Introduction to Grammatical Framework

Inari Listenmaa

Tartu 29.9.2014

Introduction

0. About myself
1. Grammatical Framework (GF)
2. Resource Grammar Library
3. Estonian grammar in GF
4. Linguistic insights from RGL

About myself

- MA in language technology, University of Helsinki 2013
- PhD student in University of Gothenburg 2013–
- Working on GF since 2010
  - written Estonian grammar with Kaarel Kaljurand
  - contributed in Finnish, Catalan, Spanish, English and Dutch grammars

I. Grammatical Framework

Grammatical Framework

- Demo!

Grammatical Framework

- Demo!

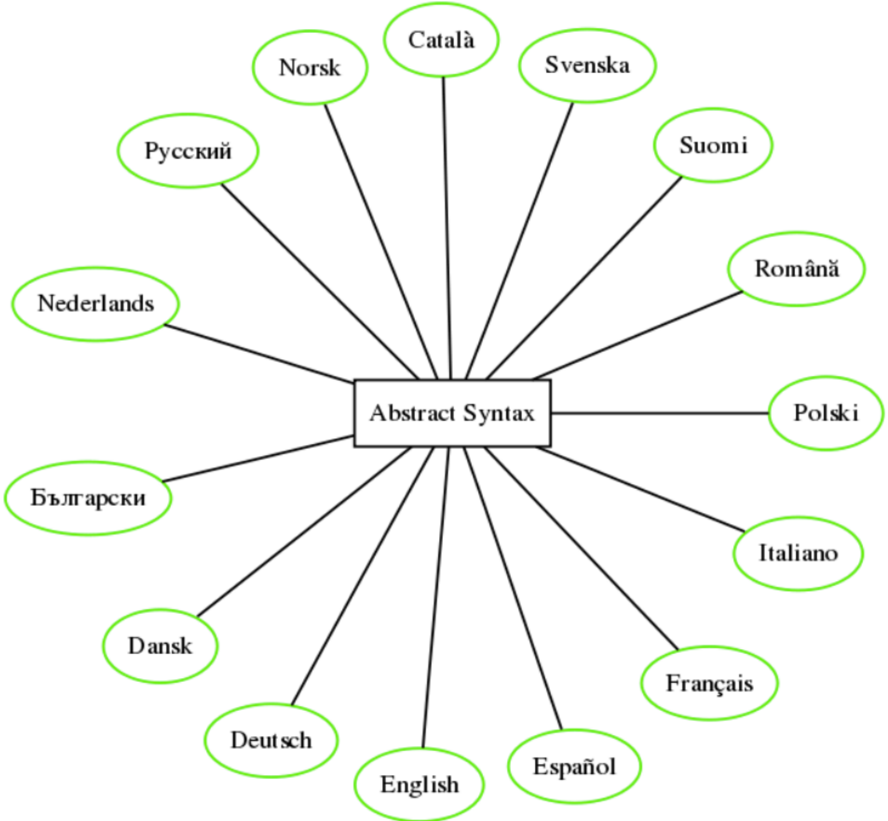
Grammatical Framework is a...

- Grammar formalism
  - like HPSG, TAG, LFG (for linguists)
  - like YACC, BNFC (for computer scientists)
- Logical framework + concrete syntax

Grammatical Framework

Abstract syntax +
concrete syntaxes

Bidirectional mapping
→ Interlingual translation!



Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum   : Recipient ;
    Friends : Recipient ;
}
```

```
concrete HelloEst of Hello = {


 lincat
   Greeting, Recipient = {s : Str} ;

 lin
   Hello rec = {s = "tere" ++ rec.s} ;
   World     = {s = "maailm"} ;
   Mum       = {s = "ema"} ;
   Friends   = {s = "sõbrad"} ;
}
```

Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum   : Recipient ;
    Friends : Recipient ;
}
```

Abstract syntax: **description** of the things you want to say

Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum   : Recipient ;
    Friends : Recipient ;
}
```

Abstract syntax: **description** of the things you want to say

startcat: start symbol

Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum   : Recipient ;
    Friends : Recipient ;
}
```

Abstract syntax: **description** of the things you want to say

startcat: start symbol

cat: categories of the grammar

Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum     : Recipient ;
    Friends : Recipient ;
}
```

Abstract syntax: **description** of the things you want to say

startcat: start symbol

cat: categories of the grammar

fun: lexical items and ways to manipulate them

Example grammar

Concrete syntax: **implementation** of the abstract syntax

```
concrete HelloEst of Hello = {


 lincat
   Greeting, Recipient = {s : Str} ;

 lin
   Hello rec  = {s = "tere" ++ rec.s} ;
   World      = {s = "maailm"} ;
   Mum        = {s = "ema"} ;
   Friends    = {s = "sõbrad"} ;
}
```

Example grammar

Concrete syntax: **implementation** of the abstract syntax

lincat: concrete type of the categories

```
concrete HelloEst of Hello = {


 lincat
   Greeting, Recipient = {s : Str} ;

 lin
   Hello rec = {s = "tere" ++ rec.s} ;
   World     = {s = "maailm"} ;
   Mum       = {s = "ema"} ;
   Friends   = {s = "sõbrad"} ;
}
```

Example grammar

Concrete syntax: **implementation** of the abstract syntax

lincat: concrete type of the categories

lin: concrete behaviour of the functions

```
concrete HelloEst of Hello = {


 lincat
   Greeting, Recipient = {s : Str} ;

 lin
   Hello rec  = {s = "tere" ++ rec.s} ;
   World      = {s = "maailm"} ;
   Mum        = {s = "ema"} ;
   Friends    = {s = "sõbrad"} ;
}
```

Example grammar

```
abstract Hello = {
  flags startcat = Greeting ;

  cat
    Greeting ; Recipient ;

  fun
    Hello : Recipient -> Greeting ;
    World : Recipient ;
    Mum   : Recipient ;
    Friends : Recipient ;
}
```

```
concrete HelloEst of Hello = {


 lincat
   Greeting, Recipient = {s : Str} ;

 lin
   Hello rec = {s = "tere" ++ rec.s} ;
   World     = {s = "maailm"} ;
   Mum       = {s = "ema"} ;
   Friends   = {s = "sõbrad"} ;
}
```

Example grammar II

```
concrete HelloIce of Hello = {

  lincat
    Greeting  = {s : Str} ;
    Recipient = {s : Str ; n : Number ;
                 g : Gender} ;

  param
    Gender = Fem | Masc | Neutr ;
    Number = Sg | Pl ;

  lin
    Hello rec = {s = case <rec.g,rec.n> of
        <Sg,Masc> => "sæll"  ++ rec.s ;
        <Sg,_>    => "sæl"   ++ rec.s ;
        <Pl,Masc> => "sælir" ++ rec.s ;
        <Pl,_>    => "sælar" ++ rec.s } ;

    World  = {s = "heimur" ; g=Masc ; n=Sg} ;

    Mum    = {s = "mamma"  ; g=Fem  ; n=Sg} ;

    Friends = {s = "vinir" ; g=Masc ; n=Pl} ;

}
```

II. Resource Grammar Library

Resource Grammar Library

- Reusable grammar implementations for 30 languages from 6 families
- Morphology
- Shared syntactic features
- Extra modules for language-specific features

Resource Grammar Library

Division of labour

- translation in general needs *semantic predicates*
- but syntactic grammar is useful as *library*

Resource Grammar Library

- Application grammar writer: import and use as a black box
- Linguist: implement it!
- 3-6 months of work, e.g. master’s project, conference paper

Morphology

User view:

➔ give 1–4 word forms to *smart paradigms*

➔ get inflection tables

```
> mkV "lugema" "lugeda" "loeb"
   Presn Sg P1 => loen
   Impf Sg P1 => lugesin
   Condit Sg P1 => loeksin
   Imper Sg => loe
   Imper Pl => lugege
   PassPresn True => loetakse
   PassPresn False => loeta
   PastPart Act => lugenud
   ...
```

Morphology

Developer view:

- ➔ read description of morphology and code it into a grammar
- ➔ find 1–4 forms that best predict the word’s inflection type

```
-- TS 57 (lugema)
-- Like 55-56 but irregular
gradation patterns
--including also marssima,
valssima
cLugema : Str -> VForms ;


-- TS 67-68 (hüppama, tõmbama)
-- strong stem in ma, b, s
-- weak stem in da, takse, ge,
nud, tud
-- t in da, takse; k in ge
cHyppama : Str -> VForms ;
```

Morphology

```
case <link,lingi> of {
  --e-deletion
  <_ + #c + "el", _ + #c + "li"> => hjk_type_IVb_audit link ;
  <_ + #c + "er", _ + #c + "ri"> => hjk_type_IVb_audit link ;

  -- More specific VII rules (which work reliably)
  <_ + "e", _ + #c + "me"> => hjk_type_VII_touge link lingi ;
  <_ + "se", _ + "ske"> => hjk_type_VII_touge link lingi ;
  <_ + "re", _ + "rde"> => hjk_type_VII_touge link lingi ;
  <_ + #v + "e", _ + "de"> => hjk_type_VII_touge link lingi ;
```

Syntax

Syntactic categories: noun phrase, clause

```
NP = {s : Case => Str ; a : Agr}
V2 = {s : Agr => Str  ; compl : Case}
Cl = {s : Str}
```

Parameters

```
Case   = Nom | Acc | Gen
Number = Sg  | Pl
Agr    = P1 Number | P2 Number | P3 Number
```

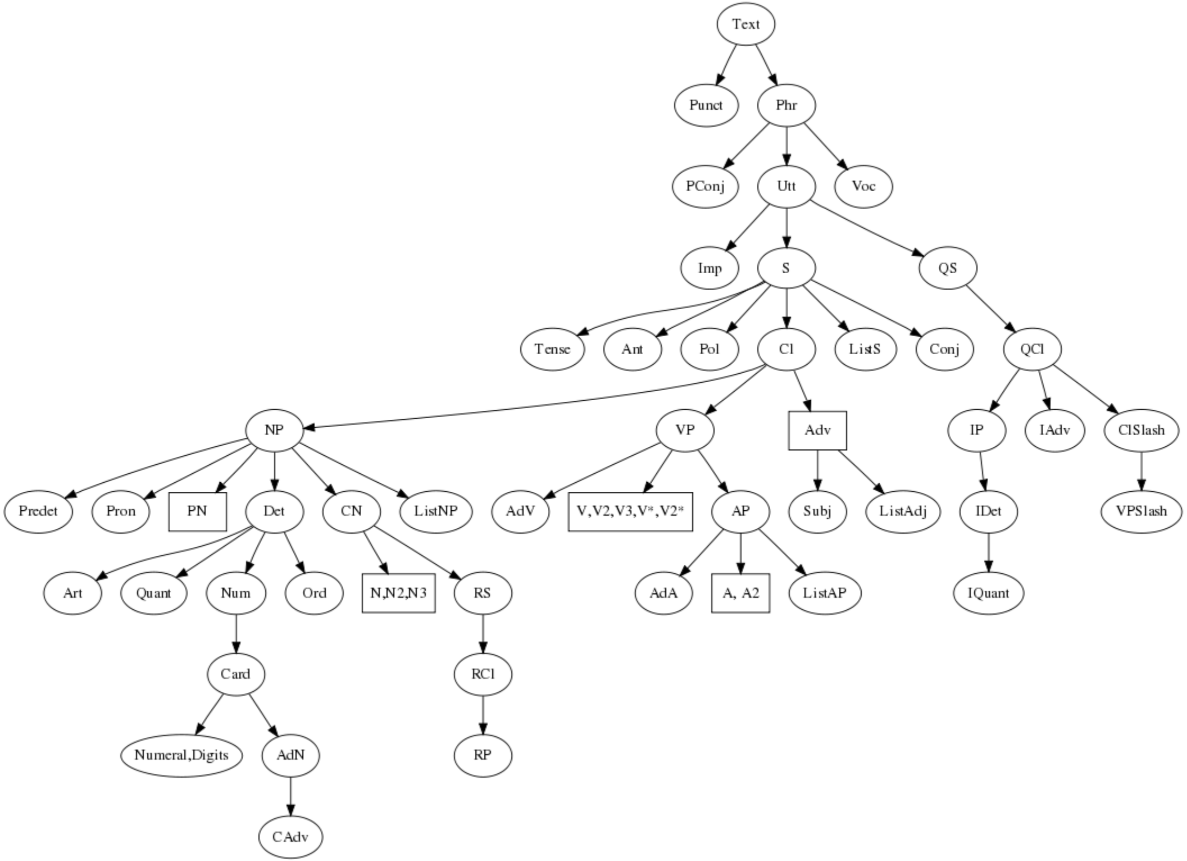
Syntax

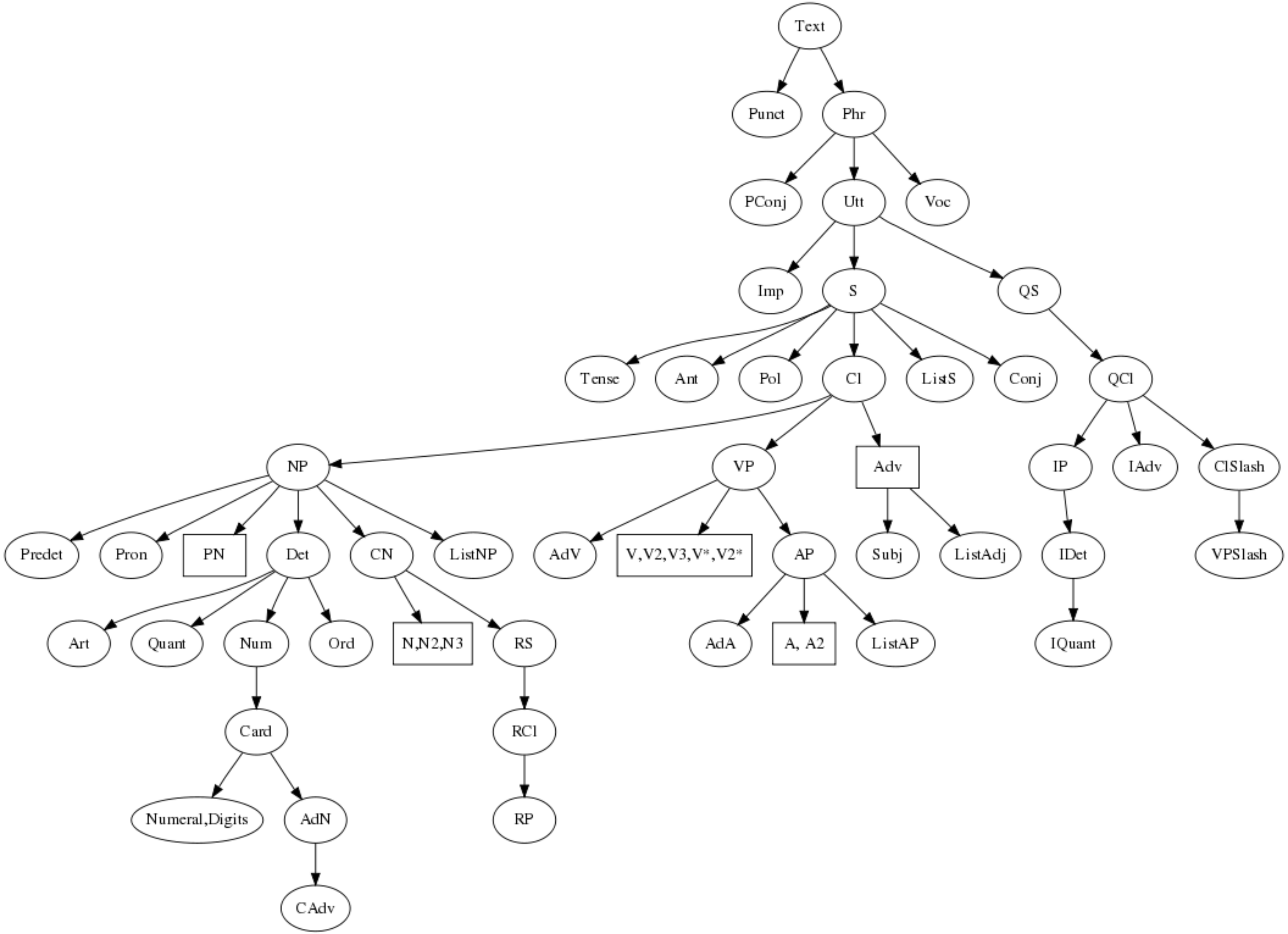
Syntactic rules

```
Pred subj verb obj = {s = subj.s ! Nom ++
                          verb.s ! subj.a ++
                          obj.s ! verb.compl}
```

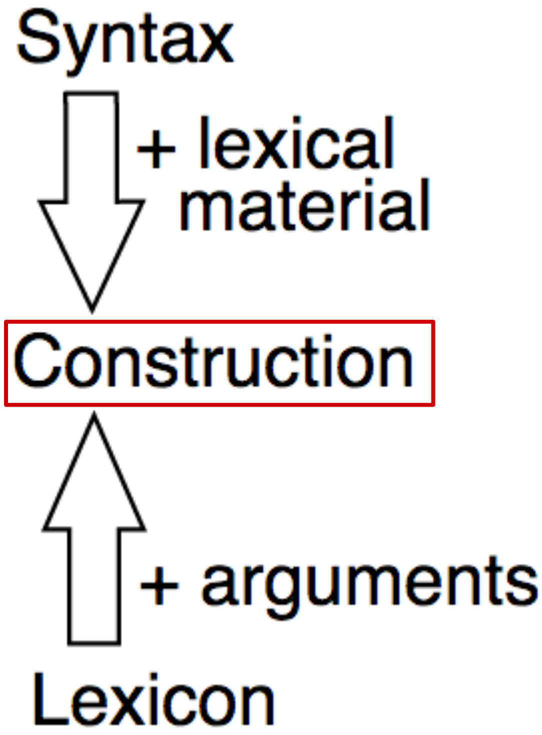
```
Pred He Love I ⇒ “he loves me”
Pred I Love He ⇒ “I love him”
```

Scary graphics




Text
Punct
Phr
PConj
Utt
Voc
Imp
S
QS
Tense
Ant
Pol
Cl
ListS
Conj
QCl
NP
VP
Adv
IP
IAdv
ClSlash
Predet
Pron
PN
Det
CN
ListNP
AdV
V,V2,V3,V*,V2*
AP
Subj
ListAdj
IDet
VPSlash
Art
Quant
Num
Ord
N,N2,N3
RS
AdA
A, A2
ListAP
IQuant
Card
RCl
Numeral,Digits
AdN
RP
CAdv

Constructions



Syntax

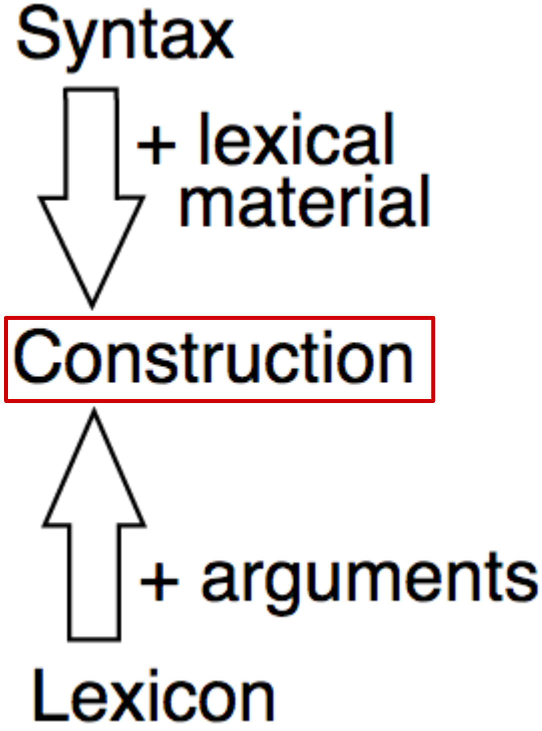
+ lexical material

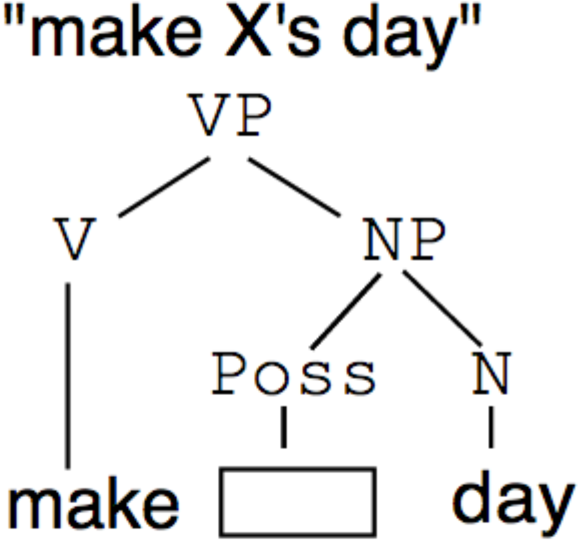
Construction

+ arguments

Lexicon

Constructions





Constructions.gf

ConstructionEng.gf

```
weather_adjCl ap =
mkCl (mkVP ap) ;

is_right_VP =
mkVP (mkA "right") ;
```

ConstructionFre.gf

```
weather_adjCl ap = mkCl
(mkVP (mkVA I.faire_V ap) ;

is_right_VP =
ComplCN avoir_V2
(mkCN (mkN "raison")) ;
```

II. Estonian grammar in RGL

Categories

Lexical categories: nouns, adjectives, verbs, …

Syntactic categories: NP, VP, clause, ...

Morphology

Nouns

- 14 cases, 2 numbers
- Implementation based on Kaalep 2012, *Eesti käänamissüsteemi seaduspärasused*
- max. 6 forms needed, other 8 based on genitive

Nouns: GF representation

```
param
  Number = Sg | Pl ;
  Case   = Nominative | Genitive | Partitive
          | Illative | Inessive | Elative
          | Allative | Adessive | Ablative
          | Translative | Essive
          | Terminative | Abessive | Comitative ;

  NForm = NCase Number Case ;

oper
  Noun : Type = {s : NForm => Str} ;
```

Nouns: paradigms

13 templates for creating 6 forms from 1 (sg nom)

```
-- if ends with 'i' ('arvuti') then last form is 'arvut' + 'e' + 'id'
-- There are ~50 such words in the WordNet.
hjk_type_IVa_aasta x =
    let
       x_e : Str = case x of {
                    _ + "i" => (init x) + "e" ;
                    _       => x }
    in
       nForms6 x x (x+"t") (x+"sse") (x+"te") (x_e+"id") ;
```

Smart paradigms

Matching words based on endings and stress patterns

```
case <(syl_type x), x, i> of {
   <_, _ + #vv + ("lik"|"nik"|"stik"), _> => hjk_type_IVb_audit x "u" ;
   <S3, _ + #v + #v + #c, i>             => hjk_type_VI_link2 x i ;
   <(S1|S3), _ + #v + #c + #c, i>        => hjk_type_VI_link2 x i ;
   <(S21|S22), _ + ("nu"|"tu"), _>        => hjk_type_IVa_aasta x ;
```

Smart paradigms

```
case <(syl_type x), x, i> of {
    <_, _ + #vv + ("lik"|"nik"|"stik"), _> => hjk_type_IVb_audit x "u" ;
    <S3, _ + #v + #v + #c, i>             => hjk_type_VI_link2 x i ;
    <(S1|S3), _ + #v + #c + #c, i>        => hjk_type_VI_link2 x i ;
    <(S21|S22), _ + ("nu"|"tu"), _>        => hjk_type_IVa_aasta x ;
```

2-arg smart paradigm: genitive as additional argument

```
mkN "lakk"        => lakk, laki, lakki, lakki, lakkide, lakkisid
mkN "lakk" "laka" => lakk, laka, lakka, lakka, lakkade, lakkasid
```

Adjectives

Most adjectives agree with nouns in case and number:

```
suure+s     linna+s
big.Sg+INE  town.Sg+INE
```

Adjectives

Most adjectives agree with nouns in case and number:

```
suure+s     linna+s
big.Sg+INE  town.Sg+INE
```

Adjectives derived from participles don’t agree as modifiers:

```
väsinud      mehe+le
tired.Sg.NOM man.Sg+ALL
```

Adjectives

Most adjectives agree with nouns in case and number:

suure+s	linna+s
big.Sg+INE	town.Sg+INE

Adjectives derived from participles do not agree as modifiers:

väsinud	mehe+le
tired.Sg.NOM	man.Sg+ALL

but inflect as predicatives:

mees	muutus	väsinu+ks
man	became	tired.Sg.TRANSL

Adjectives

Most adjectives agree with nouns in case and number:

suure+s	linna+s
big.Sg+INE	town.Sg+INE

Adjectives derived from participles do not agree as modifiers, but inflect as predicatives:

väsinud	mehe+le	mees	muutus	väsinu+ks
tired.Sg.NOM	man.Sg+ALL	man	became	tired.Sg.TRANSL

Invariable adjectives do not agree, inflect nor allow comparative or superlative:

linn	sai	valmis
town	became	ready.Sg.NOM

Adjectives: GF representation

```
param
  AForm  = AN NForm | AAdv ;
  Degree = Positive | Comparative | Superlative ;
  Infl   = Regular | Participle | Invariable ;

oper
  Adjective : Type = {s : Degree => AForm => Str ;
                      infl : Infl} ;
```

Verbs

- Inflect in voice, mood, tense, person, number
- Non-finite forms and participles that inflect like nouns
- 40 forms, incl. 11 non-finite
- Full conjugation tables from 8 forms
- Choice of forms based on Erelt et al., 2009 *Eesti keele käsiraamat*
- Smart paradigms for 1–4 arguments

Verbs: GF representation

```
param
  VForm =
   Presn Number Person | Impf Number Person
 | Condit Number Person | Quotative Voice
 | Imper Number | ImperP3 | ImperP1Pl | ImpNegPl
 | PassPresn Bool | PassImpf Bool --Positive or negative
 | PresPart Voice | PastPart Voice | Inf InfForm ;

 Person   = P1 | P2 | P3 ;
 Voice    = Active | Passive ;

 InfForm =
   InfDa | InfDes | InfMa | InfMas
 | InfMast | InfMata | InfMaks ;
```

```
oper
  Verb : Type = {
    s : VForm => Str ;
    p : Str   -- multi-word verbs
  } ;
```

Verbs: paradigms

- 15 templates for creating 8 forms from 1 forms
- 1–4-argument smart paradigms

```
cHyppama : Str -> VForms = \hyppama ->
    let
       hyppa = tk 2 hyppama ;
       hypp  = init hyppa ;
       a     = last hyppa ;
       hypa  = (weaker hypp) + a
    in vForms8
       hyppama
       (hypa + "ta")
       (hyppa + "b")
       (hypa + "takse") -- Passive
       (hypa + "ke")    -- Imperative P1 Pl
       (hyppa + "s")    -- Imperfect Sg P3
       (hypa + "nud")   -- PastPartAct
       (hypa + "tud") ; -- PastPartPass
```

Testing morphology

- % of words covered by smart paradigms
- Using Filosoft's morphological synthesizer as the gold standard
- Test vocabulary from Estonian WordNet (44k words in 29k synsets)

Testing morphology

- Test vocabulary from Estonian WordNet (44k words in 29k synsets)

Testset	Constructor	1-arg	2-arg	3-arg	4-arg
nouns	mkN	91.1	95.4	97.1	98.2
adjectives	mkN	90.0	93.6	95.2	96.9
verbs	mkV	90.5	96.6	98.3	99.7

Syntax

Syntax

- Morphology was fun
- Syntax is even more fun!

Syntax

Common abstract syntax

```
AdjCN   : AP -> CN  -> CN ;  -- big house
RelCN   : CN -> RS  -> CN ;  -- house that John bought
AdvCN   : CN -> Adv -> CN ;  -- house on the hill

UseV     : V   -> VP ;        -- sleep
ComplVV  : VV  -> VP -> VP ;  -- want to run
ComplVS  : VS  -> S  -> VP ;  -- say that she runs
```

Syntax

Estonian-specific details: adjective agreement

```
AdjCN adj noun = {
   s = \\nf => case adj.infl of {
      (Invariable      --valmis kassile; väsinud kassile
      |Participle) => adj.s ! True ! (NCase Sg Nom) ++ noun.s ! nf ;
       Regular     =>
               case nf of {
                 NCase num (Ess
                           |Abess
                           |Comit    --suure kassiga, not *suurega kassiga
                           |Termin) => adj.s ! True ! (NCase num Gen) ++ noun.s ! nf ;
                 _                 => adj.s ! True ! nf ++ noun.s ! nf --suurel kassil
               }
      } ;
```

Syntax

Estonian-specific details: choosing object case

```
Compl : Type = {s : Str ; ncase : NPForm ; isPre : Bool} ;

appCompl : Bool -> Polarity -> Compl -> NP -> Str = \isFin,pol,compl,np ->
let
  c = case compl.ncase of {
        NPAcc => case pol of {
            Neg => NPCase Part ; -- ma ei näe raamatut/sind
            Pos => case isFin of {
                True => NPAcc ;  -- ma näen raamatu/sind
                _    => case np.isPron of {
                     False => NPCase Nom ;  --tuleb see raamat lugeda
                     _     => NPAcc         --tuleb sind näha
                     }
```

Lexicon

350-word basic lexicon for all RG languages:

```
fun_AV    = mkAV (mkA (mkN "lõbus" "lõbusa" "lõbusat")) ;
garden_N  = mkN "aed" "aia" "aeda";
green_A   = mkA "roheline" ;
hate_V2   = mkV2 (mkV "vihkama" "vihata") partitive ;
know_VS   = mkVS (mkV "teadma" "teada" "teab") ; --know that S
know_VQ   = mkVQ (mkV "teadma" "teada" "teab") ; --know if QS
know_V2   = mkV2 (mkV "tundma") ;                --know someone
```

Lexicon II

80k-word monolingual lexicon

```
vaas_N = mkN "vaas" "vaasi" "vaasi"
             "vaasisse" "vaaside" "vaase" ;
vaataja_N = mkN "vaataja" "vaataja" "vaatajat" "vaatajasse"
                "vaatajate" "vaatajaid" ;
vaatama_V2 = mkV2 "vaatama" "vaadata" "vaatab" "vaadatakse"
                  "vaadake" "vaatas" "vaadanud" "vaadatud" ;
vaatamine_N = mkN "vaatamine" "vaatamise" "vaatamist"
                  "vaatamisesse" "vaatamiste" "vaatamisi" ;
```

Lexicon II

80k-word monolingual lexicon sources:

- EstWN
- the verbs of the EstCG lexicon
- database of multi-word verbs

Morfessor 2.0 used for compound word splitting of nouns

Filosoft’s morphology tools used to generate the base forms for our constructors

Lexicon III

- 65k-word multilingual lexicon
- Currently implemented by 11 languages
- TODO for Estonian
  - Then possible to do machine translation!
  - Master’s thesis project & conference/workshop paper

Lexicon III

- 65k-word multilingual lexicon
- Currently implemented by 11 languages
- TODO for Estonian
  - Then possible to do machine translation!
  - Master’s thesis project & conference/workshop paper

e.g. http://www.eki.ee/keeletehnoloogia/projektid/inglise-eesti/? Estonian WordNet?

V. Linguistic questions

General idea

- Abstract syntax: categories and functions
- Concrete syntax
    - diversity of languages
    - unity of languages

Linguistic questions

- Language similarity
- Language complexity

Language similarity

- Shared categories and functions

Language similarity

- Shared categories and functions
- Where the differences are

Language similarity

- Shared categories and functions
- Where the differences are
    - Hindi-Urdu: almost identical resource grammar, differs in lexicon, especially in technical domains (Prasad, Virk)

Language similarity

- Shared categories and functions
- Where the differences are
    - Hindi-Urdu: almost identical resource grammar, differs in lexicon, especially in technical domains (Prasad, Virk)
    - Possibility to investigate e.g. Estonian and Finnish

Language complexity

- Morphological complexity and predictability
  - Number of "non-smart" paradigms in MorphoXxx
  - Percentage of correct results for 1–4 arg smart paradigms
- Lines of code
  - Questionable; depends on programmer
  - So far all 30 languages have around same numbers

Language complexity

- Morphological complexity and predictability
  - Number of "non-smart" paradigms in MorphoXxx
  - Percentage of correct results for 1–4 arg smart paradigms
  - Aarne Ranta 2008, *How predictable is Finnish morphology? An experiment on lexicon construction.*
  - Grégoire Detrez 2012, *Smart paradigms and t predictability and complexity of inflectional*

Expert opinion

> *Importantly, I have by no means chosen the most baroque comparison possible. Partitive marking in Finnish's close sister Estonian is so much more elaborate in terms of complex interaction with its notoriously complex consonant gradations plus rampant irregularity that its very learnability seems almost questionable.*

–John McWhorter, *Linguistic simplicity and complexity*

Expert opinion

GF paradigms indicate otherwise:

- 1-arg paradigm 80 % correct in Finnish, 90 % in Estonian
- Worst-case constructor needs 10 forms in Finnish, 6 in Estonian
- Dependent on test set and implementation!

Next up:

- break
- hands-on / live coding

Extra remarks about morphology

EKK09 says that all except 27 verbs can be formed from 4 forms; *ma*, *da*, *b*, *takse.* Possibe counterexamples?

a) Forming the imperfect forms from ma stem

```
jooksma : 62
jooksma, joosta, jookseb, joostakse
Impf Sg P3 => jooksis
```

```
maitsma : 62
maitsma, maitsta, maitseb, maitstakse
Impf Sg P3 => maitses
```

Choice of vowel (e/i) is not obvious from any of the 4 forms.

Extra remarks about morphology

b) Forming the past participle (nud) from da

```
jooksma : 62
jooksma, joosta, jookseb, joostakse
PastPartAct Sg Nom => jooksnud

laskma : 64
laskma, lasta, laseb, lastakse
PastPartAct Sg Nom => lasknud
```

Here the past participle is formed with *ma* stem, not *da* stem.