Metagrammar engineering in a multi-lingual context

A thesis proposal

Antske S. Fokkens

Saarland University, DFKI Projektbüro Berlin

Outline

Outline

Metagrammar

Metagrammar

Software that can generate an implemented grammar based on given input

Metagrammar engineering

- METAGRAMMAR (Candito (1998))
  - Grammar Optimization
  - Code sharing
- GF (Ranta (2009))
  - Code sharing
  - Providing linguistic expertise
- PAWS (Black and Black (2009))
  - Support language description
- LinGO Grammar Matrix (Bender et al. (2010))
  - Code Sharing
  - Support starting new grammar
  - $\Rightarrow$ Comparison between analyses (this work)

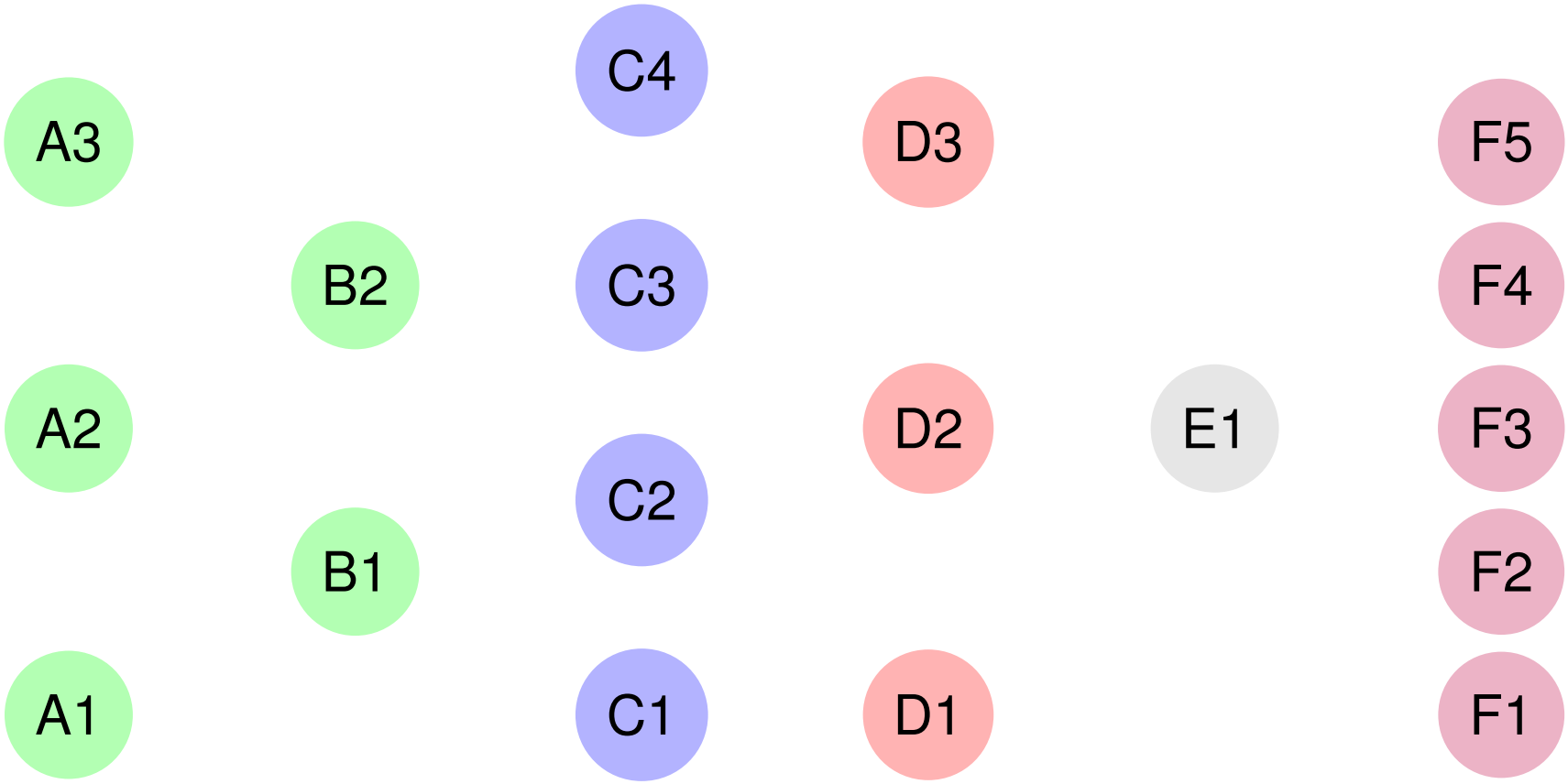
Formal grammars of natural language

Two well-known challenges of formal grammars of natural language:

1. Typically, more than one analysis can account for the data
2. Syntactic phenomena interact

→ The combination of these two challenges makes it harder to address them

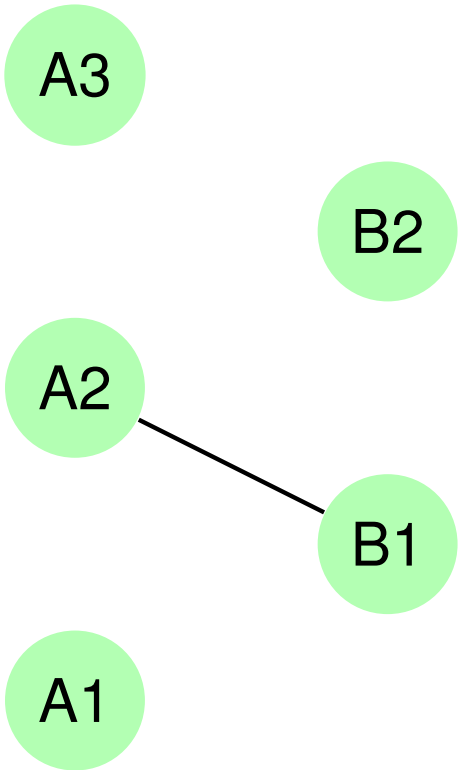
Several Possible Analyses



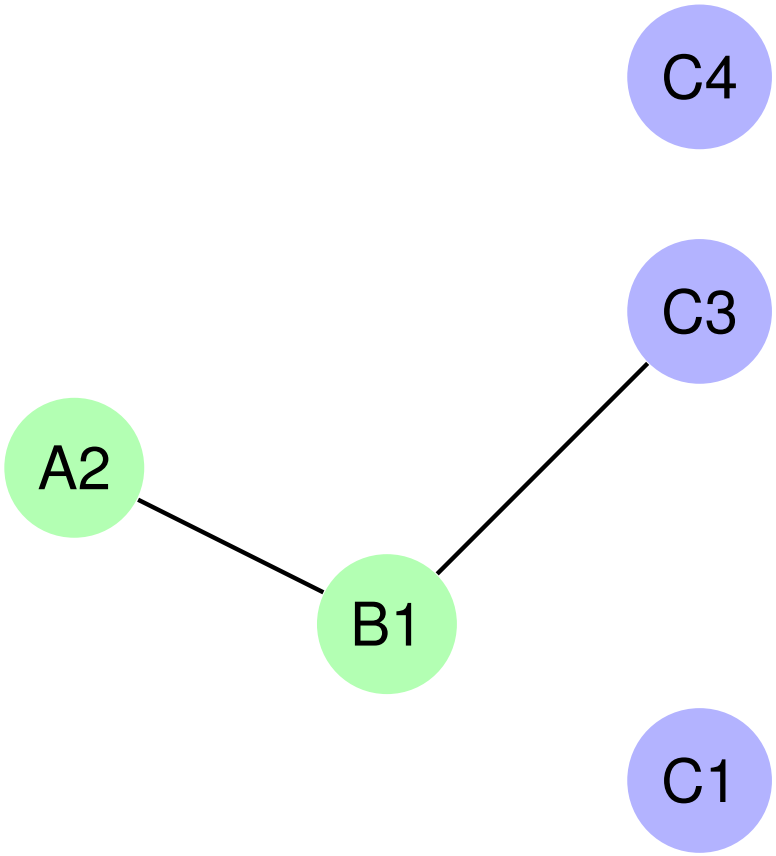
How to know what analysis to pick?

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

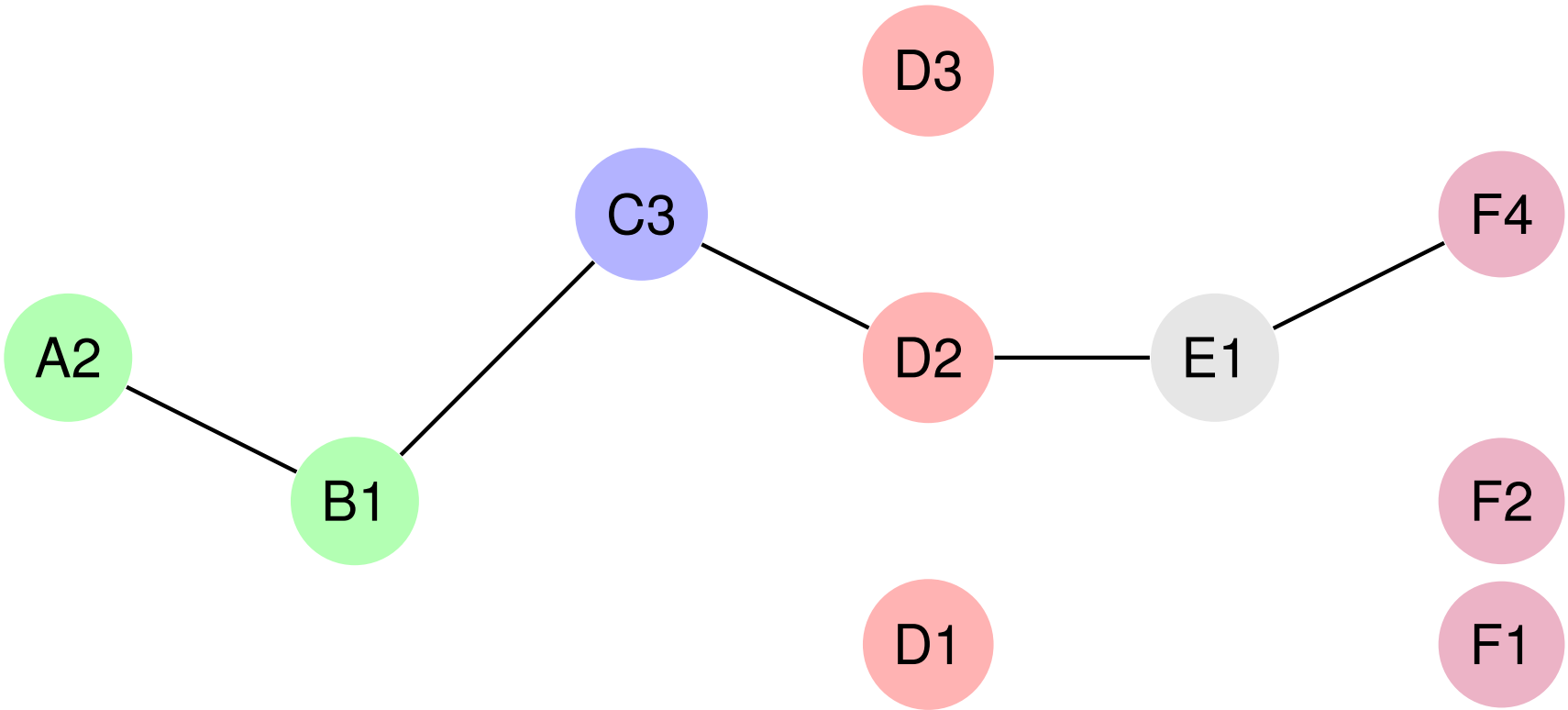
Grammar development



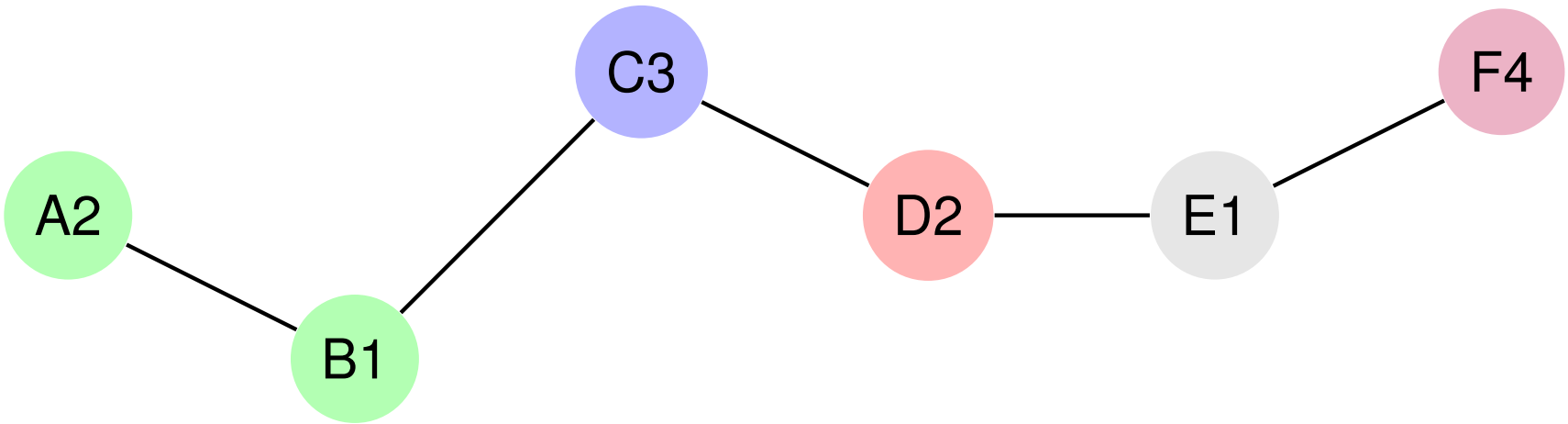
Grammar development



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

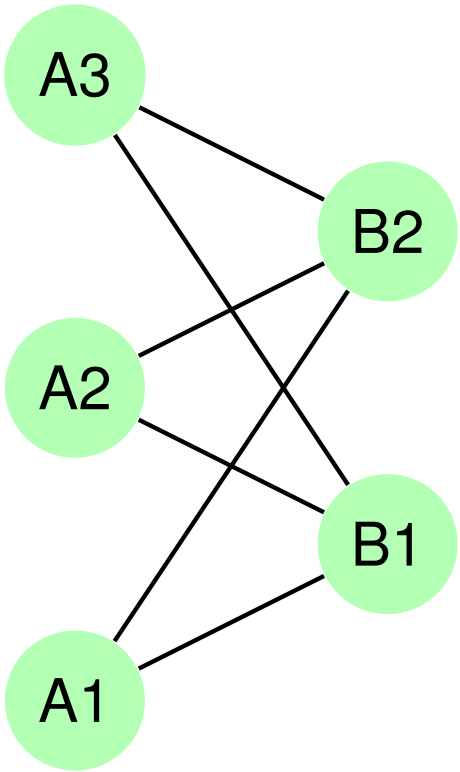
Outline

Metagrammars for systematic exploration

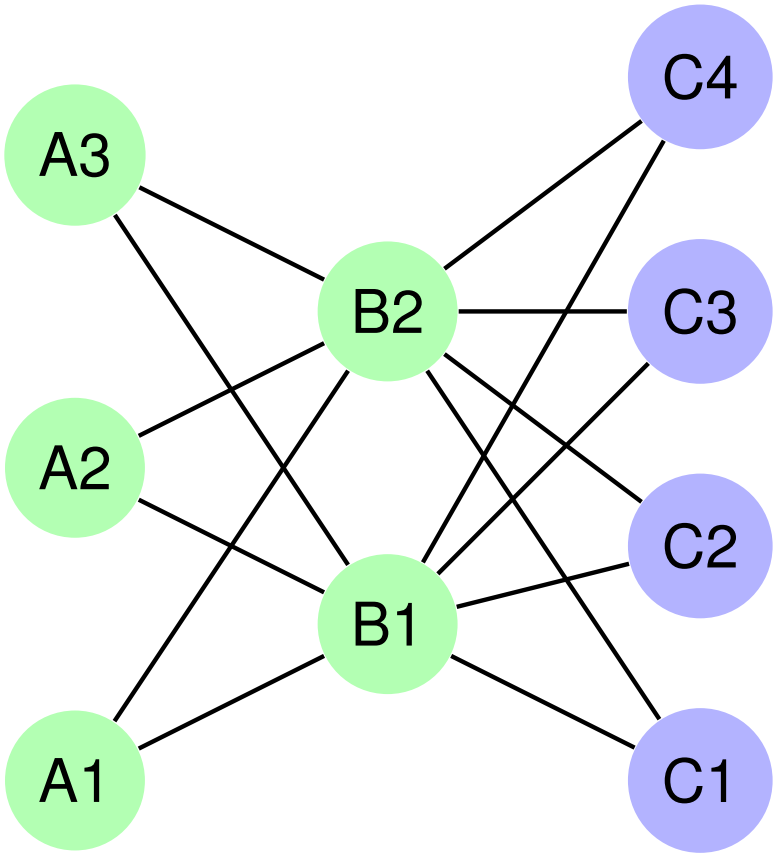
Can we keep track of choices made in the past and preserve alternative solutions?

- 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

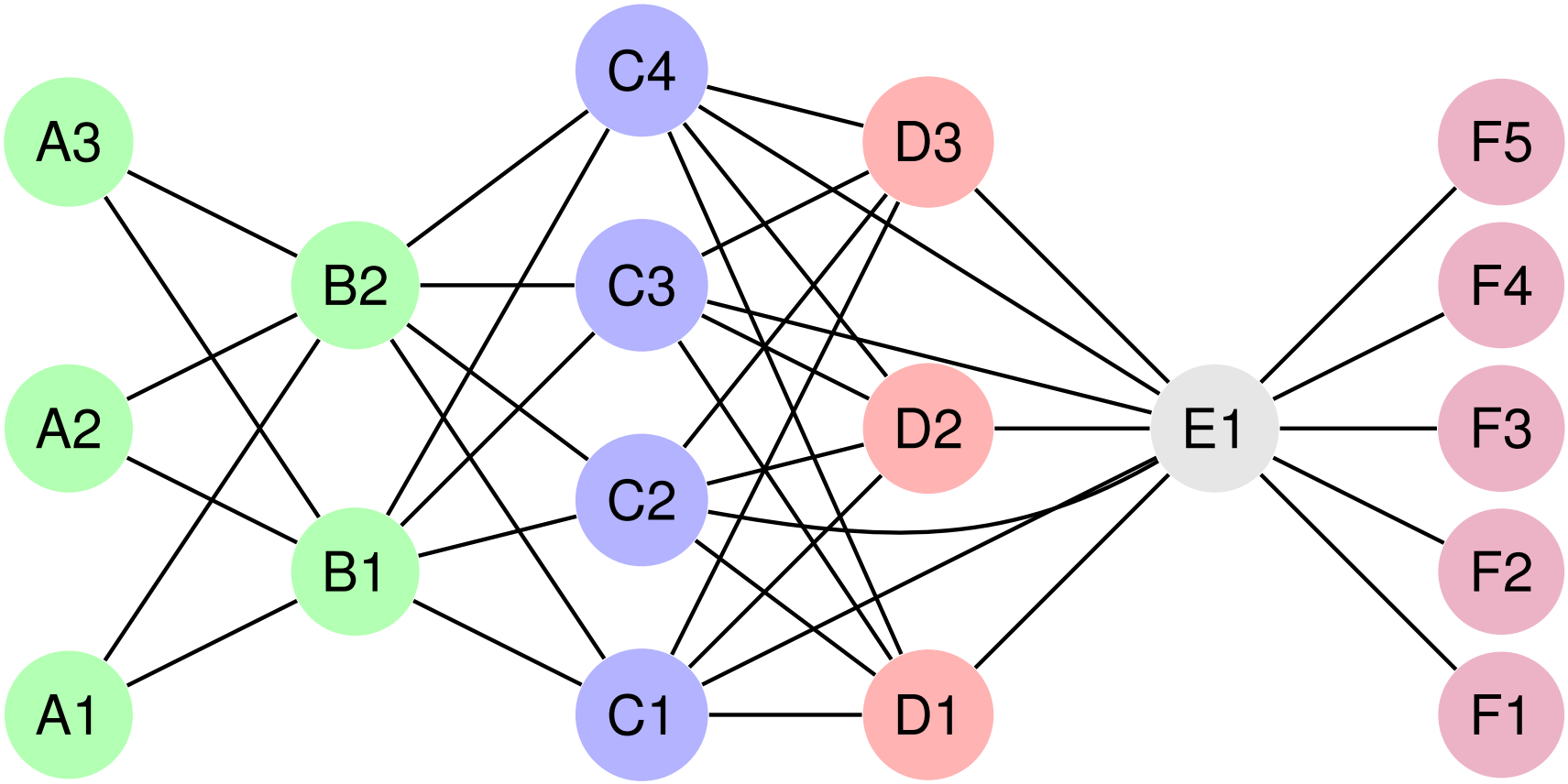
Possibilities



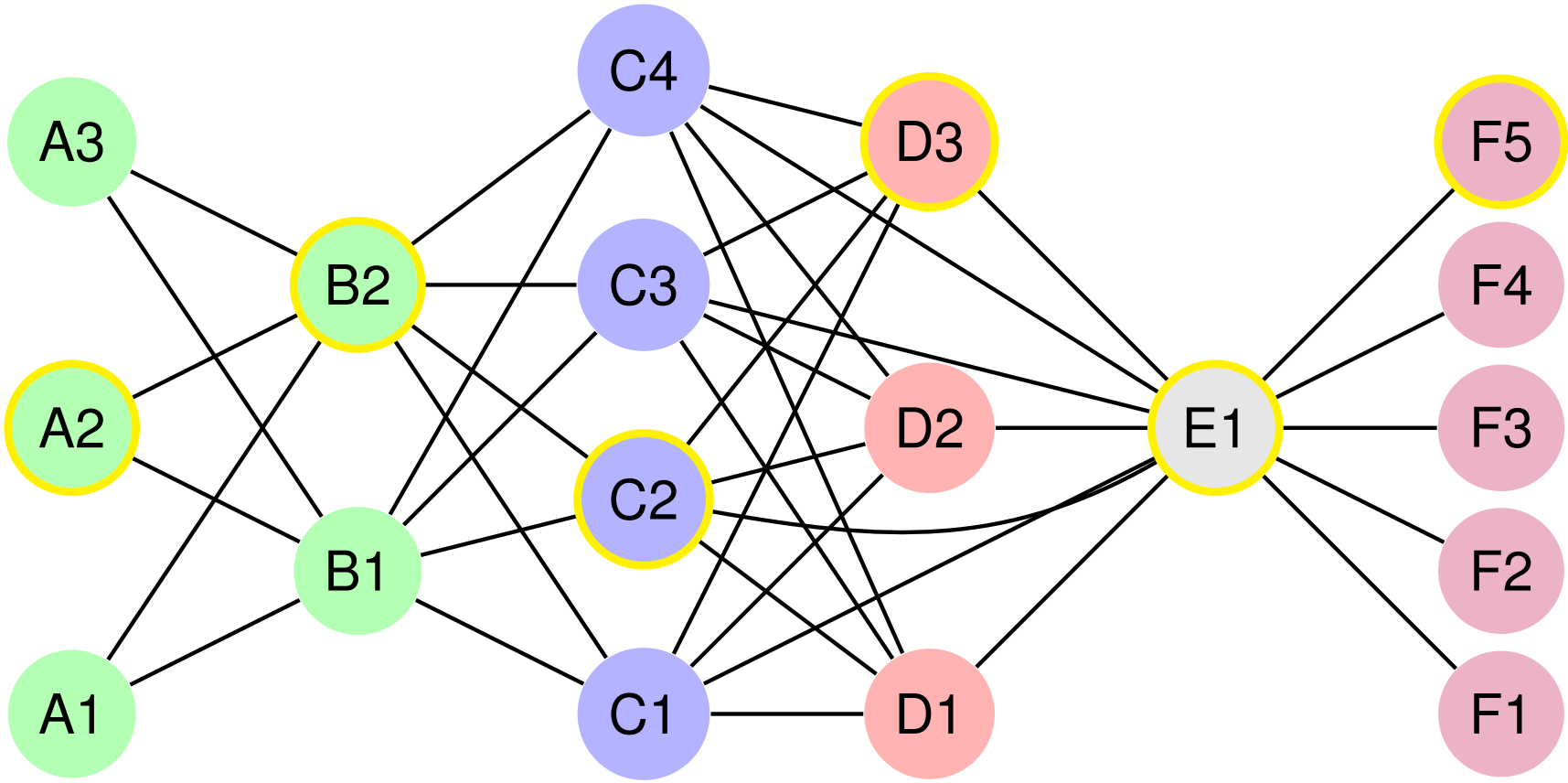
Possibilities



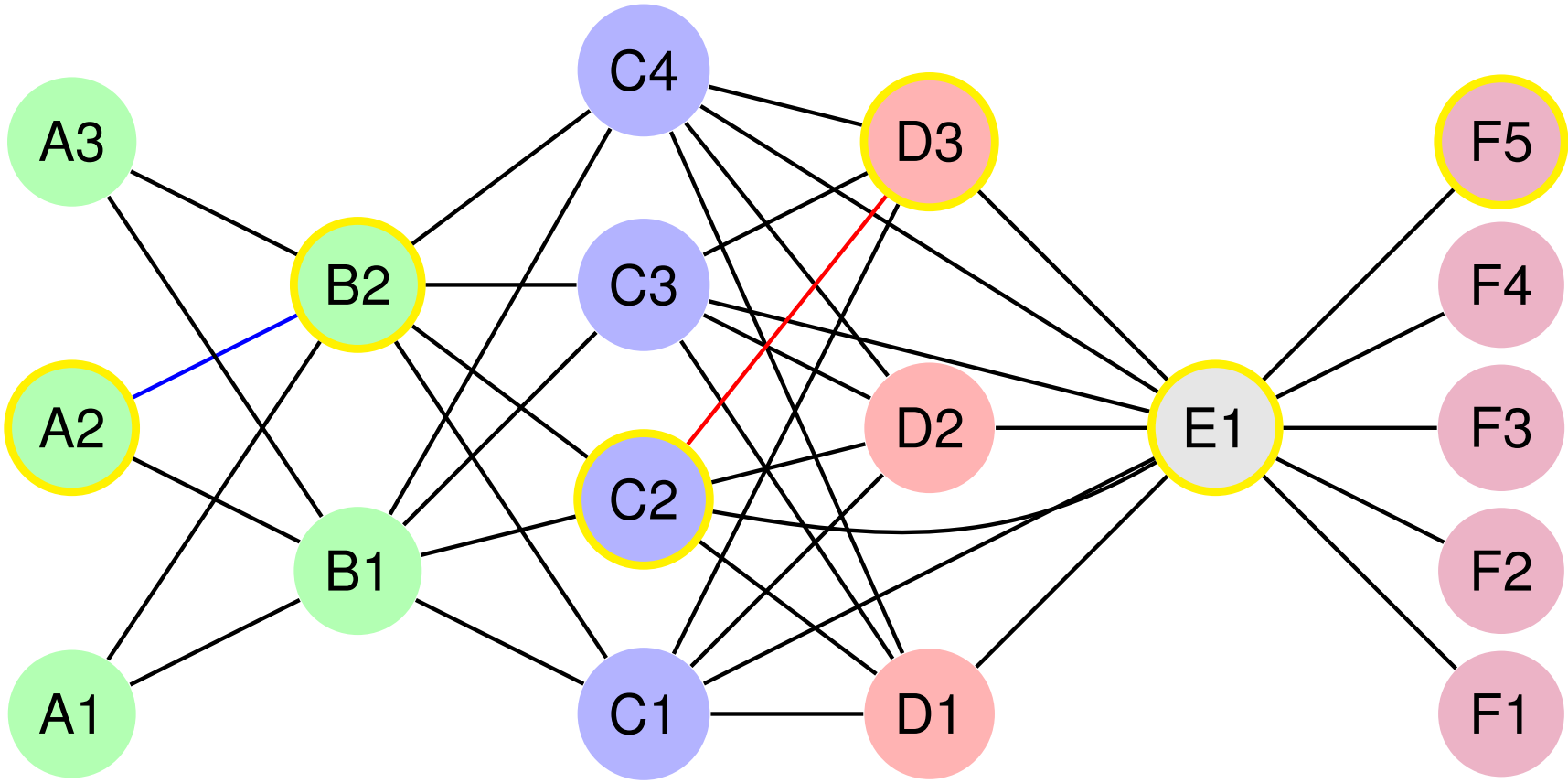
Possibilities



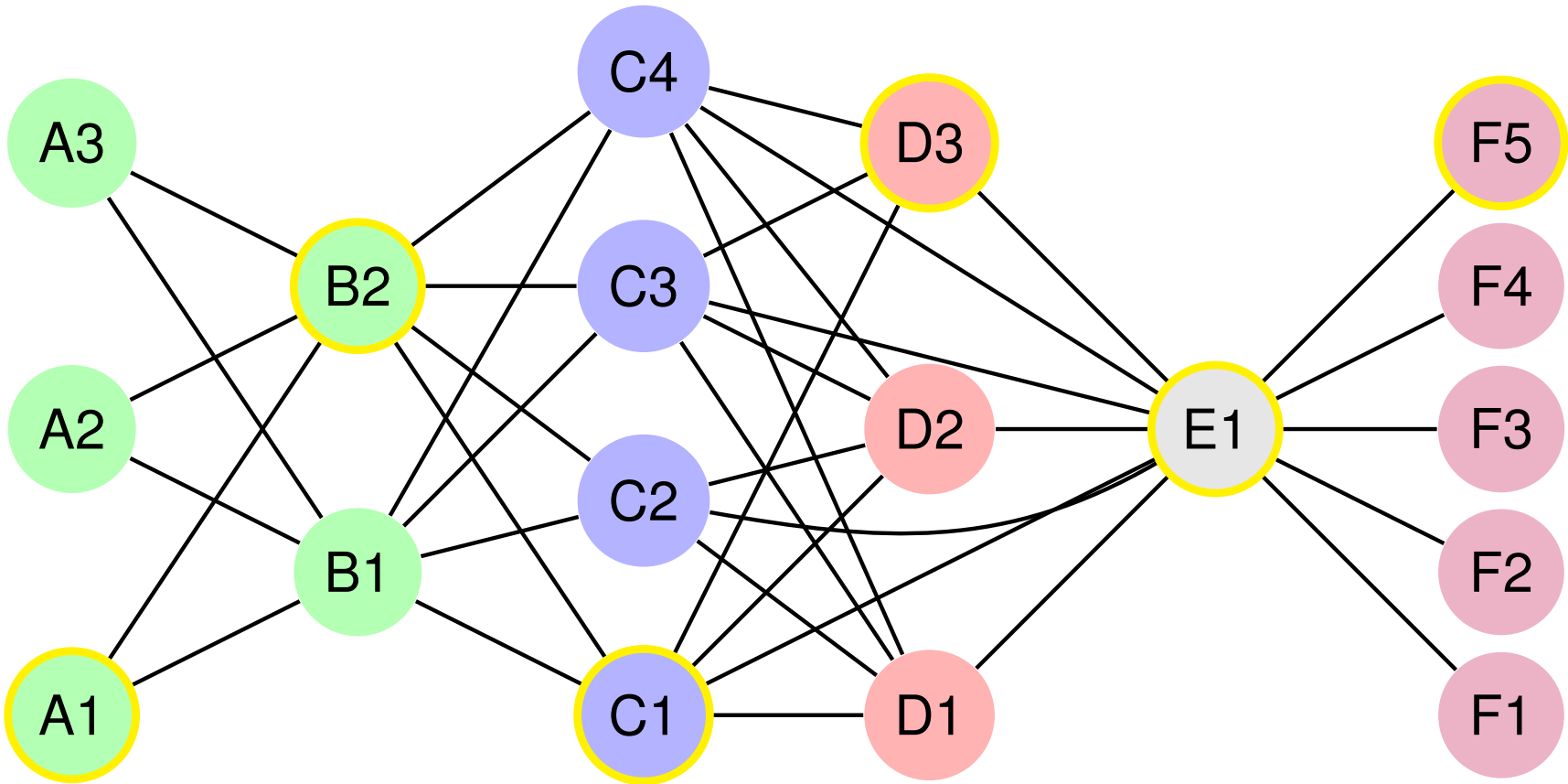
Possibilities



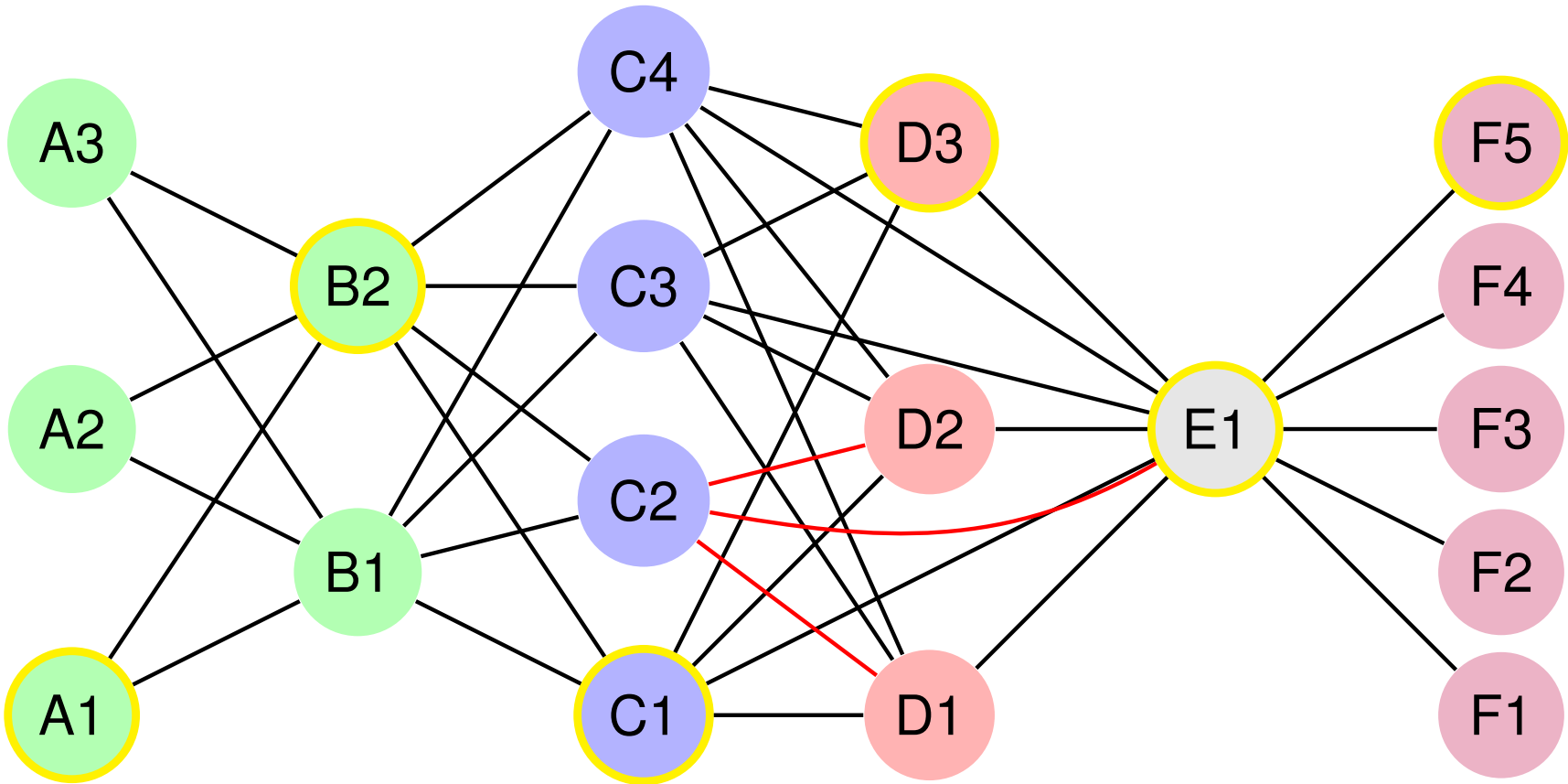
Possibilities



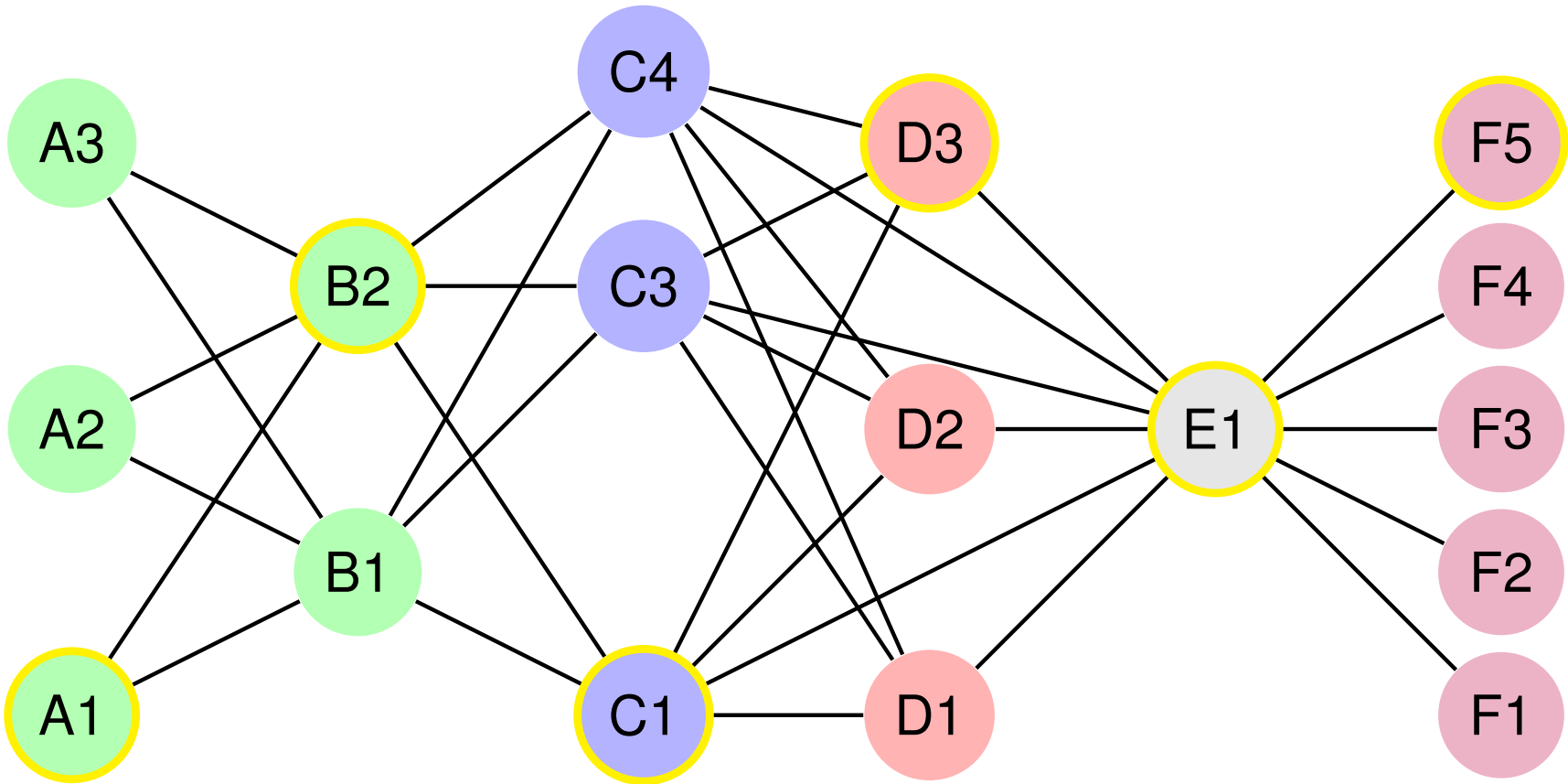
Possibilities



Possibilities



Possibilities



Outline

Thesis Plan

- Create a metagrammar for Germanic languages (except English)
- Develop the metagrammar to cover the same phenomena as Bart Cramer's grammar (Cramer (2011))
- Include alternative analyses for:
  - Auxiliary structures
  - Word order
  - Case marking
- Map lexical types to those used in Cramer's grammar
- Compare different grammatical combinations on coverage and efficiency

Outline

First steps

- Basic Metagrammar set-up for Germanic languages (other than English)
- Comparative analysis of auxiliary structures for Germanic languages (Fokkens (2011))
- Extensions of the grammars for German and Dutch

The LinGO Grammar Matrix

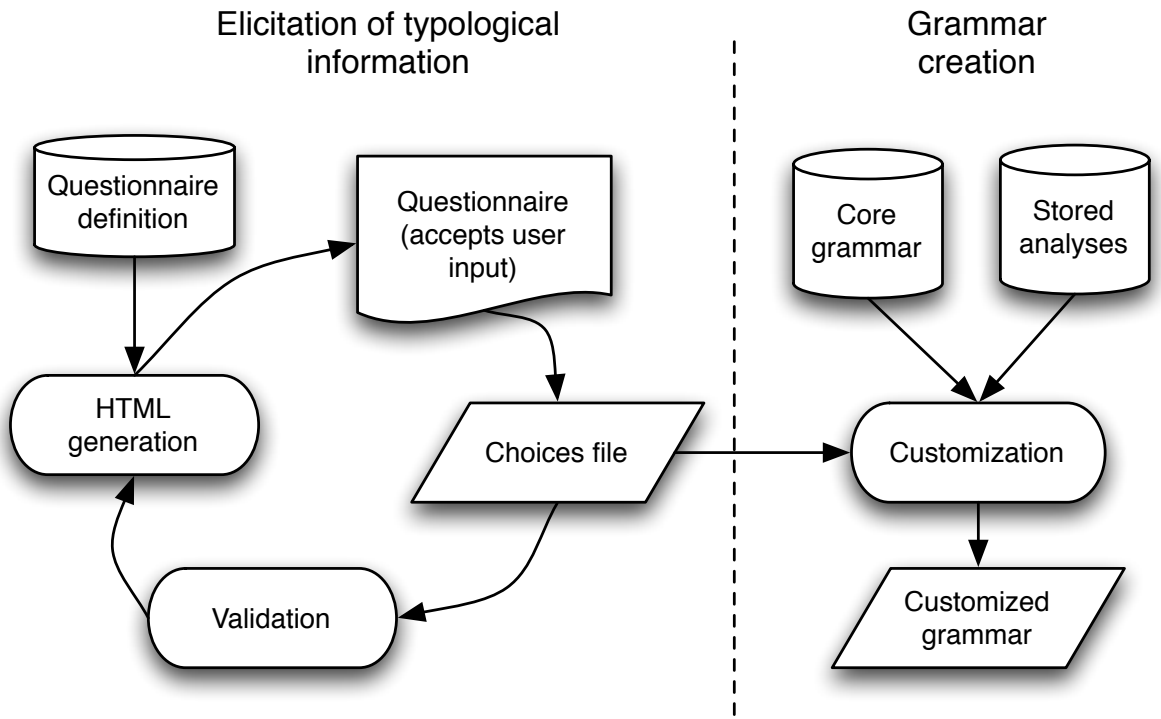


Figure: Schematic system overview

Germanic Metagrammar described in Fokkens (2011)

- An extension of the LinGO Grammar Matrix
- Contains the following additions:
  - Adapted word order analyses to capture Germanic topological fields
  - Partial VP fronting (with or without split clusters)
  - Ditransitives
  - Interaction between morphology and Dutch word order
  - Extension of coordination
- Contains both alternative analyses for auxiliary structures
- Alternative analyses combined with optional split clusters leads to four possible alternative grammars

Additional currently covered phenomena

- Modification: adverbs, adjectives, prepositions
- Negation
- Polar questions
- Raising and Control verbs
- Subordinate clauses (including German auxiliary flip)
- Copula
- Wh-questions

Coverage of Cramer's development set: 40.6% (43% of data reported in Cramer (2011))

To Do:

- Integration of a German grammar in language learning dialogue system (proof of concept)
- Cover the last 57-59.4% of the development set
- Find other phenomena covered by Cramer's grammar (TiGer development set?)
- Add Cramer's analyses to the metagrammar
- Run experiments on efficiency
- Write up

Outline

Metagrammars as methodology

- Using a metagrammar facilitates testing multiple combinations of analyses (and thus encourage the engineer to do so)
- The approach helps to increase systematic empirical exploration of analyses leading to better informed choices in grammar design

A critical note

- The interaction between analyses for different phenomena remains a challenge (even when using a metagrammar)
  $\Rightarrow$ it is likely that (slightly) different versions of an account need to be created to interact properly with alternative analyses for other phenomena
- Occasions where it is worth-while maintaining analyses in parallel need to be well-chosen

$\rightarrow$ to do: add functionality that allows to treat small changes at one place in the customization system

Advantages of using a Metagrammar

- Using a Metagrammar can speed up grammar development
- Modularity is increased in the Metagrammar: potentially easier to add new alternative accounts
- Consistency among grammars

Additional advantages of the approach

- Facilitates creation of alternative grammars depending on application
- Multilingual aspect of the approach:
  - Code sharing among similar languages
  - Comparative cross-linguistic analysis: are there differences in optimal choices among related languages?

Theoretical interest

- Philosophy:
  - “Truth” search in syntactic research
  - Problem solving methods
- Computer Science/Metaprogramming:
  - To my knowledge, not used previously for such a purpose
  - Procedural code used to generate declarative language (generally true for the Grammar Matrix)

Evaluation

- Time measurement of grammar development
  - General indication of time to get a grammar usable on a Treebank
  - Comparison with Cramer's development time
- Influence of basic analyses
  - How much of the original analyses is used?
  - How do the analyses compare to independently developed analyses?
  - Differences between Cramer's analyses with and without matrix.tdl

Additions and revisions

- Extension of v2 analysis (including options to choose between analyses)
- Observations and revisions in matrix.tdl (come to subgroup activity!):
  - Adposition's argument structure
  - Semantics of modifiers (notably adjectives)
  - Sharing of QUE and REL
  - Filler-head structures
- Germanic specific extensions may serve as basic examples for future additions

Future work

- Kaplan and Maxwell's work on automatically improving the grammar (?)
- Can only learn certain aspect of grammar design
- Interesting empirical questions:
  - How much can grammars using different analyses gain from different methods increasing efficiency?
  - Can inefficient grammars catch up with more efficient ones?
  - Is the most efficient grammar without using additional efficiency methods also the most efficient with such methods?

Thanks to Mark Johnson for his question after my talk

Acknowledgments

- Thanks to:
  Emily M. Bender, Bart Cramer, Dan Flickinger, Mike Goodman, Varya Gracheva, Joshua Growgey, Laurie Poulson, Ron Kaplan, Sanghoun Song, Hans Uszkoreit, David Wax, Yi Zhang & anonymous reviewers
- you for your attention ‿

Bibliography I

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

Black, C. A. and Black, H. A. (2009). PAWS: Parser and writer for syntax: Drafting syntactic grammars in the third wave. In *SIL Forum for Language Fieldwork*, volume 2.

Candito, M. (1998). Building parallel LTAG for French and Italian. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 211–217, Montreal, Quebec, Canada. Association for Computational Linguistics.

Cramer, B. (2011). *Improving the feasibility of precision-oriented HPSG parsing*. PhD thesis, Universität des Saarlandes.

Bibliography II

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.

Ranta, A. (2009). The GF resource grammar library. *Linguistic Issues in Language Technology*, 2(2).