Lauri Karttunen

Numbers and Finnish Numerals

Abstract

This paper describes a computational implementation of the Finnish numeral system as a single finite-state transducer that maps inflected numerals to the corresponding numbers with tags that indicate morphological features, ordinality, number and case. The transducer is bidirectional. It can be used to analyze complex numerals such as *kahdensienkymmenensienyhdeksänsien* as *29+Ord+Pl+Gen* and to generate from a numeric input such as *251+Sg+Nom* the corresponding inflected numeral *kaksisataa-viisikymmentäyks*i. The mapping from numbers to numerals in Finnish is much more complex than a similar transduction for languages such as English because in Finnish complex numerals are traditionally inflected and all parts agree in ordinality, number and case, with the exception of nominative singulars such as *kaksisataa* '200'. Nevertheless, a complete analyzer/generator for millions of Finnish numerals can be built easily with the techniques described in the book on *Finite State Morphology* by Beesley and Karttunen (2003) using the **XFST** and **LEXC** compilers.

1. Introduction

Among computational linguists there is broad agreement that morphological systems of natural languages constitute **regular** (= rational) **relations** (Kaplan & Kay 1994). That is, the mapping from a Finnish lemma such as *kala+Pl+Gen* 'fish' to the corresponding inflected form *kalojen* can be described formally by a system of regular expressions that in turn can be compiled into a single **lexical transducer**, a finite-state transducer for morphological analysis and generation. The lexical side of the transducer includes all the morphological tags such as *+Pl* (plural), *+Sg* (singular), *+Nom* (nominative), *+Par* (partitive), *+Gen* (genitive), etc. The surface side of the transducer contains all the valid inflected forms of the language such as *kalojen*. The transducer encodes all the correct mappings

A Man of Measure
Festschrift in Honour of Fred Karlsson, pp. 407–421

from lemmas to surface forms, and vice versa. In the following, we use a colon to indicate a mapping between a pair of strings. For example, *kala+Pl+Gen:kalojen* indicates that the lemma *kala+Pl+Gen* is mapped to *kalojen*, and vice versa.

The regular expressions from which a lexical transducer is derived with the help of a compiler encode the **morphosyntax** of the language, that is, the way in which words are formed from prefixes, stems and suffixes. For example, in Finnish the plural suffix, generally realized as *i* or *j*, comes after the stem and before any case ending. Rules for **morphological alternations** such as Finnish vowel harmony, consonant gradation, etc. can also be represented as regular expressions using a formalism of two-level rules (Koskenniemi 1983), classical rewrite rules (Chomsky & Halle 1968), or replace rules (Karttunen 1996, 1997). Such rules represent regular relations and they can be compiled into finite-state transducers with techniques described in Kaplan and Kay (1994).

Because regular relations are **closed**[1] under concatenation, union, and composition, complex transducers can be constructed from simpler ones with these operations. For example, if Lex1a is a transducer that contains mappings such as *2:kaksi* for 'two' and if Lex1b contains identity relations for tag sequences such as *+Sg+Par:+Sg+Par*, then the concatenation of Lex1a and Lex1b, Lex1, contains *2+Sg+Par:kaksi+Sg+Par*. If Lex2 is a transducer with mappings such as *kaksi+Sg+Par:kahta*, then the composition of Lex1 with Lex2, Lex3, contains the mapping *2+Sg +Par:kahta*. In other words, Lex3 can be defined by the regular expression **`[Lex1a Lex1b] .o. Lex2`**, where concatenation is indicated by the empty space between Lex1a and Lex1b and **.o.** stands for composition. The Lex3 transducer can be used to analyze the surface form *kahta* as the singular partitive of *2* or to generate the surface form *kahta* from the lexical specification *2+Sg+Par*.

The basic idea of the mapping between numbers and inflected Finnish numerals sketched above must of course be refined to take into account the complexities involving agreement and morphological alternations. The details are explained in the following sections. As we will see, the resulting transducer is surprisingly compact. A network containing up to a million numerals contains only a few thousand states and arcs, and a complete

[1] A set or relation *R* is said to be closed under some operation *O* when the result of performing *O* on subsets or members of *R* always yields a subset or a member of *R*. For example, the set of positive integers is closed under addition but not under subtraction.

specification for the compiler, an **XFST** script,[2] consists of a few dozen lines of text.

2. The abstract syntax and semantics of numerals

Abstracting away for a moment from the peculiarities of Finnish, the composition and the interpretation of numeral expressions is very similar in most languages. As observed by Hurford (1975) and Smith (1998), complex numeral expressions are composed of three types of components: multipliers (M), units (U), and remainders (R). For example, the numeral *kaksikymmentäyksi* '21' has the structure [kaksi '2' • kymmentä '10' + yksi '1'], schematically [M • U + R]. The numeric value of the numeral is obtained by multiplying U with M and adding R to the result. The M and R components may themselves be complex numerals, as in *kaksikymmentä-yksituhattaviisisataaneljäkymmentäkolme* '21 543'. This complex numeral has the structure

[[kaksi • kymmentä] + yksi] • tuhatta + [viisi • sataa + [[neljä • kymmentä] + kolme]],

in numbers, [[2 • 10] + 1] • 1000 + [[5 • 100] + [[4 • 10] + 3]] = 21 543. In this recursive structure, each [M • U + R] triplet is subject to the following two constraints:

1. The numeric value of R is less than the numeric value of U.

2. The numeric value of M • U is less than the numeric value of the next larger U with the exception of *sata* '100' that allows 11-19 as additional multipliers.

In Finnish (as well as English), the units increase by a factor of ten up to one thousand: *kymmenen* '10', *sata* '100', *tuhat* '1 000'; and thereafter by a factor of thousand: *miljoona* '1 000 000', *miljardi* '1 000 000 000', etc. For example, for *sata* '100', the largest regular multiplier is *yhdeksän* '9' and the largest possible remainder is *yhdeksänkymmentäyhdeksän* '99' yielding *yhdeksänsataayhdeksänkymmentäyhdeksän* '999' as the largest

[2] An **XFST** script is set of commands for the **XFST** application, an interface to the Xerox finite-state calculus, distributed with the Beesley & Karttunen (2003) book.

regular numeral based on *sata*.[3] The M and R components are optional. A missing M is equivalent to 1, a missing R is interpreted as 0.

Because of the two semantic constraints, numerals such as *satatuhatta* '100 000' and *tuhatsata* '1 100' are unambiguous. In the former case *sata* must be the multiplier; in the latter case *sata* must be the remainder, not the unit. The semantic constraints do not rule out all ill formed numerals; for example, they allow **yksisataa* '100' and **kymmenenyksi* '11'. Finnish *yksi* '1' never appears as a multiplier. As in English, the Finnish numerals for 11–19 are composed in an exceptional way.

3. Morphosyntax of complex numerals

In Finnish, numerals are marked for ordinality, number, and case. In general, Finnish numerals occur with singular nouns; in the singular genitive of 'two men', *kahden miehen*, both the numeral *kaksi* '2' and the noun *mies* 'man' are singular. However, public events such as funerals, weddings, sport events, etc. are often denoted by plural nouns. In such contexts, the modifying numeral is also in the plural. For example, in *yhdet häät* 'one wedding' the numeral *yksi* '1' and the noun *häät* 'wedding' are both plural. The hypothetical singular of 'wedding', **hää*, does not exist.

All parts of a complex numeral generally agree in ordinality, number, and case. For example, the three components of *kahdensienkymmenensienyhdeksänsien* '29th+Pl +Gen are all redundantly marked for agreement: *kahdensien* '2nd+Pl+Gen' *kymmenensien* '10th+Pl+Gen' *yhdeksänsien* '9th +Pl+Gen'.[4] In general, complex numerals are written as one word, although it is possible to separate the unit with a space and to add an optional *ja* 'and' before the remainder, as in *kaksi tuhatta ja viisisataayksi* '2501'. In the following, we ignore these options and write all numerals as single words. Allowing optional spacing would not have any significant effect on the size of the final result.

---

[3] In English, the numeral *hundred* can also be modified by multipliers from *eleven* to *nineteen*. Thus 1900 can be read either as *one thousand nine hundred* or *nineteen hundred*. This is possible in Finnish, too, but less common than in English: *tuhatyhdeksänsataa* (1000 + (9 • 100)) vs. *yhdeksäntoistasataa* ((10 + 9) • 100).

[4] In singular ordinal numerals it is possible to mark ordinality and case only on the last numeral: *kaksikymmentäviidennen* instead of *kahdennenkymmenennenviidennen* '25th+Sg+Gen'. The prefix must be a singular cardinal in the nominative case. This option could be accommodated using the tag recoding technique in Section 5.2.

One remarkable exception to the across-the-board agreement principle is the singular nominative of cardinals. Instead of the expected **kaksikymmenenyhdeksän* ‘29+Sg +Nom’, we have *kaksikymmentäyhdeksän* ‘29+Sg +Nom’ where the unit *kymmenen* ‘10’ is in the partitive case. This is not a peculiarity of numeral unit nouns. The same happens with all types of nouns. For example, in the singular nominative case of ‘two men’, *kaksi miestä*, the numeral *kaksi* is in the nominative and the noun *mies* in the partitive (cf. “two of men”). Syntactically, the complex noun phrase *kaksi miestä* is clearly a singular nominative. The numeral becomes syntactically the head of the NP and expresses its case. There is no ready explanation for this switch and why it happens only with numerals. In the nominative singular of complex numerals, the multipliers and remainders are in the nominative but all the underlined unit nouns are in the partitive: *kaksi<u>kymmentä</u>yksi<u>tuhatta</u>viisi<u>sataa</u>neljä<u>kymmentä</u>kolme* ’21 543+Sg+Nom’.

Another exceptional case is numerals such as *toistakymmentä* ‘between ten and twenty’ *viidettäsataa* ‘between four and five hundred’, *kolmattatuhatta* ‘between two and three thousand’. These approximative numerals are peculiar in several ways. They consist of an ordinal multiplier in the singular partitive case (*toista* +2^{nd}+Sg+Part, *viidettä* 5^{th}+Sg+Part, *kolmatta* ‘3^{rd}+Sg +Part) followed exceptionally by a cardinal unit numeral that is in the singular partitive, as other unit nouns with a multiplier. The expression as a whole functions as a cardinal numeral and it can be interpreted either as nominative or as partitive.[5] Complex numerals of this type cannot contain a remainder (**toistakymmentäyksi*) and they do not occur in the remainder part of other complex numerals (**kaksisataatoistakymmentä*). But they can appear as multipliers, as in *viidettäsataatuhatta* ‘between four and five hundred thousand’.

---

[5] In *Tulva tuhosi kymmenen kylää* ‘The flood destroyed ten villages’ the object noun phrase is in the nominative; in *Tulva uhkasi kymmentä kylää* ‘The flood threatened ten villages’ the object is in the partitive. The numeral *toistakymmentä* can be interpreted either way: *Tulva tuhosi/uhkasi toistakymmentä kylää* ‘The flood destroyed/threatened more than ten villages. Hakulinen & al. (p. 759) states that only the nominative reading of *toistakymmentä* exists but counterexamples can be found in corpora and on the web.

4. Morphological alternations

Finnish has a rich set of regular morphophonological alternations such as Consonant Gradation and Vowel Harmony that affect the realization of all lexical forms. There are also alternations that are specific to a particular suffix. The plural marker, for example, is realized either as *i* or as *j* depending on the environment. The strategy adopted here is to map the lexical tag *+Pl* into a provisory morphophoneme *I* that is replaced by *i* or *j* in the final lexical transducer. That is, the original lexicon contains the mapping *+Pl:I* and the final result, either *+Pl:i* or *+Pl:j* is derived by composing the intermediate lexicon with a replace rule that realizes *I* as *j* between vowels and as *i* elsewhere. Similarly, the lexical tag for the partitive in the source lexicon is mapped to a pair of morphophonemes, *+Par:TA*, and the ultimate realization of *TA* as *ta*, *tä*, *a*, or *ä* is determined by rules. There are also stem alternations. Depending on ordinality, case, and number, the stem for the numeral *kaksi* '2' is realized as *kaksi*, *kahte*, *kahde*, *kaks*, or *kah*.

In our solution, we start with two lexicons, Lex1 and Lex2. Lex1 maps digits and morphological tags into a canonical representation of the numeral with identical tags. That is, it contains mappings such as *2+Sg +Gen*:*kaksi+Sg+Gen*. The second lexicon, Lex2, maps the canonical representations of numerals into forms that in some cases contain morphophonemes whose ultimate realization is determined in the composition with the rules. For example, it contains pairs such as *kaksi+Sg+Gen:kakTEn* that comes from the concatenation of the exceptional stem *kaksi:kakTE* with the regular form of the singular genitive *+Sg+Gen:n*. The final result, *kahden*, comes about when the lexicon is composed with a third component, the rules that realize the *kTE* sequence and other morphophonemes in the appropriate way.

Figure 1 is a sketch of the process that results in the mapping *2+Sg +Gen:kahden*.

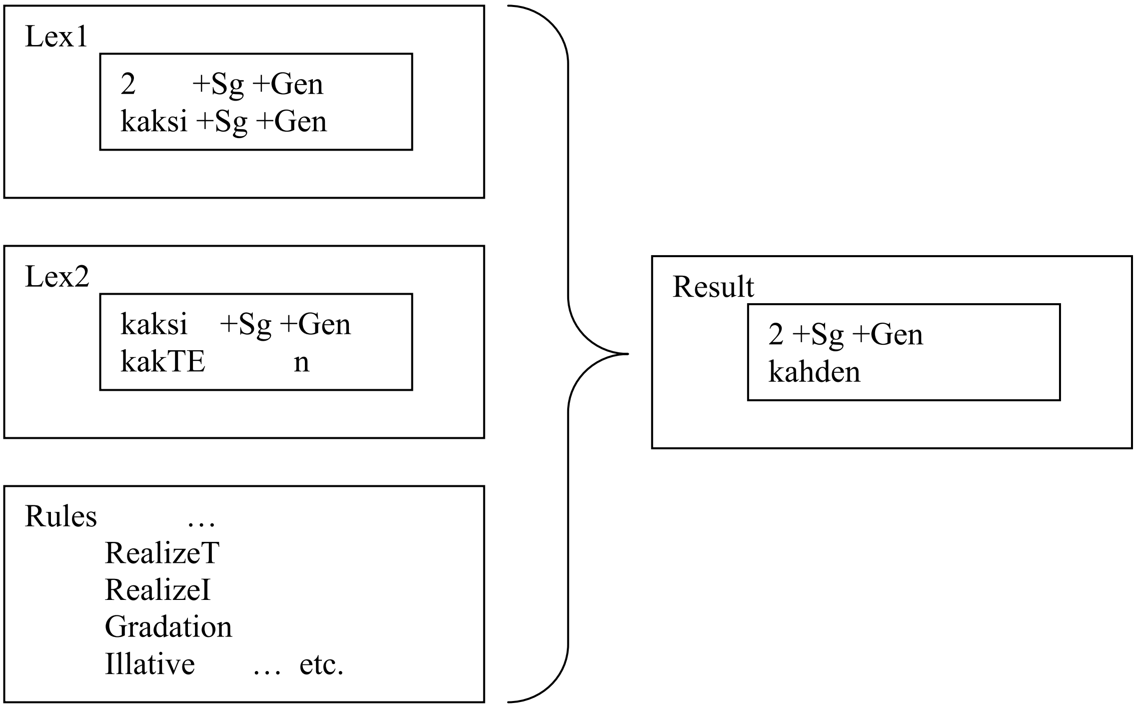

Figure 1.

Lex1 in Figure 1 is created with an XFST script, Lex2 is compiled with the LEXC compiler, the Rules component is an XFST script containing 17 ordered replace rules. The two lexicons and the rules are combined into a single lexical transducer by composition.

5. Some construction details

There is no space in this article to present and explain all the details of the scripts from which the lexicons and the rule transducers are compiled. We focus here on a couple of issues of particular interest.

5.1 Numbers to numerals mapping

The **XFST** script for Lex1 starts off with a number of definitions that introduce the optional ordinal tag, *+Ord* for **`Type`**, the tags for **`Number`** (*+Sg* or +Pl) and fourteen **`Case`** tags (+Nom, +Par, +Gen, etc.). The definition

```
define Infl Type Number Case;
```

specifies the order of these suffixes and the 56 possible combinations. For reasons that will be explained in a moment, we need a special version of **Infl** to mark the end of units such as *kymmenen* '10' and *sata* '100'.

```
define Unit 0:"!" Infl;
```

The zero in the definition of **Unit** stands for an epsilon, an empty string in the lexical representation. The exclamation mark on the lower side of Lex1 is used to trigger the realization of nominative as partitive in numerals such as *kaksikymmentä* '20+Sg +Nom'.

The following five definitions construct the mappings in Lex1 from *1...:yksi...* to *9...9...:yhdeksän...kymmenen|...yhdeksän...* where … represents all the possible tag sequences represented by **Infl**.

```
define TwoToTen [2:{kaksi}|3:{kolme}|4:{neljä}|
                 5:{viisi}|6:{kuusi}|7:{seitsemän}|
                 8:{kahdeksan}|9:{yhdeksän}] Infl;
define OneToTen [1:{yksi} Infl | TwoToTen];
define Teens [1:0 [{0}:{kymmenen} Infl |
                   OneToTen 0:{toista}]];
define Tens [TwoToTen [{0}:{kymmenen} Unit |
                        0 :{kymmenen} Unit OneToTen]];
define OneToHundred [OneToTen | Teens | Tens];
```

The vertical bar, **|**, represents the union operation. Consequently **TwoToTen** includes mappings such as **2:{kaksi}** and **3:{kolme}**. The curly brackets surrounding **{kaksi}** and **{kolme}** etc. indicate that the enclosed expression is to be interpreted as a sequence of single character symbols, *k a k s i,* and not as a multicharacter symbol such as *+Sg*. The definitions of **Teens** and **Tens** carefully distinguish between the plain **0** that stands for the empty string (epsilon) and **{0}**, for the literal digit zero. The **Teens** component of **OneToHundred** consists of the union of the two sets of sequences, **1:0 {0}:{kymmenen} Infl** and **1:0 OneToTen 0:{toista}**. The former encodes the mapping *10:kymmenen* concatenated with all the mappings in **Infl**; the latter contains mappings such as *11…:yksi…toista*, where *toista* is an uninflected component corresponding to *–teen* in English. For example, it includes the mapping

```
1 1       +Sg +Gen
  y k s i +Sg +Gen t o i s t a
```

The definition of `Tens` encodes the two restrictions in Section 2. As a unit noun, *kymmenen* '10' can be preceded by a multiplier less than 10 and followed by a remainder up to 9. `Tens` includes mappings such as

```
2      +Ord +Pl +Gen                +Ord +Pl +Gen 9           +Ord +Pl +Gen
kaksi +Ord +Pl +Gen kymmenen ! +Ord +Pl +Gen yhdeksän +Ord+ Pl +Gen
```

The possible combinations for the next larger unit, `Hundreds`, are subject to the same multiplier restrictions as *kymmenen* '10' but allowing remainders up to 99.

```
define HMult [1:0 | TwoToTen];
define Hundreds [HMult [{00}:{sata} Unit |
                        {0}:{sata} Unit OneToTen |
                         0:{sata} Unit [Teens|Tens]]];
```

The definition of `HMult` contributes an initial single digit to `Hundreds` and the corresponding numeral for all digits but 1. The number of zeros mapping to *sata* depends on the size of the remainder. If there is no remainder, we get two zeros; if the remainder is in `OneToTen`, one zero is required; if the remainder provides two digits (`Teens` and `Tens`), *sata* is paired with an empty string. Consequently, all complex numerals with *sata* as the unit are paired with a three-digit number from 100 to 999. Complex numbers based on *tuhat* '1000', *miljoona* '1000000', etc. are constructed analogously. If the optional *+Ord* tag is not present, the number is cardinal.

5.2 Agreement

In addition to encoding the phonological realization of ordinality, case, and number, we also need to make all parts of a complex numeral agree in these respects. This is done with the help of "flag diacritics" (see Beesley & Karttunen 2003, Chapter 7). Flag diacritics are special epsilon symbols such as `@U.Type.Ord@`, `@U.Number.Sg@`, `@U.Case.Nom@`, that are invisible to all finite-state operations. They have meaning only to the APPLY routine that is used to analyze or generate surface forms. These flag diacritics contain three components, an operation (`U` = unify), an attribute, (`Type`, `Number`, or `Case`) and a value such as `Ord` (ordinal), `Sg` (singular) and `Nom` (nominative). The unify operation has the effect that any flag diacritic that is encountered by the apply routine along a path must agree with any previous value of the same attribute. Although the network described in Section 5.1 in fact contains paths for illegal numerals such as **kahdeksisatojaviidennen* '205' that do not agree in ordinality, case, and number, such invalid numerals are neither recognized nor generated by the system because of mismatching flags.

The use of flag diacritics does not increase the formal power of the system beyond finite-state. In fact the **XFST** tool has an operation that removes any flag attribute and compiles the constraint it enforces directly into the state and arcs of the network.

The three components of `Infl` defined in the beginning of Section 5.1 pair each morphological tag with the corresponding flag diacritic. Thus the lexical representation of *kahdensadan* '200+Sg+Gen' actually contains flag diacritics in addition to morphological tags. The two components, *2:kaksi* and *00:sata*, are both concatenated with the same sequence of flag diacritics and tags:

```
@U.Type.Card@ @U.Num.Sg@ +Sg @U.Case.Gen@ +Gen
```

When Lex1 is composed with Lex2, the flags in Lex1 are treated as epsilons but the tags have to pick up a matching tag in Lex2. In this way, the morphological tag *+Gen* is paired with the string *n*, its surface realization.

This method of construction provides an easy solution to the problem that unit nouns surface in the partitive form in singular nominatives. In addition to the flags and morphological tags, the unit nouns *kymmenen*, *sata*, *tuhat* and *miljoona* are marked with a special symbol, `!`, that distinguishes them from other numerals. Before Lex1 is composed with Lex2, we compose Lex1 with a transducer derived from the replace rule

```
[%+Nom -> %+Par || "!" %+Sg _ .o. "!" -> 0];
```

The effect is that, in the singular nominative of unit nouns, the lexical side tag *+Nom* gets paired with *+Par* tag and the special trigger for this operation, `!`, is removed. Consequently, when Lex1 is composed with Lex2, the singular nominative unit words in Lex1, now concatenated with *+Nom:+Part* tags, are paired with singular partitives in Lex2 giving us mappings such as

```
2          +Sg +Nom      0 0     +Sg +Nom
k a k T E              #  s a t a         T A
```

where the second *+Nom* tag is paired with the partitive marker. After the composition of Lex1, Lex2 and the Rules, the redundant tags are removed from the lexical side leaving *200+Sg+Nom:kaksisataa* in the final result. (The word boundary # marks the domain of Vowel Harmony.)

The cardinality/ordinality and case mismatch in numerals such as *viidettäsataa* 'between four and five hundred' can be handled without any rules by introducing a special sequence of tags:

```
0:%+Ord %+Sg [%+Nom:%+Par | %+Par]
```

where the upper side contain the sequences *+Sg+Nom* and *+Sg+Par* and the lower side consists of *+Ord+Sg+Par*. After composition with Lex2, we have mappings such as

```
~ 5                +Sg +Nom       0 0      +Sg +Nom
 v i i TE  NTE         TA     #  s a t a          TA
```

where the lower side contains the ordinal marker *NTE* without the corresponding *+Ord* tag and the two *+Nom* tags both are paired with the partitive marker *TA*. The final result is the mapping *~500+Sg+Nom: viidettäsataa*. The initial tilde, ~, is a mark for this type of approximative numeral.

6. Implementation

The construction of the lexical transducer for Finnish numerals described in this paper takes a couple of seconds on a Macintosh laptop (1.67 GHz PowerPC G4) for numerals up to a million cardinals and ordinals inflected for case and number. With the XFST tool, it can be used for analysis (apply up) and generation (apply down), as shown below:

```
xfst[1]: apply up kaksikymmentäyksituhattaviisisataaneljäkymmentäkolme
21543+Sg+Nom

xfst[1]: apply down 29+Ord+Pl+Gen
kahdensienkymmenensienyhdeksänsien
```

```
xfst[1]: apply up kaksisadan
xfst[1]:
```

Ill formed numerals such as **kaksisadan* receive no analysis. Because agreement constraints are encoded by flag diacritics and checked by the apply routine at "run time", the transducer is quite small: 1 946 states and 3 641 arcs. The flags can be removed in **XFST** with the command 'eliminate flag' that takes as argument a name of an attribute such as `Type` that occurs in `@U.Type.Card@` and `@U.Type.Ord@`. Removing a set of flag diacritics generally increases the speed of the apply routine at the cost of increasing the size of the network because the "hardwiring" of the constraints they encode requires additional states and arcs. Table 1 below shows the effect of removing the three types of agreement flags one by one:

Flag diacritics present	Size of the transducer
`Type`, `Number`, `Case`	1 946 states, 3 641 arcs
`Number`, `Case`	2 635 states, 4 794 arcs
`Case`	3 706 states, 6 346 arcs
(All flags eliminated)	20 498 states, 26 371 arcs

Table 1

Removing all the flags yields a transducer that encodes only valid numerals and requires no runtime checking for agreement. It contains nearly 57 million inflected surface forms mapped to the corresponding numbers ranging from 1 to 1 000 000.

7. Comparison with Lingsoft's TWOL analyzer

The leading commercial morphological analyzer for Finnish, Lingsoft's **TWOL** system (http://www.lingsoft.fi/cgi-bin/fintwol), correctly checks for number and case agreement of cardinal numerals. For complex numerals it returns a sequence of components separated by `#` and the appropriate morphological tags. For example, *tuhatkolmesataa* '1300+Sg +Nom' is analyzed as `"tuhat#kolme#sataa" NUM NOM SG`. The marking of component boundaries in the Lingsoft lexical forms is unsystematic. The result for the corresponding genitive form, *tuhannenkolmensadan*, is `"tuhat#kolmesataa" NUM GEN SG`, but *sataa* '100+Sg+Par' by itself is mapped to `"sata" NUM PTV SG`. The output is not optimal for higher-level processing because the components are identified inconsistently and there is no semantic analysis. For the same reason, the system appears not to be suitable for the generation of numerals.

The Lingsoft analyzer for Finnish does not include plural forms of complex ordinal numerals. Forms such as *kahdensienkymmenensienyhdeksänsien* '29+Ord+Pl+Gen' are not recognized. All the exceptional approximative numerals other than the ones beginning with *toista* '2nd+Ord+Sg+Par' are missing. Forms such as *viidettäsataa* '~500+Sg+Nom' and *kolmattatuhatta* '~3000+Sg+Nom', etc. are systematically absent. Numerals such as *toistakymmentä* '~20+Sg+Nom' are recognized, **`"toistakymmentä" NUM NOM SG`**, but the partitive reading, '~20+Sg+Par', is not present. The case tag should be **`NOM/PTV`**. The Lingsoft analyzer accepts forms such as **toistakymmentäyksi* that are ill formed because approximative numerals cannot contain remainders.

8. Outstanding issues

As an anonymous reviewer pointed out, the occurence of discourse particles such as *-hAn*, *-kO* and *-pA* in complex numerals is not addressed. These particles occur rather freely, not just at the end of a complex numeral but also in intermediate positions: *kahtakosataatuhatta*, *kahtasataakotuhatta*, *kahtasataatuhattako* are examples of the question clitic *-kO* focusing on a different component of the numeral. In English the focus of the question would be marked with intonation: 'TWO hundred thousand?' (*kahtakosataatuhatta?*), 'Two HUNDRED thousand?' (*kahtasataakotuhatta?*). 'Two hundred THOUSAND?' (*kahtasataatuhattako*). We could easily generate clitics in any of the positions and use flag diacritics to control their order and co-occurrence restrictions. But the same issues arise in other types of complex nominals: *uttako tekniikkaa* 'NEW technology?', *uutta tekniikkaako* 'new TECHNOLOGY?'. The syntax and semantics of discourse particles is a too large a topic to be addressed here.

9. Summary

This paper presents a comprehensive description of the Finnish numeral system, extending and formalizing the descriptions found in standard grammars (Karlsson 1983, Hakulinen & al. 2005) that do not include an account of the semantics of complex numerals. The mapping from numbers to numerals provides an account of the meaning of numerals in addition to describing their surface form. The lexical transducer compiled from the description can be used both for analysis and generation. Text-to-speech applications for Finnish obviously must produce inflected numerals. For example, the abbreviated numeral *29:nsien* in *Tervetuloa 29:nsien olympialaisten avajaisiin* 'Welcome to the Opening Ceremonies of the 29th Olympic Games' must be pronounced as *kahdensienkymmenensienyhdeksänsien.*

Overall, this paper is a case study of how to decompose a complex morphological relation into a few simple components that a computational linguist can readily understand and apply. Here the Lex1 component defines the semantics, morphotactics and the syntactic case of numerals. Lex2 specifies the ordinal, numeral and case endings using morphophonemes to encode some rule-governed alternations. The fact that the syntactic case does not always match the morphological case is accounted for by mapping nominative to partitive on the lower side of Lex1 in the relevant environments before the composition with Lex2. The final realization of stems and suffixes is determined by the third component, an ordered set of replace rules.

References

Beesley, Kenneth R. & Lauri Karttunen (2003) *Finite State Morphology*. Stanford, CA: CSLI Publications.

Chomsky, Noam & Morris Halle (1968) *The Sound Pattern of English*. New York, NY: Harper and Row.

Hakulinen, Auli, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja-Riitta Heinonen & Irja Alho (2005) *Iso suomen kielioppi*. Suomalaisen Kirjallisuuden Seuran Toimituksia 950. Helsinki: Suomalaisen Kirjallisuuden Seura.

Hurford, James R. (1975) *The Linguistic Theory of Numerals*. Cambridge: Cambridge University Press.

Kaplan Ronald M. & Martin Kay (1994) Regular models of phonological rule systems. *Computational Linguistics* 20.3: 165–186.

Karlsson, Fred (1983) *Finnish Grammar*. Porvoo, Helsinki & Juva: Werner Söderström Osakeyhtiö.

Karttunen, Lauri (1996) Directed replacement. In *ACL-1996,* pp. 108–115. Santa Cruz, CA: Association for Computational Linguistics.

—— (1997) The replace operator. In Emmanuel Roche & Yves Schabes (eds.) *Finite-state Language Processing*, pp. 117–147, Cambridge, MA: The MIT Press.

Koskenniemi, Kimmo (1983) *Two-level Morphology: A general computational model for word-form recognition and generation*. Publications of the Department of General Linguistics, University of Helsinki 11. Helsinki: University of Helsinki.

Smith, Jeffrey D. (1998) English number names in HPSG. In Gert Webelhuth, Jean-Pierre Koenig, & Andreas Kathol (eds.) *Lexical and Constructional Aspects of Linguistic Explanation*, pp. 145–160. Stanford, CA: CSLI Publications.

Acknowledgement

I thank Kenneth R. Beesley, Annie Zaenen and an anonymous reviewer for their helpful comments and suggestions on an earlier version of the paper.

Contact information:

Lauri Karttunen
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
karttunen(at)parc(dot)com
http://www2.parc.com/istl/members/karttune/