Hannes Hirzel, Univ. of Zürich,

Mary Esther Kropp Dakubu, Univ. of Ghana,

Dorothee Beermann, Jonathan Brindle and Lars Hellan, NTNU:

“Porting lexicon files from Toolbox into LKB-grammars: A case study for a grammar of Ga”

Toolbox-LKB-Link

- The Toolbox lexical database for Ga
- Transformation
- The LKB-TDL file
- Specific problems: Unicode, tone
- Outlook

What is Toolbox?

- An editor and database program for lexical data
- A corpus building tool
- A morphological annotation tool
- Data is kept in UTF8 encoded textfiles
- Import / Export functions
  - Cc - consistent changes tool
- Freeware: `www.sil.org/computing/toolbox`

Data format of Toolbox files

- The lexical data is kept in flat file text files.
- The files are encoded in Unicode (UTF8)
- Hierarchies are possible
- As it is 'text only' the file format is sustainable;
  - Still usable in 10 ... 20 years
  - Processing the data is easy (Unix command line tools, Lisp list processing)

Why use Toolbox?

- Lexical data in this format is available
- Editing lexical entries in Toolbox is easier that in a TDL file
- Toolbox includes a formatting program: printout of the entries in dictionary form.

Ga

- Kwa language spoken in Ghana
- Printed dictionary available
  (ed. Mary-Esther Kropp Dakubu, Univ. of Ghana)
- Electronic dictionary in Toolbox format with 1700 entries.

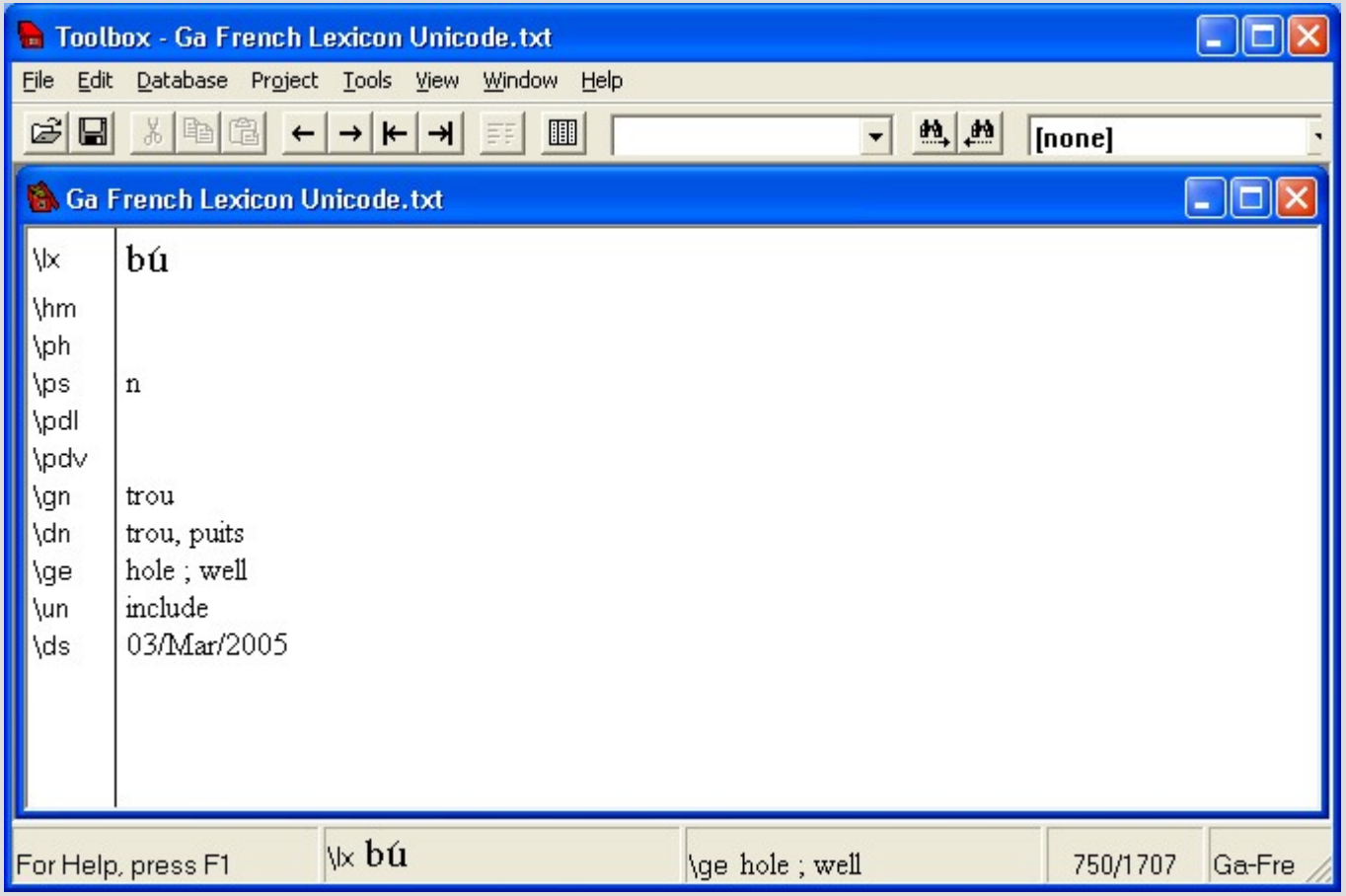
How does the Ga lexicon look like in Toolbox format

- Toolbox format = SFM format /FOSF format = tagged text

- Each field/ text element is marked with a preceding tag which begins with a backslash

\lx bú	\lx = lexeme
\ps n	\ps = part of speech
\gn trou	\gn = gloss french
\ge hole ; well	\ge = gloss english

Toolbox: Single entry view

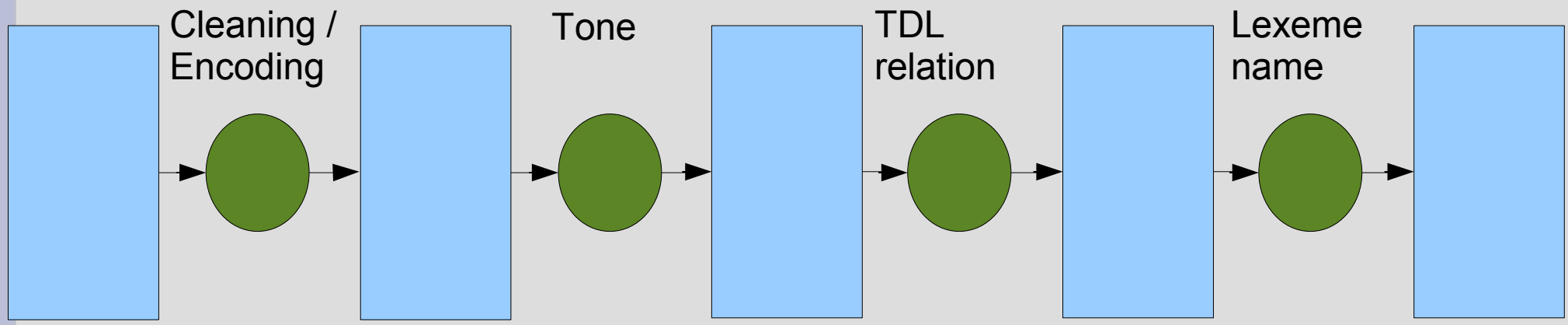


TDL lexicon

```
bú := noun-lexeme &
[STEM <"bú">,
PHON <"bú">,
ENGL-GLOSS <"hole ; well", "trou">,
SYNSEM.LKEYS.KEYREL.PRED "_bú_n_rel"].
```

Toolbox export processes

- Export processes may be chained
  - The individual process step is simple
- Cf. Unix pipes



What is cc (consistent changes)

- A little language for writing filters like sed or awk
- Exchange of strings (groups are possible)
  - `'aString'  > 'anotherString'`
- Variables
- Control structures (condition, loop)

Steps (“processes”)

1: Unicode to ASCII encoding

2: Duplicate lexeme field

The duplicate is named '\phon' (phonetics)

3: Eliminate tone

4: Construct TDL type

5: Create unique name for lexeme

(include homograph number in lexeme name)

6: Reformat for TDL

Step 1: Unicode to ASCII

"â" > "aHL"
"ã" > "aN"
"ã" > "aN"
"ã́" > "aNH"

"ε" > "E"
"έ" > "EH"
"ὲ" > "EL"
"ἓ" > "EHL"

Step 2: Duplicate lexeme field

- Specific for this Ga dictionary
  - Duplicate the entry lexeme
  - Copy the information into a \phon field
- The lexeme entry is tone marked.
- This step allows elimination of tone in the lexeme name while keeping the tone information in the phon field

Result of step 2

\lx buH

\phon buH

\ps n

\gn trou

\ge hole ; well

Step 3: Eliminate tone information in lexeme field

```
c -------------------------------------------------------
group(main)

'\lx '   > '\lx '          c copy what we have found
                           c in the input stream (i.e. '\lx') to the
                           c output stream
 use(lxGroup)              c switch to the other group.



c -------------------------------------------------------
group(lxGroup)

'H' > ''                  c H gets replaced by the empty string
                          c (i.e. The empty string)
'L' > ''                  c L: the same

                          c when we find that the next field starts,
                          c  we switch back to the main group.
'\'  > '\'
        use(main)
```

Result of step 3

\lx bu

\phon buH

\ps n

\gn trou

\ge hole ; well

Step 4: Form TDL type

```
'\ps V'   nl      >   next
'\ps v'   nl      >   dup
                      '\tdlType verb-lexeme'
                      nl

'\ps N'   nl      >   next
'\ps n'   nl      >   dup
                      '\tdlType noun-lexeme'
                      nl
```

Result of step 4

\lx bu

\phon buH

\ps n

\tdlType noun-lexeme

\gn trou

\ge hole ; well

Step 5: Add homograph number and TDL relation

```
...
"\tdlRelation "
'"_'
out(valueLexeme)
"_"
out(valuePartOfSpeech)
'_rel"'
...
```

Result of step 5

\lx bu

\phon buH

\ps n

\tdlType noun-lexeme

\gn trou

\ge hole ; well

\tdlRelation "_bu_n_rel"

Step 6: Form TDL

```
Begin   >  Initialisation
define(output_lexical_entry) > incr(cntNoOfEntries)
                                out(valueLexeme)' := '
                                out(tdlType)
                                ' &' nl
group(main)
'\lx '        >        endstore
                       do(output_lexical_entry)
                       store(valueLexeme)
'\tdlType '            >  store(tdlType)

'\tdlRelation '        > store(tdlRelation)

endfile        > do(output_lexical_entry)
                 endfile
```

Result of step 6

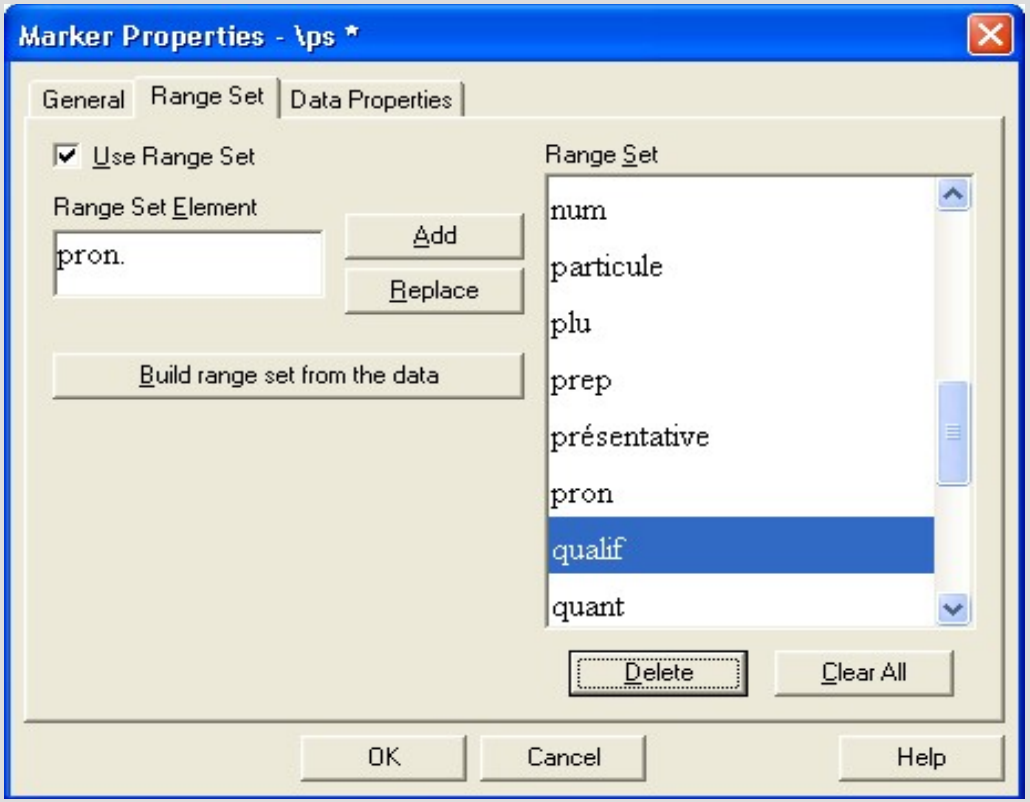
```
bu := noun-lexeme &
 [STEM <"bu">,
 PHON <"buH">,
 ENGL-GLOSS <"hole ; well", "trou">,
 SYNSEM.LKEYS.KEYREL.PRED "_bu_n_rel"].
```

What has to be adapted by the grammar writer

- Selection of fields
- Specification of TDL types
- Specification of TDL relations

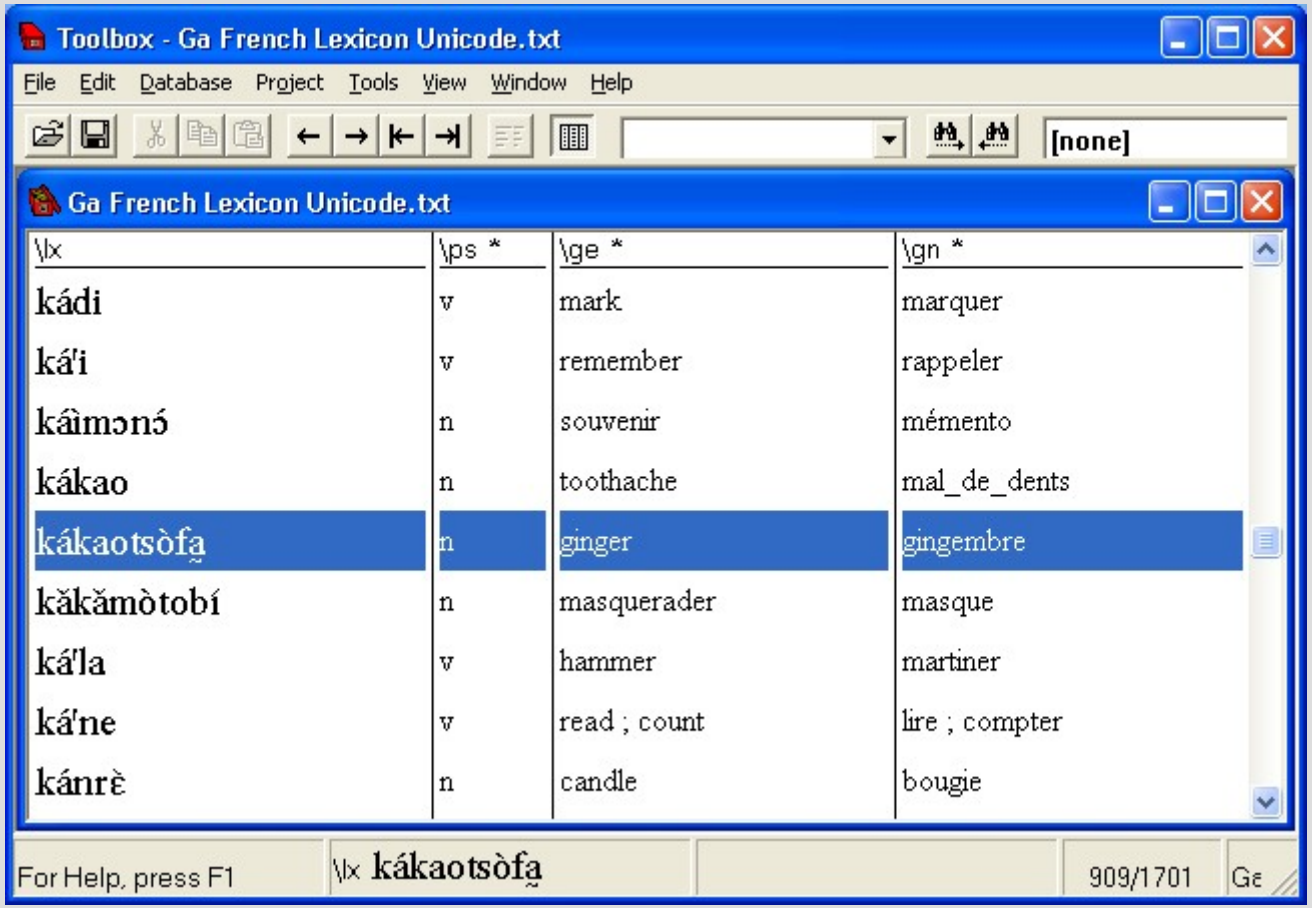
Questions: Markup hierarchies of lexicons

- How should a markup of a lexicon hierarchy look like?
- Toolbox allows consistency checks (range sets)



Cleaning the data

- It is easy to view and resort data in Toolbox



\lx	\ps *	\ge *	\gn *
kádi	v	mark	marquer
ká'i	v	remember	rappeler
kâìmɔnɔ́	n	souvenir	mémento
kákao	n	toothache	mal_de_dents
kákaotsòfa̰	n	ginger	gingembre
kăkămòtobí	n	masquerader	masque
ká'la	v	hammer	martiner
ká'ne	v	read ; count	lire ; compter
kánrɛ̀	n	candle	bougie

Summary

- Automatic acquisition of lexical data from Toolbox databases is possible
- Process can be fine-tuned by the grammar writer
  - Selection of fields
  - TDL types
  - TDL relations
- Toolbox is useful for editing large lexical databases and works with a data format which is sustainable (tagged text files - UTF8)

Outlook

- Instead of writing a Toolbox export function one could write an LKB import function
- Alternative Toolbox export function which generates SQL-insert statements for import in Postgres or other databases.
- More 'best practice' examples for type hierarchies needed.
- Using Unicode (UTF8) in LKB will facilitate working with African languages.