Emily M. Bender, Antske S. Fokkens and Safiyyah Saleem

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:

  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
   - HPSG in a Nutshell
   - Practicalities

3. Extended example: Maltese
   - Overview
   - Analyses, Part 1: Word order–PNG
   - Analyses, Part 2: Case–Argument Optionality
   - Analyses, Part 3: The Lexicon

4. Extending a grammar
   - Using the LKB
   - Regression testing with [incr tsdb()]
   - Editing tdl
   - Conclusion
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
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)
Grammar Matrix Context: DELPH-IN

Applications

- Machine translation (Oepen et al., 2007)
- Question answering from structured knowledge sources (Frank et al., 2006)
- Robust textual entailment (Bergmair, 2008)
- Knowledge extraction from scientific text (Rupp et al., 2007)
- Ontology construction (Nichols et al., 2006)
- ...
The Grammar Matrix facilitates the development of grammars for low-resource languages that can take advantage of these tools and applications designed for high-resource languages.
Outline

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

2. The Matrix Customization System
   - System Overview
   - HPSG in a Nutshell
   - Practicalities

3. Extended example: Maltese
   - Overview
   - Analyses, Part 1: Word order–PNG
   - Analyses, Part 2: Case–Argument Optionality
   - Analyses, Part 3: The Lexicon

4. Extending a grammar
   - Using the LKB
   - Regression testing with [incr tsdb()]
   - Editing tdl
   - Conclusion

Bender, Fokkens & Saleem

U. Washington & U. Saarlandes
# Components of the Customization System

- Core grammar containing cross-linguistically useful types and constraints
- Libraries: Analyses of cross-linguistic variable phenomena
- Customization system:
  - Web-based questionnaire to elicit choices among libraries
  - Validation to check that answers are coherent
  - Back-end script to output grammars
Figure: Schematic system overview (To the web page...)

Bender, Fokkens & Saleem
U. Washington & U. Saarlandes
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").
A system of signs

- Saussurean signs pair forms (orthographic/phonological as well as morphosyntactic) with meanings (semantics).
- Lexical entries, lexical rules and phrase structure rules are all signs.
- Signs are modeled with typed feature structures, where features can take on atomic as well as complex values (other feature structures, lists of feature structures).
- Phrase structure rules have one or more daughters, but always fixed arity.
- Lexical rules are non-branching rules (one daughter) which can only take other lexical rules or lexical entries as their daughters.
HPSG in a Nutshell

Unification and typed feature structures

- Feature structures are combined using *unification*.
- Unification is order-independent.
- Types are arranged into a multiple inheritance hierarchy.
- Type constraints specify appropriate features for each type as well as constraints on their values.
- The type hierarchy determines which types will unify with each other (closed-world assumption).
Familiarity with HPSG is required for extending Matrix-derived grammars, but not for creating starter grammars with the customization system.
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 amount 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.
Outline

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

2. The Matrix Customization System
   - System Overview
   - HPSG in a Nutshell
   - Practicalities

3. Extended example: Maltese
   - Overview
   - Analyses, Part 1: Word order–PNG
   - Analyses, Part 2: Case–Argument Optionality
   - Analyses, Part 3: The Lexicon

4. Extending a grammar
   - Using the LKB
   - Regression testing with [incr tsdb]
   - Editing tdl
   - Conclusion
The Maltese language

- Semitic language spoken in Malta.
- 300,000+ speakers as of 1975.
- Closely related to Moroccan 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
- Person, number, gender
- Case
- Tense/aspect
- Negation
- Coordination
- Yes-no questions
- Argument optionality
- Lexicon
Main constituent order

Most permutations of O, S, and V are possible, under the appropriate information structure interpretations.
⇒ Select “free word order” on the word order page.

Norma fetħi-t-u l-bieb. (SVO)
Norma opened-3fsg-3msg df-door-msg
“Norma opened the door.” (Fabri, 1993, 141)

Norma l-bieb fetħi-t-u. (SOV)
Fetħi-t-u Norma l-bieb. (VSO)
L-bieb Norma fetħi-t-u. (OSV)
L-bieb fetħi-t-u Norma. (OVS)
fetħi-t-u l-bieb Norma. (VOS)
Determiners

Maltese has demonstrative determiners as well as possible indefinite articles which appear pre-nominally.
⇒ Select ‘has independent determiners’
⇒ Select ‘determiner precedes the noun’

Pawlu kiteb dan il-ktieb
Pawlu wrote this df-book
“Pawlu wrote this book.” (Fabri, 1993, 60)

wafßda mara
INDEF-fsg woman “a woman” (Fabri, 2001, 155)
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 joqḥod il-Belt
John qed stay-3msg in Valletta
“John is living in Valletta” (Borg, 1981, 114b)

Word order restrictions unknown: the auxiliary directly precedes the verb in the provided examples.
⇒ Select ‘V’ complement, and auxiliary ‘before’ complement
Use of auxiliaries likely to be limited, word order might be free (possibly no obligatory cluster forming).
⇒ Select maximally one auxiliary
Singular and plural markers are present in our dataset. According to [wikipedia](http://en.wikipedia.org/wiki/Maltese_language, accessed 2010/05/13), Maltese also distinguishes dual on nouns. Since there is no evidence of dual on verbs, we will assume a non-singular type which subsumes dual and plural.

⇒ Define ‘singular’, ‘non-singular’, ‘dual’ and ‘plural’ on the Number page

**Note**

⇒ More relevant data is needed to find the correct representation of number in Maltese

Bender, Fokkens & Saleem

U. Washington & U. Saarlandes
There are pronouns and there is subject agreement for 1st, 2nd and 3rd person
⇒ Select the option 1st, 2nd, 3rd on the person page

There are (to our knowledge) no subtypes of 1st person (inclusive/exclusive distinction)
⇒ Select ‘none’ for subtypes of 1st person
Maltese distinguishes masculine and feminine gender, on nouns and in subject agreement
⇒ Define masculine and feminine on gender page
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.
  see-1SG LIL Pawlu.
  ‘I saw Pawlu.’

- Xtraj-t (*lil) il-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.
### Tense/aspect Data

<table>
<thead>
<tr>
<th>Pawlu</th>
<th>Kiteb</th>
<th>Il-Lettra</th>
<th>Maltese Verb</th>
<th>Def-Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pawlu</td>
<td>write-3ms.perfect</td>
<td>Il-Lettra</td>
<td>&quot;Pawlu wrote the letter&quot;</td>
<td>Def-Letter</td>
</tr>
<tr>
<td>Pawlu future 3ms.imperfect-write</td>
<td>Il-Lettra</td>
<td>&quot;Pawlu is going to write the letter&quot;</td>
<td>Def-Letter</td>
<td></td>
</tr>
</tbody>
</table>
Tense/aspect Analysis

⇒ Select elements from common tense hierarchy
  - past, future, present

⇒ Define types in section “viewpoint aspect”
  - imperfect, subtype of aspect
  - perfect, subtype of aspect
  - progressive, subtype of imperfect
### Negation Data

<table>
<thead>
<tr>
<th>Pawlu</th>
<th>ma</th>
<th>ðareġx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pawlu</td>
<td>ma</td>
<td>ðrġ-ae-x</td>
</tr>
<tr>
<td>Pawlu</td>
<td>neg</td>
<td>leave-3rd.masc.sing.int.vow.perf-neg</td>
</tr>
</tbody>
</table>

“Pawlu left”

*Pawlu ma ðareġ

*Pawlu ðareġx

*Pawlu ðareġx 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)
Noun phrases and verb phrases can be coordinated using the word *u*. The coordinator stands in between the coordinands.

Pawlu u Norma ḡareġu
Pawlu and Norma leave-perfect.3pl
“Pawlu and Norma left.” based on (Borg, 1981)

Pawlu kiel ilmazzita u ḡareġ.
Pawlu eat-3ms.perfect def-blutwurst.fsg and leave-perfect.3ms
“Pawlu ate the blutwurst and left.” based on (Borg, 1981)
Coordination Analysis

The marking pattern of Maltese coordinated structures is either **monosyndeton** (A B and C)
⇒ Select ‘monosyndeton’ marking pattern (or ‘polysyndeton’)
⇒ Select marked by
  - a word
  - spelled “u”
  - that comes before the coordinand
Argument Optionality

Both subjects and objects may be dropped in Maltese

jiktebha
3ms.imperfect-write-3f.obj
“He writes it” (based on (Fabri, 1993))
Analyses, Part 2: Case–Argument Optionality

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 3ms.imperfect-write-3f.obj
Pawlu writes it

Pawlu jikteb il-ittra.
Pawlu 3rd.imperfect-write def-letter.fem
Pawlu writes the letter

*Pawlu jikteb
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.
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)’
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
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.
see-1SG LIL Pawlu.
‘I saw Pawlu.’

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

Common nouns can be marked for number and definiteness

⇒ Define two slots:

■ The obligatory slot ‘number’, appearing after noun-type2 and noun-type3, with morphemes
  ■ *jiet* marking plural
  ■ a phonologically empty morpheme (leaving spelling blank) to indicate ‘singular’

■ The optional slot ‘definite’, appearing before noun-type2, noun-type3 or the number 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, but in our abstract representation, we normalize to after only.
Recall that the system does not handle morphophonology.

We represent the morphology of Maltese verbs as follows:

<table>
<thead>
<tr>
<th>Stem</th>
<th>Thematic Vowels</th>
<th>Consonant-Vowel-Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>ṣareġ</td>
<td>-</td>
<td>-aeoo -CvCvC</td>
</tr>
</tbody>
</table>
Verb inflection analysis

⇒ Define two inflection slots

- Slot 1 (for intransitives)
  - appears after the stem
  - each morpheme defines aspect, person, number, gender agreement variations

- Slot 2 (for transitives)
  - appears after the stem
  - each morpheme defines aspect, person, number, gender agreement variations
  - serves as input to object marker slot
For some inflection phenomena, it is non-trivial how to capture them with the system. E.g.:

- Morphemes appearing in the same position with different syntactic properties
- Circumfixes
A circumfix can be handled by the system, if it is defined as two morphemes:

1. Define a slot (slot1) for the part of the circumfix that precedes the stem
2. Define a slot (slot2) for the part of the circumfix that follows the stem
3. Add a constraint to slot1 that slot2 is required, and vice versa
Different syntactic constraints within slot?

- In Turkish, the paradigm for subject agreement depends on the TAM-morpheme preceding it:
  
  | stem... | -dI/-sE       | paradigm1 |
  | stem... | -mEli/-mls/-... | paradigm2 |

- Intuitively, we would define a single slot for the TAM-morphemes, and one for the agreement morphemes.

- This grammar would over-generate: the wrong agreement marker may be used.
Defining Separate Slots

Solution: define 4 slots rather than two

1. Define two slots for the final TAM-morpheme
2. Forbid the slots to co-occur
3. Do the same for the agreement morphemes
4. TAM-slot 1 will require agreement from slot 3
5. TAM-slot 2 will require agreement from slot 4

<table>
<thead>
<tr>
<th></th>
<th>slot 1</th>
<th>slot 2</th>
<th>slot 3</th>
<th>slot 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>stem</td>
<td>-dl/-sE</td>
<td>—</td>
<td>AGR1</td>
<td>—</td>
</tr>
<tr>
<td>stem</td>
<td>—</td>
<td>-mls/mEli/...</td>
<td>—</td>
<td>AGR2</td>
</tr>
</tbody>
</table>
Outline

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

2. The Matrix Customization System
   - System Overview
   - HPSG in a Nutshell
   - Practicalities

3. Extended example: Maltese
   - Overview
   - Analyses, Part 1: Word order–PNG
   - Analyses, Part 2: Case–Argument Optionality
   - Analyses, Part 3: The Lexicon

4. Extending a grammar
   - Using the LKB
   - Regression testing with [incr tsdb()]
   - Editing tdl
   - Conclusion
Using the LKB

Workflow

- Develop initial test suite
- Identify phenomena to analyze
- Treebank
- Parse full test suite
- Debug implementation
- Parse sample sentences
- Compile grammar
- Implement analysis
- Extend test suite with examples documenting analysis
- Develop analysis
- Parse full test suite

Figure: Grammar engineering workflow
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 \texttt{[incr tsdb()]} 

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

- Start \texttt{[incr tsdb()]}: \texttt{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:
  
  \[ \textit{new-type} := \textit{supertype}. \]

- This statement introduces a type (\textit{new-type}) that inherits properties of some already existing type (\textit{supertype}).

- A type may inherit properties from more than one type:
  
  \[ \textit{new-type} := \textit{supertype1} & \textit{supertype2}. \]
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`
Note that:

- FEATURE1 may be already defined, in that case it must be defined for a supertype of *new-type*, and be located at 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)

Unification

- Unification is encoded using #, e.g.

\[
\text{adjective} := \text{modifier} \&
\begin{align*}
&[ \text{SYNSEM.LOCAL.CAT} [ \text{HEAD} [ \text{CASE} \#\text{case}, \\
&\quad \text{MOD} < [ \text{LOCAL.CAT.HEAD.CASE} \#\text{case} ] > ] ]] .
\end{align*}
\]
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
LKB also checks:

- Does the supertype exist?
- Are there redundant supertypes? E.g. `head-comp-phrase` below:
  
  \[
  \text{head-initial} := \text{headed-phrase} & \ldots \\
  \text{head-comp-phrase} := \text{head-initial} \& \text{headed-phrase}.
  \]

- Does the feature-name conflict with another feature? 
  \[\Rightarrow\] 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 constraints that conflict with one of its supertype?

- Are there types that inherit from conflicting supertypes?
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 Object Verb:**

\[
\text{comp-head-rule} := \text{basic-head-compl-phrase} \& \text{head-final}. \\
\text{subj-head-rule} := \text{basic-head-subj-rule} \& \text{head-final} \& \ \\
\left[ \text{SYNSEM.LOCAL.VAL.COMPS} < > \right].
\]

The basic properties of these rules are defined in *matrix.tdl*. 
Supertype of the basic-head-comp-phrase

\[
\text{basic-head-comp-phrase} := \text{head-nexus-phrase} \land \text{basic-binary-headed-phrase} \land \\
[\text{SYNSEM} \text{phr-synsem-min} \land \\
[\text{LOCAL} [\text{CAT} [\text{VAL} [\text{SUBJ} \#\text{subj}, \\
\text{SPR} \#\text{spr}], \\
\text{POSTHEAD} \#\text{ph}, \\
\text{HC-LIGHT} \#\text{light}], \\
\text{CONT.HOOK} \#\text{hook}], \\
\text{LIGHT} \#\text{light}, \\
\text{NON-LOCAL}.\text{SLASH} \#\text{slash}] \\
\text{INFLECTED +}, \\
\text{HEAD-DTR}.\text{SYNSEM} [\text{local.cat} [\text{VAL} [\text{SUBJ} \#\text{subj}, \\
\text{SPR} \#\text{spr}], \\
\text{HC-LIGHT} \#\text{light}, \\
\text{POSTHEAD} \#\text{ph}]], \\
\text{NON-LOCAL}.\text{SLASH} \#\text{slash} \\
\text{NON-HEAD-DTR}.\text{SYNSEM} \text{canonical-synsem} \land \\
[\text{LOCAL}.\text{COORD} - ], \\
\text{C-CONT} [\text{RELS} < ! ! >, \\
\text{HCONS} < ! ! >, \\
\text{HOOK} \#\text{hook}], \\
\text{ARGS} < [\text{INFLECTED} + ], \\
[\text{INFLECTED} + ] > ].
\]
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 Irules.tdl are defined here
- In your starting grammar, most types definition will (still) 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
Editing tdl

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 your starting grammar, most types definition will (still) 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...

Bender, Fokkens & Saleem

U. Washington & U. Saarlandes
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
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.
see-1SG LIL Pawlu.
‘I saw Pawlu.’

Xtraj-t (*lil) il-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 〝hağa‘x
Pawlu ma 〝hğa‘-ae-x
paul neg leave-3rd.masc.sing.int.vow.perf-neg
Paul left
*Pawlu ma 〝hğa‘
*Pawlu 〝hğa‘x
*Pawlu 〝hğa‘x 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.
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
Editing tdl

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 Grammar Matrix customization system
- Explain the approach to producing language resources
- Illustrate how to derive the most benefit from the system
- Demonstrate how to work with and extend a starter grammar
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


Bibliography II


Conclusion

Bibliography III


