Some Fine Points of Hybrid Natural Language Processing

Peter Adolphs, DFKI GmbH, Language Technology Lab, Berlin

Stephan Oepen, Universitetet i Oslo, Department of Informatics
Ulrich Callmeier, acrolinx GmbH, Berlin
Berthold Crysmann, Universität Bonn
Dan Flickinger, Stanford University, CSLI
Bernd Kiefer, DFKI GmbH, Language Technology Lab, Saarbrücken
Motivation

- hybrid processing, integrating annotations of ‘shallow’ tools into HPSG parsing
- different tools make different assumptions
- example: PTB-style tokenizers for English
  - e.g.: Don't you! → <do, n't, you, !>
  - contracted verb forms are split
  - punctuation is split off the preceding word form
- we need to adapt annotations of different tools to the requirements of our grammar
- goal: a declarative, expressive, scalable device
Token Feature Structures

- feature structures for describing tokens
- different annotations provided as feature structures
- lattice of structured categories (token feature structures) as input to the parser
• tools may assume different tokenization (paradigm case: input from speech recognizers)

• chart: dag whose vertices are abstract objects rather than indexed token boundary positions
Chart Mapping

• chart mapping: non-monotonic rewrite mechanism on feature structure chart edges

• general format:

\[
[\text{CONTEXT} : ] \text{INPUT} \rightarrow \text{OUTPUT}
\]

• CONTEXT, INPUT, OUTPUT are sequences of feature structures (each possibly empty)

• resource-sensitive: chart edges that let a rule fire may be removed (namely, all INPUT edges)
Chart Mapping – Example

\[
\begin{align*}
\text{FORM} & \quad /\stackrel{(\ .+)}{\text{\$}}/ \quad \to \quad \text{FORM} & \quad \"n'\ t\" \\
\text{TO} & \quad 1 \quad \to \quad \text{FROM} & \quad 1 \quad \to \quad \text{FORM} & \quad \wedge 1n'\ t/ \\
\end{align*}
\]

- example: recombining split contracted forms
- rules extended with regular expression matches
- regex capture groups can be referred to in the output
- rules themselves described as feature structures, thus we can use re-entrancies
Chart Mapping – Examples

- light-weight named entity recognition

\[
\left[ \text{FORM} /^{\wedge}( [0-2]?[0-9]:[0-5][0-9])$/ \right] \rightarrow \left[ \text{CLASS \ clockTime} \right]
\]

- fixing broken tokenization

\[
\left[ \text{FORM} /^{\wedge}( .+: ([a-zA-Z0-9].*))$/ \right] \rightarrow \left[ \text{FORM \ 1/}, \  \text{FORM \ 2/} \right]
\]
Preprocessing has to provide the input chart as expected by the grammar.

This has to be ensured by specialized conversion routines without recourse to the grammar.

Changes to the grammar have to be reflected in these data adaptation routines.
Proposed Architecture (Simplified)

- proposal: token mapping performs certain preprocessing steps within the grammar

- advantages:
  - full control for the grammar writer, using the same formalism as for the grammar
  - makes assumptions by the grammar explicit
  - removes complexity from preprocessing
Hybrid Processing

- shaping the search space of the parser:
  - widening search space (e.g. unknown word handling)
  - narrowing search space (e.g. removing / postponing the processing of edges)

- constraints on the search space
  - hard: categorial conditions for introduction / removal of chart edges
  - soft: probabilistic disambiguation, prioritize parser's tasks on the agenda
Lexical Instantiation

• native and generic lexical entries (les)
• selection of appropriate generic lexical entries originally controlled by the parser (hard-coded)
• strategy:
  – map from part-of-speech tags to generic les
  – instantiate generic le for highest ranked pos tag where no native le is available
• disadvantage:
  – not flexible enough (e.g. no chain of responsibility)
  – partial lexical coverage: *We’ll bus to Paris.*
Lexical Instantiation

- proposal: try to instantiate all generic les for all tokens
- token feature structure is unified into a predefined path in the lexical entry
- selection of compatible tokens by constraints on the token feature structure
- example:

  genericname := \[\begin{align*}
  &\text{SYNSEM} & \text{noun\nobreakdash-nocomp\nobreakdash-synsem}, \\
  &\text{LOCAL.CAT.HEAD.MINORS.MIN} & \text{named\nobreakdash-rel}, \\
  &\text{TOKENS} & \left\langle \left[\text{POS.TAGS}\;\langle\;\text{NNP, \ldots}\rangle\right]\right\rangle
  \end{align*}\]
Lexical Filtering

- after lexical instantiation, native and generic les may be available in the same chart cell
- we can restrict lexical instantiation by positing constraints on the token feature structures
- but we might also want to prevent some lexical chart edges in certain contexts (set operations)
- proposal: lexical filtering phase
- same formalism as for token mapping: chart mapping rules with empty OUTPUT list
Proposed Architecture

- use feature structures to describe tokens
- chart mapping: resource-sensitive rewriting of feature structure items
- chart mapping on token fs
- generic instantiation driven by compatibility with token fs
- lexical filtering with chart mapping
Applications

- fine grained control over instantiation of generic lexical entries
- mapping external morphological information into the grammar's universe
- chart dependency filter (optimizing parsing performance)
- activate syntactic rules only for certain spans of the input (e.g., in hybrid grammar checking)
Conclusions

• versatile device for many applications
• external information is made accessible to the grammar
• pre-processing can be better controlled with grammar-specific means
• reduces the need for special code inside and outside the parser
• outlook: consolidation of our current parsers and grammars
Thank you!
Acknowledgements

- DELPH-IN community and beyond, especially Nuria Bertomeu, Ann Copestake, Remy Sanouillet, Ulrich Schäfer and Benjamin Waldron for numerous in-depth discussions

- funding:
  - ProFIT program of the German federal state of Berlin and the EFRE program of the EU (to the DFKI project Checkpoint)
  - the University of Oslo (through its scientific partnership with CSLI)