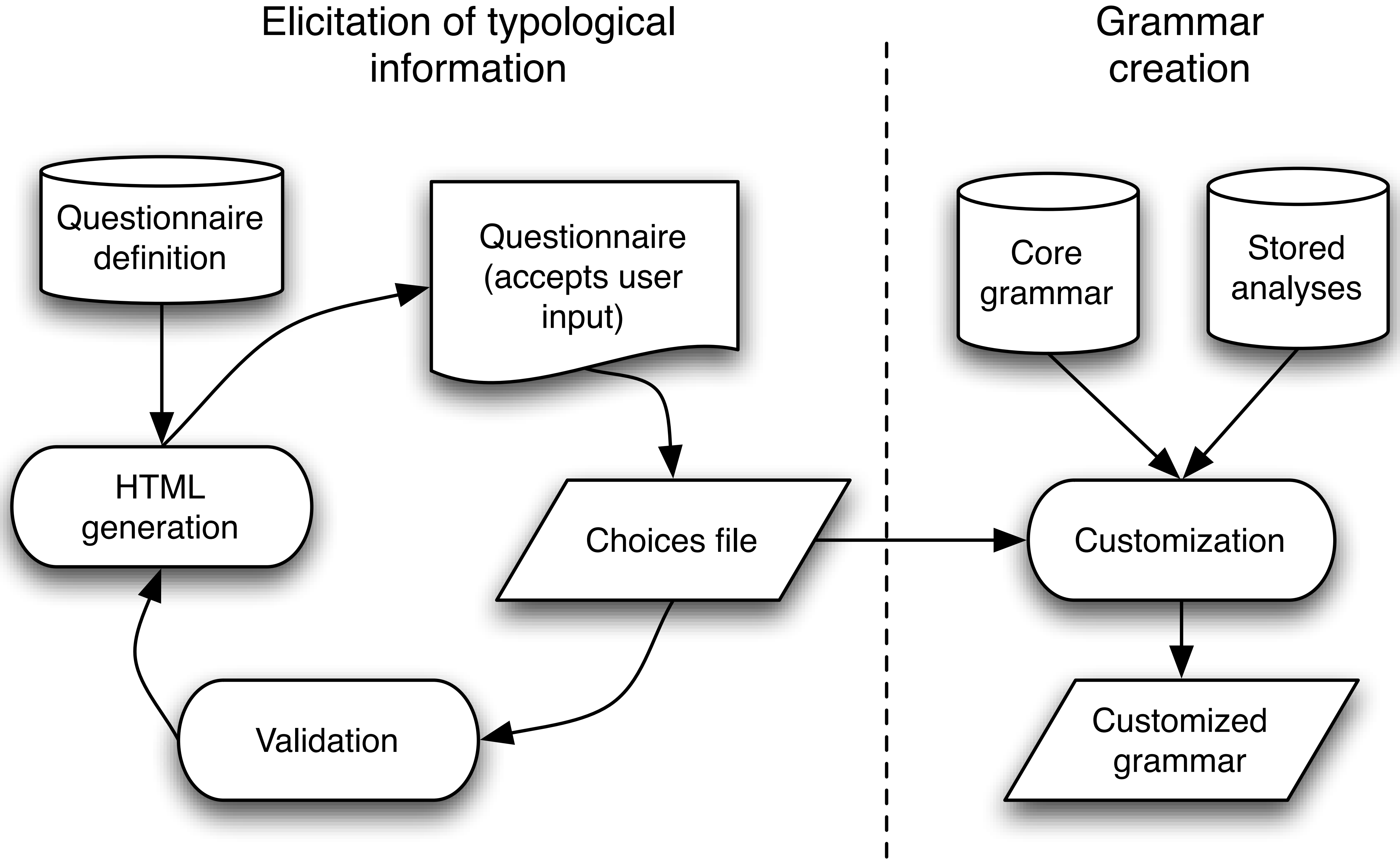
Grammar Matrix (incl morphotactics)
AGGREGATION
Test suites

Ling 567
Jan 11, 2022

Overview

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



Elicitation of typological information

Grammar creation

Questionnaire definition

Questionnaire (accepts user input)

HTML generation

Choices file

Validation

Core grammar

Stored analyses

Customization

Customized grammar

(Bender et al 2010)

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

- What is the difference between the leaf node and the lexical entry of a word?

- Are rules and phrases types?

- I'm not sure where if at all there are distinctions between lexical rules and lexical types especially in how they're written or represented in the grammar (tdl?) files.

- I am curious how the tdl files are all linked.

- I wonder why would there are separate matrix and _grammar tdl files? How are they different?

```
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/
```

Lab 1 questions

- How do the constructions/notation/format of the lexical rules in LKB correspond to our conceptions of INPUT and OUTPUT in l-rules (since we only see a single feature structure in the visualizations)?

Lab 1 questions

- Why are there only four grammar rules available when we look at unification errors?

- It was mentioned that the SUBJ-HEAD Rule is used for sentences, but when do we use BARE-NP?

- I am curious how things like the valence principle or GAP principle is implemented in this LKB formalism.

Lab 1 questions

- Why do all the features have such short names? I feel that longer names could contribute to readability.

- There are so many features!! Is every single feature useful in that it connects to another by an identity or inherited constraint, or are some of them more placeholder-y for potential future expansions of identities and associations?

- Why are there so many more features than were studied in LING 566? (What purpose does LKEYS serve?)

Lab 1 questions

- What are ARGS and C-COUNT, why would they only appear in non-lexeme type strucuture?

- What are olist and alist? What is the difference between these two?

- Why would we need SUBJ and SPEC in our VAL when we already have SPR and COMPS?

- What exactly makes the SUBJ-HEAD and HEAD-SPEC rules need to be different grammar rules (vs. a single broader Head-Specifier Rule for both constructions, for example)?

Lab 1 questions

- How do you insure that you are dragging the failed parsed string on to the right part of the AVM to get a unification failure that is the result of a missing rule and not just wrong usage of the grammar?

- Can one mechanically use LKB without knowing much about 566 and HPSG? Just from a tool usage perspective.

- When dragging the arguments for interactive unification, I wasn't sure which level of the phrase to drag (for example, if there are several NPs on top of each other). I see when hovering over them that they're associated with different lexical rules, but I'm still not clear on this. Should we always drag the topmost one (that is, just under where a larger phrase has failed to form?)

Lab 2 questions

- “In order to encourage people to get started early, this assignment requires that you post a question to the discussion on the course Canvas by Tuesday night. Examples: A question about something in these instructions that's unclear, a question about something in your grammar resources that's unclear, or a question about something in the customization system that's unclear. Or all of the above! Ask away :)”

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 LIFT 2019 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

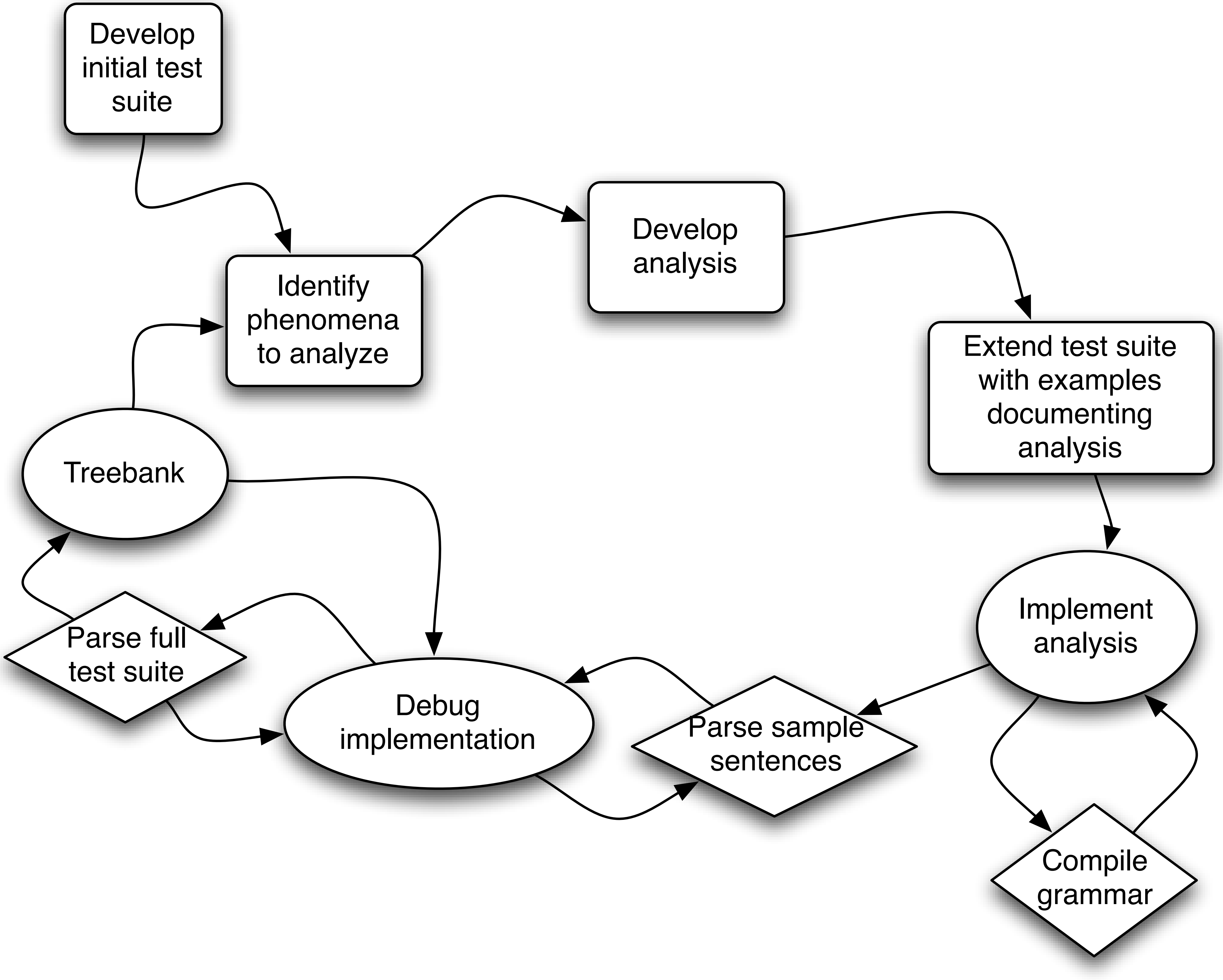
Test data

- 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



Develop initial test suite

Identify phenomena to analyze

Develop analysis

Extend test suite with examples documenting analysis

Implement analysis

Compile grammar

Parse sample sentences

Debug implementation

Parse full test suite

Treebank

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

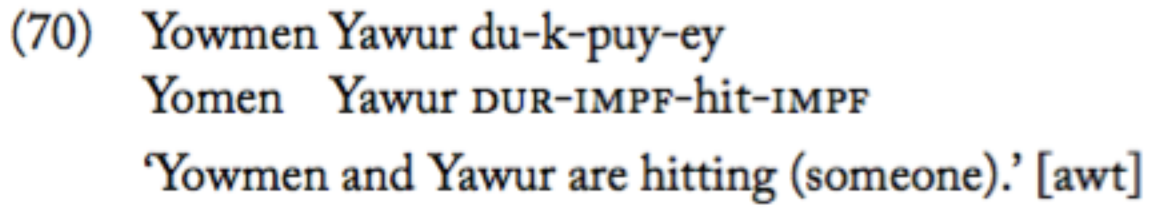
- Why keep examples simple?

- How simple is too simple?

- What kinds of things make an example not simple enough?

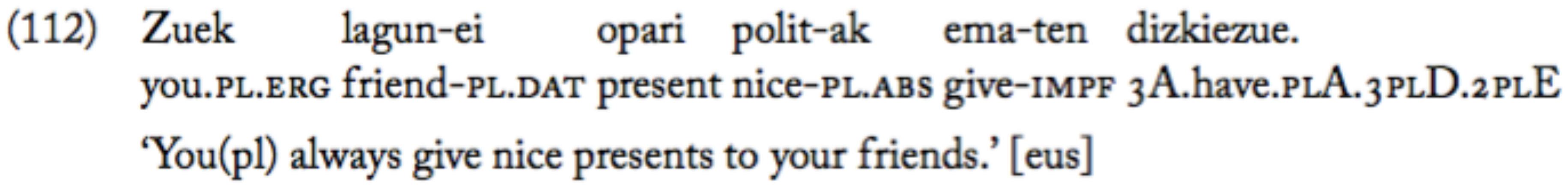
On the importance of simple examples

- 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]

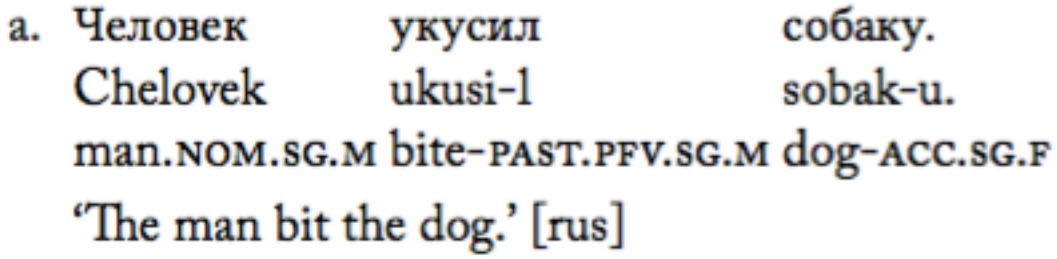
- Basque [eus] (adapted from Joppen and Wunderlich 1995:129)



(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

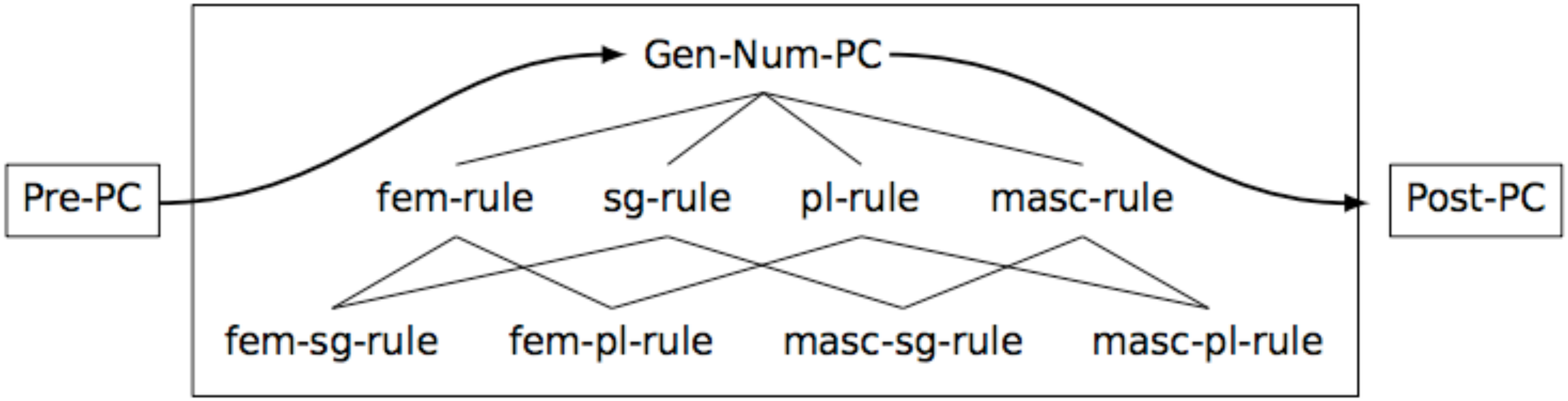


Figure 9: Example lexical rule type hierarchy in a position class

(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 ] ] ].
```

Thursday = demo day

- Send me questions by noon on Wednesday; 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