Test suites (continued)
Grammar Customization
Lab 2 Phenomena

Ling 567
January 16, 2024

Overview

- [incr tsdb()]
- Grammar customization
- Lab 2 phenomena

[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)

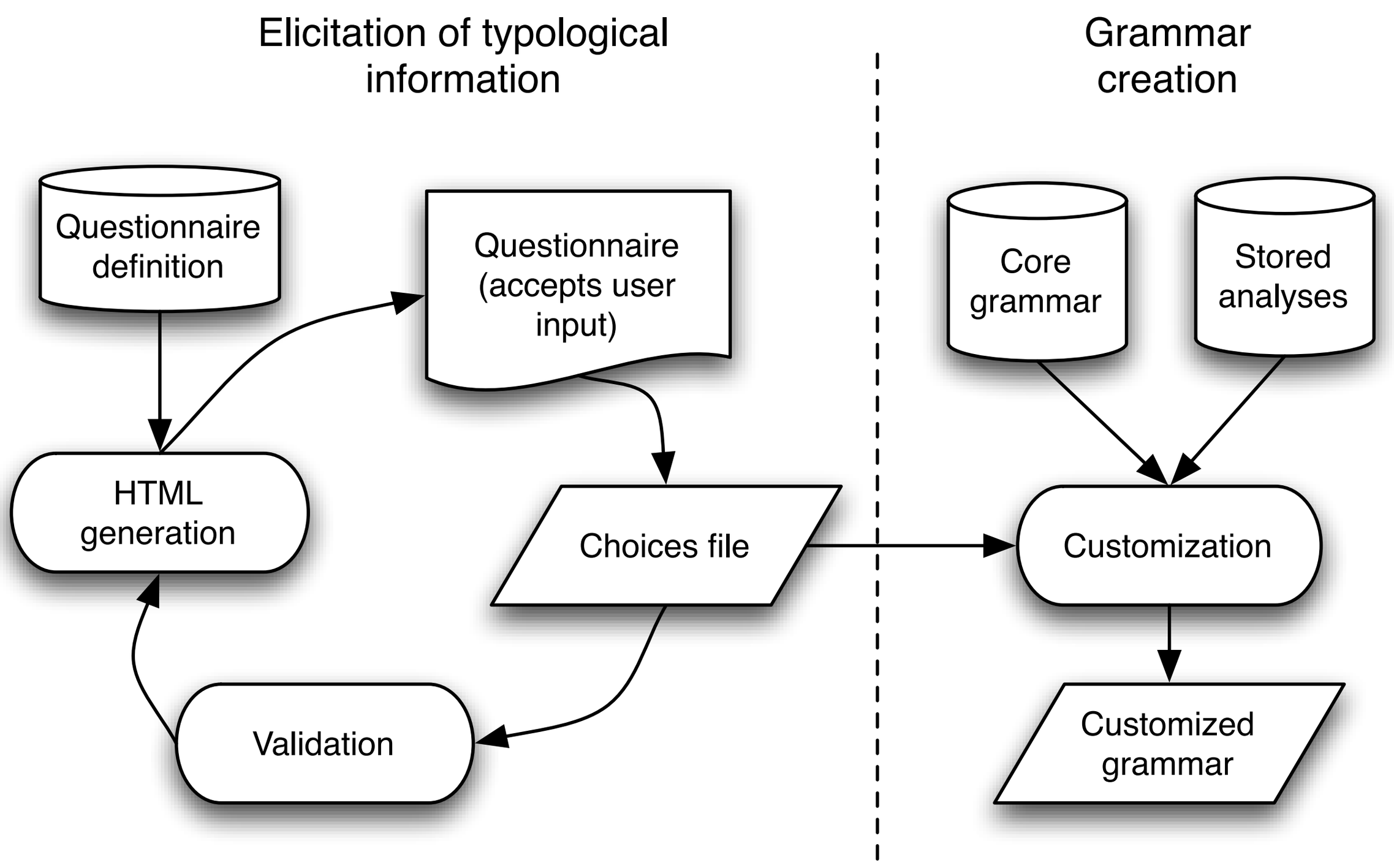
[incr tsdb()] basics

- 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

- -> Demo



(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

Word order

- Even fixed word order languages have constructions that allow other orders
- Pick a fixed (or more fixed, like V-final) word order if it seems like something special has to be going on to get the other orders
- Pick free word order (or freer, like V-final) if the choice seems to be very flexible / vaguely pragmatically driven

Lexicon

- Full form lexicon: That means separate entries for e.g. *neko-ga* (cat.NOM) and *neko-wo* (cat.ACC).
    - You'll need separate noun types for accusative and nominative nouns
    - For tense/agreement info on verbs, just leave off the constraints (no need to make extra types)
- True clitics are words to the syntax; spell them with = rather than - and instruct the LKB to strip the = but keep -
- In general, make sure the spelling your lexicon & testsuite match

Pronouns

- Personal pronouns only (not demonstratives, possessives, relative pronouns; wh pronouns under wh questions)

- Will entail working on person/number/gender and possibly case

- Looking at free pronouns only — not verbal affixes functioning as pronouns

Case

- Variation in the form of NPs based on their syntactic role in the sentence
- Might be marked on noun, on nominal dependents (determiners, adjectives) or both — or via separate case-marking adpositions
- Might be marked differently on pronouns
- Might function differently with pronouns (“split-ergativity”)

The rest of the NP

- What's up with determiners?
- Is there anything else required in NPs?

Tense

- Tense encodes the three-way relationship between speech time, event time and an anaphorically determined reference time.
    - Past/present/future
    - Past/non-past
    - Future/non-future
    - Recent/remote past/future

Aspect

- Aspect encodes the description of the internal temporal properties of the event and how it's viewed.
    - Internal temporal properties: Aktionsart or situation aspect
    - How it’s viewed: Viewpoint aspect
- Perfective/imperfective
    - NB: Perfective != perfect
- Inceptive, continuative, progressive, habitual, iterative…

Wh questions

- We are interested here in utterances where the whole utterance is a question, and the question concerns the identity of one (or more) of the arguments of the main verb. In the simplest case, there is a single clause and one argument is questioned: *Who did the child see?*

- Ungrammatical examples will depend on what constraints you find, but may involve wh elements in the wrong position, involve wh elements in the wrong form (e.g. case), or hinge on ancillary properties of the construction (e.g. lack of inversion where it is required).

- Optional: embedded wh questions and wh questions where the questioned element is an adjunct (*how, why, when, where*).

Wh questions
Find out and then document in your testsuite:

- The shape of the wh words for core arguments. Do these vary with case, animacy, gender, something else?

- The possible positions of wh words: Do they appear where an ordinary argument would? Move to the beginning of the clause? Are both of these possible?

- What happens if the questioned argument belongs to a lower clause (e.g. *Who did the observer think the child saw?*)?

- Are there any other differences between wh questions and declaratives (or yes-no questions)?

- Are there are any differences between wh questions concerning subject and non-subject arguments?

- Optional: What happens with multiple wh elements in the same clause (e.g. *Who saw what?*)

Things to discuss

- Reminders: Test early & test often!
- Keeping the scope manageable, especially in the face of tempting tables and paradigms
- When the customization system can’t handle something
- What should go in the testsuite?
- What should go in the choices file?
- What should go in your write up?

Thursday = demo day

- Send me questions by noon on Thursday; all should include:
    - Question
    - Choices file
- 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)

Questions from Lab 1: Customization system functioning

- I am not sure how we are going to generate the script for our respective languages. I believe we will use the grammar matrix customization page, but we uploaded a file to get the English file. This still seems very abstract and confusing to me. Can you explain how this process works further?

Questions from Lab 1: Processor functioning

- In the "basic-head-spec-phrase-super" in question 4, I'm not sure how the things in the definition get identified with the arguments of the unification result. I would appreciate more detail on these definitions in general, and how it is translated to the grammar rule AVM.

- I'm confused about the different types of software we are using. What IS the LKB (e.g. a database? a programming language?)? What part of what we worked with was the Grammar Matrix (just the part where we generated and downloaded the grammar on the web?)? What is Ubuntu? What is ace doing?

Questions from Lab 1: Processor functioning

- Are the tags specific for each node, such that they don't repeat between nodes when viewing subtrees of the entire structure independently? Otherwise wouldn't the tags 6 4 and 12 consolidate to the same tag number? (Those might not match with what you see, but they were all referencing the same entity.)

- In "Testing out generation" instructions, what does the choice for number of edges mean?

Questions from Lab 1: Processor functioning

- Why are we able to enter the past tense verb "danced" and have it parse? I see that there is a past tense lexical rule `past-lex-rule`, but how does the system know to associate "danced" with "dance?" In LING 566 we would have used a function `F_past(original_verb)`. Where do those functions live? How do we populate them? How does the system automatically change words for tenses when only the base form exists in the lexicon.

- It seems as though we are now treating lexical rules in a similar capacity to grammar rules (e.g. The identity described in my second .tdl snippet that would convert the lexeme for cat into the word cat). Is this perception correct? If so, would all lexical rules be visible in our syntax trees?

Questions from Lab 1: Using the software

- At times, the avm windows would become unresponsive and I would have to restart the virtualbox. I've allocated as much memory as recommended for my laptop (no more than 80%).

- When I use grep to find a type, it only shows me its supertype(s), without the associated constraint. For example, when I execute grep basic-head-subj-phrase := *.tdl, it simply shows matrix.tdl:basic- head-subj-phrase := binary-nonloc-phrase & binary-headed-phrase & head-compositional &. Do I have to manually search in the tdl file to see the constraint? I'm still not sure if this is supposed to be this way or I'm using grep in a wrong way.

- What does lkb stand for?

Questions from Lab 1:
Using the software

- Why is running lkb through emacs needed/recommended?
- Is it common for lkb to crash when inputing a string to parse?
- There's a grey square that appeared after I selected 'View > Grammar Rule' that crashes my VirtualMachine. What is that?
- The feature structures are far more intricate than the ones that we worked with in 566. Just finding the INDEX feature was difficult. Is there any way to search a feature structure within emacs? Would it be possible to use grep or something to search the data within the structure?

Questions from Lab 1:
Specific features & types, tdl syntax

- What does the < mean?

- What is ref-ind and how is this different than ref-ind event?

- Going off the above question, when we have instances like ref-ind event that result in unification error, how can we back trace the error?

- I did a lot of poking around to try to find where the NUM and PER values of the verb are identified with its subject, which caused me to find PNG. Why are PER and NUM part of PNG (and therefore part of HOOK?) rather than being part of AGR where I assumed they would be? Also, what is XARG?

Questions from Lab 1:
Specific features & types, tdl syntax

- Why do some lex types have lex-item as their supertype (e.g. intransitive-verb-lex having a supertype intransitive-lex-item), and what does it signify? Why is such a relation necessary? I would like to see this addressed.

- In the given English grammar, why are many types split into a "normal" version and a "basic" version?

- I am uncertain what the head-subj rule is. I saw it used in an example on Canvas to show unification failures with regards to an agreement mismatch, but it is not clear to me why that was tested first for say "Cat dance" instead of the head-specifier rule.

- What is the difference between the subj-head rule and the head-spec rule? : I assume that subj-head is intended for np-vp while head-spec is for det-noun, but I would still like to get clarification on this.

Questions from Lab 1:
Specific features & types, tdl syntax

- How do we locate rules and principles from 566 in this grammar? This seems to be far more elaborate than what we learned and situating our checkpoints within the new terms and expansions was difficult in this assignment. - This one hasn't been answered so much but again I think exposure is making the layout more approachable.

- I remember asking in LING 566 about the redundancy of word structures, where impossible words can be realized, and the answer was 'yes'. How are impossible words created by the grammar matrix, and where is it stored?

- Why are there so many features? Are all of them necessary? What problems arise if we just use the features that we learned in class?

Questions from Lab 1:
Specific features & types, tdl syntax

- Does the TFS type system make a distinction between a type and a value?

- What is olist? I saw this in the TDL as `olist := list` and `list := avm`, and `avm := *top*`, but what is top?

- Why are the rules defined across separate .tdl files? I cannot see the pattern that is governing their separation.

- Are the constraints in the matrix.tdl file language-independent ('universals'), and the English.tdl file contains language-dependent constraints?

Questions from Lab 1: Interactive unification

- When choosing which NP or VP to drag, are there times when it matters which one we choose? Would we want to include unification failures from both NPs in a write up?

- What would the first point of failure be in parsing "Many dogs chase cats"? It does not parse. Is this just a limitation of the small grammar we are testing out?

- For the HEAD-SPEC rule, I was trying to find anything that said that the first element we input in ARGS becomes the second element in ARGS. Are there tags that state this? Or do we just assume that it’s true because that’s what we say the rule does.

Questions from Lab 1: Better software

- Related to my software question, but slightly different: Why does it only run on a virtual machine? Where/how did you build the program? Why must it only run on a virtual machine with extremely ugly graphics? Would it be possible to transfer the data and code to a nicer looking enviornment (if someone were willing to put in the effort)?

- Is it possible to use Docker as a VM solution to run the LKB Ubuntu? I don't necessarily want to, but I was under the impression that Docker is a modern replacement for VMs.

- Have there been attempts at, or is there any interest in developing native or web apps that use LKB as a backend?

Questions from Lab 1: Chain of identities

- In the chain of identities, I was struggling to find the tags that encode the chain for the NP, S, and VP. Where are the tags that encode that chain for the index?
- What is the constraint that identifies arg1 of chase relation with XARG of ‘chase’? Does XARG always refer to something about the subject/first argument in ARG-ST?
- I notice the examples of links in the chain have unnecessary info removed from the tdl excerpts. How can this be done? (requires understanding the syntax, which I don't currently understand much).
- I followed the singular-noun-lex-rule attempting to find the #index or #args, and only found ARGS and RELS in in lex-rule of matrix.tdl. Would this be how the index is passed up or have I missed something else entirely
- I would like to step through the entire chain of identities(or most of it) for 5.