tuki (nom), Tuka (acc)
np(Agr,N) --> pron(Agr,N). % a pronoun only, e.g. ThaiR (nom)
```

A noun phrase may be more complex with a modifier phrase *(mp)* as in the following rules. The modifier phrase may occasionally occur before the noun which is indicated by the second variant. Nominal agreement between the head and the modifier is controlled by the values in the agreement complex *Agr*, the name of the complex variable *agr(Case,Number,Gender).* If there are several word meanings they are gathered in a list, e.g. [N,A] in the semantic representation to the left of the arrow.

```
np(Agr,[N,A]) --> n(Agr,N),mp(Agr,A). % buanta guthan, bruthir sin
np(Agr,[N,A]) --> mp(Agr,A),n(Agr,N). % guthan buanta
```

Modern Swedish normally only accepts attributives before the head as reflected in the following equivalent rule for Modern Swedish.

```
mnp(Agr,[N,A]) --> mmp(Agr,A),mn(Agr,N). % god make, sin broder
```

Appositions

Old Norse is famous for its appositions, placed rather freely (see below). The structure of the appositions is illustrated by the following rules, where *ap* is the apposition category. Typically, the head of such a noun phrase is a proper noun: *Ulf sun sin.* The second rule takes care of the case where there are two appositions, one apposition before and one after the head. The apposition must agree with its head, which is controlled by the variable *Agr*.

```
np(Agr,[N,A]) --> n(Agr,N),ap(Agr,A). % Tuki bruthir BiarnaR
np(Agr,[N,A,B]) --> ap(Agr,A),n(Agr,N),ap(Agr,B). % bruthur sin Tuka trak
      kuthan (acc)
```

The structure of appositions is specified by the following rules. Typically, the noun is a kin or social word, e.g. *bruthur* ‘brother; acc’, *suni* ‘sons; acc’, *filaka* ‘fellow; acc, sg or pl’. The *mp* may also be a genitive *np* or possessive pronoun. We will not go into further details here.

```
ap(Agr,[N,A]) --> n(Agr,N),mp(Agr,A). % fathir tuka, bruthir sin
ap(Agr,[N,A]) --> mp(Agr,A),n(Agr,N). % tuka father, sin father
```

The word order in Modern Swedish may be changed by the following rule, which changes the apposition into an epithet: *Toke sin broder* into *sin broder Toke*. The different functions of *sin* and a personal pronoun such as *hans* in Modern Swedish are pinpointed when experimenting with the program. Modern Swedish *Björn reste denna sten efter Toke sin broder* is somewhat strange, when referring to Björn; an alternative is *Toke hans broder*. It is, however, correct to use *sin* if the apposition is changed into a preattributive epithet as in *Björn reste denna sten efter sin broder Toke*. In that position *hans* can hardly be used if the word is to refer to Björn. Using the following rule, Swedish gets a different word order.

```
mnp(Agr,[N,A]) --> map(Agr,A),mn(Agr,N). % Björns broder Tuki
```

Removed appositions

Old Norse is famous for its use of appositions removed from their heads (postponed appositions). We show how this can be handled in our grammar by the following rule which includes a postponed subject apposition after the adverbial phrase. Note that the apposition is required to agree with the subject noun phrase and that the value of the apposition included in the variable *Ap* is inserted after the value *N1* of the subject *np* in the functional representation just as if it was not removed.

```
s(d,N1,[subj([N1,Ap]),pred(V),obj(N2),advl(A1)]) -->
      np(agr(nom,Nu,G),N1),vfin(agr(_,Nu,_),V),np(agr(acc,_,_),N2), advp(A1),
      ap(agr(nom,Nu,G),Ap).
```

This rule takes care of a sentence such as: *Tuki raisti stain thansi aftiR Tuma fathir Bjarnar,* where *fathir Bjarnar* is the postponed subject apposition whose content value by this rule is represented in the same way in the functional representation as in: *Tuki fathir Bjarnar raisti stain thansi aftiR Tuma.* In the sentence *Tuki raisti stain thansi aftiR Tuma fathur sin* the apposition *fathur sin* is in the accusative and must therefore belong to the object *Tumi (Tuma).* The grammar will analyze such cases correctly.

The word *sin* gets the meaning representation *m(refl,_)*, and the further interpretation of *sin* is considered a matter of semantic interpretation. Such an interpretation rule could state that a reflexive marker should be identified with (substituted by) the value of the subject of the sentence, often a proper name which identifies the referent.

Coordinated noun phrases

The following are rules for coordinated noun phrases. Such noun phrases coordinated with *auk* are quite common. The second rule takes care of the case where a personal pronoun *(thaiR)* sums up and stresses the coordination. Such cases *(Asbiurn auk Loke thaiR …)* occur only in subject position. The third rule illustrates how an apposition (in plural) to a coordinated noun phrase is handled. Note that the agreement number of a coordinated *np* is *pl*.

By using *np* as the second constituent our rules can also handle multiple coordinations, e.g. *Biurn auk Tuki auk Suin …* The semantic representation of coordination is a list including the meaning of the *np*s coordinated and the word *and*.

```
np(agr(Ca,pl,_),[N1,and,N2]) -->
      n(agr(Ca,_,_),N1),[auk],np(agr(Ca,_,_),N2). % Tuki auk Suin

np(agr(nom,pl,_),[N1,and,N2]) -->
      n(agr(nom,_,G1),N1),[auk],np(agr(nom,_,G2),N2),
      pron(agr(nom,pl,G3),N3). % Tuki auk Suin thaiR

np(agr(Ca,pl,_),[N1,and,N2,E]) --> n(agr(Ca,_,_),N1),[auk],
      np(agr(Ca,_,_),N2),ap(agr(Ca,pl,_),E). % Tuki auk Suin suni        BiarnaR
```

One Swedish equivalent is the following:

```
mnp(agr(Ca,pl,_),[N1,and,N2]) -->
      mn(agr(Ca,_,_),N1),[och],mnp(agr(Ca,_,_),N2). % Toke och Sven
```

Adverbial phrases

The general adverbial expression is an adverb or a prepositional phrase *(aftiR Tuma)* but some standard adverbial phrases of several words may be regarded as unit idiomatic expressions, e.g. *i uiking* and given a unit semantic representation. The preposition requires the following *np* to have a certain case (only accusative implemented).

```
advp(m(inviking,_)) --> [i,uiking].
advp(m(intheeast,_)) --> ([i,austr];[a,ustarla]).
advp(m(well,_)) --> [uel].
advp(m(manly,_)) --> [trikila].
advp([P,N]) --> p(Agr,P),np(Agr,N).
advp([]) --> []. % no adverbial
```

Some prepositions

```
p(agr(acc,_,_),m(after,_)) --> ([aftiR];[iftiR]; [aiftiR];[uftiR];[aft]).
p(agr(acc,_,_),m(in,_)) --> [i].
p(agr(acc,_,_),m(for,_)) --> [at].
p(agr(acc,_,_),m(on,_)) --> [a].
```

The following are some equivalent Modern Swedish rules

```
madvp([P,N]) --> mp(Agr,P),mnp(Agr,N).
madvp(m(inviking,_)) --> [i,viking].
madvp(m(intheeast,_)) --> ([i,öster).
madvp(m(well,_)) --> [väl].
madvp(m(manly,_)) --> [manligen].

mp(agr(acc,_,_),m(after,_)) --> [efter].
mp(agr(acc,_,_),m(in,_)) --> [i].
mp(agr(acc,_,_),m(for,_)) --> [för].
mp(agr(acc,_,_),m(on,_)) --> [på].
```

Representation of word meanings

In Swetra grammar all word meanings are written on the form *m(M,Gr),* where *m* stands for meaning. The word meanings (values of the variable *M*) are given in Swetra grammar as ‘machinese’ English-like words and the grammatical meaning as values of the variable *Gr*. For nouns *Gr* takes the values *sg* and *pl*, for verbs *pres, past, imp(erative), conj(unctive), nonf(inite).* The word meaning (universal semantic representation) of *fathir* is thus written: *m(father, sg)* and the meaning of *raisti* is written *m(raise, past)*. The case of the words is not represented in the word meanings of Swetra. Case is considered as a surface phenomenon varying with the syntax of the particular language. Adjectives, adverbs, prepositions and conjuntions have no value (underscore) of *Gr*, but are represented in the same formula for technical reasons. The semantic representation of *after* is *m(after,_)*. Only the surface category includes information about the rection of the preposition. The following are some lexical items. We have not written any general morphological rules for this domain.

Some lexical rules

```
a(agr(nom,sg,m),m(good,_)) --> [kuther]. % nominative form singular
a(agr(acc,sg,m),m(good,_)) --> [kuthan]. % accusative form singular
a(agr(acc,pl,m),m(good,_)) --> [kutha]. % accusative form plural
n(agr(nom,sg,m),m(tuki,prop)) --> [tuki].
n(agr(acc,sg,m),m(tuki,prop))--> [tuka].
n(agr(nom,sg,m),m(sigmund,prop))--> ([sikmutr];[sikmuntr).
```

```
n(agr(acc,sg,m),m(sigmund,prop))--> ([sikmut];[sigmunt]).
n(agr(nom,sg,m),m(kunar,prop))--> [kunar].
n(agr(acc,sg,m),m(kunar,prop))--> [kunar].
n(agr(nom,sg,m),m(father,sg)) --> [fathir].
n(agr(acc,sg,m),m(father,sg)) --> [fathur].
n(agr(nom,sg,m),m(son,sg)) --> [sun].
n(agr(gen,sg,m),m(son,sg)) --> [sunaR].
n(agr(acc,sg,m),m(son,sg)) --> [sun].
n(agr(nom,pl,m),m(son,pl)) --> [sunir].
n(agr(acc,pl,m),m(son,pl)) --> [suni].
n(agr(nom,sg,m),m(husband,sg)) --> ([buanti];[buati]).
n(agr(acc,sg,m),m(husband,sg)) --> ([buanta];[buata]).
n(agr(nom,sg,m),m(fellow,sg)) --> [filaki].
n(agr(acc,sg,m),m(fellow,sg)) --> [filaka].
n(agr(acc,pl,m),m(fellow,pl)) --> [filaka].
n(agr(acc,sg,m),m(stone,sg)) --> ([stain];[stin]).
n(agr(acc,pl,m),m(stone,pl))--> ([stina];[staina]).

pron(agr(nom,pl,m),m(m,pl)) --> ([thaiR];[thiR]).
pron(agr(nom,sg,m),m(m,sg)) --> ([haa];[ha];[saR]).
pron(agr(acc,_,_),m(refl,_)) --> [sik].
```

A rule interpreting unknown words

The following rule may be used when trying to interpret unknown inscriptions. It states that an unknown word (X) in a noun position may be identified as a proper noun in the nominative, accusative or genitive. This rule must be used with care otherwise words may be wrongly identified as proper names.

```
n(agr(_,sg,_),m(X,prop))--> [X].
```

Some verb forms

```
vfin(agr(_,sg,_),m(raise,past)) --> ([risthi];[raisthi];[risti];[raisti]).
vfin(agr(_,pl,_),m(raise,past)) --> ([risthu];[raisthu];[ristu];[raistu]).
vfin(agr(_,sg,_),m(set,past)) --> [sati].
vfin(agr(_,pl,_),m(set,past)) --> [satu].
aux(agr(_,sg,_),m(let,past)) --> [lit].
aux(agr(_,pl,_),m(let,past)) --> [litu].
finvi(agr(_,sg,_),m(fall,past)) --> [fil].
finvi(agr(_,pl,_),m(fall,past)) --> [filu].
finvi(agr(_,sg,_),m(go,past)) --> [fur].
finvi(agr(_,pl,_),m(go,past)) --> [furu].
cop(agr(_,sg,_),m(be,past)) --> ([uaR];[huar];[uas]).
cop(agr(_,pl,_),m(be,past)) --> [uaRu].
cop(agr(_,sg,_),m(be,pres)) --> [iR].
cop(agr(_,pl,_),m(be,pres)) --> [iRu].
```

Print-outs of interpretations of some inscriptions

The inscriptions mentioned are referred to by their usual Swedish names. The first two inscriptions are given as runes as well which are then transliterated. The transliterated inscription *(Inscr)* is then entered into the sentence call: *s(Mode,Topic,Funct,Inscr,[]))* in order to get the *Mode*, the *Topic* and the functional analysis *(Funct)*. It is possible to use the grammar rules in order to generate as well by inserting a functional representation and ask for a transliterated version and then a runic transcription of it. For some inscriptions it is possible to get several solutions, but we will not discuss this matter here. (3759, etc. are technical numbers.)

Dagstorp

ßikmuTr: ßaTi:ßTain:›anßi:ifTiZ:klakZ:fa›ur:ßin

sikmutr sati stain thansi iftiR klakR fathur sin

subj(m(sigmund, prop)) pred(m(set, past)) obj([m(stone, sg), m(this, _3759)]) advl([m(after, _3339), [m(klak, prop), [m(father, sg), m(refl, _873)]]])

Sigmund satte denna sten efter sin fader Klak

Skårby 1 Marsvinsholm

kaulfZ:auk:auTiZ:›aiZ:ßaTu:ßTain:›anßi:afTiZTuma:bru›ur:ßin

kaulfR auk autiR thaiR satu stain thansi aftiR tuma bruthur sin

subj([m(kaulf, prop), and, m(autiR, prop)]) pred(m(set, past)) obj([m(stone, sg), m(this, _3726)]) advl([m(after, _3306), [m(tumi, prop), [m(brother, sg), m(refl, _7161)]]])

Kaulv och Auter satte denna sten efter sin broder Tume

Vallkärra

tufi raisthi staina thisi aftiR kamal buanta sin auk asar sun hans

subj(m(tufi, prop)) pred(m(raise, past)) obj([m(stone, pl), m(these, _1122)]) advl([m(after, _723), [[m(gamal, prop), [m(husband, sg), m(refl, _4926)]], and, [m(asar, prop), [m(son, sg), m(ma, sg)]]]])

Tove reste dessa stenar efter sin husbonde Gammal och hans son Assar

Täby

iarlabaki lit raisa stain thisa at sik kuikuan

subj(m(iarlabaki, prop)) pred([m(let, past), m(raise, inf)]) obj([m(stone, sg), m(this, _1890)]) advl([m(for, _1341), [himselfalive]])

Jarlabanke lät resa denna sten åt sig själv i livet

Hunnestad 1
asbiurn auk tumi thaiR satu stain thansi aftiR rui auk laikfrut suni kuna hantaR

subj([m(asbiurn, prop), and, m(tumi, prop)]) pred(m(set, past)) obj([m(stone, sg), m(this, _1086)]) advl([m(after, _606), [m(rui, prop), and, m(laikfrud, prop), [m(son, pl), [m(kuna, prop), m(hanta, prop)]]]])

Asbiörn och Tume satte denna sten efter Gunne Hands söner Roi och Lekfrod

Skivarp
tumi raisthi stain thansi aftiR rua filaka sin

subj(m(tumi, prop)) pred(m(raise, past)) obj([m(stone, sg), m(this, _4320)]) advl([m(after, _3900), [m(rui, prop), [m(fellow, sg), m(refl, _1260)]]])

Tume reste denna sten efter Roi sin kamrat

Gårdstånga 1
austi auk kunar raisthu staina thisi aftiR knut auk biurn filaka sina

subj([m(austi, prop), and, m(kunar, prop)]) pred(m(raise, past)) obj([m(stone, pl), m(these, _1320)]) advl([m(after, _921), [m(knut, prop), and, m(biurn, prop), [m(fellow, pl), m(refl, _1251)]]])

Austi och Gunnar reste dessa stenar efter sina kamrater Knut och Björn

Kyrkstigen 1
runa rista lit rahnualtr

subj(m(ragnuald, prop)) pred([m(let, past), m(carve, inf)]) obj(m(rune, pl)) advl([])

Runor rista lät Ragnvald

Kyrkstigen 2
huar a kriklanti

subj(m(m, sg)) pred(m(be, past)) advl([]) advl(m(igrekland, _879))

Han var i Grekland

Kyrkstigen 3
uas lis forunki

subj([m(m, sg), [m(chief, sg), m(gang, sg)]]) pred(m(be, past)) advl([])

Han var gängs ledare

Hällestad
saR flu aiki at ubsalum

subj(m(m, sg)), pred(m(flee, past)), advl(m(nix, _3429)),
advl(m(atuppsala, _3372))

Han flydde ej vid Uppsala

References

Jansson, B. F. 1963. (2nd ed. 1976). *Runinskrifter i Sverige*. Stockholm: AWE/Gebers.

Rask, Lars. 1990. *Runläseboken.* Stockholm: Utbildningsförlaget Brevskolan.

Sigurd, Bengt. 1994. *Computerized grammars for analysis and machine translation.* Lund: Lund University press.