The LinGO Grammar Matrix

Rapid Grammar Development for Hypothesis Testing

Emily M. Bender and Antske S. Fokkens

University of Washington & Saarland University

Acknowledgements

- This material is based upon work supported by the National Science Foundation under Grant No. 0644097. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation
- Deutsche Forschungsgemeinschaft for funding a 2 month stay at the University of Washington
- This tutorial presents joint work with:

  Safiyyah Saleem, Scott Drellishak, Michael Wayne Goodman, Daniel P. Mills and Laurie Poulson

Outline

1 Introduction
- Multilingual Grammar Engineering
- Related Work
- DELPH-IN

2 The Matrix Customization System
- System Overview
- Notes on HPSG, Analyses and practicalities

3 Extended example: Maltese
- Word order and Auxiliaries
- Case, Negation, Argument Optionality
- Analyses, Part 3: The Lexicon

4 Extending a grammar
- Using the LKB and [incr tsdb()]
- Editing tdl
- Conclusion

Outline

1 Introduction
- Multilingual Grammar Engineering
- Related Work
- DELPH-IN

2 The Matrix Customization System
- System Overview
- Notes on HPSG, Analyses and practicalities

3 Extended example: Maltese
- Word order and Auxiliaries
- Case, Negation, Argument Optionality
- Analyses, Part 3: The Lexicon

4 Extending a grammar
- Using the LKB and [incr tsdb()]
- Editing tdl
- Conclusion

The Matrix Customization System

The LinGO Matrix Customization System is a tool that provides start-up implementations for linguistically motivated precision grammars

- From an engineering point of view it supports code-sharing leading to
  - a significant reduction in grammar engineering effort
  - more consistency across grammars
- From a scientific point of view
  - it supports syntactic research for hypothesis testing
  - it encourages research that combines typology with formal syntactic analysis

Tutorial Goals

- Introduce the LinGO Grammar Matrix system
- Illustrate how to derive the most benefit from the system
- Demonstrate how to work with and extend a starter grammar
- Exemplify the methodology of grammar engineering for linguistic hypothesis testing

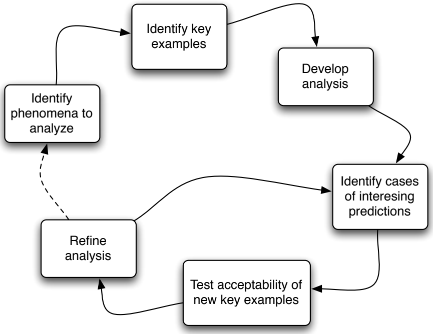
Test suites: Best Practices

- Use IGT format and Leipzig Glossing Rules (Bickel et al., 2008)
- Include both test *suites* and test *corpora*
  - Test suites: Simple, constructed examples illustrating specific phenomena
  - Test corpora: Naturally occurring text
- Expect to iteratively improve and extend test suites alongside implemented grammars

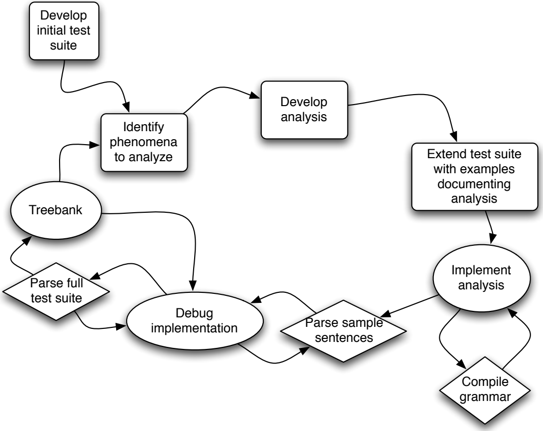
Why Grammar Engineering?

- Natural language grammars are complex.
- Our models of natural language grammars are therefore also complex.
- Grammar engineering allows us to have the computer do the work of checking the models for consistency.
- ... and to test against a much broader range of examples.

Pen and Paper Syntax Workflow


Identify key examples
Develop analysis
Identify cases of interesing predictions
Test acceptability of new key examples
Refine analysis
Identify phenomena to analyze

Grammar Engineering Workflow



Multilingual Grammar Engineering

Main Ideas:

- Reduce the efforts of creating new grammars by using knowledge from those already created
- Create consistency between grammars of different languages
  - Compatibility with downstream components
- Research on crosslinguistic similarity

Related Work

Multilingual Grammar Engineering:

- ParGram (LFG) (Butt et al., 2002; King et al., 2005)
- CoreGram (HPSG) (Müller, 2009)
- GF (Ranta, 2007)
- MetaGrammar project (LTAG) (de la Clergerie, 2005)
- OpenCCG (Baldridge et al., 2007)
- KPML (Bateman et al., 2005)
- MedSLT (Bouillon et al., 2006)
- PAWS (PC-PATR) (Black, 2004; Black and Black, 2009)

Related Work

Automatic Elicitation:

- PAWS (PC-PATR) (Black, 2004; Black and Black, 2009)
- Avenue (Probst et al., 2001; Monson et al., 2008)
- Expedition (Sheremetyeva and Nirenburg, 2000; McShane and Nirenburg, 2003)

Grammar Matrix Context: DELPH-IN

- DELPH-IN (www.delph-in.net) is a collaboration of researchers working on deep linguistic processing.
- The DELPH-IN member sites contribute open-source software and linguistic resources.
- The reference formalism used in DELPH-IN is based on HPSG (Pollard and Sag, 1994) and uses MRS (Copestake et al., 2005) for parse output and basis for generation.
- (Most) grammars are written in tdl (type description language) — interpreted by LKB and PET
- [incr tsdb()] (Oepen, 2001) for regression testing and treebanking

Grammar Matrix Context: DELPH-IN

Large and medium scale grammars:

- ERG (English) (Flickinger, 2000)
- Jacy (Japanese) (Siegel and Bender, 2002)
- GG (German) (Müller and Kasper, 2000)
- NorSource (Norwegian) (Hellan and Haugereid, 2003)
- Modern Greek (Kordoni and Neu, 2005)
- Spanish (Marimon et al., 2007)
- Portuguese (Branco and Costa, 2008)
- Korean (Kim and Yang, 2003)

Grammar Matrix Context: DELPH-IN

Grammar development and deployment tools:

- LKB grammar development environment (Copestake, 2002)
- PET fast parser (Callmeier, 2002)
- [incr tsdb()] competence and performance profiling platform (Oepen, 2001)
- Parse- and realization-ranking (Toutanova et al., 2005; Velldal, 2008)
- Unknown word handling (Blunsom and Baldwin, 2006; Zhang and Kordoni, 2006)
- Tools for merging information from deep and shallow processing (Callmeier et al., 2004; Schäfer, 2007)

. . . and a wide variety of applications.

Outline

1 Introduction
- Multilingual Grammar Engineering
- Related Work
- DELPH-IN

2 The Matrix Customization System
- System Overview
- Notes on HPSG, Analyses and practicalities

3 Extended example: Maltese
- Word order and Auxiliaries
- Case, Negation, Argument Optionality
- Analyses, Part 3: The Lexicon

4 Extending a grammar
- Using the LKB and [incr tsdb()]
- Editing tdl
- Conclusion

Components of the Customization System

- Core grammar containing cross-linguistically useful types and constraints
- Libraries: Analyses of cross-linguistic variable phenomena
- Customization sytem:
  - Web-based questionnaire to elicit choices among libraries
  - Validation to check that answers are coherent
  - Back-end script to output grammars

System Overview

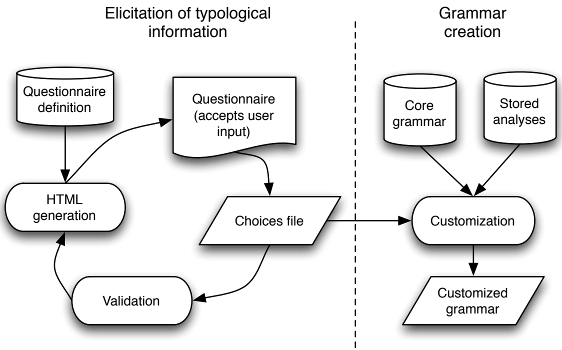


Figure: Schematic system overview (To the web page. . .)

Libraries

- Conceptually the subpart of the customization system which treats one phenomenon
- Library development begins with defining the phenomenon.
- Libraries interact with each other.
- A typical library involves both syntactic and lexical/morphological information.
  - In the customization system, libraries usually correspond to one subpage, plus information on the lexicon page.
  - Choices on the subpage enable options on the lexicon page.
- Some libraries offer closed menus of preset choices, others offer more flexibility (“metamodeling”).

HPSG Design Choices

- No relation constraints
- Closed-world type hierarchy
- No defeasible constraints
- Rules have a fixed arity

Analyses

- Words and lexical rules have an ARG-ST. Signs have the attributes SUBJ, COMPS and SPR attributes under VAL
- No adjuncts as arguments (yet)
- Lexical case-marking
- The Agreement Library does semantic agreement
- Lexical rules are non-branching productions
- Typically more schemata than in theoretical HPSG

A Note on Morphology

We find it desirable to separate morphophonology from morphosyntax (cf. Bender and Good, 2005). The customization system only supports strictly concatenative morphology without any phonological rules, while the LKB supports a small about of morphological rules.

Your test suites should be consistent in their orthography with what you enter in the lexicon page (spelling of stems and affixes). We encourage you to use a regularized, underlying form for both, such as would be the output of a finite-state morphological analyzer.

General best practice

- Data first: Prepare a test suite, preferably in IGT format following the Leipzig glossing rules (`http://www.eva.mpg.de/lingua/resources/glossing-rules.php`)
- Incremental development:
    - Answer only the required questions first, and then test (e.g., with test by generation).
    - Try one sample morpheme first before filling out large paradigms.
    - Periodically save your choices file.
- Take advantage of validation system—red asterisks indicate what needs to be corrected; hover over them for further information.

Test suites: Best Practices (repeated)

- Use IGT format and Leipzig Glossing Rules (Bickel et al., 2008)
- Include both test *suites* and test *corpora*
  - Test suites: Simple, constructed examples illustrating specific phenomena
  - Test corpora: Naturally occurring text
- Expect to iteratively improve and extend test suites alongside implemented grammars

Test suites

- Examples as would be used in linguistic papers
- Try to use few words
- Include examples of simple(r) phenomena to test how new implementations interact
- Negative examples (see next slide)

Negative examples

- Important for testing the grammar (use more than in your paper!)
- Make sure all words in negative examples are also included in some positive example
- Each phenomenon should (at least) be tested in a negative example with exactly one error
- Don’t be surprised if your negative examples become positive examples as you increase the grammar

Outline

1 Introduction
- Multilingual Grammar Engineering
- Related Work
- DELPH-IN

2 The Matrix Customization System
- System Overview
- Notes on HPSG, Analyses and practicalities

3 Extended example: Maltese
- Word order and Auxiliaries
- Case, Negation, Argument Optionality
- Analyses, Part 3: The Lexicon

4 Extending a grammar
- Using the LKB and [incr tsdb()]
- Editing tdl
- Conclusion

The Maltese language

- Semitic language spoken in Malta.
- 300,000+ speakers as of 1975.
- Closely related to Morrocan Spoken Arabic, with influence from Italian (Lewis, 2009).
- Described in (Fabri, 1993; Müller, 2009; Borg, 1981).
- Our testsuite draws heavily on one provided by Müller, consisting primarily of examples from Fabri 1993.
- It contains 59 examples, focused on illustrating the phenomena which can be handled through the customization system.

Phenomena

- *Word order and auxiliaries*
- Person, number, gender
- *Case*
- Tense/aspect
- *Negation*
- Coordination
- *Argument optionality*
- *Lexicon*

Word order

- We analyse Maltese as having free (i.e., pragmatically defined) major constituent order.
- Maltese also has determiners which precede the nouns they combine with.
- Further details in appendix slides (and in the choices file).

Auxiliaries I

Future is formed using the auxiliary *se*. The verbs *kien* (be) and *qed* (imperfect) can be analyzed as auxiliaries.
⇒ Select 'yes' has auxiliaries

Auxiliaries II

jkun	sar	it-tamar	
be-fut-3msg	become-past-3msg	df-date-pl	

“The dates will have ripened.” (Borg, 1981, 154)

Ġanni	qed	joqħod	il-Belt
John	*qed*	stay-3msg	in Valletta

“John is living in Valletta” (Borg, 1981, 114b)

Word order restrictions unkown: the auxiliary directly precedes the verb in the provided examples.

⇒ Select ‘V’ complement, and auxiliary ‘before’ complement

Auxiliaries III

Use of auxiliaries likely to be limited, word order might be free (possibly no obligatory cluster forming).
⇒ Select maximally one auxiliary

Case data

Maltese marks human direct objects and all indirect objects with *lil* (Fabri, 1993; Müller, 2009). Non-human NPs may not appear with *lil* in direct object position. (Pronouns are subject to a slightly different pattern.)

Raj-t	*(lil)	Pawlu.
Raj-CCvCt	lil	Pawlu
see-1SG	LIL	Pawlu.

‘I saw Pawlu.’

Xtraj-t	(*lil)	il-ktieb
Xtraj-CCvCt	lil	l-ktieb
buy-1SG	LIL	DEF-book

‘I bought the book.’

Case Analysis

⇒ Select 'Nominative-accusative' case system and define nominative and accusative cases.
⇒ Define dative as an additional case.

⇒ On 'Other features' page, define HUMAN and NTYPE as semantic features.

Negation Data

Pawlu	ma	ħareġx
Pawlu	ma	ħrġ-aeoo-CvCvC-x
Pawlu	neg	leave-3rd.masc.sing.int.vow.perf-neg

“Pawlu left”

*Pawlu ma ħareġ

*Pawlu ħareġx

*Pawlu ħareġx ma

Negation is formed by the adverb *ma*, which precedes the verb in combination with the suffix *-x*. Both are required.

Negation Analysis

The customization system cannot handle doubly marked negation at present. The easiest way to get this in the grammar is to define the adverb and add the properties of the morpheme manually

⇒ For sentential negation select:

- an independent modifier
- modifying V
- appearing before the item it modifies

⇒ A dummy slot for the morpheme *x* can be defined on the lexicon page (without properties for now)

Argument Optionality

Both subjects and objects may be dropped in Maltese

jiktebha
jvCCvC-ktb-ieie-ha
3ms.imperfect-write-3f.obj
“He writes it” (based on (Fabri, 1993))

Subject Dropping

Verbs agree with their subject in person, number and gender. The subject may be dropped in any context.

Select:

- Subject dropping may occur with any verb
- If the subject is dropped ⇒ subject marker required
- If the subject is overt ⇒ subject marker required
- Subject dropping occurs in all contexts

Object Dropping Data

When the object is dropped, an object marker is required. This marker is optional when the object is overt.

Pawlu	jiktebha
Pawlu	jvCCvC-ktb-ieie-ha
Pawlu	3ms.imperfect-write-3f.obj

Pawlu writes it

Pawlu	jikteb	il-ittra.
Pawlu	jvCCvC-ktb-ieie	l-ittra
Pawlu	3rd.imperfect-write	def-letter.fem

Pawlu writes the letter

*Pawlu	jikteb
Pawlu	jvCCvC-ktb-ieie
Pawlu	3ms.imperfect-write

Object Dropping Analysis

Select

- Object dropping may occur
  - with any verb
- If the object is dropped, an object marker on the verb is
  - required
- If the object is overt, an object marker on the verb is
  - optional
- Object dropping may occur in
  - all contexts

The Lexicon Page

- Allows the user to define types of nouns, verbs, determiners and adpositions
- Types are based on syntactic properties (one or more stems with related predicate must be defined for each class)
- Inflection (supported for nouns, verbs and determiners) is also defined on the lexicon page

Nouns

The following properties of nouns play a role in Maltese grammar

- Human versus non-human referent
- Grammatical gender masculine and feminine

⇒ Define three noun types:

- Nouns referring to humans (proper names)
- Nouns with feminine grammatical gender not referring to humans
- Nouns with masculine grammatical gender not referring to humans

Pronouns

- There is no special place to define pronouns on the lexicon page.
- They can be defined as noun types
- Each pronoun forms its own individual type
- Person, number, gender (and other relevant features) are defined as properties of the type

Main Verbs

Maltese has a nominative-accusative case marking pattern.
⇒ Define a verb type ‘intransitive’ with argument structure ‘intransitive(nom)’
⇒ Define a verb type ‘transitive’ with argument structure ‘transitive(nom-acc)’

Auxiliaries

se, *kien* and *qed* can be analyzed as auxiliaries. They contribute to the tense and aspect of the clause.
⇒ Define three auxiliary types. All three:

- Contribute ‘no predicate’
- Require their subject NP to bear the case assigned by its complement
- Take a complement in finite form

⇒ Each auxiliary type contributes different features to tense and aspect

Case Data (revisited)

Maltese marks human direct objects and all indirect objects with *lil* (Fabri, 1993; Müller, 2009). Non-human NPs may not appear with *lil* in direct object position. (Pronouns are subject to a slightly different pattern.)

Raj-t	*(lil)	Pawlu.
Raj-CCvCt	lil	Pawlu
see-1SG	LIL	Pawlu.
‘I saw Pawlu.’		

Xtraj-t	(*lil)	il-ktieb
Xtraj-CCvCt	lil	l-ktieb
buy-1SG	LIL	DEF-book
‘I bought the book.’		

Case-marking Adpositions

- The customization system cannot capture all aspects of the behavior of *lil*
  - The system assumes that case marking adpositions bear the same case as their complement nouns
  - The adposition can either be obligatory (for all nouns) or optional
- We can capture the fact that *lil* may not co-occur with nouns referring to non-humans

Case-marking Adpositions Analysis

⇒ Define a case-marking adposition

- with spelling *lil*
- which is optional
- and stands before the NP

⇒ Add features

- case = acc
- human = plus
- ntype = non-pro

Inflection

- Inflection is defined through “slots”
- For each slot, it is possible to define:
- Position(s):
  - Are the morphemes of the slot prefixes or a suffixes?
  - Where do they attach? (more than one input may be defined)
- Co-occurrence constraints:
  - Do morphemes from the slot require morphemes from some other slot?
  - Do morphemes from the slot prohibit morphemes from some other slot?

Verb inflection

- Maltese verbs are marked for aspect and the subject's person, number and gender.
- These properties are mainly captured by consonant-vowel patterns, plus additional consonants or vowels
- The additional phonemes may precede or follow the stem, leading us to posit prefixes and suffixes in our abstract representation.

Morphophonological processes

- Recall that the system does not handle morphophonology
- We represent the morphology of Maltese verbs as follows:

stem	thematic vowels	consonant-vowel-pattern
ħareġ		
ħrġ	-aeoo	-CvCvC

Verb inflection analysis

- Two PNG/aspect inflection slots
  - One before, one after the stem
  - Each contain morphemes with aspect, person, number, gender agreement constraints
  - Both serve as input to the object marker slot
- Object marker slot
  - Contains over object marker morphemes
  - (Customization system will also provide zero-marked “no dropping” morpheme)
  - Required by transitive verbs, incompatible with intransitives

Outline

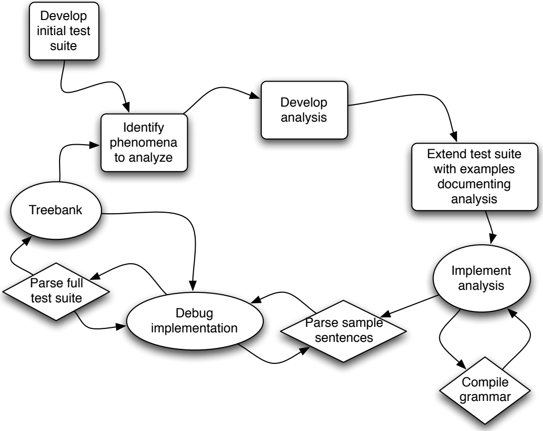
1 Introduction
- Multilingual Grammar Engineering
- Related Work
- DELPH-IN

2 The Matrix Customization System
- System Overview
- Notes on HPSG, Analyses and practicalities

3 Extended example: Maltese
- Word order and Auxiliaries
- Case, Negation, Argument Optionality
- Analyses, Part 3: The Lexicon

4 Extending a grammar
- Using the LKB and [incr tsdb()]
- Editing tdl
- Conclusion

Workflow (Reprise)



A First Session

- Start emacs: emacs &
- Start the LKB: M-x lkb
- Load the grammar: C-c g (or through the menu)
- Parse an item: C-c p (or through the menu)
- Explore parse chart

Regression testing with [incr tsdb()]

The LKB has batch testing facilities, but they are very basic. [incr tsdb()] allows detailed exploration of differences between test runs.

- Start [incr tsdb()]: M-x itsdb
- Set database root
- Set skeleton root
- Create skeletons
- Create instance
- Process all items

Ways to explore the data

- Browse | Results: Which examples parsed.
  - Red items can be clicked, to view structures or to send to the LKB for interactive parsing.
- Browse | Test items: Interactive parsing, of any example.
- Analyze | Competence: Overview of coverage and overgeneration.
- Compare | Competence: Comparison of coverage and overgeneration between two test suite profiles.
- Compare | Detail: Which items have different (number of) analyses.
- Options | Tsql condition: Restrict output to a subset of the data.

Understanding the grammar

- Individual components of the grammar are divided over a set of files (more later)
- The grammar is written in tdl (type description language)

⇒ The following slides provide an overview of tdl and the components of the grammar

Type Description Language in a nutshell (1)

How to define types?

- The following syntax is used to define a type:

  new-type := *supertype*.
- This statement introduces a type (*new-type*) that inherits properties of some already existing type (*supertype*)
- A type may inherit properties from more than one type:

  new-type := *supertype1* & *supertype2*.

TDL in a nutshell (2)

Adding new properties to a type

- In addition to inheriting properties from already existing types, a new type may introduce properties of its own, e.g.

  new-type := *supertype1* & *supertype2* &
  [PATH.FEATURE1 value1].

  assigns *value1* to FEATURE1

TDL in a nutshell (3)

Note that:

- FEATURE1 may be already defined, in that case, it must either be defined as a feature of a supertype of *new-type*, and be located at PATH, or it must be an appropriate feature of the value of PATH
- FEATURE1 may be new, in which case no other feature with the same name may exist in the grammar
- *value1* must be defined as a type

TDL in a nutshell (4)

Reentrancy

- Reentrancies are encoded using #, e.g.

```
adjective := modifier &
  [ SYNSEM.LOCAL.CAT [ HEAD [ CASE #case,
                              MOD < [ LOCAL.CAT.HEAD.CASE #case ] > ] ] ].
```

tdl primer

Type definition with errors (example)

```
type-identifier := supertype1 & supertype2 &
[  FEATURE1 type1,
   FEATURE2 #coref,
   FEATURE3 [ FEATURE4 type2,
              FEATURE5 type3 ].
```

⇒ LKB warns about bracketing error and coreference used only once

tdl primer

Type definition corrected

```
type-identifier := supertype1 & supertype2 &
[  FEATURE1 type1,
   FEATURE2 #coref,
   FEATURE3 [ FEATURE4 #coref & type2,
                        FEATURE5 type3 ]].
```

LKB also checks:

- Does the supertype exist?
- Are there redundant supertypes? E.g. *head-comp-phrase* below:

  head-initial := *headed-phrase* &...
  head-comp-phrase := *head-initial* & *headed-phrase*.
- Does the feature-name conflict with another feature?
  ⇒ also triggered when a feature is defined at the wrong location
- Is the value assigned to the feature the appropriate type?
- Are there types that contain any constaints that conflict with one of its supertype?

Type and Instance Files

- Type files:
  - matrix.tdl, head-types.tdl: Matrix core grammar
  - my_language.tdl: language-specific type definitions
- Instance files:
  - lexicon.tdl: Lexical entries
  - irules.tdl: Spelling-changing lexical rules
  - lrules.tdl: Non-spelling changing lexical rules
  - rules.tdl: Phrase structure rules

Additional (collateral) Files

- roots.tdl: Initial symbol definitions
- labels.tdl: Node abbreviation definitions
- lkb/script: Load file
- lkb/globals.lsp, lkb/mrsglobals.lisp: Language-specific LKB parameters
- pet.tdl, my_language-pet.tdl: PET configuration files

Exploring the grammar

- Most relevant properties of the grammar are defined in the matrix.tdl and my_language.tdl file
- First steps in exploring the grammar:
  - Examine the types in my_language.tdl
  - (Examine their supertypes in matrix.tdl)
  - Explore the types matrix.tdl has to offer

The Matrix Core

- The Core Grammar *matrix.tdl* is meant to be used as the basis of all Matrix Grammars. It provides:
  1. Basic features and devices used in HPSG grammars (e.g. phrase, word, category, lists)
  2. Basic grammar rules (e.g. unary/binary rules, head-subject/head-complement/head-specifier, head-final/head-initial)
  3. Semantic structures and constraints ensuring semantic compositionality, in the style of MRS (Copestake et al., 2005)
  4. Some more advanced features (e.g. simple part of speech inventory, argument extraction, coordination)

Example: what you find in my_language.tdl

Implementation for a language with word order
Subject **O**bject **V**erb:

> *comp-head-rule := basic-head-compl-phrase & head-final.*
>
> *subj-head-rule := basic-head-subj-rule & head-final &*
> [SYNSEM.LOCAL.VAL.COMPS < >].

The basic properties of these rules are defined in *matrix.tdl*.

Supertype of the basic-head-comp-phrase

```
basic-head-comp-phrase := head-nexus-phrase & basic-binary-headed-phrase &
[ SYNSEM phr-synsem-min &
        [ LOCAL [ CAT [ VAL [ SUBJ #subj,
                              SPR #spr ],
                        POSTHEAD #ph,
                        HC-LIGHT #light ],
                        CONT.HOOK #hook],
        LIGHT #light,
        NON-LOCAL.SLASH #slash]
INFLECTED +,
HEAD-DTR.SYNSEM [local.cat [ VAL [ SUBJ #subj,
                                   SPR #spr ],
                             HC-LIGHT #light,
                             POSTHEAD #ph ]],
                 NON-LOCAL.SLASH #slash
NON-HEAD-DTR.SYNSEM canonical-synsem &
                    [ LOCAL.COORD - ],
C-CONT [ RELS <! !>,
         HCONS <! !>,
         HOOK #hook ],
ARGS < [ INFLECTED + ],
       [ INFLECTED + ] > ].
```

The role of matrix.tdl when extending your Grammar

- The matrix core saves you the trouble of worrying about many details.
- It contains several useful types that are not instantiated by the libraries at present.
- You may need to examine matrix.tdl to understand the behavior of your grammar.
- Types in matrix.tdl may provide useful examples of how to implement aspects of your analysis.

my_language.tdl

- Contains specific types for the language you are working with
- Most (or all) types that are instantiated in rules.tdl, lexicon.tdl. irules.tdl, and lrules.tdl are defined here.
- In starter grammar, most types definitation will be relatively simple
- The bulk of grammar engineering will be done in this file
- Easiest start: extend an analysis provided by the customization system that does not capture the grammar completely

my_language.tdl

- Contains specific types for the language you are working with
- Most (or all) types that are instantiated in rules.tdl, lexicon.tdl. irules.tdl, and lrules.tdl are defined here.
- In starter grammar, most types definitation will be relatively simple
- The bulk of grammar engineering will be done in this file
- Easiest start: extend an analysis provided by the customization system that does not capture the grammar completely

so let’s get started...

Phenomena to be implemented

Recall that there were two phenomena that could not be handled completely with the customization system:

1. A case marker that only appears on human direct objects
2. Negation is marked by an adverb in combination with a suffix on the verb

Case Data (revisited)

Maltese marks human direct objects and all indirect objects with *lil* (Fabri, 1993; Müller, 2009). Non-human NPs may not appear with *lil* in direct object position. (Pronouns are subject to a slightly different pattern.)

Raj-t	*(lil)	Pawlu.
Raj-CCvCt	lil	Pawlu
see-1SG	LIL	Pawlu.

‘I saw Pawlu.’

Xtraj-t	(*lil)	il-ktieb
Xtraj-CCvCt	lil	l-ktieb
buy-1SG	LIL	DEF-book

‘I bought the book.’

Customization System Output

- *lil* correctly only attaches to human nouns
- But human nouns can be objects without *lil*.
- ⇒ Overgeneration.
- Case marking adpositions identify their own CASE value with their complements'.

Improved Analysis

- Make case marking adpositions have independent case value from their complements.
- Make proper nouns inherently [CASE *nom*].

Negation, revisited

Pawlu	ma	ħareġx
Pawlu	ma	ħrġ-aeoo-CvCvC-x
paul	neg	leave-3rd.masc.sing.int.vow.perf-neg

Paul left

*Pawlu ma ħareġ

*Pawlu ħareġx

*Pawlu ħareġx ma

Negation is formed by the adverb *ma*, which precedes the verb in combination with the suffix *-x*. Both are required

Customization system output

- Independent adverb, which attaches to the left of V.
- Meaningless suffix *-x*.
- ⇒ Nothing in this analysis requires both of these to co-occur.

Improved analysis

There are two main techniques to improve on the basic analysis

1. Using a feature to assure that *ma* and *-x* co-occur
2. Treat *ma* like a selected adverb

Let’s look at both techniques in more detail

Using a feature (version 1)

- Introduce a feature e.g. [NEG *bool*]: *ma* requires the verb to be [NEG +]
- -*x* assigns [NEG +] to the verbs it attaches to
- a zero morpheme in the same inflection slot as -*x* makes verbs [NEG −]

⇒ This way, *ma* will always co-occur with -*x*, but -*x* may still occur without *ma*

Using a feature (version 2)

- Introduce the feature [NEG *luk*], with possible values +, -, *na*, *na-or*-+, and *na-or*-–
- a zero morpheme in the same inflection slot as -*x* makes features [NEG –]
- -*x* makes verbs [NEG +]
- *ma* requires verbs to be [NEG +], but changes this value into [NEG *na*]
- The head of a clause may not be [NEG +]

⇒ This captures the data without over-generation

⇒ Draw-back: this requires many additional constraints in the grammar

ma as a selected adverb

- The morpheme -*x* adds *ma* to the verbs COMPS list

⇒ *ma* is required when -*x* occurs, and it can only occur when -*x* is present

- We need to restrict the grammar so that *ma*
  - only precedes the verb
  - only attaches to lexical Vs

Tutorial Goals

- Introduce the LinGO Grammar Matrix system
- Illustrate how to derive the most benefit from the system
- Demonstrate how to work with and extend a starter grammar
- Exemplify the methodology of grammar engineering for linguistic hypothesis testing

To learn more. . .

- UW Ling 567 course web page:

  `http://courses.washington.edu/ling567`

- Matrix mailing list:

  `matrix@lists.delph-in.net`

- Our approach to data-driven cross-linguistic hypothesis testing relies on feedback from users.

- We are always interested to know how the system is being used, what's confusing, what's clear.

  ⇒ Please feel free to ask questions!

Bibliography I

Baldridge, J., Chatterjee, S., Palmer, A., and Wing, B. (2007). DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG. In King, T. H. and Bender, E. M., editors, *Proceedings of the GEAF07 Workshop*, pages 5–25. CSLI.

Bateman, J. A., Kruijff-Korbayová, I., and Kruijff, G.-J. (2005). Multilingual resource sharing across both related and unrelated languages: An implemented, open-source framework for practical natural language generation. *Research on Language and Computation, Special Issue on Shared Representations in Multilingual Grammar Engineering*, 3(2):191–219.

Bender, E. M. and Good, J. (2005). Implementation for discovery: A bipartite lexicon to support morphological and syntactic analysis. In *Proceedings from the Panels of the Forty-First Meeting of the Chicago Linguistic Society: Volume 41-2.*

Bickel, B., Comrie, B., and Haspelmath, M. (2008). The Leipzig glossing rules. conventions for interlinear morpheme by morpheme glosses. Max Planck Institute for Evolutionary Anthropology and Department of Linguistics, University of Leipzig.

Black, C. A. (2004). Parser and writer for syntax. Paper presented at the International Conference on Translation with Computer-Assisted Technology: Changes in Research, Teaching, Evaluation, and Practice, University of Rome “La Sapienza”, April 2004.

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.

Blunsom, P. and Baldwin, T. (2006). Multilingual deep lexical acquisition for hpsgs via supertagging. In *Proceedings of EMNLP*, volume 6, pages 164–171.

Borg, A. J. (1981). *A Study of Aspect in Maltese*. Karoma Publishers, Inc, Ann Arbor, USA.

Bibliography II

Bouillon, P., Rayner, M., Vall, B. N., Starlander, M., Santaholma, M., Nakao, Y., and Chatzichrisafis, N. (2006). Une grammaire partagée multi-tâche pour le traitement de la parole : application aux langues romanes. *TAL (Traitement Automatique des Langues)*, 47.

Branco, A. and Costa, F. (2008). A computational grammar for deep linguistic processing of portuguese: Lxgram, version a.4.1. Technical report. Technical Report, University of Lisbon, Department of Informatics.

Butt, M., Dyvik, H., King, T. H., Masuichi, H., and Rohrer, C. (2002). The parallel grammar project. 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 1–7.

Callmeier, U. (2002). Preprocessing and encoding techniques in pet. In Oepen, S., Flickinger, D., Tsujii, J., and Uszkoreit, H., editors, *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*. CSLI Publications, Stanford, CA.

Callmeier, U., Eisele, A., Schäfer, U., and Siegel, M. (2004). The deepthought core architecture framework. In *Proceedings of LREC 04*, Lisbon, Portugal.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005). Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(4):281–332.

de la Clergerie, É. V. (2005). From metagrammars to factorized TAG/TIG parsers. In *Proceedings of IWPT'05*, pages 190–191.

Fabri, R. (1993). *Kongruenz und die Grammatik des Maltesischen*. Linguistische Arbeiten. Niemeyer Verlag, Tübingen, Germany.

Bibliography III

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28.

Hellan, L. and Haugereid, P. (2003). NorSource: An exercise in Matrix grammar-building design. In Bender, E. M., Flickinger, D., Fouvry, F., and Siegel, M., editors, *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 41–48, Vienna, Austria.

Kim, J.-B. and Yang, J. (2003). Korean phrase structure grammar and its implementations into the LKB system. In *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation*, pages 88–97.

King, T. H., Forst, M., Kuhn, J., and Butt, M. (2005). The feature space in parallel grammar writing. *Research on Language and Computation, Special Issue on Shared Representations in Multilingual Grammar Engineering*, 3(2):139–163.

Kordoni, V. and Neu, J. (2005). Deep analysis of Modern Greek. In Su, K.-Y., Tsujii, J., and Lee, J.-H., editors, *Lecture Notes in Computer Science*, volume 3248, pages 674–683. Springer-Verlag, Berlin.

Lewis, M. P., editor (2009). *Ethnologue: Languages of the World*. SIL International, Dallas, TX, sixteenth edition. Online version: http://www.ethnologue.com/.

Marimon, M., Bel, N., and Seghezzi, N. (2007). Test-suite construction for a Spanish grammar. In King, T. H. and Bender, E. M., editors, *Proceedings of the GEAF 2007 Workshop*, Stanford, CA. CSLI Publications.

McShane, M. and Nirenburg, S. (2003). Parameterizing and eliciting text elements across languages for use in natural language processing systems. *Machine Translation*, 18:129–165.

Bibliography IV

Monson, C., Llitjós, A. F., Ambati, V., Levin, L., Lavie, A., Alvarez, A., Aranovich, R., Carbonell, J., Frederking, R., Peterson, E., and Probst, K. (2008). Linguistic structure and bilingual informants help induce machine translation of lesser-resourced languages. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Müller, S. (2009). Towards an HPSG analysis of Maltese. In Comrie, B., Fabri, R., Hume, B., Mifsud, M., Stolz, T., and Vanhove, M., editors, *Introducing Maltese linguistics. Papers from the 1st International Conference on Maltese Linguistics (Bremen/Germany, 18–20 October, 2007)*, volume 113 of *Studies in Language Companion Series*, pages 83–112. John Benjamins Publishing Co., Amsterdam, Philadelphia.

Müller, S. and Kasper, W. (2000). HPSG analysis of German. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer, Berlin.

Oepen, S. (2001). [incr tsdb()] — Competence and performance laboratory. User manual. Technical report, Saarbrücken, Germany.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press and CSLI Publications, Chicago, IL and Stanford, CA.

Probst, K., Brown, R., Carbonell, J., Lavie, A., Levin, L., and Peterson, E. (2001). Design and implementation of controlled elicitation for machine translation of low-density languages. In *Workshop MT2010 at Machine Translation Summit VIII*, pages 189–192.

Ranta, A. (2007). Modular grammar engineering in GF. *Research on Language & Computation*, 5(2):133–158.

Schäfer, U. (2007). *Integrating Deep and Shallow Natural Language Processing Components – Representations and Hybrid Architectures*. PhD thesis, Faculty of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany.

Bibliography V

Sheremetyeva, S. and Nirenburg, S. (2000). Acquisition of a language computational model for NLP. In *Proceedings of COLING'2000*, Saarbrücken, Germany.

Siegel, M. and Bender, E. M. (2002). Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Toutanova, K., Manning, C. D., Flickinger, D., and Oepen, S. (2005). Stochastic HPSG parse disambiguation using the Redwoods corpus. *Research on Language & Computation*, 3(1):83–105.

Velldal, E. (2008). *Empirical Realization Ranking*. PhD thesis, University of Oslo, Department of Informatics.

Zhang, Y. and Kordoni, V. (2006). Automated deep lexical acquisition for robust open texts processing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 275–280.