CLIMB: Code generation for grammar development

Antske Fokkens

Saarland University

Outline

Outline

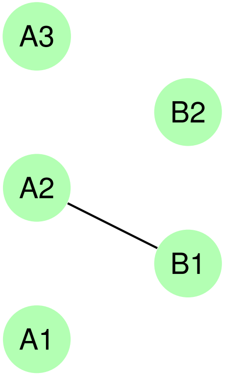
Multiple hypotheses

- Theories (whether scientific or philosophical) aim to explain phenomena
- But what if there is more than one hypothesis that can explain the phenomenon?
- Ideally, all possible hypotheses are maintained until evidence is found that exclude them
- The scenario of multiple hypotheses is common in syntax and grammar engineering; maintaining them until evidence is found to make a decision is not

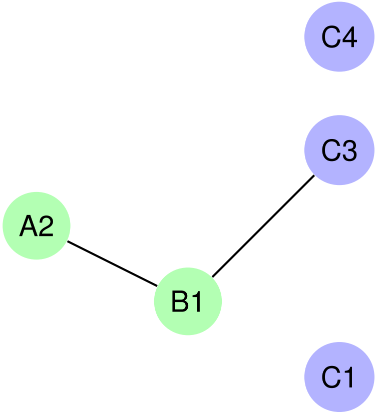
How to know what analysis to pick?

- Ability to account for data
- Interaction with other analyses
- Theoretical soundness: how well does the analysis fit to general theoretical assumptions
- Elegance/simplicity
- Efficiency

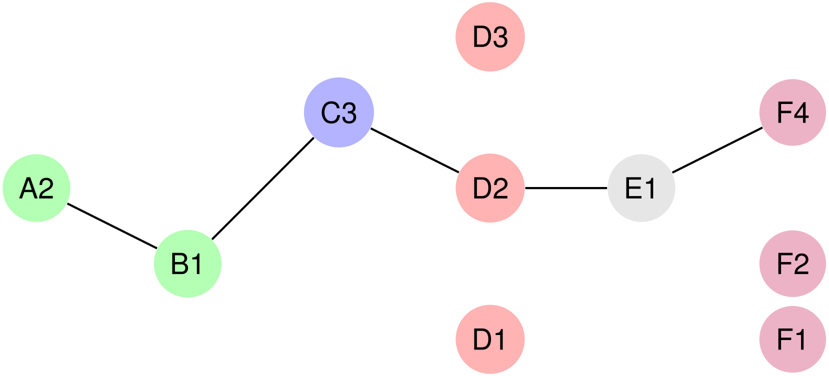
Grammar development



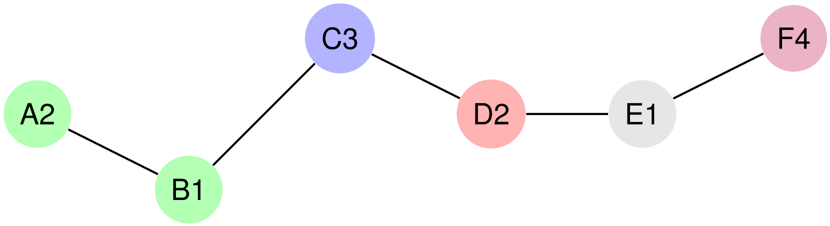
Grammar development



Grammar development



Grammar development



Practice: Select best analysis according to criteria given current knowledge

Interaction

- Often, there is no conclusive evidence indicating what “the” correct analysis is
- **Phenomena interact**: what if an analysis chosen in the past excludes the optimal solution for a new phenomenon to be added?
- Analyses can be revised based on new evidence, but this becomes less and less likely as time passes (chosen analysis deeply embedded, alternatives forgotten)

$\Rightarrow$ The order in which phenomena are treated may have a major impact on the resulting grammar

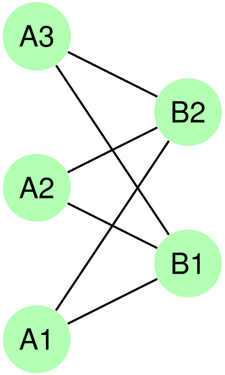
Outline

Metagrammars for systematic exploration (Fokkens, 2011)

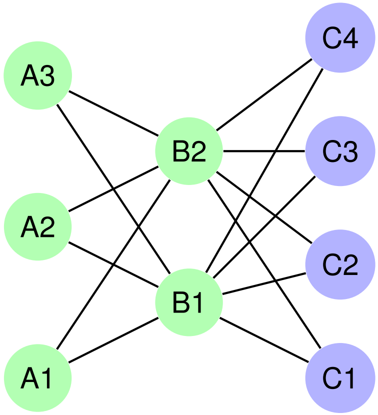
Can we keep track of choices made in the past and preserve alternative solutions?

- Instead of directly implementing a grammar, analyses can be stored in a metagrammar
- The metagrammar can generate grammars with alternative analyses that cover the same phenomena
- Different alternatives from the past can be tried out, when new phenomena are added to the grammar

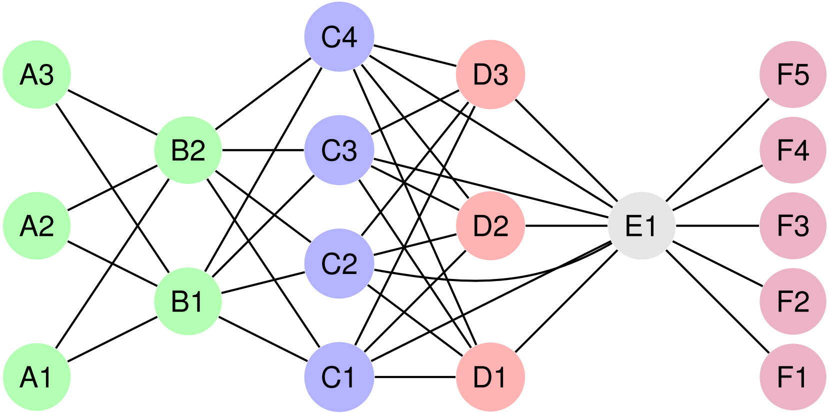
Possibilities



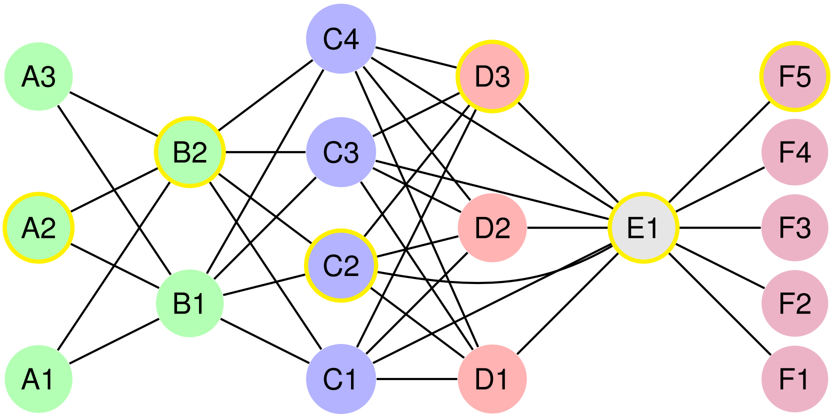
Possibilities



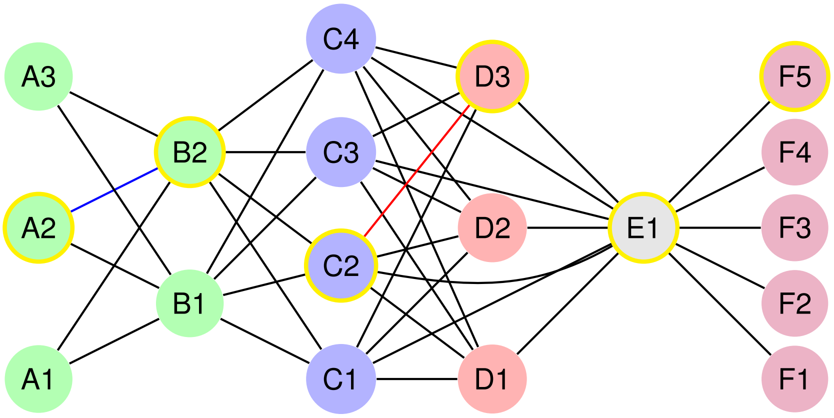
Possibilities



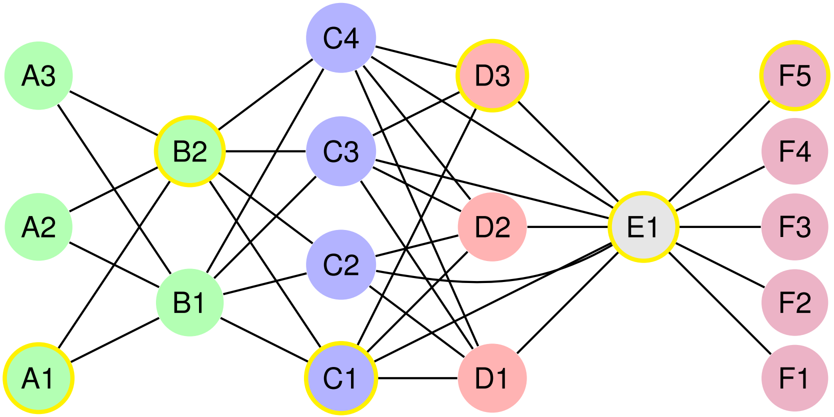
Possibilities



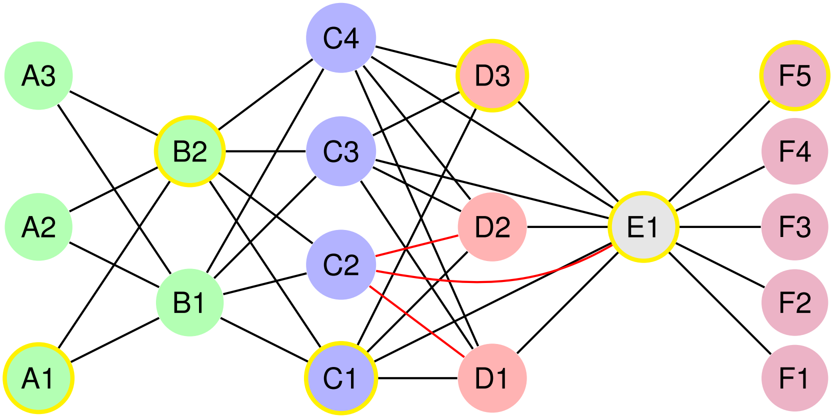
Possibilities



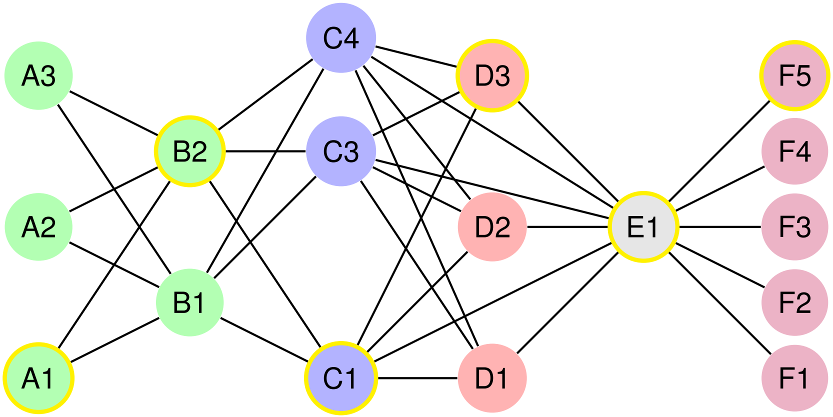
Possibilities



Possibilities



Possibilities



Outline

The LinGO Grammar Matrix (Bender et al., 2002, 2010)

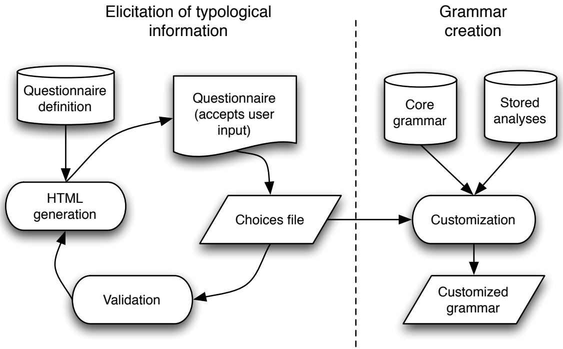
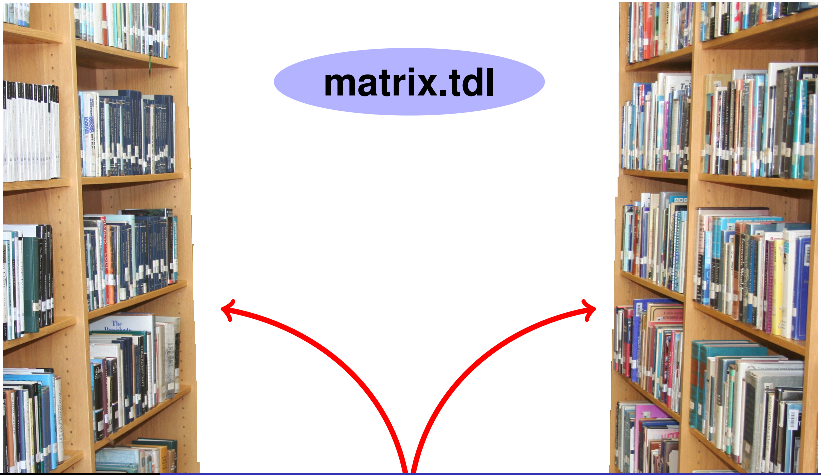


Figure: Schematic system overview

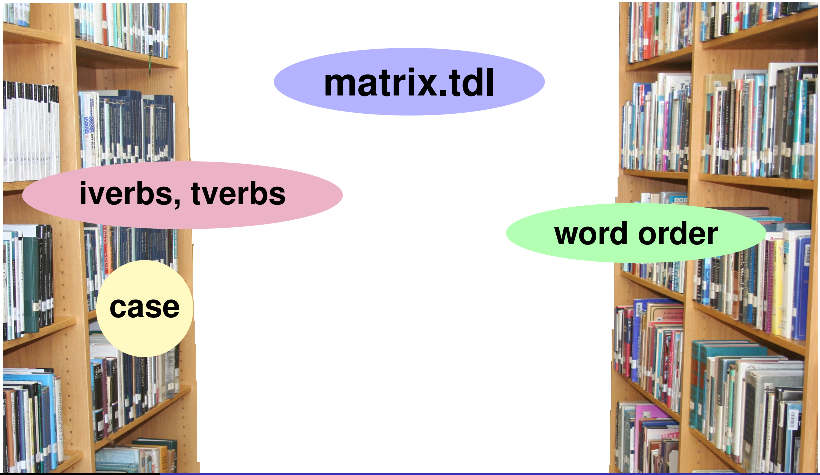
Grammar Matrix Workflow

matrix.tdl

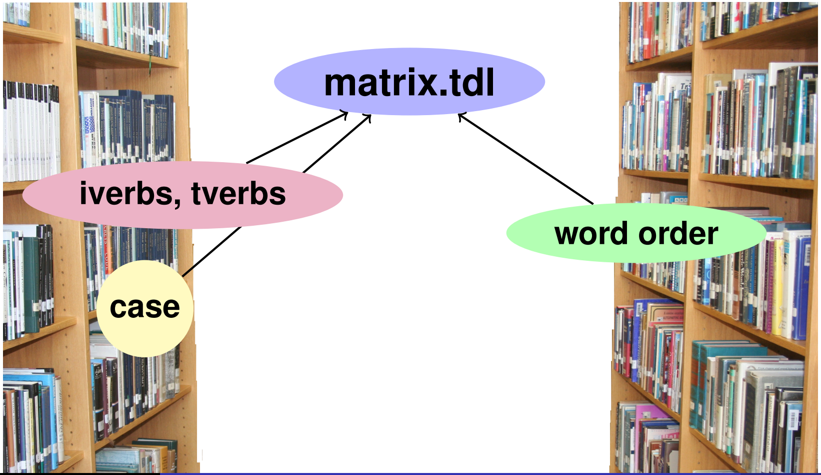
Grammar Matrix Workflow



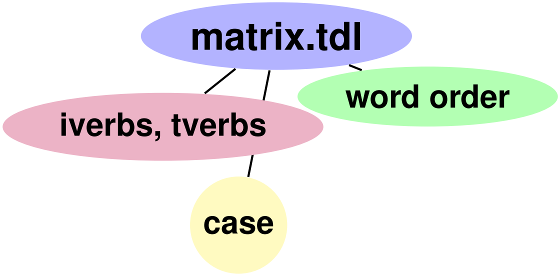
Grammar Matrix Workflow



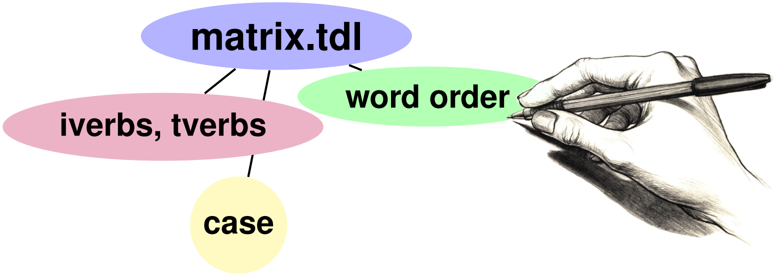
Grammar Matrix Workflow



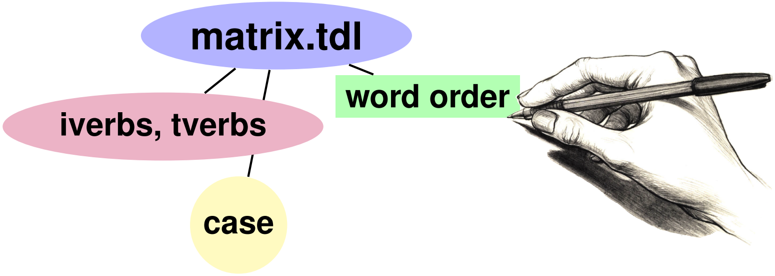
Grammar Matrix Workflow



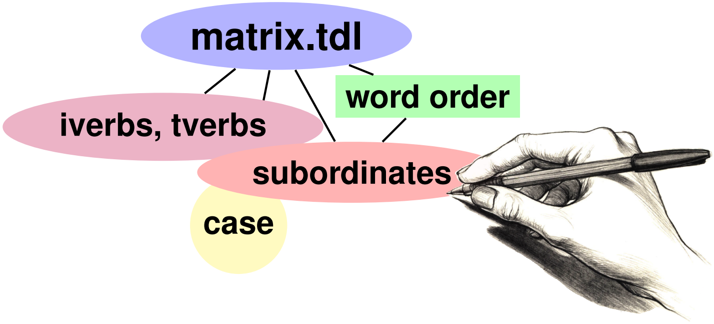
Grammar Matrix Workflow



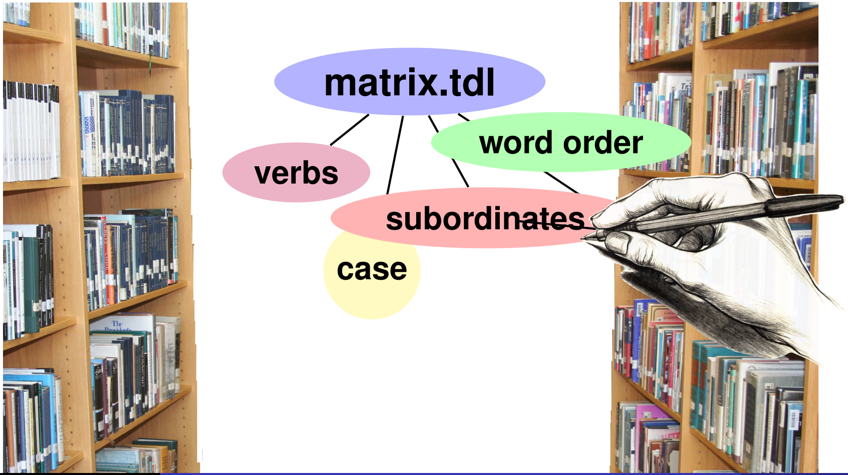
Grammar Matrix Workflow



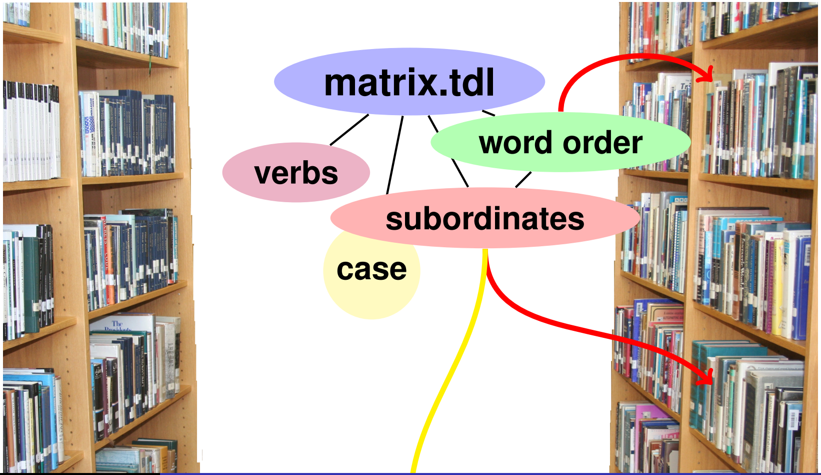
Grammar Matrix Workflow



Metagrammar engineering



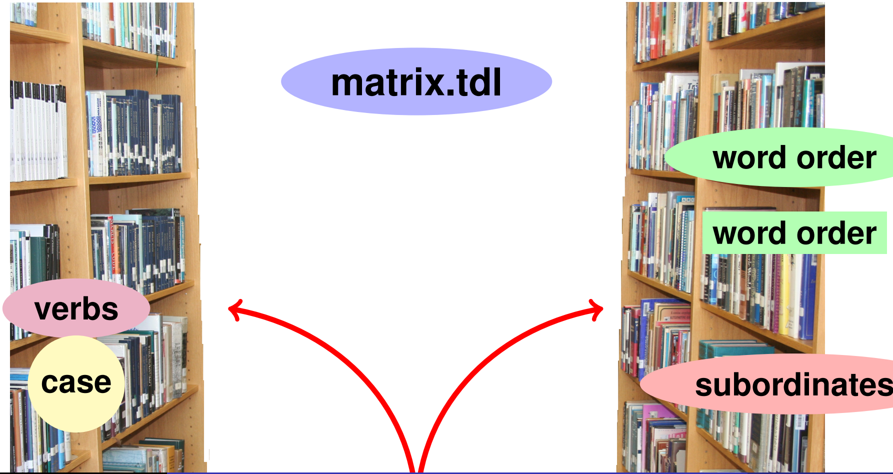
Metagrammar engineering



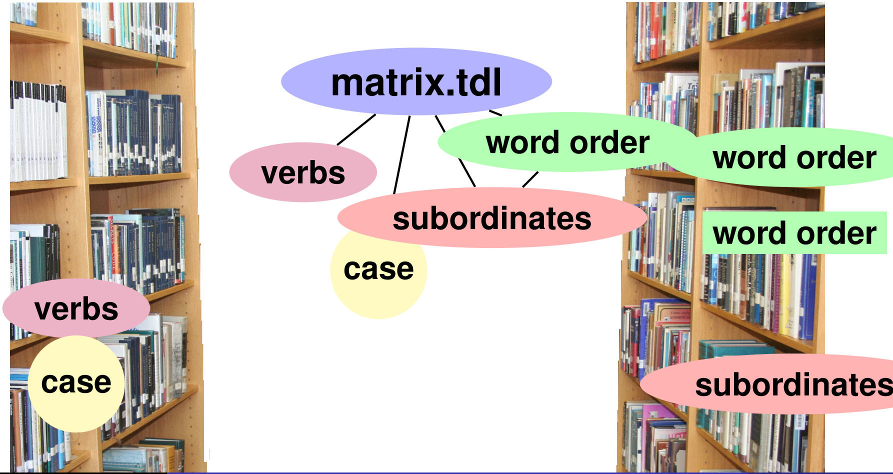
Metagrammar engineering

matrix.tdl

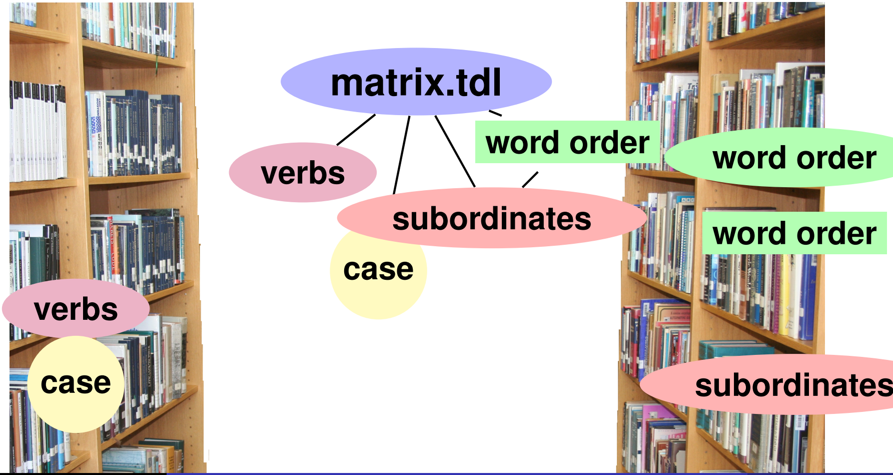
Metagrammar engineering



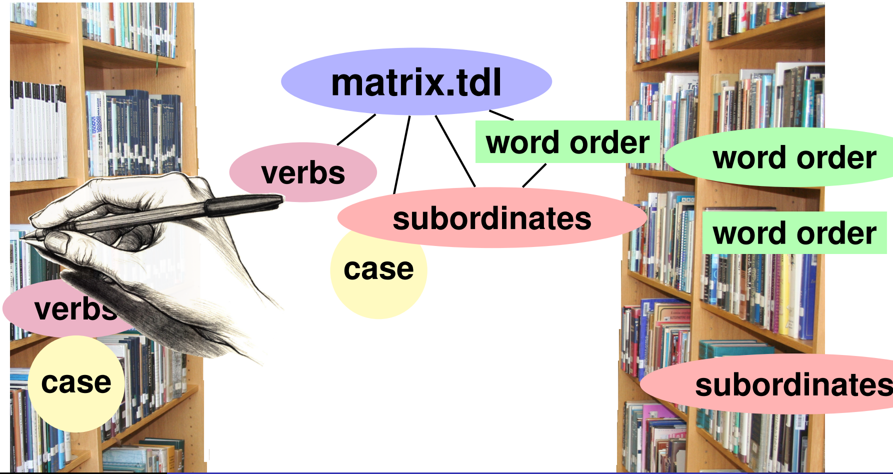
Metagrammar engineering



Metagrammar engineering



Metagrammar engineering



The Grammar Matrix and CLIMB

The Grammar Matrix and CLIMB (Comparative Libraries of Implementations with a Matrix Basis) (Fokkens et al., 2012) complement each other

Grammar Matrix	**CLIMB**
- Support to start new grammar	- Supports long term development
- Accessible to new users	- For expert developers
- Focus on wide typological coverage	- Focus on high coverage in one or few language(s)

Advantages of CLIMB

- Systematic comparison of implementations
- Facilitates maintainance of different versions of the grammar (e.g. for different applications)
- Phenomena-based organization of the grammar
- Can provide direct feedback to Grammar Matrix

Outline

Germanic CLIMB, Goals

- Test impact of the methodology on long(ish) term grammar development
- Compare analyses for word order and auxiliaries in German
- Keeping the complete Grammar Matrix set-up: what changes are required in the customization system (extensions versus revisions)

Germanic CLIMB, Current state

- Coverage of Cheetah's development set obtained in 6 months (compared to 1 year reported in Cramer (2011))
- Efficiency (preliminary result) between GG and Cheetah
- Revisions of several analyses in matrix.tdl:
    - adjectives
    - modification
    - wh-questions
    - long distance dependencies
    - relative clauses
    - adpositions

Slavic CLIMB, Motivation

- The initial set-up of SlaviCore (Avgustinova and Zhang, 2009):
  - matrix.tdl
  - slavic.tdl (static file with types useful for Slavic languages)
  - my_slavic_language.tdl
- Adapting SlaviCore to CLIMB:
  - Increase flexibility for sharing analyses across Slavic languages
  - Through automatic generation of parts of the hierarchy, Avgustinova (2007)'s theory on Slavic grammar analyses can be tested based on observation in individual languages

Slavic CLIMB, current state

- The Russian Research Grammar (Avgustinova and Zhang, 2010, RRG) has been ported into CLIMB
- **Observation**: much could be done by changing ‘choices’: $\Rightarrow$ customization system could be exploited more!
- Many implementations are currently purely technical
- Linguistically motivated organisation has started taking implementations from BURGER (Osenova, 2010) into account
- ‘Clean’ implementation for case

Chinese CLIMB (future project)

- Combine analyses from independently created grammars
- Shared development at different sites (Uds/DFKI and NTU)

Outline

Feature path abbreviation

- Slavic CLIMB contains a basic algorithm that allows users to abbreviate paths (writing out the full path in TDL)
- The algorithm goes backwards through the path until it finds the preceeding feature, or hits *sign*
- It complains when the path is ambiguous
- The first element of a list needs to be stated explicitly (future work: allow users to define defaults)

Feature path abbreviations, basic example

- The Head Feature Principle:

$$\begin{bmatrix} \text{HEAD } \boxed{1} \\ \text{HEAD-DTR.HEAD } \boxed{1} \end{bmatrix}$$

- Not possible (SUBJ has *list* as value):
  - SUBJ.HEAD *noun*
- Must be (for now):
  - SUBJ.FIRST.LOCAL.HEAD *noun*

Code generation for larger grammars

- Can be helpful for large revisions in the grammar
- The basic idea:
  - A library specifies unique types for old and new analysis $\Rightarrow$ which types should be removed or inserted
  - This library also specifies types that change
  - Algorithm goes through type hierarchy removing, inserting and changing types based on the chosen analysis
- All changes and revisions related to the new analysis are in one place: easier to revise
- Basic algorithm to flip back and forth between analyses remains: easier to compare analyses at different stages

Spring cleaning (Fokkens et al., 2011)

- The spring cleaning algorithm goes through a type hierarchy and removes types that do not have any impact on the grammar, i.e. types that
  - are not instantiated types
  - do not define a feature or value of an instantiated type
  - do not define a lower bound between two relevant types
- The original structure of the grammar is maintained

Spring cleaning (future work)

- Intermediate stages between spring cleaning and grammar compression
  - Yi Zhang implemented an algorithm that identifies which types are computationally relevant for competence
  - Because features are moved up to supertypes, this can have a negative impact on performance (Petter Haugereid)
- Grammar comparision:
  - The spring cleaning algorithm takes a close look at the structure of the type hierarchy
  - It can be extended to compare two type hierarchies that are highly similar
  - Goal: identify changes made to the grammar after revision $\Rightarrow$ create an interface supporting metagrammar development

Outline

Conclusion

- CLIMB allows engineers to:
  - compare implementations systematically
  - organize implementations according to phenomena (not type)
  - easily maintain different versions of the grammar
- There are two CLIMB projects, plus one more planned:
  - Germanic CLIMB
  - Slavic CLIMB (initial stages)
  - Chinese CLIMB (future work)
- CLIMB comes with some basic algorithms:
  - Path abbreviation
  - Code generation for larger grammars
  - Spring cleaning, plans for grammar comparison

This Summit: CLIMB Tutorial

- Tutorial
  - Basic introduction to CLIMB
  - CLIMB implementation & organization
- Discussion:
  - Lowering the hurdle to start using CLIMB
  - Architecture, basic set-up

Acknowledgments

- Thanks to:
  Tania Avgustinova, Emily M. Bender, Francis Bond, Bart Cramer, Rebecca Dridan, Dan Flickinger, Michael Goodman, Joshua Growgey, Laurie Poulson, Ron Kaplan, Sanghoun Song, Hans Uszkoreit, David Wax, Yi Zhang
- you for your attention $\ddot\smile$

Bibliography I

Avgustinova, T. (2007). *Language Family Oriented Perspective in Multilingual Grammar Design*. Linguistik International: Band 17. Peter Lang - Eurpopäischer Verlag der Wissenschaft, Frankfurt am Main, Germany.

Avgustinova, T. and Zhang, Y. (2009). Parallel grammar engineering for Slavic languages. In *Proceedings of GEAF*, Singapore.

Avgustinova, T. and Zhang, Y. (2010). Conversion of a Russian dependency treebank into HPSG derivations. In *Proceedings of TLT'9*.

Bender, E. M., Drellishak, S., Fokkens, A., Poulson, L., and Saleem, S. (2010). Grammar customization. *Research on Language & Computation*, 8(1):23–72.

Bibliography II

Bender, E. M., Flickinger, D., and Oepen, S. (2002). The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In Carroll, J., Oostdijk, N., and Sutcliffe, R., editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Cramer, B. (2011). *Improving the feasibility of precision-oriented HPSG parsing*. PhD thesis, Saarland University.

Fokkens, A. (2011). Metagrammar engineering: Towards systematic exploration of implemented grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Portland, Oregon, USA. Association for Computational Linguistics.

Bibliography III

Fokkens, A., Avgustinova, T., and Zhang, Y. (2012). CLIMB grammars: three projects using metagrammar engineering. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, Instanbul, Turkey. European Language Resources Association (ELRA).

Fokkens, A., Zhang, Y., and Bender, E. M. (2011). Spring cleaning and grammar compression: Two techniques for detection of redundancy in HPSG grammars. In *Proceedings of the 25th PACLIC*, Singapore, Singapore.

Osenova, P. (2010). BUlgarian Resource Grammar â Efficient and Realistic (BURGER).