From a dictionary towards the Hungarian Constructicon

eLex2023
27–29 June 2023

Bálint Sass

Hungarian Research Centre for Linguistics, Institute for Lexicology
sass.balint@nytud.hu

‘asztal’ (table)

‘fehér’ (white)

'fehér asztal' (white table)
literally

'fehér asztal' ('white table' = **table set for meals**)
a construction!

outline

1. definition of construction
2. single-element constructions
3. treat all constructions in a unified way
4. initial dictionary $\rightarrow$ constructicon
5. querying the constructicon
6. the dynamic toolbox
7. examples
8. entry-query links
9. availability

outline

1. definition of **cxn**
2. single-element cxns
3. treat all cxns in a unified way
4. initial dictionary $\rightarrow$ **ccn**
5. querying the ccn
6. the dynamic toolbox
7. examples
8. entry-query links
9. availability

abbreviations

- cxn = construction
- ccn = constructicon
  inventory of cxns
- hcxn = head-construction
  an entry in a ccn
  $\sim$ by analogy to the term *headword*
- ccn-hu = the Hungarian Constructicon

definition of *construction* (cxn)

"learned pairings of form and function"

or

everything which is not fully compositional

(Goldberg, 2006)

- morphemes
- words
- fixed continuous combinations: *'fehér asztal'*
- more complex cxns (future work):
  having free slots, not continuous, variable word order ...

single-element units *are* cxns

cxns are often imagined as a multi-element unit
'get rid of X', 'sweep X under the rug', 'ad hoc'…

we emphasize:

single-element units are cxns as well

- words are cxns
  e.g. *'asztal'* (table)
- morphemes are cxns
  e.g. *'-ra/-re'* (onto) ← a Hungarian case marker

not a novel idea, still Goldberg (2006)

'asztal' (table)

a cxn!

'fehér' (white)

a cxn!

'fehér asztal' (white table)
literally – **not a cxn!**
just a compositional combination of two cxns

‘fehér asztal’ (‘white table’ = table set for meals)
a cxn, a non-compositional combination of two cxns

equal care for all kinds of cxns!

a step forward: treat all kinds of cxns in the same way

i.e. **include all cxns into one lexical resource**
regardless the number of their sub-elements

→ **the basic unit will be the cxn**

→ we integrate dictionaries and ccns into one resource

→ this allows us to grasp all connections between these units

equal care for all kinds of cxns!

a step forward: treat all kinds of cxns in the same way

i.e. **include all cxns into one lexical resource**
regardless the number of their sub-elements

→ **the basic unit will be the cxn**

→ we integrate dictionaries and ccns into one resource

→ this allows us to grasp all connections between these units

- *'asztal'*
- *'fehér'*
- *'fehér asztal'* – literally
- *'fehér asztal'* – idiomatically

equal care for all kinds of cxns!

a step forward: treat all kinds of cxns in the same way

i.e. **include all cxns into one lexical resource**
regardless the number of their sub-elements

→ **the basic unit will be the cxn**

→ we integrate dictionaries and ccns into one resource

→ this allows us to grasp all connections between these units

- *'asztal'* ✓
- *'fehér'* ✓
- *'fehér asztal'* – literally ✗
- *'fehér asztal'* – idiomatically ✓

this is an **important aspect** of our approach

lifting out cxns

consequently: **all cxns should be hcxns in our resource**

start from an existing dictionary, process the XML database, and
lift out cxns *hidden inside the "collocation" part of the entries*
= lift out the XML *subtree* representing the cxn and
create a new individual entry (a new hcxn) for it on its own

form of new hcxn := textual form of the original cxn

add cross-references
from the original place of the cxn to the newly created entry
$\rightarrow$ 14000 new entries added to the original 73000
result: both single-element and multi-element cxns as hcxns

a conceptually simple but **important step** of processing

lifting out cxns

```
<entry form="fehér"> (white)
  <def>...</def> (having the colour of fresh snow)
  ...
    <coll>
      <phr>fehér asztal</phr> (white table)
      <def>...</def> (table set for meals)
    </coll>
    ...
```

→

```
<entry form="fehér"> (white)
  <def>...</def> (having the colour of fresh snow)
  ...
    <xrcoll>fehér asztal</xrcoll>
    ...

<entry form="fehér asztal"> (white table)
  <def>...</def> (table set for meals)
</entry>
```

completeness and correctness

after the lifting step

- ✗ we do not think that ccn-hu is *complete* in any sense,
  it just contains quite a large amount of cxns:
  73000 single-element and 14000 multi-element units
- ✗ starting from a dictionary, the majority of hcxns will be correct,
  but clearly, ccn-hu may contain hcxns which are not *correct*,
  i.e. do not meet the requirement of non-compositionality
  – consider e.g. *'vasgolyó'* (iron ball)

(issues about improving the initial dictionary is outside the scope of this talk)

as a working hypothesis we consider the ccn-hu complete and correct,
i.e. we take it that a unit is included *if and only if* it is a cxn

how does querying work?

common solution for ccns (e.g. Swedish or Russian):
the user can choose from a predefined list of cxns

our approach is different:

we do not want to limit what the user can ask
we do not expect the user:

- to know what is a cxn and what is not
- to know the canonical form of the cxn
  – i.e. how cxns are formally represented in the ccn

  just let the user enter arbitrary query text and leave it to the cnn to find out which cxn (or cxns) are to be shown to the user

→ *task:* we have to extract cxns from the query text,
we have to break down the query text into cxns

the dynamic toolbox

the solution is: *the dynamic toolbox*

1. **analysed search**
2. **cxn-identification**
3. **dynamic referencing**
4. **virtual entries**

the dynamic toolbox allows the ccn
to give an answer to any queries to the best of its ability

→ linguistic analysis is needed for processing user queries

the dynamic toolbox algorithm

1. is the query a hcxn? → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

- **break down** the query into its linguistic elements: 2. & 5.
- **assemble** the relevant cxns from the elements: 3. & 6.

the dynamic toolbox algorithm

example 1/8

'asztal' (table)

1. **is the query a hcxn?** → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words . . .
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ real entry

(monomorphemic cxn, simple word)

the dynamic toolbox algorithm

'asztalos' ('table + *-s* suffix' = carpenter)

1. **is the query a hcxn?** → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ real entry

(non-compositional multimorphemic cxn)

the dynamic toolbox algorithm

'asztalra' ('table+onto' = onto table)

1. is the query a hcxn? → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. **morpho-analyse** *'asztal + ra'*
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ virtual entry: *'asztal'* (table) + *'-ra/-re'* (onto)

(compositional suffixed word → 2 cxns)

the dynamic toolbox algorithm

'faasztal' (wooden table)

1. is the query a hcxn? → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. **morpho-analyse** *'fa + asztal'*
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ virtual entry: *'fa'* (wooden) + *'asztal'* (table)

(compositional compound → 2 cxns)

the dynamic toolbox algorithm

'asztalfiókba' ('table+drawer+into' = into table drawer)

1. is the query a hcxn? → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. **morpho-analyse** *'asztal + fiók + ba'*
6. **merge morphemes to identify cxns** → ✓ *'asztalfiók'* is a hcxn
7. remaining morphemes → ✓

→ virtual entry: *'asztalfiók'* (table drawer) + *'-ba/-be'* (into)

(suffixed non-compositional compound → 2 cxns)

the dynamic toolbox algorithm

'fehér asztal' ('white table' = table set for meals)

1. **is the query a hcxn?** → ✓
2. tokenize
3. merge words to identify cxns → ✓
4. for remaining words …
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ real entry

(fixed continuous multiword cxn)

the dynamic toolbox algorithm

'három asztal' (three tables)

1. is the query a hcxn? → ✓
2. **tokenize** *'három + asztal'*
3. merge words to identify cxns → ✓
4. for remaining words …
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ virtual entry: *'három'* (three) + *'asztal'*

(compositional combination of 2 simple words → 2 cxns)

the dynamic toolbox algorithm

'fehér asztal mellett' ('white table around' = around table set for meals)

1. is the query a hcxn? → ✓
2. **tokenize** — *'fehér* + *asztal* + *mellett'*
3. **merge words to identify cxns** → ✓ — *'fehér asztal'* is a hcxn
4. for remaining words …
5. morpho-analyse
6. merge morphemes to identify cxns → ✓
7. remaining morphemes → ✓

→ virtual entry: *'fehér asztal'* + *'mellett'*

(compositional combination of a multiword cxn and a simple word → 2 cxns)

future work

handling more complex cxns
having free slots, not continuous, variable word order

so far: extract continuous word/morpheme sequences

from now on: extract **arbitrarily arranged** combinations of morphemes

possible steps: to achieve this . . .

1. a formal representation should be developed
   which may be built on dependency parsing of hcnxs and queries,
2. hcxns should be converted to this canonical representation,
3. an algorithm should be developed which is able to efficiently match
   the user query against the canonical form of cxns in the ccn

future work – examples

- **slot**
  'nem az ő asztala' ('not his table' = none of his business)

  → 2 cxns:
  'nem SLOT *asztal*+PS' ('not SLOT's table' = none of SLOT's business)
  'ő' (he)

- **not continuous**
  'munkában nem vesz részt'
  ('work+in not take part' = does not take part in work)

  → 3 cxns:
  'vesz részt SLOT+*ban'* (take part in SLOT)
  'munka' (work)
  'nem' (not)

entry-query links

a cross-referencing system

every word in the text of entries
functions as a link
to start a query that looks up the word itself in the ccn

'sárga' (yellow) $\leftrightarrow$ *'citrom'* (lemon)

thanks to the dynamic toolbox, every word form can be linked

database

the database of ccn-hu is structured as follows

- list of cxns
- a structured entry (an XML tree) for each cxn containing...
  – form and definition for lifted entries
  – whole dictionary entry for original headwords
- cross-references
  – from the original headword to the lifted cxn
  – entry-query links

availability

http://ccn.nytud.hu

username: `eLex2023`
password: `letssee`

feel free to try some queries and
contact me if you have questions or comments

Bálint Sass
sass.balint@nytud.hu

summary

1. definition of cxn	learned, non-compositional pattern
2. single-element units	. . . are cxns
3. treat all cxns in a unified way	multi-element ✓ **single-element** ✓
4. initial dictionary → ccn	lift out cxns → own hcxn for each
5. querying the ccn	**handle arbitrary query text**
6. **the dynamic toolbox**	ana-search, cxn-ident, dyn-ref, virt-entry
7. examples	how the dynamic toolbox works
8. entry-query links	cross-references for all words
9. availability	**`http://ccn.nytud.hu`** (`eLex2023/letssee`)

From a dictionary towards the Hungarian Constructicon

eLex2023
27–29 June 2023

Bálint Sass

Hungarian Research Centre for Linguistics, Institute for Lexicology
sass.balint@nytud.hu