Computational Syntax Exam

LT2214, University of Gothenburg

31 May 2024

This exam has four questions. Each question is worth 15 points. You need 30 points for the mark G, 45 for VG.
As your help, we provide two tables: UD labels and GF constructs.
Teacher: Aarne Ranta
Email: aarne.ranta@cse.gu.se
Phone: 031-772 10 82 (also connects to mobile phone)

relation	explanation	dependent-head	example
`acl`	clausal modifier of noun	S-N	*the moon as we* ***see*** *it*
`acl:relcl`	relative clause modifier	S-N	*the moon that we* ***see***
`advcl`	adverbial clause modifier	S-C	*I leave if she* ***goes***
`advmod`	adverbial modifier	ADV-C	*he sleeps* ***now***
`amod`	adjectival modifier	ADJ-N	***black*** *cat*
`appos`	appositional modifier	N-N	*Macron, the* ***president***
`aux`	auxiliary	AUX-S	***does*** *he sing*
`case`	case marking	ADP-N	***on*** *the moon*
`cc`	coordinating conjunction	CCONJ-C	***and*** *dogs*
`ccomp`	clausal complement	S-C	*I know that he* ***runs***
`compound`	compound	N-N	***data*** *science*
`conj`	conjunct	C-C	*cats and* ***dogs***
`cop`	copula	AUX-S	*he* ***is*** *old*
`csubj`	clausal subject	S-S	*that is* ***moves*** *is clear*
`dep`	unspecified dependency	C-C	(if nothing else works)
`det`	determiner	DET-N	***the*** *cat*
`expl`	expletive	PRON-S	***there*** *is hope*
`fixed`	fixed multiword expression	ADP-C	*because* ***of***
`flat`	flat multiword expression	PROPN-PROPN	*Adam* ***Smith***
`iobj`	indirect object	N-VERB	*she gave* ***us*** *a hint*
`mark`	marker	PART/SCONJ-S	***to*** *go*
`nmod`	nominal modifier	NOUN-NOUN	*man on the* ***moon***
`nmod:poss`	possessive modifier	N-NOUN	***my*** *cat*
`nsubj`	nominal subject	N-S	***John*** *walks*
`nsubj:pass`	nominal subject of passive	N-VERB	***John*** *was seen*
`nummod`	numeric modifier	NUM-N	***five*** *cats*
`obj`	object	N-VERB	*she sees* ***us***
`obl`	oblique nominal	N-S	*she comes with* ***us***
`parataxis`	parataxis	VERB-VERB	*I said:* ***come*** *here*
`punct`	punctuation	PUNCT-S	*I see* ***!***
`root`	root	S-	*John **walks*
`xcomp`	open clausal complement	S-S	*I want to* ***go***

Syntactic relations used in UD standard 2, together with their typical uses. In the examples, the dependent is **boldfaced** and its head underlined. S means a sentence-like, N a noun-like, and C any kind of phrase.

Construct	Notation	Example
Abstract syntax module	`abstract`	`abstract Lang =...`
Concrete syntax module	`concrete`	`concrete LangEng of Lang Lang =...`
Resource module	`resource`	`resource ResEng Lang =...`
Module extension	`**`	`abstract Lang = Noun,Verb **...`
Module opening	`open`	`resource ResEng = open Prelude in...`
Abstract syntax category	`cat`	`cat NP`
Abstract syntax function	`fun`	`fun Pred : NP -> VP -> S`
Linearization type	`lincat`	`lincat N = {s : Number => Str}`
Linearization rule	`lin`	`lin Pred np vp = np.s!Nom ++ vp!np.a`
Parameter type	`param`	`param Case = Nom | Acc`
Auxiliary operation	`oper`	`oper addS : Str -> Str = \x -> x + "s"`
String concatenation	`++`	`"loves" ++ "Mary"`
Token concatenation	`+`	`"Maria" +"m" ⇓ "Mariam"`
Function type	`->`	`NP -> VP -> S`
Function application	`f a b`	`Pred np vp`
Function abstraction	`\ ->`	`\x,y -> x + y`
Table type	`=>`	`Case => Str`
Table	`table`	`table {Nom =>"she" ; Acc =>"her"}`
Selection from table	`!`	`she_NP.s ! Acc ⇓ "her"`
Table with one branch	`\\ =>`	`\\p,q => np ! q ! p`
Record type	`{...:...}`	`{s : Str ; g : Gender}`
Record	`{...=...}`	`{s = "doctor" ; g = Fem}`
Projection from record	`.`	`{s = "doctor"}.s ⇓ "doctor"`
Record update	`**`	`doctor_N ** {g = Masc}`
Case expression	`case`	`case np.a of {Ag n _ => cn.s ! n}`
Tuple type	`*`	`Number * Case`
Tuple	`<,>`	`<Sg,Dat>`
Comment	`--`	`-- comment till the end of line`
Comment	`{- -}`	`{- comment of any length -}`

This is a reading guide for GF notation. The first five rows are about modules, the next six list the different kinds of rules. The rest are expressions for types and objects. The notation $e \Downarrow v$ means that expression e is computed to value v.

Question 1

Show the Universal Dependency tree of the following sentence:

UD is a framework for consistent annotation of grammar across different human languages.

Show it in two formats:

- as CoNLLu with fields ID, FORM, LEMMA, POS, DEPREL, HEAD
- as a graphical figure with arrows and dependency labels.

Question 2

Draw a phrase structure tree for the example sentence of Question 1, maintaining as much as possible of the same structure as your UD tree. This means that each head together with its dependents should form a subtree. Use familiar category labels such as S, NP, VP, CN, AP, Det, Prep, V.

Write a context-free grammar that recognizes the sentence with the tree that you drew.

Question 3

Here is a fragment of an abstract syntax of sentences consisting of a subject, a verb, and an object:

```
abstract Sentence = {
cat
  S ; NP ; VP ; V2 ;
fun
  PredVP : NP -> VP -> S ;    -- add subject to verb phrase
  ComplV2 : V2 -> NP -> VP ;  -- verb phrase from verb and object
}
```

In concrete syntax, any of the following constituent orders are possible:

- SVO: subject-verb-object
- SOV: subject-object-verb
- VSO: verb-subject-object
- VOS: verb-object-subject
- OVS: object-verb-subject
- OSV: object-subject-verb

Write concrete syntaxes for each of these orders, named `SentenceSVO`, etc. You can assume that subjects, verbs, and objects are just strings (with no morphological features or agreement), but you may still need to use records as linearization types.

Question 4

Now consider one of the constituent orders of the previous question, SOV, as used in Latin. Write a concrete syntax `SentenceLat` with the following morphological parameters and agreement rules:

- Case, with values Nom, Acc, Gen, Dat, Abl.
- Number, with values Sg, Pl.
- Person, with values Per1, Per2, Per3.
- NP has inherent Number and Person, and variable Case.
- VP has variable Number and Person.
- V2 has variable Number and Person, and inherent Case (for the object).
- The subject of a sentence has case Nom.
- The object of a sentence gets its case from the verb.
- The verb of a sentence gets its number and person from the subject.

For example, the following sentences can be formed:

```
Ioannes                  Mariam       amat
Ioannes (Sg, Per3)(Nom)  Maria (Acc)  amare (Sg, Per3)(Acc)
"John loves Mary"

vos                  mihi       nocetis
vos (Pl, Per2)(Nom)  ego (Dat)  nocere (Pl, Per2)(Dat)
"you harm me"
```

Make sure to include all `lincat`, `lin`, and `param` definitions required to implement this. But your solution need only cover the syntactic functions PredVP and ComplV2, no words.