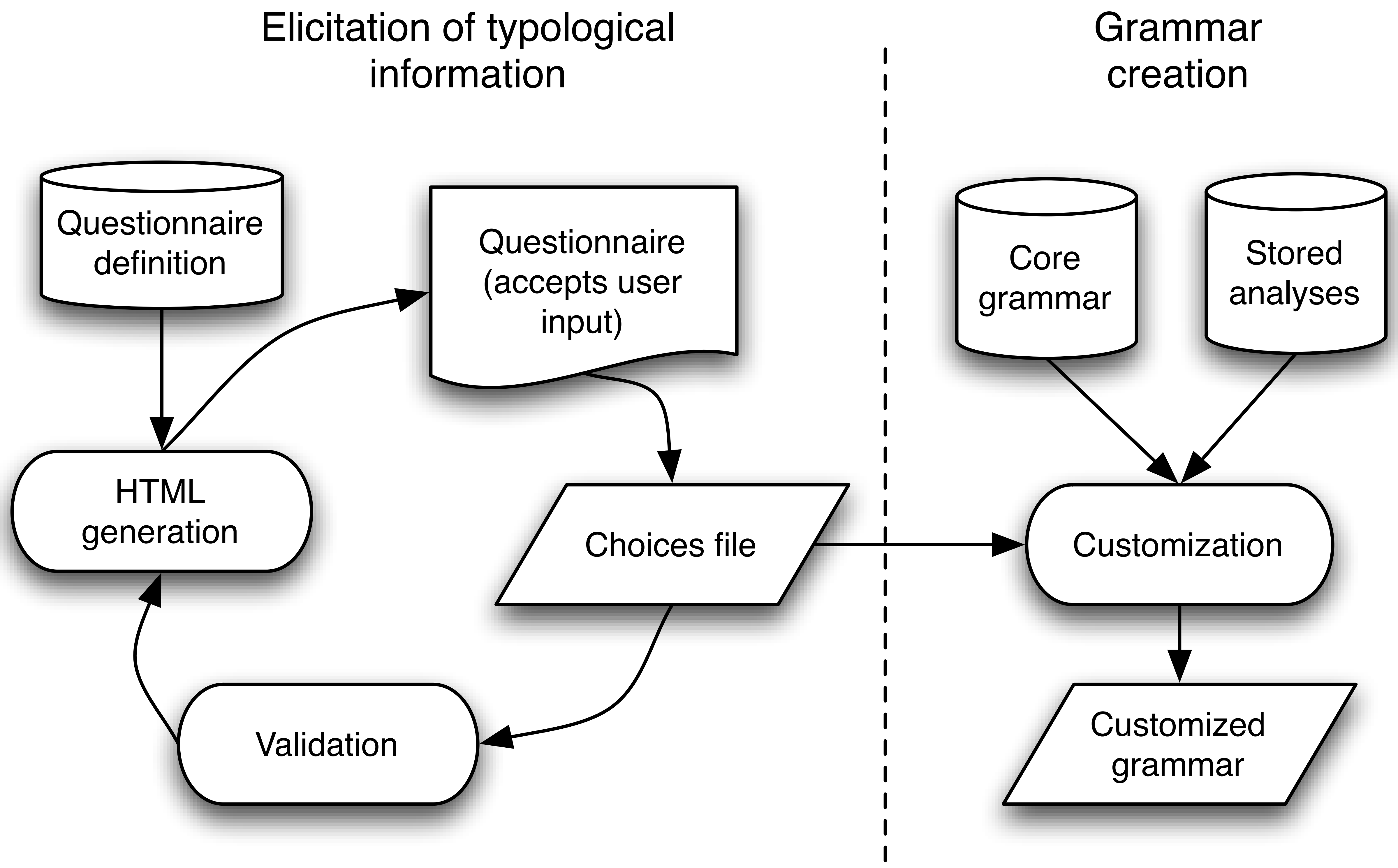
Grammar Matrix (incl morphotactics) AGGREGATION Test suites

Ling 567
Jan 12, 2021

Overview

- Grammar Matrix customization system
- Questions from Lab 1
- AGGREGATION
- Testsuites & [incr tsdb()]
- Morphotactics in the Grammar Matrix



(Bender et al 2010)

```
567_english.tdl
head-types.tdl
irules.tdl
labels.tdl
lexicon.tdl
lrules.tdl
matrix.tdl
mtr.tdl
pet.tdl
roots.tdl
rules.tdl
```

```
LICENSE
METADATA
README
Version.lsp
567_english-pet.tdl
ace/
choices
irregs.tab
lkb/
pet/
repp/
semi.vpm
test_sentences
trigger.mtr
tsdb/
```

Creating a library for the customization system

- Choose phenomenon
- Review typological on phenomenon
- Refine definition of phenomenon
- Conceptualize range of variation within phenomenon
- Review HPSG (& broader syntactic) literature on phenomenon
- Pin down target MRSs
- Develop HPSG analyses for each variant
- Implement analyses in tdl
- Develop questionnaire
- Run regression tests
- Test with pseudo-languages
- Test with illustrative languages
- Test with held-out languages
- Add tests to regression tests
- Add to MatrixDoc pages

Overview

- Grammar Matrix customization system
- Questions from Lab 1
- AGGREGATION
- Testsuites & [incr tsdb()]
- Morphotactics in the Grammar Matrix

Lab 1 questions: Grammar Matrix

- What process leads to a constraint like 'norm-hook-lex-item,' for example, being developed? Many of the constraints seem very specific and I'm interested in how much data is needed to motivate a new constraint.

- How can we know when the current grammar structure (the matrix) may need a new feature or identity?

- How will extensions to the grammar work. Will we be eventually writing ourselves into the tdl files?

Lab 1 questions: Grammar Matrix

- On the LinGO Grammar Matrix, what are supertypes for gender and number and what languages feature them.

- How was the decision to separate rules into the "generic" grammar (matrix.tdl) versus the English grammar/language-specific grammar made?

- What are the motivations for the design choices in this grammar. I'm sure I will understand this naturally as I use it more, but for example, I still don't fully understand the value of pulling the first thing on the RELS list to LKEYS.KEYRELS.

Lab 1 questions: Working with unfamiliar languages

- Because I have analyzed English syntax before, I always knew what rule to apply during the lab. What happens when implementing a grammar for a language you are less familiar with and maybe you do not even know the rule you want to use to unify something? How could you debug both unification errors and which rule to use? Does this ever happen or is it more straightforward than I think?

- So far, this lab could be done with a lot of intuition because I already know English grammar. I am a bit worried that I will be very lost when it's a completely new language. Is there any strategy for avoiding entering a bad trace of grammar identities? I am afraid that we'd go down a rabbit hole and can't get out...

- How will we reconcile our unfamiliarity with our research languages when making high-level guesses such as which grammar rule to unify with? Will it come given reading the field linguist material about the language?

Lab 1 questions: New features (so many)
See: http://moin.delph-in.net/wiki/GeFaqFeatureGeometry

- How will I become familiar with all of these features???

- I'm still quite a bit confused about how C-CONT works. In my chain of identities I used it pretty much as the CONT of the parent phrase of its daughters, but I'm not sure if that's correct or not. What is the purpose of this distinction between LOCAL.CONT and C-CONT?

- Can we go over some of the main counterparts of the 566 features? Or perhaps more importantly, can we go over some of the most important features we're working with now that don't have 566 counterparts? And likewise some rules?

- Why is an element referred to as “sign"?

Lab 1 questions: Ginormous feature structures

- How to navigate huge feature structures?

- Why are these feature structure so much bigger than those that we saw in 566? I assume it is because this grammar has much better coverage of language phenomena, but the size increase still surprises me. How can the coverage of a grammar be measured?

- In the code, is the suffix -min convention for basic avm feature structures? Are there other such conventions that we should be utilizing?

Lab 1 questions: Feature-structure representation of rules

- I'm having a bit of a difficult time picking apart the grammar rules; they aren't as cut and dry as what we used in 566. I guess it would be helpful for me to see something in the form of ```TFS1 -> H-TFS2 TFS3```, where I can see how the structures unify to form a new feature structure. Does LKB have any functionality to visualize this?

-

Lab 1 questions: Interactive unification when & how

- When I was testing unification in steps 2-4 I was not 100% sure how what we were doing was going to fit into our eventual workflow. Like, it felt like we had to have a theory ahead of time, guessing which grammar rule was the problem and which entries went into which slots; if we guessed wrong, it wouldn't necessarily reflect what was *actually* failing in the parse, right? Why isn't there a way to just click on the parse chart and say "show parse failure"?

Lab 1 questions: LKB

- Does LKB enforce any principles on its own, without specification in matrix.tdl or other files?
- If you click on the N above 'cat', what is the difference between 'Feature Structure', 'Unfilled Feature Structure' , Unfilled Feature Structure - Edge 10', 'Rule CAT' etc?
- Is there a way to parse a whole file of sentences at once? I think there should be but I wonder if we can show this in the lecture.
- Going forward, should I worry about LKB performance issues given more complex, more ambiguous sentences?

Lab 1 questions: Tracing chain of identities

- Why does all the tracing to get to a link not count as chain of identity?
- It's not entirely clear to me how to choose the type that is actually contributing the constraint I'm looking for. If I see what I'm looking for, should I just stop and name that type? Or should I keep looking further into its supers to make sure that the constraint isn't already entailed?
- I am trying to find the type that links the second element of the RELS list on the head-spec-phrase to the RELS value on the non-head-daughter of the head-spec-phrase. I found a type that links the HOOKS of these two things, but not the RELS. Furthermore, it's not clear to me how the second element of the RELS list is identified, because when I went looking I found it in the middle of a huge append list, and I don't know how to make sense of why it is on the position in that append list that it is.

Lab 1 questions: tdl

- Similarly, it would be nice to go over some of the combinatorial operators. Does the & work like a unification? It seems to have boolean-esque properties in some cases (like INDEX ref-ind & #ind1) but append-y type properties in other cases?

- Wrt the lists, I was wondering if they were all linked lists - feels very car/cdr-ish with the FIRST/REST syntax, which wouldn't be surprising if this is lisp-based.

Overview

- Grammar Matrix customization system
- Questions from Lab 1
- AGGREGATION
- Testsuites & [incr tsdb()]
- Morphotactics in the Grammar Matrix

AGGREGATION: Research goals

- Precision implemented grammars are a kind of structured annotation over linguistic data (cf. Good 2004, Bender et al 2012).
- They map surface strings to semantic representations and vice-versa.
- They can be used in the development of *grammar checkers* and *treebanks*, making them useful for language documentation and revitalization (Bender et al 2012)
- But they are expensive to build.
- The AGGREGATION project asks whether existing products of documentary linguistic research (IGT collections) can be used to boot-strap the development of precision implemented grammars.

AGGREGATION: Recent developments

- See ComputEL-3 slides

Overview

- Grammar Matrix customization system
- Questions from Lab 1
- AGGREGATION
- Testsuites & [incr tsdb()]
- Morphotactics in the Grammar Matrix

Evaluation and Computational Linguistics

- Why is evaluation so prominent in computational linguistics?
- Why is it not so prominent in other subfields of linguistics?
- What about CS?

Intrinsic v. extrinsic evaluation

- Intrinsic: How well does this system perform its own task, including generalizing to new data?
- Extrinsic: To what extent does this system contribute to the solution of some problem?
- Examples of intrinsic and extrinsic evaluation of parsers?

Test data

- Test suites
    - Hand constructed examples
    - Positive and negative examples
    - Controlled vocabulary
    - Controlled ambiguity
    - Careful grammatical coverage

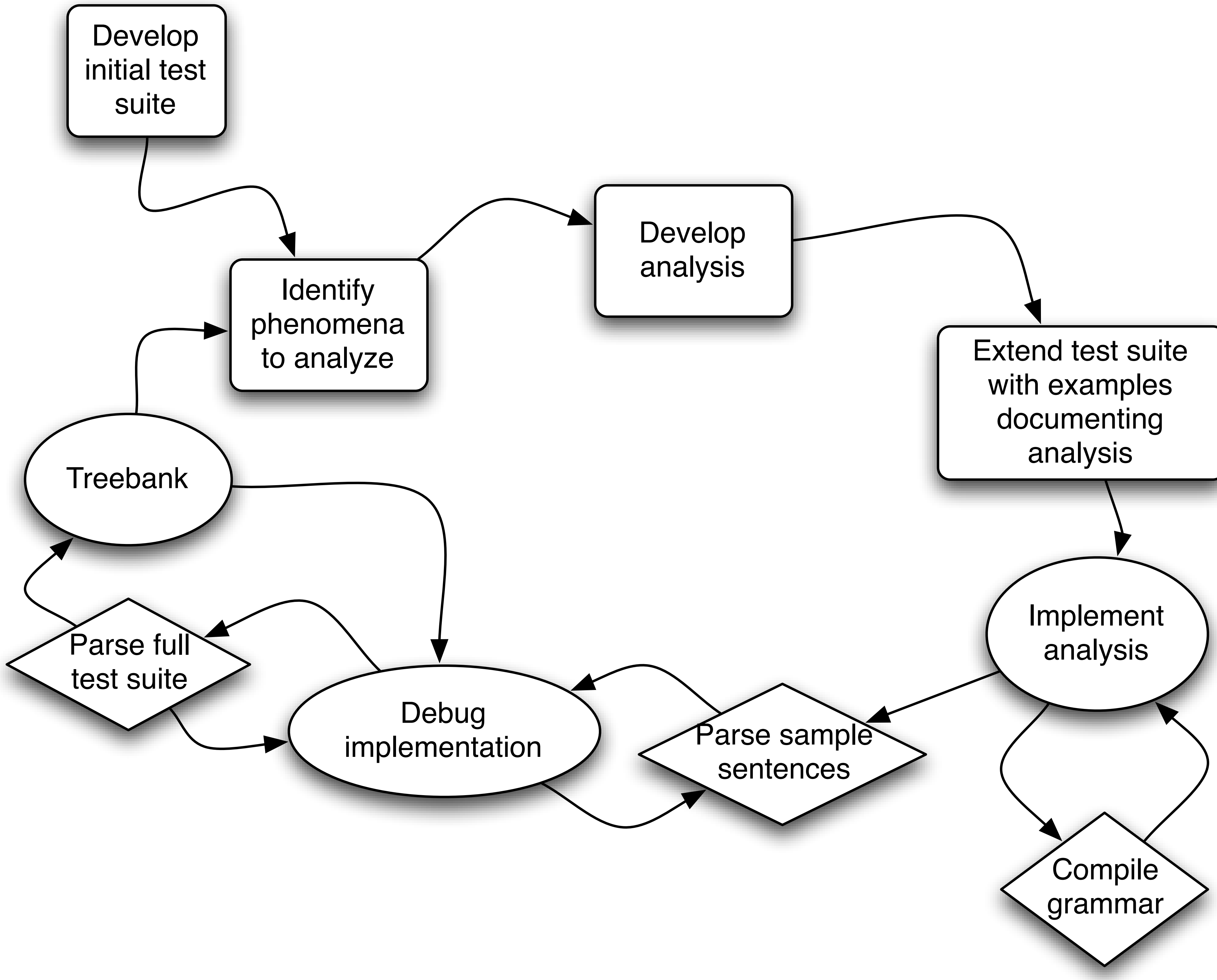
Test data

- Test corpora
    - Naturally occurring
    - More open vocabulary
    - Haphazard ungrammatical examples
    - Application-focused

Uses of test data

- How far do I have left to go?
  - Internal metric
  - Objective comparison of different systems
- Where have I been?
  - Regression testing
  - Documentation

Grammar engineering workflow



Evaluating precision grammars

- Coverage over some corpus
  - Which corpus?
  - Challenges of lexical acquisition
- Coverage of phenomena
  - How does one choose phenomena?
- Comparison across languages

Levels of adequacy

- grammaticality
- “right” structure
- “right” dependencies
- “right” full semantics
- only legit parses (how can you tell?)
- some set of parses including the preferred one
- preferred parse only/within first N

Typical 567 test suites

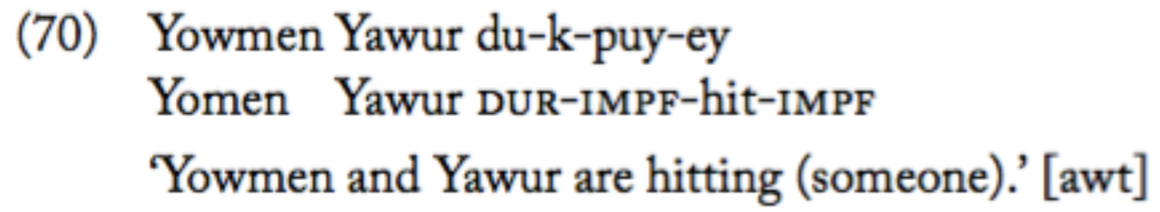
- Map out territory we hope to cover
- Include both positive and negative examples
- Serve as an exercise in understanding the description of the language
    - IGT format
    - Creating examples where necessary

On the importance of simple examples

- Why keep examples simple?
- How simple is too simple?
- What kinds of things make an example not simple enough?

On the importance of simple examples

- Awtuw [awt] (Feldman 1986:67)

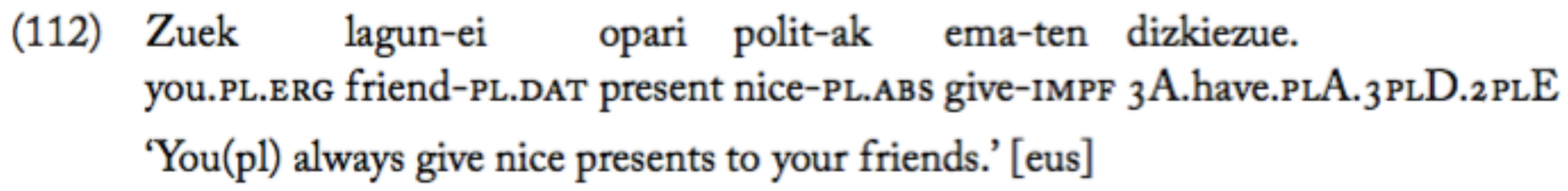


(70)

Yowmen	Yawur	du-k-puy-ey
Yomen	Yawur	DUR-IMPF-hit-IMPF

'Yowmen and Yawur are hitting (someone).' [awt]

- Basque [eus] (adapted from Joppen and Wunderlich 1995:129)



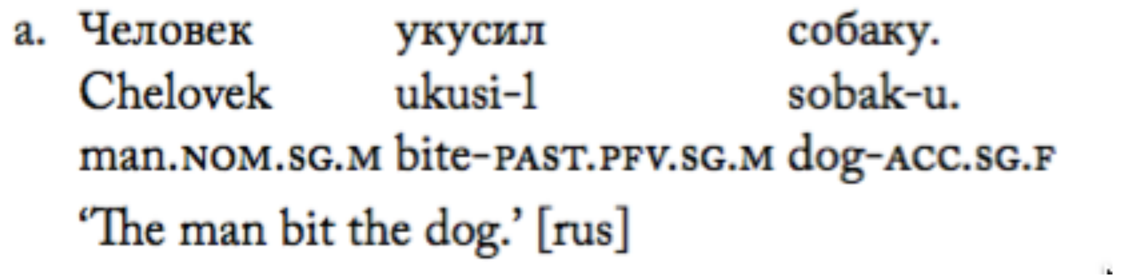
(112)

Zuek	lagun-ei	opari	polit-ak	ema-ten	dizkiezue.
you.PL.ERG	friend-PL.DAT	present	nice-PL.ABS	give-IMPF	3A.have.PLA.3PLD.2PLE

'You(pl) always give nice presents to your friends.' [eus]

On the importance of simple examples

- Russian [rus] (Bender 2013:92)



a.	Человек	укусил	собаку.
	Chelovek	ukusi-l	sobak-u.
	man.NOM.SG.M	bite-PAST.PFV.SG.M	dog-ACC.SG.F

‘The man bit the dog.’ [rus]

But this year we have test corpora!

- Might include both elicited and naturally occurring examples
- Lots more data to play with (yay!)
- Will be messy: Spoken language, lots of interacting phenomena, possibility of inconsistent transcription & glossing
- But more satisfying because it’s way more authentic
- Possibly too large: Okay to cut down or break into smaller chunks if processing is slow
    - Possibly consider using ace & art for batch processing

[incr tsdb()] basics

- [incr tsdb()] stores test suite profiles as (plain text) relational databases: Each is a directory with a fixed set of files in it.
- Most files are empty.
- A profile that has not been processed has only two non-empty files: item (the items to be processed) and relations (always the same)
- Once the profile has been processed, the result of the processing is stored in some of the other files (in particular, parse and result)

[incr tsdb()] basics

- A test suite *skeleton* consists of just the item and relations files and can be used to create new test suite profiles
- [incr tsdb()] allows the user to compare two profiles to see how they differ
- It can also produce graphs plotting summary data from many profiles to visualize grammar evolution over time
- -> If time: Demo

Overview

- Grammar Matrix customization system
- Questions from Lab 1
- AGGREGATION
- Testsuites & [incr tsdb()]
- Morphotactics in the Grammar Matrix

Morphology: Basics

- Morpheme: The smallest meaningful unit of language/smallest pairing of “form” and “meaning”
- But:
    - “form” can be lots of things, including empty but also messy changes to word form
    - “meaning” can be just syntactic features
- Morphotactics: Which morphemes can combine, in what order
- Morphophonology: Relationship between underlying word forms and surface forms
- Morphosyntax: Relationship between morphemes and syntactic and semantic features

Morphology: Example

slolmáyaye
slol-ma-ya-yÁ
know-1SG.PAT-2SG.AGT-know

‘you know/knew me’ [lkt]

- Infixation, vowel harmony: Morphophonology
- Relative order of PAT and AGT marker, optionality of same: Morphotactics
- Mapping to constraints that the patient argument be 1sg and the agent 1pl: Morphosyntax
- Actually parsing the string: priceless!

What morphophonolgy can the LKB & the customization system handle?

	LKB	Customization System
polite concatenative morphology	✓	✓
zero morphemes	✓	✓
morphologically conditioned allomorphy	✓	✓
phon. chnages at morpheme boundary	✓	
ablaut		
infixation		
vowel harmony		
suppletion		

Assume a morphophonological analyzer...

- Morphophonological analyzers map surface forms to underlying strings of morphemes
- FSTs are up to the task (except for open-class reduplication)
  - XFST (Beesley & Karttunen 2003) is a very linguist-friendly set up; FOMA (Holden & Algeria 2010) is a open-source package with similar functionality
- But you don’t need to build one for this class!
- Use the morpheme segmented line of your IGT to represent what it would map to, and then (if you have any interesting morphophonology) have that line be the target for your grammar.

Morphophonology/morphosyntax boundary: Where to draw the line?

- Underlying morphemes can be represented as a sequence of phonemes or as symbols representing morphological features.
- A canonical XFST-derived analyzer will also include POS tags as a morphological feature in the underlying form.
- From the point of view of the LKB:
  - The POS tag adds nothing
  - Spelling the morphemes as morphological features adds nothing: we still need a lexical rule that maps those strings to constraints on avms

Morphophonology/morphosyntax boundary: Where to draw the line?

- On the other hand: for XFST/FOMA, the POS tags (and maybe features) can be useful intermediate stages in processing
- The features can make it easier to create gloss lines automatically.
- On the third hand: using sequences of morphemes might make LKB input/output comprehensible to speakers
- So what should the upper tape have?

Basic concepts

- Position class: A supertype to lexical rules which fit in the same slot
- Lexical rule type: *lex-rule* and its subtypes, all have DTR feature
- Lexical rule instance: A grammar entity (manipulatable by the LKB) which inherits from a lexical rule type and specifies a spelling change (including no change).
- Forbids constraint: A specification in the customization system stating that a stem lexical rule type (including a position class) cannot co-occur with another lexical rule type, instance, pc or stem.
- Requires constraint: A specification in the customization system stating that a stem lexical rule type (including a position class) must co-occur with another lexical rule type, instance, pc or stem.

Position classes, inputs and lexical rule hierarchies

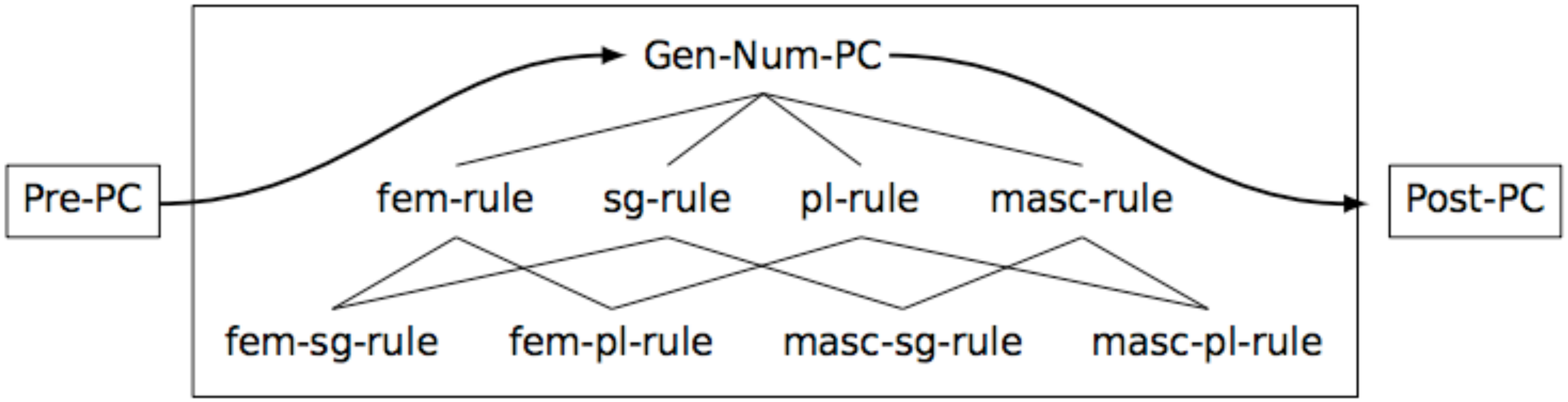


Figure 9: Example lexical rule type hierarchy in a position class

(Goodman 2013)

To define a position class

- Required:
    - Whether or not it is obligatory
    - Possible inputs and prefix/suffix
        - = position in the string
- Optional:
    - Requires/forbids constraints

To define a lex rule type

- Required
    - Nothing (though defaults fill in)
- Optional
    - Name
    - Supertype (if it doesn’t inherit directly from its position class)
    - Feature/value pairs (optional, but this is usually the point!)
    - Requires/forbids constraints

To define a lex rule instance

- Required
  - Affix v. no affix
  - Spelling for affix
- Optional
  - Nothing

tdl files

- matrix.tdl: Supertypes for lex-rules, which handle the copying up of everything you’re not changing
- my_language.tdl: Position classes and lex rule types defined through the customization system; features for inside INFLECTED
- lrules.tdl: Instances for non-spelling-changing lex rules (zero morphemes)
- irules.tdl: Instances for spelling-changing lex rules

Handling of morphotactics

- Rule order handled through super types and typing the DTR feature
- Requires/forbids through the INFLECTED feature

```
case-lex-rule-super := representative-rule-dtr &
                       add-only-no-ccont-rule &
                       noun-telic-rule-dtr &
[ INFLECTED [ CASE-FLAG +,
          INNER-NEGATION-FLAG #inner-negation,
          NUMBERED-FLAG #numbered ],
  DTR case-rule-dtr &
     [ INFLECTED [ INNER-NEGATION-FLAG
                              #inner-negation,
              NUMBERED-FLAG #numbered ] ] ].
```

Wednesday = demo day

- Send me questions by noon on Thursday; all should include:
  - Question
  - Choices file
  - Data:
    - Testsuite profile
    - IGT that should parse if we can just fix the thing
    - … or should stop parsing, if we can just fix the thing, in the case of ungrammatical examples