A LAF/GrAF based Encoding Scheme

for underspecified Representations of syntactic Annotations

Manuel Kountz[‡], Ulrich Heid[*], Kerstin Eckart[*]

Computational Linguistics: GRK 609[‡] / IMS[*], Universität Stuttgart, Germany

{kountzml|heid|eckartkn}@ims.uni-stuttgart.de

Ambiguities

Example: *Karl sieht nur Schrott in seinem Wagen.*
      Karl sees only scrap in(side) his car.

- **Structural alternatives**: Attachment of preposition *in*
  sieht - in, *Schrott - in*
  (Set aside attachment of the adverb *nur*)

- **Labelling alternatives**
  PP *in seinem Wagen* is attached to verb . . .
  - as argument: *K sieht S in A = K views A as S*
  - as adjunct: *K sieht S in A = K sees S and K is in A*

- **Interdependent alternatives**
 If PP is attached to noun (*Schrott - in*), in-PP must be adjunct

Requirements

- **Informational Efficiency**
  Encode *all information present* in the readings,
  and, at the same time, *reduce redundancy* in the encoding

- **Multifunctionality:** Usable for *several kinds of representations*
  e.g. based on various *syntactic* formalisms
  (dependency structures, TiGer);
  but also to encode *semantic* representations

- **Conformance to standards:**
  rely on LAF/GrAF, to ease *exchange*, *reuse*,
  and applicability of *standard tools & toolkits*
  (LAF: Ide & Romary '06, GrAF: Ide & Suderman '07)

Existing Approaches

for handling ambiguities
in corpus annotation

- **AND-OR-Trees**
  cf. Dörre '97, used e.g. in GrAF
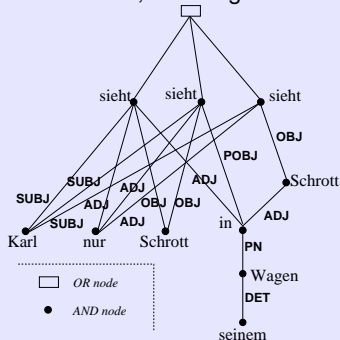
- intrinsically self-contained
- *but* many edges, additional nodes
- ⇒ much data to store
  (partially redundant)

- **Treebanks**
  manually disambiguated
  ⇒ treebank structures as such
  can't represent ambiguities

- **Separate tiers for readings**
  using multi-layer annotation:
  (e.g. NITE, Carletta et al. '04)
  ⇒ highly redundant
  ⇒ non-canonical use
  of representational means
  - How many layers?
  - What do they stand for?

Proposed Extension to LAF/GrAF

LAF/GrAF represents **graphs** as List of nodes, list of edges

- labelled with feature structures

```
10 <!-- nodes -->
11 <node id="v1">        <!-- sieht  -->
12     <f name="cat" value="V"/>
13 </node>
14 <node id="v2">        <!-- Karl   -->
15     <f name="cat" value="NE"/>
16 </node>
17 <node id="v3">        <!-- nur    -->
18     <f name="cat" value="ADV"/>
19 </node>
20 <node id="v4">        <!-- Schrott -->
21     <f name="cat" value="N"/>
22 </node>
23 <node id="v5">        <!-- in     -->
24     <f name="cat" value="PREP"/>
25 </node>
26 <node id="v6">        <!-- seinem -->
27     <f name="cat" value="PPOSS"/>
28 </node>
29 <node id="v7">        <!-- Wagen  -->
30     <f name="cat" value="N"/>
31 </node>
```

```
32 <!-- dependency edges -->
33 <edge type="dep-rel"
          id="e1" from="v2" to="v1">
34     <f name="role" value="SUBJ"/>
35 </edge>
36 <edge type="dep-rel"
          id="e2" from="v3" to="v1">
37     <f name="role" value="ADJ"/>
38 </edge>
39 <edge type="dep-rel"
          id="e3" from="v7" to="v5">
40     <f name="role" value="PN"/>
41 </edge>
42 <edge type="dep-rel"
          id="e4" from="v4" to="v1">
43     <f name="role" value="OBJ"/>
44 </edge>
45 <edge type="dep-rel"
          id="e5" from="v6" to="v7">
46     <f name="role" value="DET"/>
47 </edge>
```

```
48 <constraint-list>
49   <structural-constraint
50       id="c1" type="BelowAppropriate"
51       gov="v1" fragment="v5" />
52   <labelling-constraint
53       id="c2" type="LabelSet"
54       reference="c1"
55       labels="ADJ POBJ" />
56   <constraint-interdependency
57       id="c3" type="Enforces"
58       a="c2" aValue="POBJ"
59       b="c1" bValue="v1" />
60 </constraint-list>
```

We add to this data model a list of **constraints**

- **Structural constraints** control arrangement of *fragments*.
- **Labelling constraints** control labelling of nodes and edges.
- **Constraint inderdependencies**
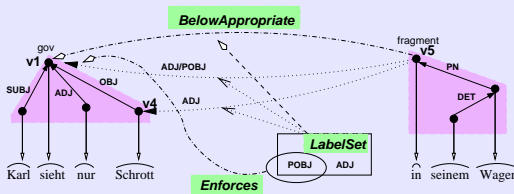
Adaptation

This schema can be adapted to particular representations by **defining new constraints**:

Precondition: Representation available for fragments
1. define constraint *name & type*
2. clarify *semantics*, i.e. how to construct appropriate structures

Application Examples

Underspecified representations of *syntactic structures*

- **Dependency structures** along with proposed encoding of non-underspecified dependency structures in LAF/GrAF

- **TiGer**-style representation of syntactic ambiguities
  (both in paper)

Encoding *semantic* USRs: e.g.

- **MRSs**: define qeq constraint & format for predicate logic formulas (structure ≈ trees). Semantics of qeq already specified. (Copestake et al. '05)

- **UDRSs**: devise leq constraint; need encoding for DRSs (structure: hierarchical)

Bibliography

SABINE BRANTS, STEFANIE DIPPER, SILVIA HANSEN, WOLFGANG LEZIUS & GEORGE SMITH (2002). The TIGER Treebank. Proceedings of the Workshop on Treebanks and Linguistic Theories.

ANN COPESTAKE, DAN FLICKINGER, CARL POLLARD & IVAN A. SAG (2005). Minimal Recursion Semantics: An Introduction. Research on Language & Computation 3.4. 281–332.

JOCHEN DÖRRE (1997). Efficient Construction of Underspecified Semantics under Massive Ambiguity. Meeting of the Association for Computational Linguistics. 386–393.

NANCY IDE & LAURENT ROMARY (2006). Representing Linguistic Corpora and Their Annotations. 5th International Conference on Language Resources and Evaluation, Genoa/Italy.

NANCY IDE & K. SUDERMAN (2007). GrAF: A Graph-based Format for Linguistic Annotations. Linguistic Annotation Workshop. Prague, Czech Republic: (28th—29th Jun 2007).

UWE REYLE (1993). Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction. Journal of Semantics 10.2. 123–179.

JEAN CARLETTA, D. MCKELVIE, A. ISARD, A. MENGEL, M. KLEIN & M. B. MØLLER (2004). A generic approach to software support for linguistic annotation using XML. In G. SAMPSON & D. MCCARTHY (Eds.) Corpus Linguistics: Readings in a widening Discipline.