Equipping TRALE with a Morphological Analyzer

Preliminary Concepts for a BA Thesis Project

Niels Ott

nott@sfs.uni-tuebingen.de

University of Tübingen, Seminar für Sprachwissenschaft

July 10, 2006

Outline

This talk will be about:

- Why using a morphological analyzer with TRALE?
- Morphological features vs. lexical entries.
- How people do this for Lexical-Functional Grammar.
- Concepts for my project.
- Open issues and remarks.

Student's Motivation

- I just found this TRALE. And there even is a wide-coverage grammar available! 

Student's Motivation

- I just found this TRALE. And there even is a wide-coverage grammar available! 
- `| ?- rec[johnny,hates,jazz,music].`

Student's Motivation

- I just found this TRALE. And there even is a wide-coverage grammar available! 
- `| ?- rec[johnny,hates,jazz,music].`
- `STRING:`
  `0 johnny 1 hates 2 jazz 3 music 4`

  `ERROR: The following words are unknown:`
  `johnny hates jazz music`

  `no`

  → 

Problem: “Wide-Coverage”

- “Wide-coverage” for grammars does not necessarily refer to lexical entries.
- Those grammars may be able to analyze a large amount of grammatical phenomena. . .
- . . . but often times the user must specify all the words manually.
- This obviously inhibits parsing of *arbitrary* texts.

Proposed Solution

Replace manually defined lexical entries in the grammar by the output of a morphological analyzer.

$\longrightarrow$ Wide-coverage for lexical entries.

Scientific Motivation

Things one could do with lexical wide-coverage:

- Annotate treebanks semi-automatically using TRALE.
- Concept: A parser suggests multiple analysis, humans choose the correct one.
- Example: This has been done for the Tiger treebank and LFG by Zinsmeister et al. (2001).

Scientific Motivation

Things one could do with lexical wide-coverage:

- Test wide-coverage grammars:
- Using “real world” language instead of constructed examples.

Scientific Motivation

Things one could do with lexical wide-coverage:

- Use HPSG as underlying structure for language analysis in other NLP applications.
- E.g. NP recognition, anaphora resolution, information retrieval.
- Often, simple phrase-structure grammars are used.
- HPSG can model the language more precisely.

Lexical Entries

Lexical entries in TRALE can contain several types of information:

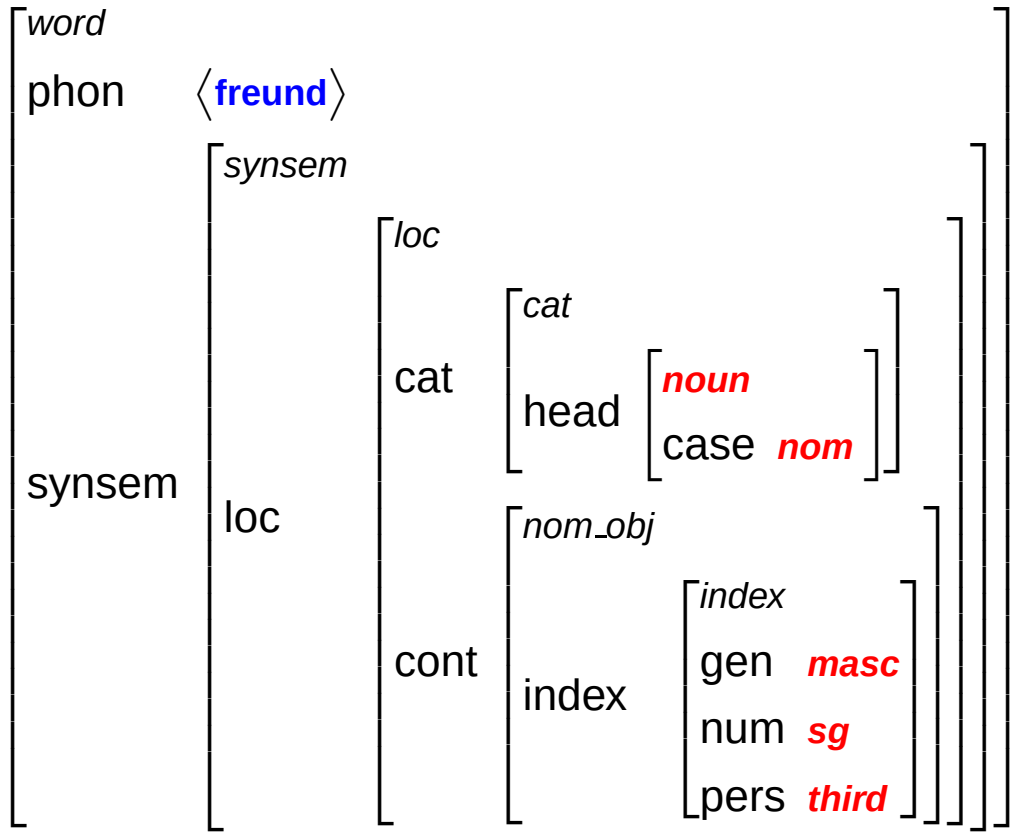
- Part-of-speech
- Case
- Number
- Gender
- Subcategorization information
- Semantic relations (e.g. *like_rel, give_rel*)
- . . . (depending on the linguistic theory)

Morphological Features

Morphological features represent what can be seen from the “outer shape” of a word:

- Case
- Number
- Gender
- Part-of-speech
- (And/or other features, depending on the language.)
- (And there are a lot of ambiguities.)

Morph. Features ↔ Lexical Entry



(Partial lexical entry of *freund* (friend), in the style of the Core Fragment of Richter 2005.)

Morph. Features $\leftrightarrow$ Lexical Entry

Morphological features can be mapped to TRALE feature structures on the word level.

(Plus *ambiguity*: In most cases, one word will lead to more than one feature structure as there is more than one morphological analysis.)

Drawbacks

One cannot obtain all desired TRALE features from morphological features:

- Relations like *like_rel* are not available from morphology.
- Subcategorization information is not present as well.
- . . . (Other information people like to have in their linguistic theory.)

Mapping

- How to map morphological features to TRALE feature structures?
- Let’s take a look what other people do.

LFG

A few properties of Lexical-Functional Grammar (LFG):

- LFG makes use of feature structure (called *f-structure* for for functional structure).
- It also makes use of phrase structure rules (called *c-structure* for constituent structure).
- The c-structure rules are equipped with rules operating on the features. These rules are called *f-annotation*.

LFG

A few properties of Lexical-Functional Grammar (LFG):

- Subcategorization information and features needed for agreement are defined in the lexicon.
- The common representation for LFG f-structures is the attribute-value matrix (AVM).
- To keep things simple, we will only consider the lexicon in LFG.

XLE

- The Xerox Linguistics Environment (XLE) is a popular platform that allows the implementation of LFG grammars.
- Unlike TRALE, XLE has a built-in morphological analyzer.

(Further information on XLE: Crouch et al., 2006;
Kaplan and Newman, 1997.)

LFG: Lexical Entries

AVM notation:

$$\begin{bmatrix} \text{PRED} & \text{'<freund>'} \\ \text{NUM} & \text{sg} \\ \text{CASE} & \text{nom} \\ \text{GEN} & \text{masc} \\ \text{PERS} & \text{3rd} \end{bmatrix}$$

XLE notation (no analyzer involved):

```
freund  N  * (^PRED) = '%stem'
             (^NUM)  = sg
             (^CASE) = nom
             (^GEN)  = masc
             (^PERS) = 3rd.
```

XLE and the Analyzer

First, the grammar writer specifies what should happen for a particular part-of-speech:

```
N --> N_BASE
      N_SFX_BASE
      N_NUM_SFX_BASE
      N_PERS_SFX_BASE
      N_CASE_SFX_BASE
      N_GEND_SFX_BASE.
```

Note that _BASE is just an obligatory suffix.

The essential information is: “If something is an `N`, then it needs an `N_SFX` and a `NUM_SFX` and . . . ”

XLE and the Analyzer

Secondly there is an entry in the lexicon for unkown words:

```
-Lunkown N XLE (^PRED) = '%stem'.
```

"For unkown words: If we found out it is of POS `N`, make its `PRED` the stem of the word."

(It is a tradition of LFG to represent the abstract concept of the word in the predicate feature, not the actual phonological form. So they take the stems only.)

XLE and the Analyzer

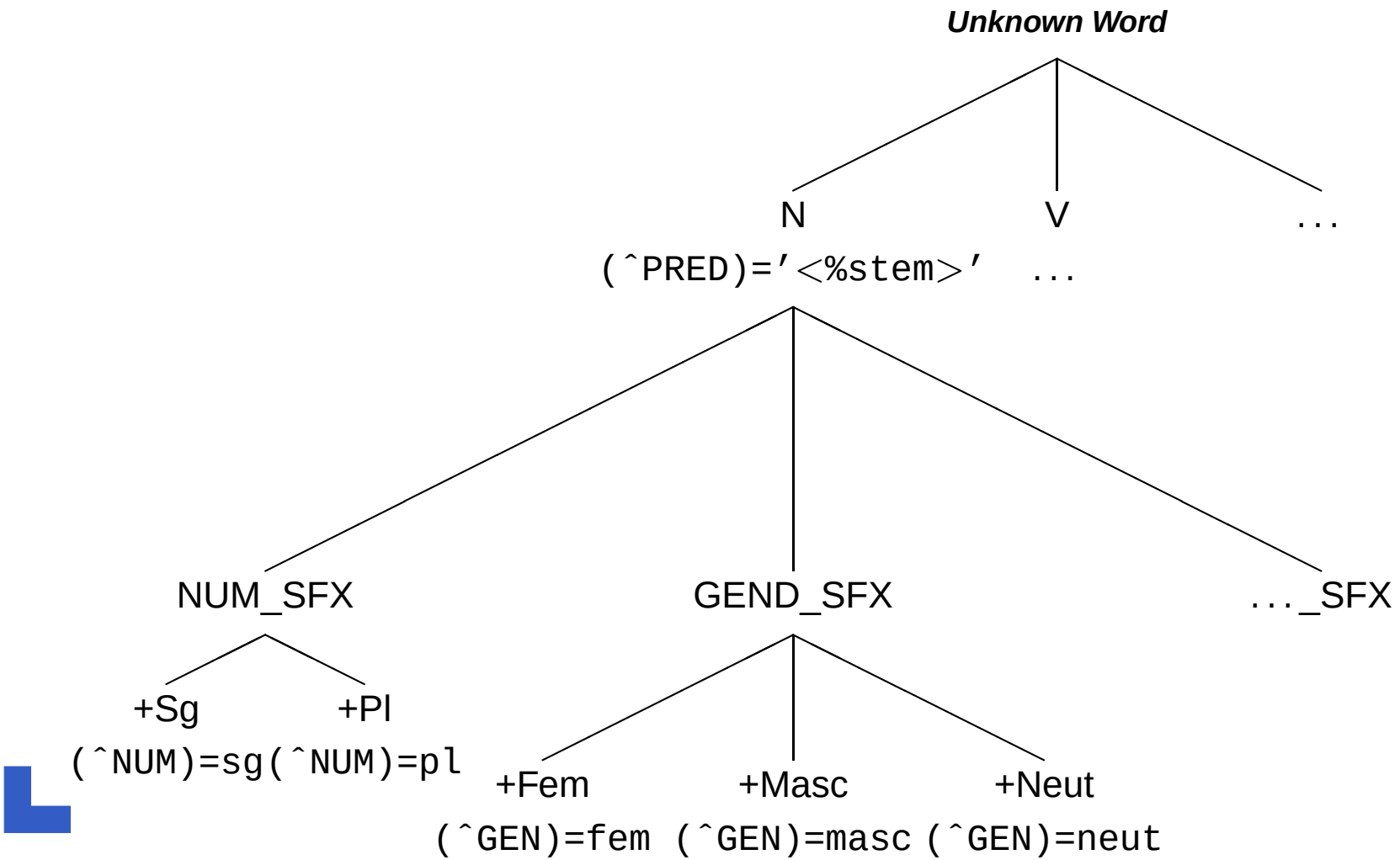
Thirdly, there are rules mapping the morphological features to LFG features:

```
+Noun  N_SFX    XLE (^PERS 3rd).     "3rd person by default"
+Sg    NUM_SFX  XLE (^NUM sg).
+Pl    NUM_SFX  XLE (^NUM pl).
+Fem   GEND_SFX XLE (^GEN fem).
+Masc  GEND_SFX XLE (^GEN masc).
+Neut  GEND_SFX XLE (^GEN neut).
"and so on"
```

Columns:

1. Output of the morphological analyzer.
2. Classification of morphological feature.
4. Features for LFG f-structures.

XLE and the Analyzer: Overview



Simplification

XLE includes a lot of complex functionality. A very simplified version can be formulated as follows:

```
IF   "analyzer said +Noun"
THEN "produce LFG feature: (^PRED) = '<%stem>'"

IF   "analyzer said +Noun" AND "analyzer said +Pl"
THEN "produce LFG feature: (^NUM) = pl"

IF   "analyzer said +Noun"" AND "analyzer said +Sg"
THEN "produce LFG feature: (^NUM) = sg"

... (and so on)
```

Intermediate Results

- The project: Equip TRALE with a morphological analyzer.
- The analyzer will produce morphological features.
- These features can be mapped to LFG features in XLE.
- This should work for HPSG feature structures in TRALE as well.
- The formalism can be simpler than the one in XLE.

The Analyzer: MMorph

- MMorph is used as a morphological analyzer.
- This tool can be licensed from the Deutsches Forschungszentrum für künstliche Intelligenz, Saarbrücken (DFKI).
- It has a lot of functions but it can simply serve as an analyzer.

(For further information on MMorph, see Petitpierre and Russell, 1995; Lehmann, 2003.)

Analyzer Output...

The output of MMorph looks as follows:

```
> Freund
"Freund" = "Freund" Noun[ gender=masc number=singular case=acc
                          spelling=unchanged ]
"Freund" = "Freund" Noun[ gender=masc number=singular case=dat
                          spelling=unchanged ]
"Freund" = "Freund" Noun[ gender=masc number=singular case=nom
                          spelling=unchanged ]
```

→ There are three analyses of *Freund* (friend).
→ Given information: Lemma, PoS + morph. features.

...what it should become in TRALE

$$
\left[\begin{array}{ll}
\textit{word} & \\
\text{phon} & \langle \textbf{freund} \rangle \\
\text{synsem} & \left[\begin{array}{ll}
\textit{synsem} & \\
\text{loc} & \left[\begin{array}{ll}
\textit{loc} & \\
\text{cat} & \left[\begin{array}{ll}
\textit{cat} & \\
\text{head} & \left[\begin{array}{ll} \textbf{\textit{noun}} & \\ \text{case} & \textbf{\textit{nom}} \vee \textbf{\textit{acc}} \vee \textbf{\textit{dat}} \end{array}\right] \\
\text{cont} & \left[\begin{array}{ll}
\textit{nom\_obj} & \\
\text{index} & \left[\begin{array}{ll} \textit{index} & \\ \text{gen} & \textbf{\textit{masc}} \\ \text{num} & \textbf{\textit{sg}} \\ \text{pers} & \textbf{\textit{third}} \end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

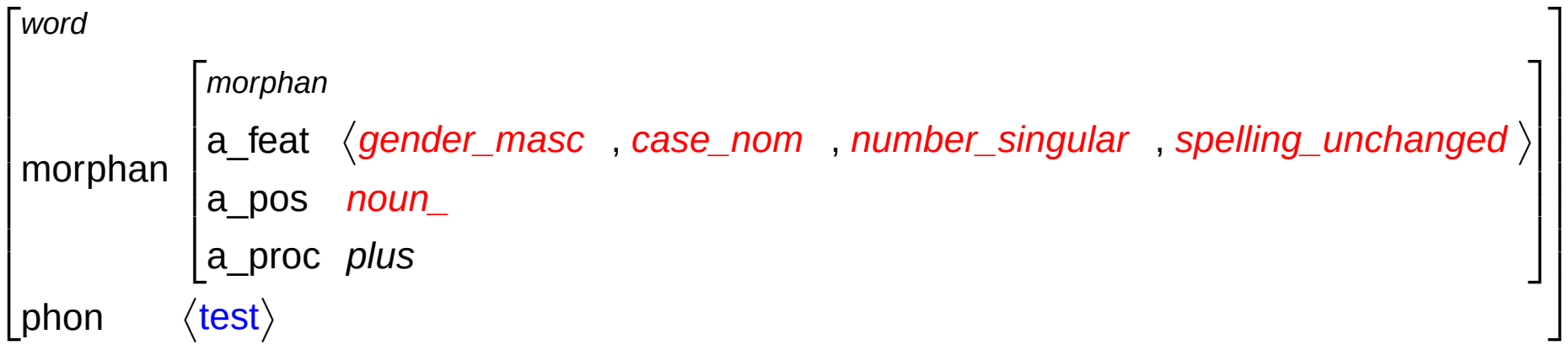
$\longrightarrow$ Several possibilities for case, but no information on person from the analyzer.

Storing MMorph output into FS

Concept: Store the output of the analyzer in a feature structure!

- Once the output is in the feature structure, one can operate on it using TRALE constraints.
- Motivation: Grammar writers do not want to consider how the analyzer actually is “plugged in”, they want to use it the way they are used to: With TRALE syntax.

Storing MMorph output into FS



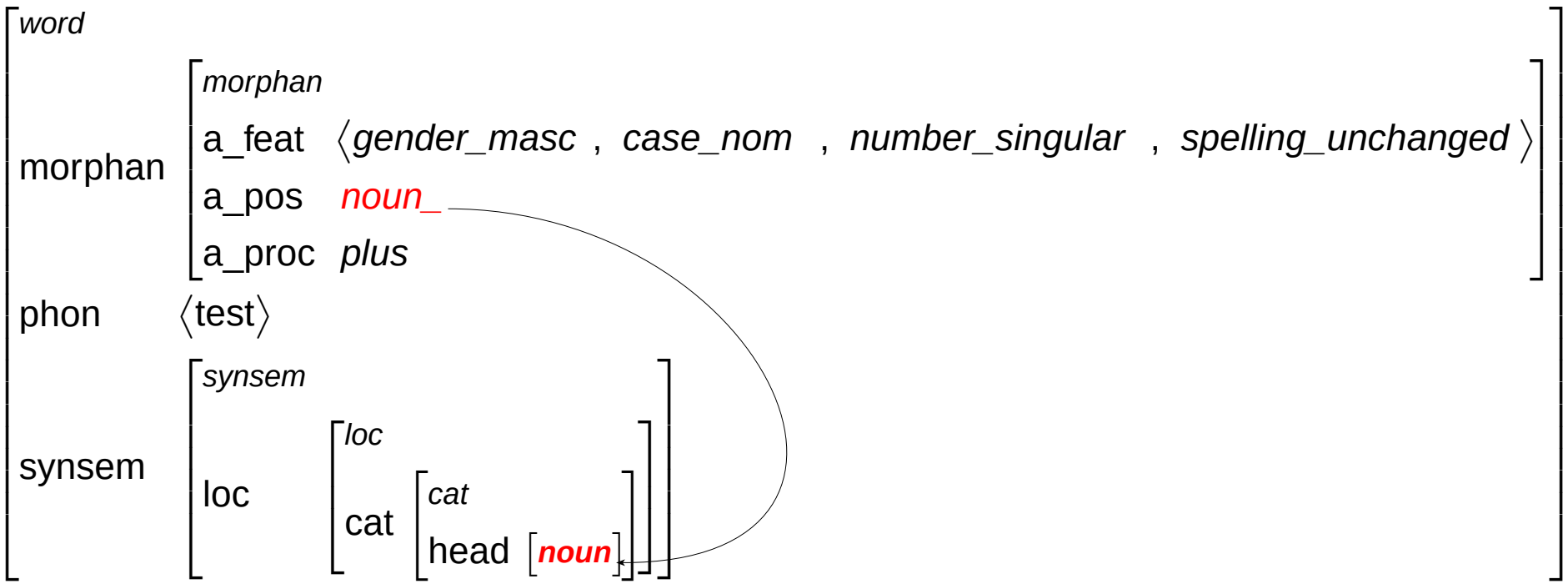
Issue: All analyzer outputs need to be introduced in the signature. One can use the a_/1 predicates. This is something I need to explore further.

Example Constraints

Transforming from the technically motivated *morphan* to HPSG-motivated structures:

```
(word, morphan:a_pos:noun_)
*>
        (word, synsem:loc:cat:head:noun)
.
```

Example Constraints



Remaining Problem

How to create a TRALE feature structure like in *morphan* from the output of an external programm (MMorph)?

Background

- TRALE is an extension to ALE.
- These two systems are implemented in Prolog.
- One needs to take a closer look at Prolog to “plug in” a morphological analyzer.
- Furthermore, it necesarry to know about a few internal details of TRALE.

Prolog (Oversimplified)

Prolog deals with facts about objects:

```
likes(peter,mary).
likes(mary,peter).
```

There are also clauses, which could be described as “abstract facts”:

```
can_fall_in_love(X,Y) :-
        likes(X,Y),
        likes(Y,X).
```

(Only people who like *each other* can fall in love.)

Prolog (Oversimplified)

After having defined the facts, one can ask Prolog questions about them on the prompt, e.g. in a toy world:

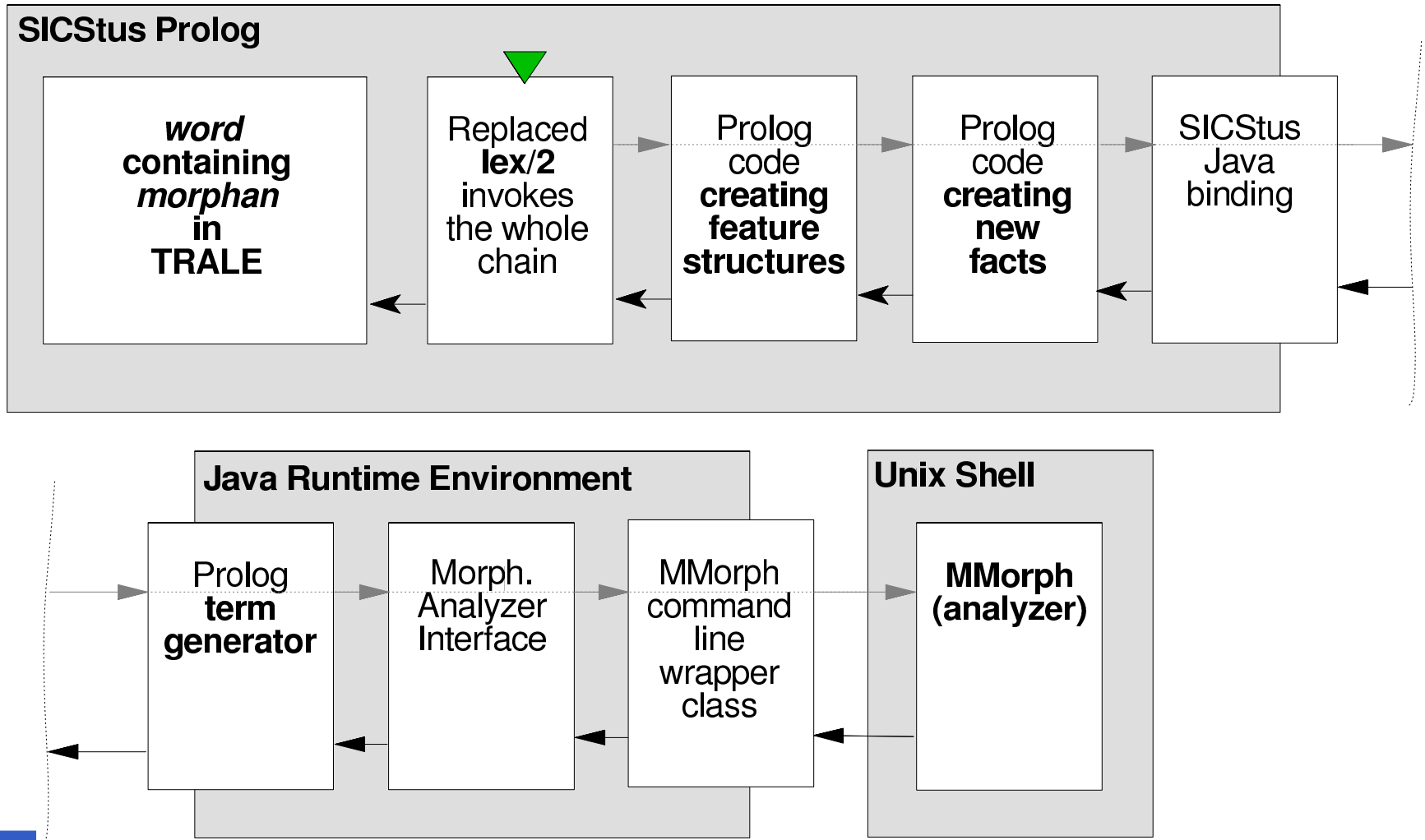
```
?- woman(anna).
Yes

?- man(peter).
Yes

?- can_marry(anna,peter).
No

| are_related(anna,peter).
Yes
```

(Accessible introduction on Prolog: Clocksin and Mellish, 2003.)

Prolog, TRALE

- Using TRALE is a very abstract way of specifying facts and asking questions.
- `| ?- lex freund.` can be seen as the question: “Does my HPSG grammar license a structure for the word *freund*?”
- . . . with the side effect to get a feature structure displayed in Grisu in case the answer is `Yes`.

From MMorph to TRALE



Existing Solutions

Parts of the framework presented have been implemented by Przepiórkowski (P.C.):

- He used the C-binding of SICStus Prolog to attach an analyzer for Polish.
- Furthermore he constructed a rather complex way of mapping the features.
- Right at this point I cannot present this mapping as I have a hard time understanding it.

Existing Solutions

Differing concepts by Przepiórkowski (P.C.):

- As the Polish analyzer comes with a C-library, there is no need to interface with the Unix shell.
- The mapping between morphological features and TRALE feature structures is done entirely in Prolog.

Existing Solutions

Concepts taken over from Przepiórkowski (P.C.):

- External code (C, resp. Java) creates Prolog facts with morphological information.
- There is a mapping between Prolog facts and TRALE feature structures.
- (My approach: Create resp. fill the *morphan* sort and continue in TRALE from there.)
- Overriding the `lex/2` predicate.

Overriding `lex/2`

- `lex/1` is what users call by asking e.g.
  `| ?- lex freund.`
- `lex/2` is an internal predicate that investiages the lexical entries from the compiled grammar.
- Important difference: `lex/1` creates Grisu output, `lex/2` creates a data structure for further use.
- Overriding `lex/2` is a way to replace the internal handling of lexical entries by one's own version.

Example (Simplified)

```
"Freundes" = "Freund" Noun[ gender=masc number=singular case=gen
             spelling=unchanged ]
```

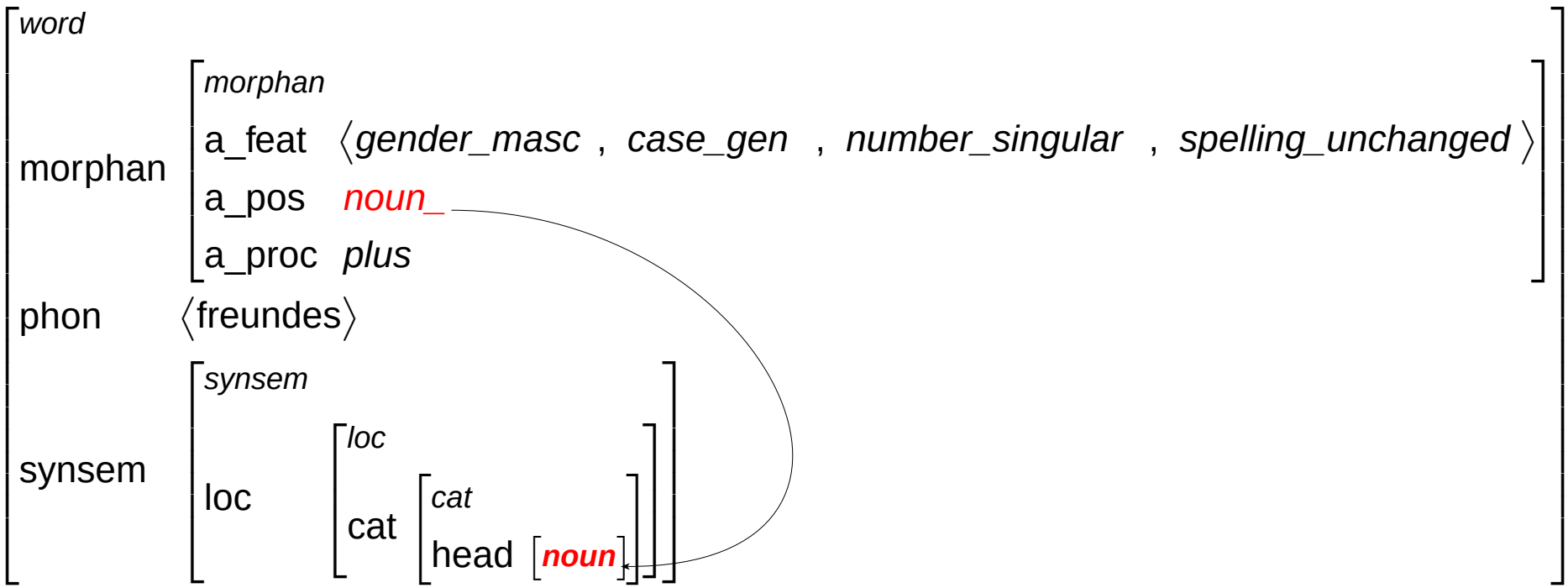
↓

```
m_analysis(freundes, noun_, [gender_masc, number_singular
                             case_gen, spelling_unchanged] ).
```

↓

```
(phon:[a_ freundes],
 morphan:(
     a_proc:plus,
     a_pos:noun_,
     a_feat:[gender_masc,number_singular,case_gen,spelling_unchanged])
).
```

Example (Again Simplified)



(Yes, this is similar to *test* from an earlier slide.)

Summing Up: Issues

- Not to be solved within my thesis project: Where to get subcategorization and semantic relation information (e.g. *like_rel*) from?
- Will the manually specified lexicon still be available after `lex/2` has been replaced?
- Performance: This point will be ignored.
- Ambiguous analyzer outputs will be transported through the processing chain, eventually producing multiple (resp. many!) TRALE outputs.

Summing Up: Gains

- The project will be a step towards parsing of arbitrary text with TRALE.
- There are many applications for this, e.g. semi-automatic treebank annotation, various kinds of NLP systems, etc.

Summing Up: BA Thesis

- Challenge: There is a lot to write about and a lot more not to write about.

References

Bresnan, Joan (2001). *Lexical-Functional Syntax*. Oxford: Blackwell Publising.

Butt, Miriam, King, Tracy Halloway, Niño, Maria-Eugenia, and Segond, Frédérique (1999). *A Grammar Writer's Cookbook*. Stanford: CSLI Publications.

Clocksin, William F. and Mellish, Christopher S. (2003). *Programming in Prolog*. Berlin: Springer-Verlag, 5th edn.

Crouch, Dick, Dalrymple, Mary, Kaplan, Ron, King, Tracy, Maxwell, John, and Newman, Paula (2006). *XLE Documentation*. Palo Alto Research Center. User's manual apparently available on line only.
URL `http://www2.parc.com/istl/groups/nltt/xle/doc/`

Kaplan, Ronald M. and Newman, Paula S. (1997). *Lexical Resource Reconciliation in the Xerox Linguistic Enironment*. In *Proceedings of the Workshop Computational Environments for Grammar Development and Linguistic Engineering*, pp. 54–61. Madrid.
URL `http://citeseer.ist.psu.edu/ronald97lexical.html`

Lehmann, Sabine (2003). *Multext German Morphology*. ISSCO, University of Geneva. Documentation of the German morphology coming with MMorph version 3.0.0, as distributed by the Deutsches Forschungszentrum für künstliche Intelligenz (DFKI), Saarbrücken.
URL `http://www.dfki.de/LT-DEMO/datensatz.php3?system_id=282`

Penn, Gerald and Haji-Abdolhosseini, Mohammad (2003). *The Attribute Logic Engine: User's Guide – with TRALE extensions*. Revision of the manual: Version 4.0 beta.
URL `http://www.ale.cs.toronto.edu/docs/`

Petitpierre, Dominique and Russell, Graham (1995). *MMORPH – The Multext Morphology Program*. ISSCO, University of Geneva. Manual for MMorph version 2.3 as distributed by ISSCO.
URL http://www.issco.unige.ch/projects/MULTEXT.html

Pollard, Carl and Sag, Ivan (1994). *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Przepiórkowski, Adam (P.C.). *Re: TRALE + Morphological Analyzer*. Institute of Computer Science, Polish Academy of Sciences, Warsaw. Personal Communication (E-Mail) during summer 2006.

Richter, Frank (2005). *A Web-based Course in Grammar Formalisms and Parsing*. Published on the WWW only.
URL http://milca.sfs.uni-tuebingen.de/A4/Course/PDF/gramandpars.pdf

Zinsmeister, Heike, Kuhn, Jonas, Schrader, Bettina, and Dipper, Stefanie (2001). *TIGER TRANSFER – From LFG Structures to the TIGER Treebank*. Tech. rep., Institut f. Maschinelle Sprachverarbeitung, Stuttgart.
URL http://www.ims.uni-stuttgart.de/projekte/TIGER/paper/transfer/tiger-tra

Thank You!

This is the end. Questions?