Nordic Journal of African Studies 8(2): 139-157 (1999)

SALAMA
Swahili Language Manager

ARVI HURSKAINEN
University of Helsinki

SALAMA is an acronym for *Swahili Language Manager*[1]. It is a computerised working environment, where it is possible, with the help of a set of programs and user-defined utilities, to perform a multitude of tasks. For introducing SALAMA, it is perhaps more interesting to describe its aims and applications first, and then give an outlay of its components. Appendix 1 gives an overall view of the structure of the system, and of some of its applications. In this paper it is possible to give only a brief and condensed description of how the system is constructed. It is without any technical detail as to how the system is implemented. For those interested in more detailed description of the components, a list of relevant publications is added in the end of the paper. References to those are also made in text.

WHY SALAMA?

SALAMA has grown gradually from a rudimentary morphological parsing program into a comprehensive language management system. In other words, there was no original plan for creating SALAMA. The accomplishment of one phase has given impetus to another effort, and by building one block on another, an end product of formidable properties has resulted.

At present, SALAMA has facilities for carrying out such task as:
Spelling checker of Standard Swahili text (implemented on Word 97 and later versions)
Hyphenator for automatic hyphenisation of Swahili text (implemented on Word 97 and later versions)
Morphological analyser, with information on such features as: part-of-speech (word class), tags for inflectional and derivational morphemes, lemma, etymology of loan-words, tags for domain-specific terminology, gloss in English, etc.
Lemmatiser
Morphological disambiguator

1 SALAMA was introduced and demonstrated for the first time in the 20th International Biennial Conference of the African Language Association of Southern Africa (ALASA), in July 5-9, 1999.

Semantic (word sense) **disambiguator**
Heuristic 'guesser' for resolving remaining ambiguities
Syntactic mapper for providing text with syntactic information (shallow/surface syntax)
Grammatical converter into InterLingua (needed in translation applications)

Broadly speaking, SALAMA holds facilities for marking and making explicit linguistically relevant information on running text. This has direct effects on information management and retrieval, which are thus vastly enhanced, compared with string search, or search utilizing regular expressions.

HISTORY OF SALAMA

The first attempts for writing a morphological description of Swahili were made on the course on Two-Level morphology, arranged by Kimmo Koskenniemi in 1985 in Helsinki. Two years earlier had appeared his dissertation *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production,* which became later a standard in language description. The first rough version of SWATWOL (Swahili Two-Level Morphology) was constructed during this course. The work was continued over the years, and in 1992 the first description of this system was published (Hurskainen 1992). Already this version had performance comparable with the best morphological parsing systems of that time. Research on finite-state transducers has continued to be quite active over the years (Karttunen 1994; Pulman and Hepple 1993).

SWATWOL was then improved and particularly its lexicon was enlarged by a number of methods. Various word-lists and dictionaries formed the basis of the lexicon, but it was continually enlarged also with the material from various types of texts, which were collected to DAHE (Dar-Helsinki Archives of Swahili). The SWATWOL lexicon was thus improved by each new text by providing new words and by pointing out possible defects in the lexicon.

A new phase in developing SALAMA was reached when SWACGP (Swahili Constraint Grammar Parser) was released (Hurskainen 1996). This parser performs disambiguation operations on ambiguous readings and it also performs syntactic mapping. Fred Karlsson (1990, 1995a, b) has been instrumental in developing the theoretical base of the implementable constraint grammar parser, and Pasi Tapanainen (1996) has written the parsing program as well as created an efficient environment for writing and testing rules. While writing and testing disambiguation rules, it has been possible to enhance the morphological lexicon further.

The work on SALAMA is presently focussed on writing algorithms for rules needed in word-sense disambiguation, and of figuring out the operations needed in the conversion of the output of syntactic analysis into InterLingua (see below).

COMPONENTS OF SALAMA

SALAMA can be viewed as a working environment, where one can perform a number of different kinds of tasks. There are two major language-specific components, SWATWOL and SWACGP, which perform the hard tasks of morphological analysis, disambiguation, and syntactic mapping. In addition, the system allows the user to tune the system according to the tasks which it is expected to perform. Both the input and output may be modified in several ways.

SWASENT is a pre-processing program which performs certain formalization operations on the text. Such operations include: separation of punctuation marks and diacritics from words; reduction of upper-case letters to lower-case and marking those letters that always should remain upper-case; joining some 'words', written separate but being grammatically one word, together as one unit; arranging the text so that each sentence is on a separate line, etc.

WLIST is a shell script which verticalises text, placing one word per line. There are several task-specific versions of this program.

SWATWOL is an acronym for Swahili Two-Level Parser, a morphological analyser of Swahili language. The analysis program itself is TWOL (Two-Level Parser), designed by Kimmo Koskenniemi (1983), and it takes the two-level rule file as well as the dictionary file as input. In normal use, these two files are compiled into a single automaton. The morphophonological two-level rules take care of surface variation, which occurs mostly in morpheme boundaries. The dictionary contains presently about 25,000 morpheme entries. However, the number of words it recognizes is several thousands bigger, because the formation of nouns derived from verb roots is implemented directly from the verb roots, without writing them as separate entries. Also the derivation of verbs from basic verb roots is implemented by the combinatory method, without writing each derived verb stem as a separate entry. This solution decreases the size of the lexicon and makes the function of the system faster. The total number of words recognized by the lexicon, if derived verbs are also counted as separate words, is at least 45,000 (for an early description of SWATWOL, see Appendix 2).

There are several versions of the dictionary, suited for different kinds of tasks. Versions for testing the coverage of various printed dictionaries is one type of application (Hurskainen 1994 , 1999). One version identifies the etymology of words. There is also a test version designed for semantic analysis.

The output of SWATWOL can be modified in several ways according to need. The modification can be performed by using selected flags which modify the operation of the program, or by selecting a suitable version of the lexicon. Often it is useful to apply both possibilities.

SWACGP is a constraint grammar parser, which performs disambiguation operations on the readings produced by SWATWOL. It also functions as a syntactic mapping program, providing word-forms with surface-syntactic tags. SWACGP consists of the language-independent parsing program CGP (Karlsson 1990; Karlsson et al 1995; Tapanainen 1996; Voutilainen et al 1992; Voutilainen and Tapanainen 1993) and the language-specific rule file, which it takes as input. The rule file has presently about 1,200 rules. By optimizing and generalizing part of the rules, it is perhaps possible to reduce the number of rules to some extent.

In the average, about 50 percent of Swahili word-forms are at least two-ways ambiguous. By running SWACGP, ambiguity is reduced to 8% of word-form tokens in fiction prose text, and to 5% in newspaper texts (Hurskainen 1996: 572). By refining rules and by writing a number of specific rules it is possible to increase the performance to some extent.

SWA-GUESS is a heuristic guessing program written in flex (a Unix utility). It disambiguates most of the ambiguous interpretations, for which it is not possible to write constraint grammar rules. The percentage of ambiguity left after running SWA-GUESS is less than 3%. But one has to be aware that SWA-GUESS may also make a wrong guess, although most of the guesses are correct. It should be noted that a full 100 percent disambiguation is in practice impossible, because in texts there are sometimes words, which even the human brain cannot disambiguate. It depends on the purpose of the application whether such words should be left ambiguous, or whether guessing should be performed also on those problematic words.

In this phase, the output contains maximally the following features: the surface word-forms, including punctuation marks and diacritics; base-form (lemma); part-of-speech specification; full morphological information; grammatical roles of verbs; some semantic tagging; etymological information; syntactic tags; and glosses in English.

SWAFDP is an acronym for Swahili Functional Dependency Parser, which is superior to SWACGP in that it performs full syntactic analysis, not only shallow parsing as SWACGP does. SWAFDP (Järvinen and Tapanainen 1997, 1998; Tapanainen and Järvinen 1997) was developed to cope with problems, for which earlier analysis systems did not have built-in facilities to cope with. The parser which performs full syntactic analysis makes it possible to build syntactic trees, where the dependence structure of each word is precisely shown. This facility is particularly useful, even necessary, if the system is used for developing language translation applications.

SWA-TO-InterLingua is a grammatical converter, which modifies the output of SWAGUESS into the format expected by the InterLingua. This program is based

on the idea that by utilizing the morphological, syntactic and semantic information of a source language it is possible to 'translate' the text into a language which is here termed InterLingua. This language is not a living language as such. It utilizes semantic equivalences of the source language and English, but only in their lexical forms. All the rest, i.e. part-of-speech specification, morphological and syntactic tags etc. are given as such.

The conversion is carried out by two types of rules.

(1) *Tag transformation rules* change the tags used in the source language into tags required by the InterLingua. What are then the tags of the InterLingua and what makes them different from the tags of the source language? For the sake of convenience and for avoiding unnecessary extra learning, the tags of the InterLingua are much the same as the tags used in the morphological and syntactic analysis of English. Since there is no general agreement on the tag-set used for marking morphological and syntactic features of English, the tags of the InterLingua have to be taken as tentative in this phase. Tag transformation rules concern more morphological tags than syntactic ones, which are much the same in Swahili and InterLingua. For example, both are basically SVO languages, although Swahili applies also the SOV principle when the object is a pronoun (see above). In addition, for animate objects it applies double marking, placing also the pronominal morpheme in front of the verb root. Word order in question sentences is also basically the same, as well as in passive constructions.

(2) *Constituent re-ordering rules* will arrange the constituents of the source language into the order required by the InterLingua. For example, if the source language is a SOV language, the constituents of the sentence are reordered to meet the SVO structure.[2] Re-ordering rules are not applied in parallel; they are ordered rules. The preference of rule application is a bit complicated, because in the identification of some basic constituents the order is from long to short, and on the sentence level it is from short to long. Noun phrase is a typical example of such basic constituents. In the identification of noun phrases, the system tries to find the longest legal noun phrases of the sentence. For example, if the sentence contains a structure such as NOUN+POSS+DEM+ADJ+NUM (e.g. viti vyangu hivi vizuri viwili) , the system applies the rule which handles the noun phrase with maximal length, and it re-orders it into DEM+POSS+NUM+ADJ+NOUN (in InterLingua: these my two good chairs). There are several other types of noun phrases and each of them is tested, the long ones first.

2 The identification of noun phrases is a major problem in language analysis, and work in this field has been done for example by Atro Voutilainen (1995), and by the team working with dependency grammar (Tapanainen and Järvinen 1994).

Here are some more noun phrase rules:

NOUN+POSS+DEM+ADJ >	DEM+POSS+ADJ+NOUN
viti vyangu hivi vizuri	these my good chairs
NOUN+POSS+DEM >	DEM+POSS+NOUN
viti vyangu hivi	these my chairs
NOUN+POSS+ >	POSS+NOUN
viti vyangu	my chairs
NOUN+POSS+DEM+NUM >	DEM+POSS+NUM+NOUN
viti vyangu hivi viwili	these my two chairs
NOUN+POSS+NUM >	POSS+NUM+NOUN
viti vyangu viwili	my two chairs
NOUN+NUM >	NUM+NOUN
viti viwili	two chairs
NOUN+POSS+ADJ+NUM >	POSS+NUM+ADJ+NOUN
viti vyangu vizuri viwili	my two good chairs
NOUN+ADJ+NUM >	NUM+ADJ+NOUN
viti vizuri viwili	two good chairs
NOUN+ADJ >	ADJ+NOUN
viti vizuri	(the) good chairs
NOUN+DEM+ADJ+NUM >	DEM+NUM+ADJ+NOUN
viti hivi vizuri viwili	these two good chairs
NOUN+DEM+NUM >	DEM+NUM+NOUN
viti hivi viwili	these two chairs
NOUN+DEM >	DEM+NOUN
viti hivi	these chairs
NOUN+DEM+ADJ >	DEM+ADJ+NOUN
viti hivi vizuri	these good chairs
NOUN+POSS+ADJ >	POSS+ADJ+NOUN
viti vyangu vizuri	my good chairs

When noun phrases and other low-level constituent clusters have been identified, rules for setting equivalence between major syntactic modules will be applied. In the phase where noun phrases already have been identified and the modifiers of the nouns have been fixed to the head, i.e. the noun, the number of remaining un-identified constituents has decreased considerably. On the basic level of syntax, Swahili and InterLingua follow quite similar rules. Both are SVO languages, and the indirect object precedes the direct object. The major difference is that in Swahili the pronominal object is prefixed to the verb, while in InterLingua it follows the verb. This can, however, be handled by the re-ordering rules.

THE ROLE OF CORPUS IN CONSTRUCTING SALAMA

SALAMA has been developed along with the accumulation of Swahili text archives. Since 1988 texts from various sources have been collected into the archives. Texts from newspapers were keyed in and books with reasonable print quality were scanned and edited. A joint research project with the University of Dar-es-Salaam resulted in a wealth of spoken texts, later transcribed into electronic form, and in a number of word-lists from various speech areas of Swahili. Texts were also received through exchange from colleagues. Some texts in computer form were kindly donated by some agencies, such as the text of the Swahili Bible by the United Bible Society, Nairobi. For a couple of years it has also been possible to extract newspaper and news texts from Internet. The material in the archives has been pre-processed and partially coded, according to the type of material concerned.

Swahili archives has been the testing material in developing SALAMA. Particularly SWATWOL has benefited from the archives, and so has SWATWOL helped in editing texts. All new texts have been analysed by SWATWOL, and new hitherto unknown words have been added into the dictionary. At the same time SWATWOL has shown the misspelled words in text, and helped in final editing. In fact the archives has been indispensable in tuning SALAMA to meet the actual needs of different types of texts.

CONCLUSION

Efficient information management and retrieval requires that the information stored in the text is made unambuguously into a searchable format. In other words, what is implicit in text has to be made explicit, so that tools designed for string search can be used effectively for retrieving features, which are not readily available in normal text. This requires a detailed analysis of language, including morphological,

syntactic and semantic analysis. The system presented above is a consistent environment for developing many kinds of applications for research as well as for practical purposes.

REFERENCES

Hurskainen, A. 1992.
A Two-Level Computer Formalism for the Analysis of Bantu Morphology: An Application to Swahili. **Nordic Journal of African Studies** 1(1): 87-122.

1994 *Kamusi ya Kiswahili Sanifu in test: A computer system for analyzing dictionaries and for retrieving lexical data*. **Afrikanistische Arbeitspapiere** 37 (Swahili Forum I): 169-179.

1996 Disambiguation of morphological analysis in Bantu languages. In *COLING-96, Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, August 5-9, 1996. Pp. 568-573.

1999 *Salim K. Bakhressa, Kamusi ya Maana na Matumizi*. Nairobi: Oxford University Press. Book review. **Journal of African Languages and Linguistics** 20 (to appear). Book review.

Järvinen, T. and Tapanainen, P. 1997.
A Dependency Parser for English. **Technical Reports**, No. TR-1. Department of General Linguistics. University of Helsinki.

1998 Towards an implementable dependency grammar. In Sylvain Kahane and Alain Polguère (eds.), *Proceedings of the Workshop 'Processing of Dependency-Based Grammars'*, Université de Montréal, Quebec, Canada, August 15th , 1998. Pp. 1-10.

Karlsson, F. 1990.
Constraint Grammar as a framework for parsing running text. In Hans Karlgren (ed.), *COLING-90. Papers presented to the 13th International Conference on Computational Linguistics*. Vol. 3, pp. 168-173, Helsinki.

1995a Designing a parser for unrestricted text. In Karlsson et al (eds.), *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin. Pp. 1-40.

1995b The formalism and environment of Constraint Grammar Parsing. In Karlsson et al (eds.), *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin. Pp. 41-88.

Karlsson, F., A. Voutilainen, J. Heikkilä and A. Anttila (eds.) 1995.
Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin.

Karttunen , L. 1994.

Constructing Lexical Transducers. In *COLING-94. Papers presented to the 15th International Conference on Computational Linguistics*. Vol. 1, pp. 406-411, Kyoto.

Koskenniemi. K. 1983

Two-level morphology: A general computational model for word-form recognition and production. Publications No. 11. Department of General Linguistics, University of Helsinki.

Pulman, S. and Hepple, M. 1993.

A feature-based formalism for two-level phonology: a description and implementation. **Computer Speech and Language** 7.

Tapanainen, P. 1996.

The Constraint Grammar Parser CG-2. Publications No. 27. Department of General Linguistics, University of Helsinki.

Tapanainen, P. and Järvinen, T. 1994.

Syntactic analysis of natural language using linguistic rules and corpus-based patterns. In *COLING-94. Papers presented to the 15th International Conference on Computational Linguistics*. Vol. 1, pp. 629-634, Kyoto.

1997 A non-projective dependency parser. In *Proceedings of the 5th Conference of Applied Natural Language Processing*, March 31st - April 3rd, Washington D.C., USA. Pp. 64-71.

Voutilainen, A. 1995.

NPtool, a detector of English noun phrases. In *Proceedings of Workshop on Very Large Corpora,* held June 22, 1993 at Ohio State University.

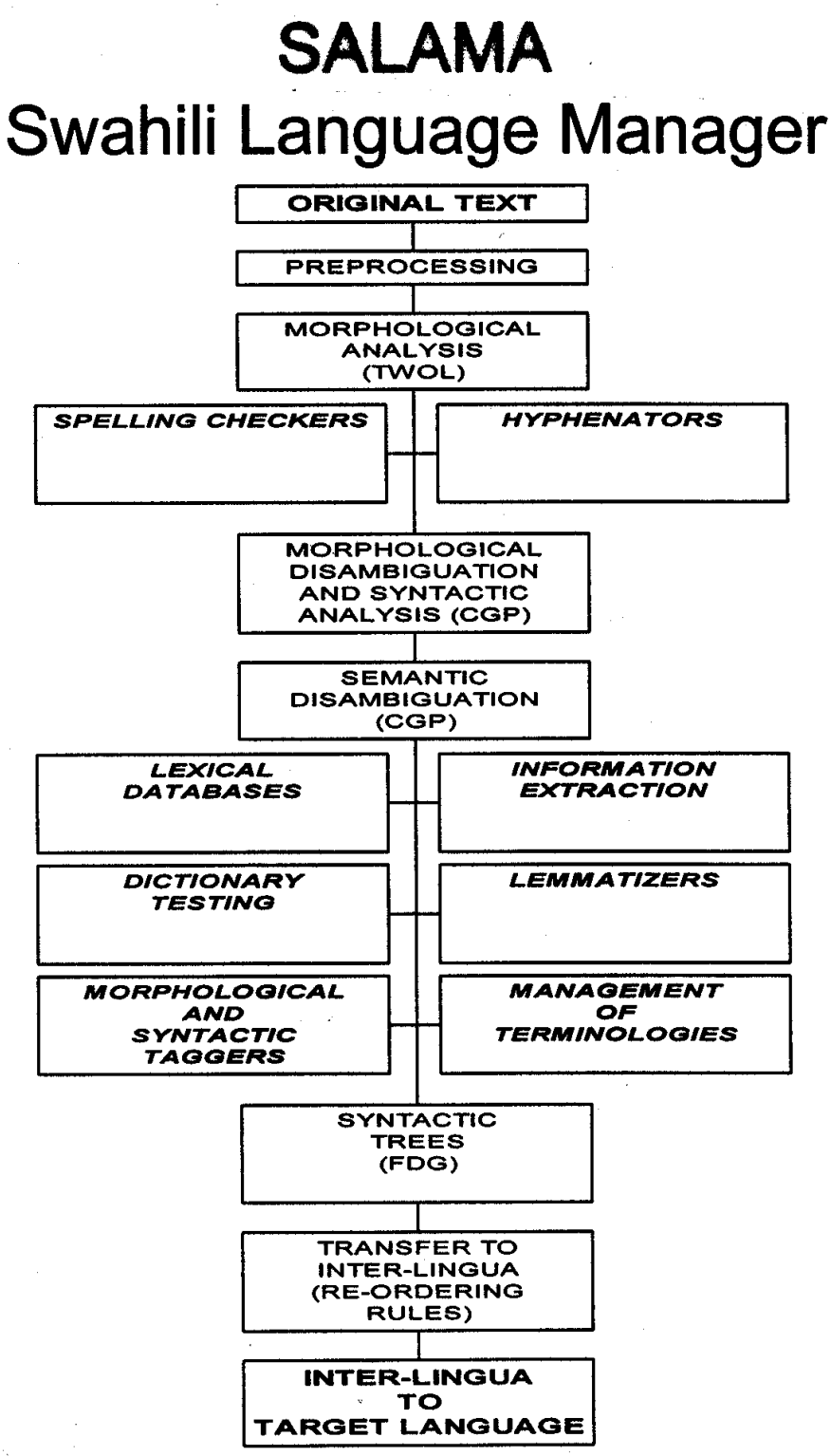
Voutilainen, A., J. Heikkilä and A. Anttila, 1992.

Constraint Grammar of English - A Performance-Oriented Introduction. Publications No. 21. Department of General Linguistics, University of Helsinki.

Voutilainen, A. and Tapanainen, P. 1993.

Ambiguity resolution in a reductionistic parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics. EACL-93.* pp. 394-403, Utrecht, Netherlands.

APPENDIX 1


SALAMA
Swahili Language Manager
ORIGINAL TEXT
PREPROCESSING
MORPHOLOGICAL ANALYSIS (TWOL)
SPELLING CHECKERS
HYPHENATORS
MORPHOLOGICAL DISAMBIGUATION AND SYNTACTIC ANALYSIS (CGP)
SEMANTIC DISAMBIGUATION (CGP)
LEXICAL DATABASES
INFORMATION EXTRACTION
DICTIONARY TESTING
LEMMATIZERS
MORPHOLOGICAL AND SYNTACTIC TAGGERS
MANAGEMENT OF TERMINOLOGIES
SYNTACTIC TREES (FDG)
TRANSFER TO INTER-LINGUA (RE-ORDERING RULES)
INTER-LINGUA TO TARGET LANGUAGE

APPENDIX 2

Demo of SALAMA
Swahili Language Manager

Original sentence

Nyakati hizi kila mtafiti analazimika kupambana na wafadhili ili apate mbinu za kufanya kazi ya utafiti.

Phase 1: Normalization

First the text is normalised with a program, which performs a number of modifications to the original text. The purpose of this program is to make explicit and systematize all features in text. For example, capital letters may occur in text in various roles. A word-initial capital letter may indicate a proper name, while in the beginning of a sentence it is found in any word. Words written with capital letters may be acronyms (e.g. *SALAMA*), but sometimes whole sentences are found written in capital letters. A full stop may indicate the end of a sentence, or it may be part of abbreviation (for example *e.g.*). Diacritics and punctuation marks are normally not part of words. All such features in text are identified and marked systematically. Note that all capital letters are converted to lower case, but in the letters where the capital letter is an orthographical feature, information of it is retained by placing and asterisk in front of the letter. Below is the sentence after having been modified by the text normalisation program.

*nyakati
hizi
kila
mtafiti
analazimika
kupambana
na
wafadhili
ili
apate
mbinu
za
kufanya
kazi
ya
utafiti
.$

Phase 2: Morphological analysis

The text is analysed with the Two-Level Morphological Parser, originally designed by Kimmo Koskenniemi (1983). Note that each word-form gets all possible interpretations, also those which are not correct in the context. First is given the base form of the word. Then follow a number of 'tags', i.e. abbreviations of lexical and grammatical features, such as IMP (imperfective), V (verb), SV (verb with one argument = subject + verb), SVO (subject + verb + object) AR (Arabic origin), 7/8-SG (noun class 7/8 in singular), etc. Also an English gloss is given.

```
"<*nyakati>"
    "wakati"  11/10-PL N AR ' time '
"<hizi>"
    "hizi"  IMP V AR SV ' put to shame '
    "hizi"  <kwisha V AR SV ' put to shame '
    "hizi"  PRON DEM :hV 9/10-PL ' this '
"<kila>"
    "kila"  ' every ' A-UNINFL
"<mtafiti>"
    "tafiti"  SBJN VFIN ½-SG3 OBJ V AR SV SVO ' do research '
    "tafiti"  SBJN VFIN ½-PL2-SP V AR SV SVO ' do research '
    "mtafiti"  ½-SG N AR ' researcher '
"<analazimika>"
    "lazimika"  ½-SG3-SP VFIN PR:na V AR SV ' be necessary / be forced to '
STAT
"<kupambana>"
    "pambana"  INF  V SV SVO ' decorate / fight with ' REC
"<na>"
    "na"  CC ' and / with ' @CC
"<wafadhili>"
    "fadhili"  SBJN VFIN ½-PL2 OBJ V AR SV SVO ' be generous / donate '
    "fadhili"  SBJN VFIN ½-PL3 OBJ V AR SV SVO ' be generous / donate '
    "fadhili"  SBJN VFIN ½-PL3-SP V AR SV SVO ' be generous / donate '
    "fadhili"  ½-SG2-SP VFIN PR:a V AR SV SVO ' be generous / donate '
    "fadhili"  3/4-SG-SP VFIN PR:a V AR SV SVO ' be generous / donate '
    "fadhili"  11-SG-SP VFIN PR:a V AR SV SVO ' be generous / donate '
    "fadhili"  ½-PL3-SP VFIN PR:a V AR SV SVO ' be generous / donate '
    "mfadhili"  ½-PL N AR ' donor / sponsor / patron '
"<ili>"
    "ili"  AR **CLB CONJ  ' so that / in order to ' @CS
"<apate>"
    "pata"  SBJN VFIN ½-SG3-SP V SV SVO ' get '
"<mbinu>"
    "mbinu"  9/10-NI-SG N ' means '
```

```
    "mbinu"  9/10-NI-PL N ' means '
"<za>"
    "za"  9/10-PL GEN-CON
"<kufanya>"
    "fanya"  INF  V SV SVO ' do / make '
"<kazi>"
    "kazi"  9/10-0-SG N ' work '
    "kazi"  9/10-0-PL N ' work '
"<ya>"
    "ya"  3/4-PL GEN-CON
    "ya"  9/10-SG GEN-CON
    "ya"  5/6-PL GEN-CON
    "ya"  5/6-PL ' to be '
"<utafiti>"
    "tafiti"  SBJN VFIN 3/4-SG OBJ V AR SV SVO ' do research '
    "tafiti"  SBJN VFIN 11-SG OBJ V AR SV SVO ' do research '
    "tafiti"  SBJN VFIN ½-SG2-SP V AR SV SVO ' do research '
    "tafiti"  SBJN VFIN 3/4-SG-SP V AR SV SVO ' do research '
    "tafiti"  SBJN VFIN 11-SG-SP V AR SV SVO ' do research '
    "utafiti"  11-SG N AR HC ' research '
"<.$>"
```

Phase 3: Constraint Grammar Parsing

Constraint Grammar Parser (CGP), originally designed by Fred Karlsson and implemented by Pasi Tapanainen, performs two important functions. The first is the elimination of such interpretations from the morphological analysis which are not correct in that context. This operation is performed by the morpho-syntactic disambiguation rules, which operate mainly by selecting the correct one, or by constraining the wrong interpretations. If need be, also syntactic tags are added. The tag, preceded by @, shows the syntactic function of the word, and also shows the direction (right or left) of its immediate head. The distance of the head is not, however, indicated.

```
"<*nyakati>"
    "wakati"  11/10-PL N AR ' time ' @TIME
"<hizi>"
    "hizi"  PRON DEM :hV 9/10-PL ' this ' @<ND
"<kila>"
    "kila"  AR ' every ' A-UNINFL  @AD-A>
"<mtafiti>"
    "mtafiti"  ½-SG N AR ' researcher ' @SUBJ
"<analazimika>"
    "lazimika"  ½-SG3-SP VFIN PR:na V AR SV ' be necessary / be forced to '
```

```
 STAT  @FMAINV
"<kupambana>"
      "pambana"  INF  V SV SVO ' decorate / fight with ' REC  @-FMAINV-n
"<na>"
      "na"  CC ' and / with ' @CC
"<wafadhili>"
      "mfadhili"  ½-PL N AR ' donor / sponsor / patron ' @I-OBJ
"<ili>"
      "ili"  AR **CLB CONJ  ' so that / in order to ' @CS
"<apate>"
      "pata"  SBJN VFIN ½-SG3-SP V SV SVO ' get ' @FMAINVtr>
"<mbinu>"
      "mbinu"  9/10-NI-PL N ' means ' @OBJ
"<za>"
      "za"  9/10-PL GEN-CON @<NOM
"<kufanya>"
      "fanya"  INF  V SV SVO ' do / make ' @-FMAINV-n
"<kazi>"
      "kazi"  9/10-0-SG N ' work ' @OBJ
"<ya>"
      "ya"  9/10-SG GEN-CON @<NOM
"<utafiti>"
      "utafiti"  11-SG N AR HC ' research ' @<P
"<.$>"
```

Phase 4: Semantic disambiguation

Although morphological ambiguity can be satisfactorily resolved with morpho-syntactic disambiguation rules, a large amount of ambiguity still remains in semantic functions of the words. Therefore word-sense disambiguation has to be performed, especially if the system will be used for Machine Translation (MT) applications. Since semantic features of words can be marked into the Two-Level lexicon, word-sense disambiguation can be performed with the Constraint Grammar Parser. The result after word-sense disambiguation is shown below.

```
"<*nyakati>"
      "wakati"  11/10-PL N AR ' time ' @TIME
"<hizi>"
      "hizi"  PRON DEM :hV 9/10-PL ' this ' @<ND
"<kila>"
      "kila"  AR ' every ' A-UNINFL  @AD-A>
"<mtafiti>"
      "mtafiti"  ½-SG N AR ' researcher ' @SUBJ
"<analazimika>"
```

```
    "lazimika"  ½-SG3-SP VFIN PR:na V AR SV ' be forced to ' STAT
@FMAINV
"<kupambana>"
    "pambana"  INF  V SV SVO ' fight with ' REC  @-FMAINV-n
"<na>"
    "na"  CC ' with ' @CC
"<wafadhili>"
    "mfadhili"  ½-PL N AR ' sponsor  ' @I-OBJ
"<ili>"
    "ili"  AR **CLB CONJ ' in order to ' @CS
"<apate>"
    "pata"  SBJN VFIN ½-SG3-SP V SV SVO ' get ' @FMAINVtr>
"<mbinu>"
    "mbinu"  9/10-NI-PL N ' means ' @OBJ
"<za>"
    "za"  9/10-PL GEN-CON @<NOM
"<kufanya>"
    "fanya"  INF  V SV SVO ' do  ' @-FMAINV-n
"<kazi>"
    "kazi"  9/10-0-SG N ' work ' @OBJ
"<ya>"
    "ya"  9/10-SG GEN-CON @<NOM
"<utafiti>"
    "utafiti"  11-SG N AR HC ' research ' @<P
```

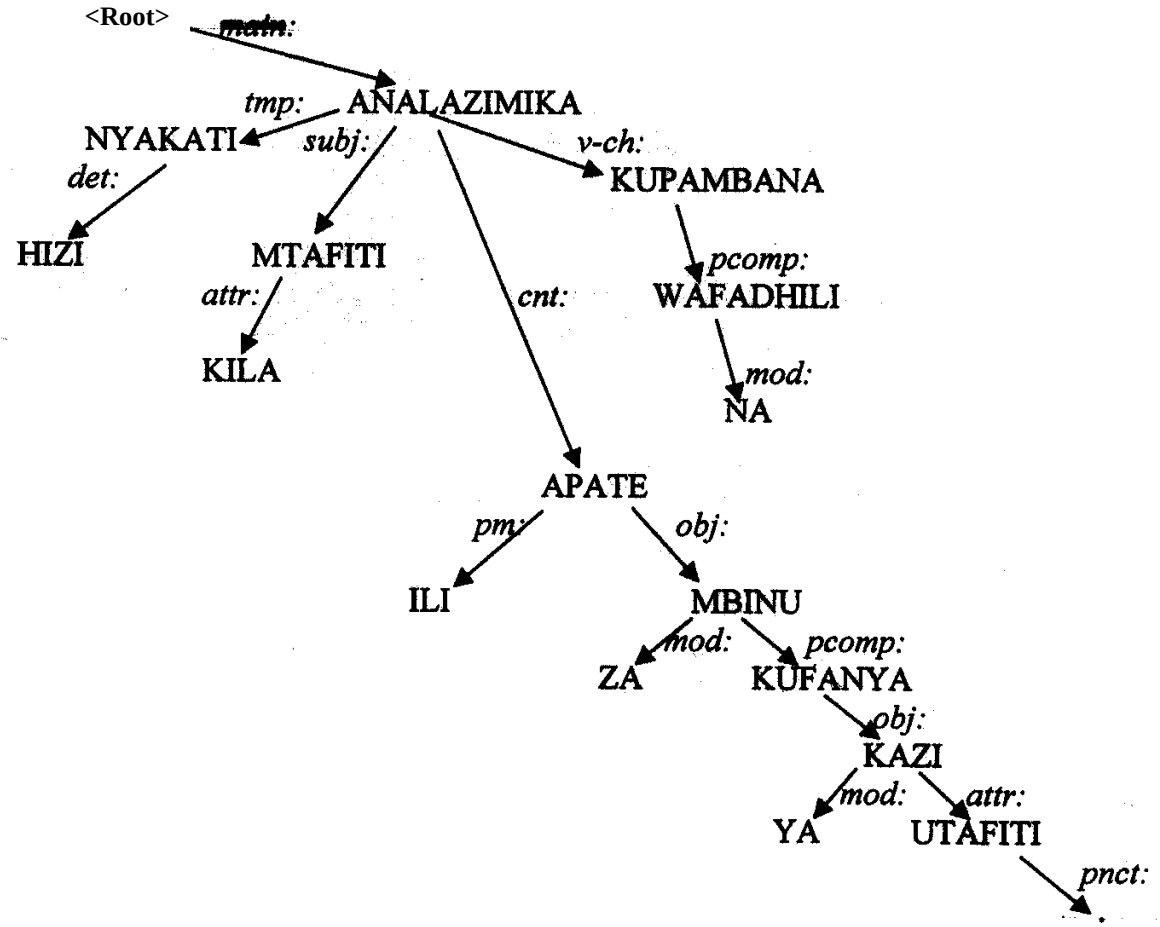
Phase 5: Syntactic tree

The limitation of the Constraint Grammar Parser is that it is not able to build syntactic trees, because it does not show the precise dependence structure of the constituents. For this purpose, a Functional Dependency Grammar (FDG) parser (see above) is used. The result of this parser is shown below. Functional labels (tags ending with a colon) show syntactic functions of the words, but, more importantly, links are built between various constituents (i.e. words) in the sentence. In the representation below, each word of the sentence is numbered, and in the end of each functional label there is the number of the word to which it is linked. The result is a syntactic tree. The description may be difficult to read, but for computational processing it is ideal. Also morpho-syntactic tags are assigned to words.

a. Syntactic tree in a machine-readable format.

```
0
1  *nyakati              wakati      tmp:>5      @TIME 11/10-PL N AR ' time
                                                 '
2  hizi          hizi    det:>1      @<ND PRON:DEM hV 9/10-PL ' this '
3  kila          kila    det:>4      @AD-A> AR A-UNINFL ' every '
4  mtafiti       mtafiti subj:>5     @SUBJ ½-SG N AR ' researcher '
5  analazimika   lazimika main:>0    @FMAINV ½-SG3-SP VFIN:PRna V
                                      AR SV ' be forced to ' STAT
6  kupambana     pambana mod:>5      @-FMAINV-n INF V SV SVO '
                                      fight with ' REC
7  na            na      cc:>8       @CC CC ' with '
8  wafadhili     mfadhili obj:>6     @I-OBJ ½-PL N AR ' sponsor '
9  ili           ili     pm:>10      @CS AR**CLB CONJ ' in order to '
10 apate         pata    cnt:>6      @FMAINVtr> SBJN VFIN ½-SG3-SP V
                                      SV SVO ' get '
11 mbinu         mbinu   obj:>10     @OBJ 9/10-NI-PL N ' means '
12 za            za      mod:>11     @<NOM 9/10-PL GEN-CON
13 kufanya       fanya   pcomp:>11       @-FMAINV-n INF V SV SVO ' do '
14 kazi          kazi    obj:>13     @OBJ 9/10-0-SG N ' work '
15 ya            ya      **:>16      @<NOM/10-SG GEN-CON
16 utafiti       utafiti attr:>14    @<P 11-SG N AR HC ' research '
   .$
```

b. Syntactic tree in visual output.



Phase 6: Swahili to InterLingua

In this phase, the morpho-syntactic information of Swahili is transformed into the format required by InterLingua. This takes place in two phases. First, the constituent re-ordering rules are applied to meet the constituent order of the InterLingua. Then the Swahili tags are modified to fit the tag-set of the InterLingua. Note that the tags indicating the noun class specification and concordial marking will be deleted, because such features do not apply in InterLingua.

a. Tag transformation rules are applied.

N PL ' time ' @TIME
PRON DEM PL ' this ' @<ND
A-UNINFL' every ' @AD-A>
SG N AR ' researcher ' @SUBJ
SG3-SP VFIN PR:na V AR SV ' be forced to ' STAT @FMAINV
INF V SV SVO ' fight with ' REC @-FMAINV-n
CC ' with ' @CC
PL N ' sponsor ' @I-OBJ
**CLB CONJ ' in order to ' @CS
SBJN VFIN ½-SG3-SP V SV SVO ' get ' @FMAINVtr>
PL N ' means ' @OBJ
PL GEN-CON @<NOM
INF V SV SVO ' do ' @-FMAINV-n
SG N ' work ' @OBJ
SG GEN-CON @<NOM
SG N ' research ' @<P
"<.$>"

b. Then constituent re-ordering rules are applied.

In this phase the elements of the constituents are re-ordered to meet the grammatical
structure of the Inter-Lingua.

PRON DEM PL ' this ' @>ND
N PL ' time ' @TIME
A-UNINFL ' every ' @AD-A>
SG N AR ' researcher ' @SUBJ
SG3-SP VFIN PR:na V AR SV ' be forced to ' STAT @FMAINV
INF V SV SVO ' fight with ' REC @-FMAINV-n
CC ' with ' @CC
PL N ' sponsor ' @I-OBJ
**CLB CONJ ' in order to ' @CS

SBJN VFIN ½-SG3-SP V SV SVO ' get ' @FMAINVtr>
PL N ' means ' @OBJ
(PL GEN-CON @<NOM)
INF V SV SVO ' do ' @-FMAINV-n
(SG GEN-CON @<NOM)
SG N ' research ' @>P
SG N ' work ' @OBJ
"<.$>"

Phase 7: InterLingua to natural language

These times every researcher is forced to fight with sponsors in order to get means to do research work.