**Participants:**

Bettina

Max

Maria

Christian

**Defining new module vocabulary**

<https://docs.google.com/document/d/1r6P9pTibsIy1iug1ptRtI-h9oVoA9N2wwDJJVaXt56s/edit?usp=sharing>

**Automatic generation of ontolex:Form instances**

Consider the case of the English plural noun. As a basic entry it could be represented as:

<#cat> a ontolex:Word ;

ontolex:canonicalForm [ontolex:writtenRep "cat"@en ;

lexinfo:number lexinfo:singular ;

morph:consistsOf [a morph:StemMorph ;

morph:representation "cat"@en]] ;

ontolex:otherForm [ontolex:writtenRep "cats"@en ;

lexinfo:number lexinfo:singular ;

morph:consistsOf [a morph:StemMorph ;

morph:representation "cat"@en] ,

[a morph:AffixMorph ;

morph:representation "s"@en]] .

This could be replaced with a productive pattern which uses the InflectionParadigm

element in place of the Form to represent the morphology as follows:

<#cat> a ontolex:Word ;

ontolex:canonicalForm [ontolex:writtenRep "cat"@en] ;

ontolex:morphologicalPattern <#regular\_english\_noun> .

<#english\_plural> morph:belongsToMorphPattern <#regular\_english\_noun> ;

lexinfo:number lexinfo:plural ;

morph:consistsOf [a morph:StemMorph] ,

[a morph:AffixMorph ; morph:representation "s"@en] .

<#english\_genitive> morph:belongsToMorphPattern <#regular\_english\_noun> ;

lexinfo:number lexinfo:singular; lexinfo:case lexinfo:genitive ;

morph:consistsOf [a morph:StemMorph] ,

[a morph:AffixMorph ; morph:representation "’s"@en] .

In this modelling the elements that the InflectionParadigm consists of are similar to those of the form, however, the representation of the stem is not given and instead can be inferred from the lexical entry. We are currently working on sets of restrictions to make this more flexible and a method for representing transfix and simulfix morphs in a generic manner.

CC: a MorphPattern should not support allomorphs within a single MorphPattern

We have to accommodate for three different use cases:

1. Inflection table that contains all possible endings of the stem with allomorphs but no disambiguation between then

|  | **Third declension paradigm**  **(*i*-stem nouns)** | | | | |
| --- | --- | --- | --- | --- | --- |
| **Masculine &**  **Feminine** | | **Neuter** | |  |
| **Singular** | **Plural** | **Singular** | **Plural** |  |
| **Nominative** | — | -ēs | — | -ia |  |
| **Accusative** | -em  -im | -ēs  -īs |  |
| **Genitive** | -is | -um  (-ium) | -is | -um  (-ium) |  |
| **Dative** | -ī | -ibus | -ī | -ibus |  |
| **Ablative** | -e  -ī |  |

1. The need to represent all forms of a lexical entry. The full paradigm of one particular word (no clue if that word is a lexical entry or form)

| ***[amnis, amnis](https://en.wiktionary.org/wiki/amnis#Latin)***  **stream, torrent m. (pure)** | | ***[pars, partis](https://en.wiktionary.org/wiki/pars#Latin)***  **part, piece f. (mixed)** | | ***[animal, animālis](https://en.wiktionary.org/wiki/animal#Latin)***  **animal, living being n. (pure)** | |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parisyllabic rule** | | **Double consonant rule** | | **Special neuter ending** | |  |
| **Singular** | **Plural** | **Singular** | **Plural** | **Singular** | **Plural** |  |
| **Nominative** | amnis[[i]](https://en.wikipedia.org/wiki/Latin_declension#cite_note-nomsg2-25) | amnēs | pars[[i]](https://en.wikipedia.org/wiki/Latin_declension#cite_note-nomsg2-25) | partēs | animal[[i]](https://en.wikipedia.org/wiki/Latin_declension#cite_note-nomsg2-25) | animālia |
| **Accusative** | amnem  amnim | amnēs  amnīs | partem  partim | partēs  partīs |
| **Genitive** | amnis | amnium | partis | partium | animālis | animālium |
| **Dative** | amnī | amnibus | partī | partibus | animālī | animālibus |
| **Ablative** | amne  amnī | parte  partī |

1. For generation process we need a table with any ambiguity

Goal to generate all ontolex:Form instances that apply to a particular paradigm

MI: Note, that different languages and different linguistic traditions describe morphology and paradigms in different ways. Most notably, there are 3 established models of morphology:

1. Item and Arrangement, IA
2. Item and Process, IP
3. Word and Paradigm, WP

*Paradigm tables*, like the Latin one above are common for the 3rd one, WP. They are rooted in the *European linguistic traditions*, since they are suited well for traditional Indo-European languages like Ancient Greek and Sanskrit. Since there are a lot of fusion, it is quite difficult and often quite useless to try and separate affixes and endings.

For *agglutinative languages*, it is more common to *separate affixes and simply give them as morphological information (so, like the first table)*, so, IA model. And it is often tricky to give full forms in paradigms because there are a lot of combinations (even for inflection). Example: a common Turkic verbal inflection has more than 100 forms. One can argue that some of these forms are derivational, but another one can argue against that.

Also, for some agglutinative languages and other languages with complex morphophonological processes, it is often very difficult to reconstruct forms from tables like these.

Take, for instance, Turkic once again, or Uralic languages, with vowel synharmony. Vowels in affixes affect vowels in the root. This can be, in principle, be modeled with several variants of a root, but this is not the only case these processes happen.

Morphologies for these languages are often described with various IP models. Computationally, usually with some variant of finite state transducers (xfst, foma, 2-level morphology (pc-kimmo), etc.)

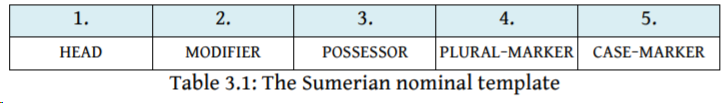
The problem with this approach is that for specifying even a small part of morphology, sometimes a lot of rules should be implemented. Also, the rules are not necessarily regular, most are context-free, and sometimes even context-free.

Example: Paul Meurer implementation of Abkhaz morphology spans over thousands lines of rules, and it takes 64Gb or RAM to compile the automata (and a lot (but less) to load).

**An approach for the automatic generation (CC + Max)**

* All names suggested below are preliminary. Unclear to us whether they overlap with previously proposed notions such as “Paradigm” or “Pattern”.
* *morph:Paradigm*: theoretically motivated type of inflection (~ “*-i-stem nouns”* in the first Latin example above). Cf. morphemes, theoretical construct, may contain allomorphs, exceptions, etc. Can have multiple inflection types linked to it (at least one).
  + Idea: a paradigm is a generalization over different inflection types. In particular, different assimilation rules may apply to different stem forms. In a paradigm, these are not distinguished
* *morph:InflectionType*: a group of flexia for a group of words. No allomorphy. (~ “*Parisyllabic rule*”, “*double consonant rule*”, etc.in the second Latin example table above)
  + Idea: If we associate an inflection type with a stem, we can unambiguously generate all forms
  + Paradigms are collections of inflection types
* *morph:Rule* (?pattern?)
  + Inflection types are collection of rules, that associate grammatical features (say, case and number) with a particular string transformation
  + Datatype property *morph:pattern*: A transformation instruction (Perl-like regex) that describes how to generate the inflected form from the stem, e.g., “*s/is$/em*/” for generating *amnem* (second Latin example above) from *amnis*
    - If there is more than one morph:pattern, this means that two equivalent transformations can apply, e.g., for *amnim* from *amnis*. Normally, this should be one.
  + Rules roughly correspond to the entries of the first Latin table above.
* If paradigms are not fully fleshed out but described by examples, create an inflection type for each example (say, *amnis*, *pars* etc. for the second Latin example above)
  + in the inflection type we have datatype properties
    - *morph:example* (no more than 1): one dictionary form of the word that is inflected according to this inflection type (string)
  + Rules (or pattern, at least 1):
    - Grammatical categories (nom, sg, etc.)
    - PCRE-compatible Regex as a string
    - Example (string)

**Christian: Ideas from a follow-up discussion with Max, 2019-06-26**

* We discussed InflectionType for inflected languages so far, only, using Latin (etc.) as an example.
* Suggested refinement to account for agglutinative languages
  + For an inflective language with fused/opaque morphemes that map to groups of grammatical features rather than individual features, one inflection type can represent a contextual variant of a paradigm. If a stem is associated with one specific inflection type, all its forms can be generated.
  + For agglutinative languages (or inflected languages with less fusion), where a clear form-function mapping between individual morphemes can be observed, and a fixed order applies (e.g., Turkish, Sumerian), InflectionType can be used to represent a specific “Slot” associated with one particular feature. Cf. Sumerian (Gábor Zólyomi 2016, An introduction to the grammar of Sumerian, <https://core.ac.uk/download/pdf/51326989.pdf>, Version 4/21/16):

1: Stem (actually, the syntactic head, see 2)

2: modifier (in Zolyomi’s model, adjectives and other modifiers are placed here, so this is a syntactic rather than a morphological position, but the following morphemes are attached to the modifier, if any such does exist)

3: -0 (none), -ŋu (1.Sg.Poss), -zu (2.Sg.Poss), -ane (3.Sg.H.Poss), etc.

4: -0 (sg), -enê (pl)

5: -0 (abs), -e (erg), -ak (gen), etc.

Examples: (actually generated forms in [...])

lugalŋu [lugal=ŋu=0=0] “my lord” (abs.sg.)

lugalene [lugal=0=enê=0] “lords” (abs.pl.)

lugalza [lugal=zu=0=ak] “your lord” (gen.sg.)

Within the morphology module, slots 3-5 would need to be covered only, where slots 3, 4 and 5 are represented by one InflectionType each. These inflection types then come with a natural order, i.e., some *next* property. Note that this is the order of application, not the sequential order of morphemes. The first element would always be the stem, and the order should reflect the relative distance of a morpheme from the stem, regardless of whether morphemes precede or follow the stem (or both, or neither). In this way, the “order of application” allows to support circumfixes and other “non-sequential” morphology.

Note: the generated forms in the example above represent “deep” morphology as they are subject to subsequent assimilation processes (resp., in this case, a mapping to an orthographic representation). Proper full form generation can be implemented by inflection typ(e variant)s that condition each other, e.g., from every slot 4 inflection type a link to both a final variant of slot 5 (with -a for genitive) and a non-final variant of slot 5 (with -ak for genitive).

Note2: another, and possibly more elegant implementation would be to implement assimilation rules by regular expressions in the last inflection type(s), e.g., with a replacement rule like “s/=ak$/a/”. If we encourage such a use of “inflection type”, we should probably rename “inflection type” to “rule set” or the like.

Note3: the -0 morphemes above correspond to a regular expression that does not modify the stem ;) Used informal representation for illustration only.

**Todo**: example from a non-assimilating agglutinative language (Finnish?, Inuktitut?)

Note4: the Sumerian example is interesting, because our modelling supports recursive morphology: a genitive NP (with its own slots 0-5) can be a (syntactic) modifier, but then carry the (morphological) slots of its syntactic head.

*e2 lugal-na* “his master’s temple” (abs.sg.)

*[e [lugal=ane=0=ak]=0=0=0]*

*[house [king=3.Sg.H.Poss=Sg=Gen]=no-Poss=Sg=Abs]*

*e2 lugal-na-ke* “of his master’s temple” (gen.sg.)

*[e [lugal=ane=0=ak]=0=0=e]*

*[house [king=3.Sg.H.Poss=Sg=Gen]=no-Poss=Sg=Gen]*

In our modelling, we can just state that after slot 5, another slot 3 can occur, without providing all possible combinations of affixes up to recursion depth, say, 5.

* “Rule” must be related to Morph in some way; the term “Rule” isn’t great either. Can be just say this \*is\* a Morph? Is there anything in the current Morph definition that prevents this?
  + Cf. the note on assimilation rules above, if those are implemented in the way suggested in Note2, this might be a good reason not to conflate morph and rule.