**Participants:**

Bettina Klimek (BK)

Christian Chiarcos (CC)

Jakub Šimek

Max Ionov

Stefania Racioppa (DFKI)

John McCrae

Matteo Pellegrini

Katerina Gkirtzou

1. **Module draft 4.4**

****

Adaptations of module draft 4.4 to be included for next telco:

* CC: move object of morph:InflectionRule -morph:generates-> to morph:Morph
* JM: remove cardinality restriction of exactly 1 on morph:generates
* look at representation of blank node class in LIAM (<http://lemon-model.net/liam> website is down, github.com/jmccrae/lemon-model.net [which file?])
* MI: collect controversial names/properties and vote upon
* controversial names
  + generates (for WordFormationRule) => morph:derives
  + generates (for InflectionRule -> morph:Morph) => morph:inflects
  + generates (for InflectionRule -> morph:Form) [?] (skip)
  + consistsOf
  + replace both :generates properties with 1) morph:inflects domain :InflectionRule and range :Morph and 2) morph:derives domain :WordFormationRule and range :LexicalEntry
  + replace (original) :inflects with :rule and :hasParadigm with :paradigm
* for generation, we would need a workflow diagram in the final report

1. **Max: Finnish example with generated morph:Morph resources**

# Words

<#kissa> a ontolex:Word ;

ontolex:canonicalForm [ontolex:writtenRep "kissa"@fi];

morph:inflectionType <#finnish\_noun\_type\_kissa\_number>, <#finnish\_noun\_type\_kissa\_number\_obl> . # the first category after the stem

# Inflection types

# They are combined with the `:next` property.

# This is actually quite usual for linguists: the same approach is used in the `lexc` formalism widely used in computational morphology (incl. xfst and foma)

<#finnish\_noun\_type\_kissa\_number> a morph:InflectionType ;

morph:hasParadigm <#regular\_finnish\_noun> ;

morph:next <#finnish\_noun\_type\_kissa\_case> ;

morph:inflects <#finnish\_noun\_type\_kissa\_sg>, <#finnish\_noun\_type\_kissa\_pl> .

<#finnish\_noun\_type\_kissa\_number\_obl> a morph:InflectionType ;

morph:hasParadigm <#regular\_finnish\_noun> ;

morph:next <#finnish\_noun\_type\_kissa\_case> .

<#finnish\_noun\_type\_kissa\_case> a morph:InflectionType ;

morph:hasParadigm <#regular\_finnish\_noun> ;

morph:inflects <#finnish\_noun\_type\_kissa\_nom>, <#finnish\_noun\_type\_kissa\_ine> .

<#finnish\_noun\_type\_kissa\_nom> a morph:InflectionRule ;

morph:inflectionType <#finnish\_noun\_type\_kissa\_case> ;

morph:generates [ lexinfo:case lexinfo:nominative ] ;

morph:replacement [ morph:source "$"; morph:target "" ] .

<#finnish\_noun\_type\_kissa\_sg> a morph:InflectionRule ;

morph:inflectionType <#finnish\_noun\_type\_kissa\_number> ;

morph:generates [ lexinfo:number lexinfo:singular ] ;

morph:replacement [morph:source "$", morph:target ""] . # we can actually create nodes for widely used replacements, like \_no replacement\_

<#finnish\_noun\_type\_kissa\_obl> a morph:InflectionRule ;

morph:inflectionType <#finnish\_noun\_type\_kissa\_number\_obl> ;

morph:replacement [morph:source "t$", "oi"] .

<#finnish\_noun\_type\_kissa\_ine> a morph:InflectionRule ;

morph:inflectionType <#finnish\_noun\_type\_kissa\_case> ;

morph:generates [ lexinfo:case lexinfo:inessive ]; # it's a convenient example even though the inessive of a cat is a bit weird

morph:replacement [morph:source "$", morph:target "ssa"] .

<#finnish\_noun\_type\_kissa\_pl> a morph:InflectionRule ;

morph:inflectionType <#finnish\_noun\_type\_kissa\_number> ;

morph:generates [ lexinfo:number lexinfo:plural ] ;

morph:replacement [morph:source "$", morph:target "t"] .

##### generated #####

<#finnish\_noun\_type\_kissa\_morph\_nom> a morph:Morph .

<#finnish\_noun\_type\_kissa\_morph\_ine> a morph:Morph .

<#finnish\_noun\_type\_kissa\_morph\_sg> a morph:Morph .

<#finnish\_noun\_type\_kissa\_morph\_pl> a morph:Morph .

<#kissa\_1> a ontolex:Form ;

ontolex:writtenRep "kissa"@fi ;

lexinfo:case lexinfo:nominative ;

lexinfo:number lexinfo:singular ;

morph:consistsOf <#finnish\_noun\_type\_kissa\_morph\_sg>, <#finnish\_noun\_type\_kissa\_morph\_nom> .

<#kissa\_2> a ontolex:Form ;

ontolex:writtenRep "kissat"@fi ;

lexinfo:case lexinfo:nominative ;

lexinfo:number lexinfo:plural ;

morph:consistsOf <#finnish\_noun\_type\_kissa\_morph\_pl>, <#finnish\_noun\_type\_kissa\_morph\_nom> .

<#kissa\_3> a ontolex:Form ;

ontolex:writtenRep "kissa"@fi ;

lexinfo:case lexinfo:inessive ;

lexinfo:number lexinfo:singular ;

morph:consistsOf <#finnish\_noun\_type\_kissa\_morph\_sg>, <#finnish\_noun\_type\_kissa\_morph\_ine> .

<#kissa\_4> a ontolex:Form ;

ontolex:writtenRep "kissaissa"@fi ;

lexinfo:case lexinfo:inessive ;

lexinfo:number lexinfo:plural ;

morph:consistsOf <#finnish\_noun\_type\_kissa\_morph\_pl>, <#finnish\_noun\_type\_kissa\_morph\_ine> .

→ this procedure is applicable when no morph:Morph resources are available in the dataset

* the property morph:generates contains a set of grammatical categories and has a blank node as object, which is a prototype
* additional statements ontolex:LexicalEntry ontolex:lexicalForm ontolex:Form have to be generated as well

Todos for next telco (create workflow diagrams together):

Bettina:

* move diagram drafts to git
* have a look into LIAM representation of blank nodes

Max: create

* create workflow diagram (and share it to Bettina) for the process of creating the finnish example data above with the starting point of having no morph:Morph resources and generating:
  + ontolex:Form resources
  + ontolex:Morph resources
  + triples: ontolex:LexicalEntry ontolex:lexicalForm ontolex:Form
  + triples: ontolex:Form morph:consistsOf morph:Morph