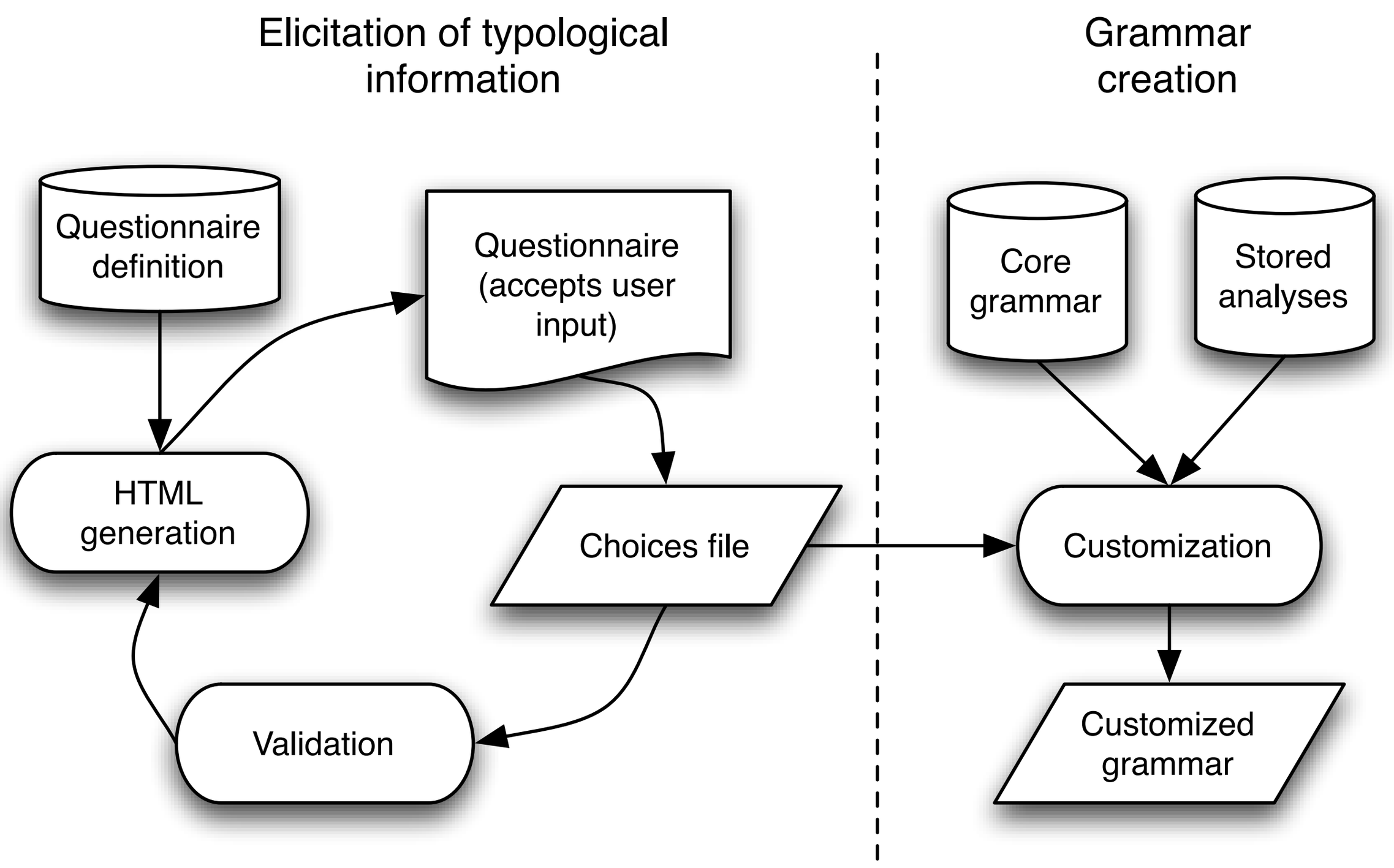
Grammar Matrix;
Test suites, [incr tsdb()]

Ling 567
Jan 13, 2025

Outline

- Grammar Matrix
- Questions from Lab 1
- Testsuites
- If time: [incr tsdb()] demo
- Morphotactics in the Grammar Matrix (preview)



(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 & customization logic
- 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

Questions from Lab 1: Software/workflow

- When I tried to load grammar using the LKB Top menu, I would run into some error and get a “process inferior-lisp killed” message. But I was able to load grammar by entering the path. Why?

- Do the .tdl files always have to be named the way they do for LKB to figure out what is what?

- How does LKB actually go through all these files to load the grammar?

- Is there a best practice or a way that more experienced individuals approach getting acquainted with all these files?

- How do I push files from my guest OS to my host OS (using MAC)?

- Edges vs. nodes??

Questions from Lab 1: Grammar Matrix

- How should we decide whether a rule goes into the matrix or into a specific language's set of rules? If a grammatical phenomenon holds in 90% of languages, can we treat it into matrix and then just add patches for the remaining cases? Or does our framework allow each language to modify the universal grammar as needed like using :+? Can we define :-?

- I looked through the tdl files and am not sure how to interpret the content in head-types.tdl.

- What are labels? And how are they associated with the feature structure.

Questions from Lab 1: Grammar Matrix

- I noticed at the end of matrix.tdl there is a PHON section that is commented out. Does any of the existing languages use this section? I don't know yet.

- What is the difference between CONT and C-CONT? Does it function similarly to the relationship between SPR/COMPS/GAP and ARG-ST?

- What degree of familiarity of matrix.tdl should we have in order to be able to proceed well in our further assignments (and take good care of the target language)? Also, are there any types that we need to pay attention to in particular so as not to fall into traps?

Questions from Lab 1: tdl

- Are the punctuation and indentation in type definition strictly required?

- What is the point of the snippets in files like rules.tdl? Is the shorter notation just for simplicity? Or does it have a function?

Questions from Lab 1: types

- Some types only inherit from their supertypes and don’t add anything new. Why do we need such types?

- Occasionally when looking for sentences that do not parse, the red highlight in the unification failure window would be on 'cons null.' What does this mean? It seems like an error on my part but I'm also not sure where it's coming from.

- How do the different types of lists function?

- When are we supposed to use cons vs. cons-copy, and null vs. null-copy?

- How is the Wrapper type LIST used in comparison to the Computation type APPEND- RESULT, as they both seem to hold the result of appending alists?

Questions from Lab 1: Software wishlist

- Is there a way to identify the tag identity that causes a unification failure directly (i.e. with a dedicated tool) via the LKB? I used the tools listed on the lab assignment and discussed in class (e.g., grep and examining local AVMs), and I would still like to hear more about this.

- Is there a way to search by feature names (similar to when reading files via emacs) in the AVM view of LKB? I haven't found a way.

- Viewing the parse chart and then pulling up type hierarchies was very helpful for trying to get a grasp on how these specific hierarchies are interacting- is there any less clunky way to view them in emacs? I'm using the lui interface and still trying to find a good size for the font and window that doesn't make the messy ones look like grammar spaghetti. :(

Questions from Lab 1: Chain of identities

- In the chain of identities, how is the index of "cat" identified with the arg0? I went digging in the lexical type definitions and did not find an answer.

- Which rules indicate that SYNSEM.LOCAL.CONT.HOOK of "many dogs" is the same as C-CONT.HOOK of "dogs"? Although they are intuitively consistent and I have checked in LKB that they are indeed the same, I cannot find specific rules to prove this.

Questions from Lab 1: Analyses

- What should the rule for nouns being used in a general sense (dogs, cats, hamsters, etc) look like? I'm still thinking about this.

- Why is it necessary to consider the pragmatic feature like information structure in ICONS when constructing grammar?

- Why do we distinguish between subj-head and spec-head?

- What are some example KEY values, so that, for example, a verb can select for some PP by sharing the same KEY value with PP?

- What's the ARG0 of _chase_v_rel, or any X-relation? Is it something like self?

Outline

- Grammar Matrix
- Questions from Lab 1
- Testsuites
- If time: [incr tsdb()] demo
- Morphotactics in the Grammar Matrix (preview)

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

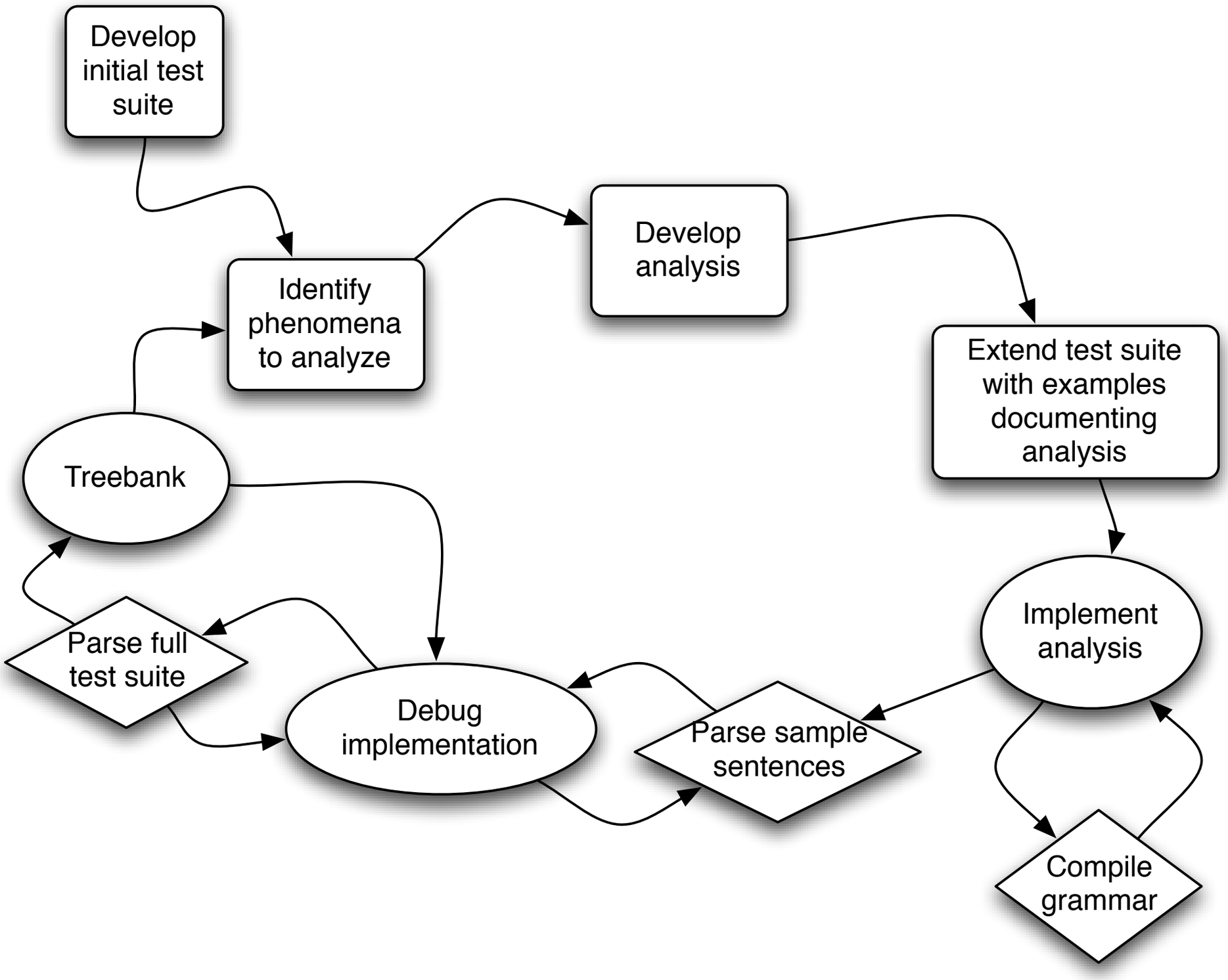
Test data

- 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]

[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

Outline

- Grammar Matrix
- Questions from Lab 1
- Testsuites
- If time: [incr tsdb()] demo
- Morphotactics in the Grammar Matrix (preview)

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

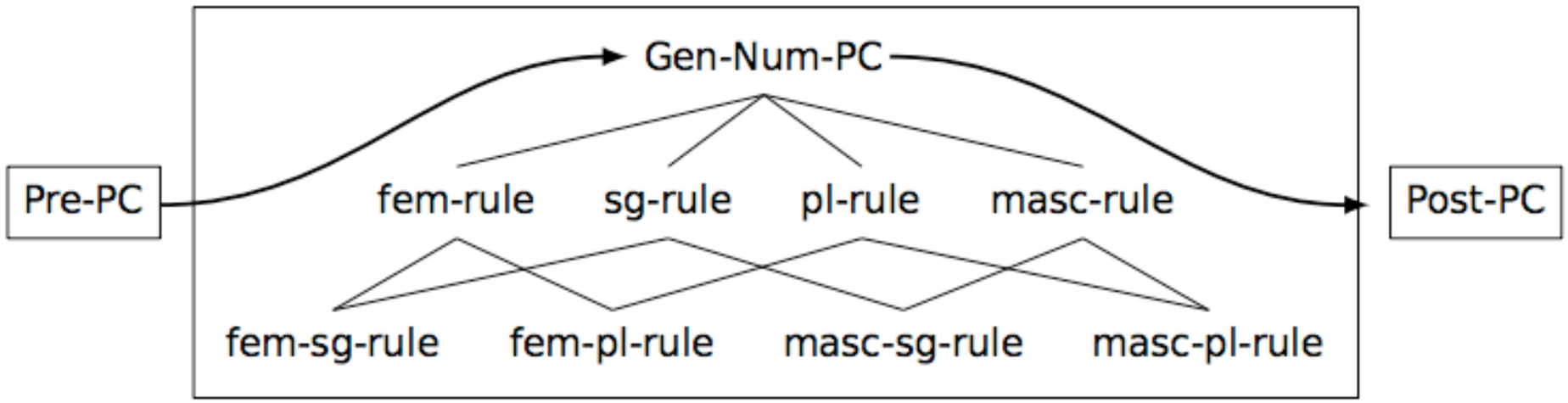


Figure 9: Example lexical rule type hierarchy in a position class

(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 ] ] ].
```

Lab 2 notes

- Post questions by Tuesday night
- Hover over red asterisks
- [incr tsdb()] woes:
  - Do format Index.lisp precisely as shown
  - Don’t put tsdb/ in a shared folder
  - Don’t let Windows corrupt your item files
  - LD_LIBRARY_PATH—see Canvas

Wednesday = demo day

- Send me questions by 2pm 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