Dependency Parsing for Medical Language and Concept Representation

Friedrich Steimann
Institut für Rechnergestützte Wissensverarbeitung
Universität Hannover
steimann@acm.org

Abstract: The theory of conceptual structures serves as a common basis for natural language processing and medical concept representation. We present a PROLOG-based formalization of dependency grammar that can accommodate conceptual structures in its dependency rules. First results indicate that this formalization provides an operational basis for the implementation of medical language parsers and for the design of medical concept representation languages.

Keywords: dependency grammar, medical concept representation language, conceptual graphs, natural language processing, feature unification

1. Introduction

Sowa noticed a similarity between existential graphs (an early form of conceptual graphs) and the graphs of dependency grammar ([20], p.8). Although he does not detail his observation, one can easily verify that the nodes of both graph types denote concepts. To illustrate this, the dependency tree of the sentence

"*There is an area of increased uptake at the right lumbosacral junction.*" (1)

and its conceptual graph are contrasted in Fig. 1a and b.

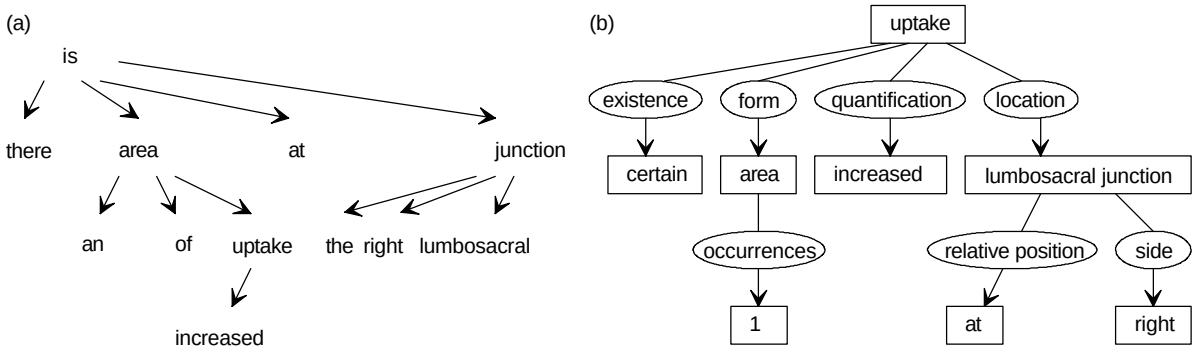

Figure 1: Two graphical representations of sentence (1)
a) its dependency tree
b) its conceptual graph

We deem the noted similarity worth further exploration. In particular, we present and discuss a PROLOG-based definition and implementaion of dependency grammar that can (a) parse sentences of a medical language, (b) generate meaningful conceptual graphs from a canonical basis, and (c) act as a conceptual parser facilitating the transformation from one form of medical content representation into the other. We report on our experience with the described framework and draw a practically relevant conclusion.

2. Parsing medical texts with dependency grammar

Dependency theory is an old linguistic theory. It is based on the assumption that every word of a sentence has slots to be filled by others, called its dependents. Dependency grammar was

first formalized by Tesnière [24] and soon after by Gaifman [10] and Hayes [11], but has since almost been forgotten. A recent treatise of dependency parsing is Fraser's dissertation [7].

According to our own formalization [21, 22] which is based on Hellwig's Dependency Unification Grammar (DUG) [12], a grammar designed to parse sentence (1) could have the following form:

```
s:> is.
is :> there, self, area, at, junction.
area :> an, self, of, uptake.
uptake :> increased, self.
junction :> the, right, lumbosacral, self.
there :> self. an :> self. of :> self. increased :> self.
at :> self. the :> self. right :> self. lumbosacral :> self.
```

The grammar is translated into Horn clauses (automatically inserting the necessary input and output variables and the *accept* predicates) by a simple preprocessor described in [22].

The first rule of the grammar is the start rule. Execution of this rule causes *is*, the root of the dependency structure, to be accepted from the input sentence, dividing it into a left and right remainder. Next, the rule for *is* is called, and the words in the body of this rule, the dependents of *is*, are accepted in left to right order. The special symbol *self* marks the position of the head of the rule among its dependents. The dependents left of *self* are accepted from the left remainder, and the words right of *self* are accepted from the right remainder. After having accepted a dependent, the parser recurses into the corresponding rule, effectively performing a depth-first search. For instance, after having accepted *there*, the parser branches to the rule for *there*. This rule terminates the recursion, because *there* does not have any dependents.

Application of the given grammar to sentence (1) produces the following parse tree in indented form:

```
is
   there
   area
      an
      of
      uptake
         increased
   at
   junction
      the
      right
      lumbosacral
```

which is the textual equivalent (neglecting word order) of Fig 1a.

The generative power of the grammar is greatly increased if its atoms, the words, are replaced by *feature structures*, a more flexible variant of first order terms that has widely been adopted as a standard in computational linguistics [15, 17]. We go one step further and employ *feature terms*, instances of record-like types organized in a subsumption-based taxonomy. The unification of feature terms restricts the terms to their greatest common subtype, and their features to the corresponding subtypes specified in the type definition ([1, 5, 13, 16, 18, 19, 21, 25], *cf* [14] for a discussion of PROLOG-based implementations of conceptual graphs). We note that feature term unification is equally well-suited to enforce grammatical agreement rules and the subsumption principles of concepts systems.

Rewritten, the grammar could have the following form:

```
% types
verb, noun[num=>int], adj, prep < word.
participle < adj.
be < verb.
area, uptake, junction < noun.
right, lumbosacral < adj.
```

```
increased < participle.
at < prep.
certain < existence.
right < side.
% dependency rules
s:> v(verb, existence).
v(be[number=>N], certain) :> v(there, .), self,
     v(noun[number=>N], finding), v(prep, .), v(noun, location).
v(area[number=>1], X) :> v(an, .), self, v(of, .), v(noun, X).
v(uptake, finding) :> v(adj, quantification), self.
v(junction, location) :> v(the, .), v(adj, side),
     v(adj, 'location modifier'), self.
v(there, .) :> self. v(an, .) :> self. v(of, .) :> self.
v(increased, quantification). v(at, .) :> self. v(the, .) :> self.
v(right, right) :> self. v(lumbosacral, 'location modifier') :> self.
```

where *v*/2 is to aggregate the lexical and the semantic facet of a word and the dot (.) in place of an argument serves as a place holder for an irrelevant term.

After adding the following statements:

```
uncertain < existence.
near < prep.
'lumbar spine' < noun.
v(be, uncertain) :> v(there, .), v(seem[number=>N], .), v(to, .), self,
     v(noun[number=>N], finding), v(prep, .), v(noun, location).
v('lumbar spine', location) :> v(the, .), self.
```

the grammar accepts

"*There seems to be increased uptake near the lumbar spine*." (2)

as well as sentence (1).

Note how anatomic knowledge is directly captured in the dependency rules: the lumbosacral junction can, while the lumbar spine cannot, be modified by a *side* attribute. Several other particularities of the given grammar are design issues, for example whether or not to model a noun and its modifier as a single term (as for *lumbar spine*) or as a term and its dependent (as for *lumbosacral junction*).

The described grammar and parsing algorithm have some favourable properties which are summarized as follows. Firstly, the depth of the search is always limited by the number of words remaining in the sentence. This is so because a rule can only be called after its head has been removed from the sentence. Secondly, the rule selection is driven by the input sentence, which greatly reduces the number of alternatives on backtracking. This is true for the very same reason: from the host of rules that make up a grammar only those will be tried whose heads conform with the word just accepted. Finally, because accepting a word splits the input sentence and subsequent searching for dependents is constrained to one remainder, the parser follows a divide-and-conquer strategy, which makes it very efficient.

3. Generating conceptual graphs with dependency rules

A dependency tree makes visible the grammatical structure of a sentence. Its content or meaning, however, is represented in a different form. According to the notation of conceptual structures [20], the *de facto* standard for medical concept representation, the content of sentence (1) could be represented by:

```
[uptake] -
   (existence) -> [certain]
   (form) -> [area] -
      (occurrences) -> [1],
   (quantification) -> [increased]
   (location) -> [lumbosacral junction] -
      (relative position) -> [at]
```

```
      (side) -> [right],,.
```

which is the textual equivalent of Fig. 1b. It corresponds to the following dependency tree:

```
uptake
  there is
  area of
    an
  increased
  lumbosacral junction
    at
    the right
```

Both structures are related by the following grammar:

```
uptake < finding.
certain < certainty.

s :> v(uptake, finding).

v(uptake,
  uptake[
    existence=>X1:certainty,
    form=>X2,
    quantification=>X3,
    location=>X4
  ]) :>
    v('there is', X1),
    v('area of', X2),
    v('increased', X3),
    self,
    v('lumbosacral junction', X4).

v('area of',
  area[
    occurences=>X
  ]) :>
    v(an, X),
    self.

v('lumbosacral junction',
  'lumbosacral junction'[
    'relative position'=>X1,
    side -> X2
  ]) :>
    v(at, X1),
    v(the right, X2),
    self.

v('there is', certain) :> self.
v(an, 1) :> self.
v(increased, increased) :> self.
v(at, at) :> self.
v('the right', right) :> self.
```

It is capable of parsing and generating sentence (1) and its conceptual graph in parallel.

From a linguist's point of view this grammar may appear unorthodox. However, given that the generated conceptual graph is an adequate representation of the medical content of sentence (1), it should none the less be acceptable. It seems that the main difference from a normal dependency grammar is that this version emphasizes the semantic rather than the syntactic dependencies. In fact, as the reader familiar with the theory of conceptual structures will have noticed, the grammar accommodates (a subset of) a canonical basis[1]. Its PROLOG implementation is a conceptual parser.

[1] A *canonical basis* [Sowa 84] is a collection of conceptual graphs encoding the elementary relationships of a domain. These graphs can be combined into more complex ones by applying *canonical*

Although our example is very simple and without modifications will not extend to more complex situations, it shows that dependency parsing can act as an operational framework for canonical graphs and the medical concept representation languages based thereon. In particular, it shows that:

- a restricted subset of canonical graphs can be modelled as order-sorted feature types[2];
- the canonical formation rules on these graphs are covered by feature term unification (*cf* [26]); and
- the generation of the closure of a canonical basis is a special application of dependency grammar.

4. Preliminary results and discussion

We have used the described formalization of dependency grammar to implement the Canon Group's core merged model of radiology findings [6, 9]. Attempts to generate all canonical graphs derivable from the model showed that it contained several design flaws [23]. In particular, in its current form it embodies hidden sources of infinite recursion. For example, the subtype relations:

```
pleural_effusion < rad_finding.
pleural_effusion < effusion.
effusion < observation.
```

and the canonical graph

```
[rad_finding] -
     (has_observation) -> [observation]
     ...
```

allow the canonical derivation:

```
[pleural_effusion] -
  (has_obervation) -> [pleural_effusion] -
    (has_observation) -> [pleural_effusion] -
      etc. ad infinitum
```

which is clearly not a desired expression. However, bugs like this, which result from a natural desire to be general, are very difficult to detect in the paper form of a specification.

We have arrived at similar results with our own attempts to formalize excerpts of Latin (disregarding word order) and toy grammars of English and German [21, 22]. In particular, we found that any grammar we thought to be reasonably powerful and compact was in fact highly underconstrained. This is so because in order to be compact, a dependency grammar has to make generous use of unbound variables (wildcards). Many and particularly insufficiently restricted variables, however, entail the generation of a large number of meaningless and ungrammatical sentences.

We conclude that the design of a dependency grammar (and a canonical basis for that matter) will always be a trade-off between two extremes: a few rules with high generative power, appropriately constrained by a well-devised framework of subsumption principles (cf. [3] for a medically oriented treatise); and the brute force of a lexicon-based grammar, more or less listing every possible usage of each word of the language (or concept of the domain). A recent tendency towards lexical orientation across all grammar formalisms suggests that practical implementations will be closer to the latter extreme. Dependency grammar and conceptual parsing based thereon should therefore be a convenient operational basis. One has to be prepared, however, to face as many as 10,000 words even in a limited medical subdomain [2]. We

formation rules. Together with a careful design of the canonical basis these rules guarantee that the derived graphs are meaningful within the given domain. Canonical bases specify languages; they have been chosen as the formal basis of several medical concept representation language projects (eg [2, 4 , 8]).

[2] The restrictions depend on the employed feature types.

conjecture that the number of concepts in such a domain is not much smaller; even with the most sophisticated tools of today, the specification of a realistic grammar will remain hard work.

5. Conclusion

The specification of medical language and concept representation is difficult and, without the aid of formal evaluation tools, error-prone. We have presented a uniform framework allowing the implementation and execution of dependency rules and canonical graphs on PROLOG machines. This framework greatly facilitates the parsing and generation of natural language sentences and conceptual structures and should therefore prove a useful design tool.

6. Acknowledgements

The author's work has in part been supported by the Universitätsgesellschaft Hildesheim e.V.

7. References

[1] H. Aït-Kaci and R. Nasr, LOGIN: A logic programming language with built-in inheritance, *The Journal of Logic Programming* 3 (1986) 186–215.

[2] R.H. Baud, A.M. Rassinoux, J.R. Scherrer, Natural language processing and semantic representation of medical texts, *Meth Inform Med* 31 (1994) 117–125.

[3] J. Bernauer, Subsumption principles underlying medical concept systems and their formal reconstruction, in: *Proc. of the 18th SCAMC* (Am Med Informatics Assoc 1994), 140–144.

[4] K.E. Campbell, A.K. Das, M.A. Musen, A logical foundation for representation of clinical data, *J Am Med Informatics Assoc* 1 (1994) 218–232.

[5] B. Carpenter, *The Logic of Typed Feature Structures* (Cambridge University Press, Cambridge 1992).

[6] D.A. Evans, J.J. Cimino, W.R. Hersh, S.M. Huff, D.G. Bell, Toward a medical-concept representation language, *J Am Med Informatics Assoc* 1 (1994) 207–217.

[7] N. Fraser, Dependency Parsing, PhD Thesis, University College London, London, 1993.

[8] C. Friedman, J.J. Cimino, S.B. Johnson, A schema for representing medical language applied to clinical radiology, *J Am Med Informatics Assoc* 1 (1994) 233–248.

[9] C. Friedman, S.M. Huff, W.R. Hersh, E. Pattison-Gordon, J.J. Cimino. The Canon Group's effort: Working toward a merged model. *J Am Med Informatics Assoc* 2 (1995) 4–18.

[10] H. Gaifman, Dependency systems and phrase-structure systems, *Information and Control* 8 (1965) 304–337.

[11] D.G. Hays, Dependency theory: a formalism and some observations, *Language* 40 (1964) 511–525.

[12] P. Hellwig, Dependency Unification Grammar, in: *Proceedings of the 11th International Conference in Computational Linguistics* (COLING´86) (Bonn, August 1986) 195–198.

[13] O. Herzog, C.R. Rollinger, (eds), *Text understanding in LILOG: integrating computational linguistics and artificial intelligence*, Lecture Notes in Artificial Intelligence 546 (Springer Verlag, Berlin, 1991).

[14] A. Kabbaj, C. Frasson, M. Kaltenbach, J.-Y. Djamen, A conceptual and contextual object-oriented logic programming: the Prolog++ language, in: *Conceptual Structures: Current Practices*, Proc. 2nd Int. Conf. Conceptual Structures (Springer-Verlag 1994) 251–274.

[15] K. Knight, Unification: a multidisciplinary survey, *ACM Computing Surveys* 21 (1989) 105–113.
[16] M. Schmitt-Schauss, *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Lecture Notes on Artificial Intelligence 395 (Springer Verlag, 1989).
[17] S.M. Shieber, *An Introduction to Unification-Based Approaches to Grammar*, CLSI Lecture Notes No. 4 (Stanford 1986).
[18] G. Smolka and H. Aït-Kaci, Inheritace hierachies: semantics and unification, *J. Symbolic Computation* 7 (1989) 343–370.
[19] G. Smolka, Feature-constraint logics for unification grammars, *J. Logic Programming* 12 (1992) 51–87.
[20] J.F. Sowa, *Conceptual Structures* (Addison-Wesley, Reading, MA, 1984).
[21] F. Steimann, Ordnungssortierte feature-Logik und Dependenzgrammatiken in der Computerlinguistik, Diplomarbeit Universität Karlsruhe, Fakultät für Informatik (Universität Karlsruhe 1991).
[22] F. Steimann, C. Brzoska, Dependency unification grammar for PROLOG, *Computational Linguistics* 21 (1995) 95–102.
[23] F. Steimann, Canonical conceptual graphs: problems and mines of solutions (Universität Hildesheim 1996).
[24] L. Tesnière, *Eléments de Syntaxe Structurale* (Librairie C. Klincksieck, Paris, 1959).
[25] U. Waldmann, Unification in order-sorted signatures, Forschungsbericht Nr. 298 (Universität Dortmund, 1989).
[26] M. Willems, Projection and unification for conceptual graphs, in: *Conceptual Structures: Applicatios, Implementation and Theory*, Proc. 3rd Int. Conf. Conceptual Structures (Springer-Verlag 1995).