Canonical Conceptual Graphs: Problems and Mines of Solutions

Friedrich Steimann
Department of Medical Informatics
University of Hildesheim, Germany

With the first version of their core merged model [Friedman] the Canon Group [Evans] delivered a "tangible product" that, while being general enough to serve a wide range of applications, is specific enough to allow formal evaluation. The following collection of problems is the result of such an evaluation; together with the proposed solutions it is to be understood as a modest contribution towards a more widely accepted standard of medical knowledge sharing.

1. Problems

Problem #1: multiple classification of concepts

The core merged model includes the following subtype relationships:

```
effusion < observation.
pleural_effusion < {rad_finding, effusion}.
```

where $a < \{b, c\}$ means $a < b$, $a < c$. The canonical conceptual graph (CCG)

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

allows the *canonical derivation* [Sowa] of

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

which is clearly not a desired expression.

The problem arises from the fact that *pleural_effusion* plays different roles (*role types* [Sowa], not to be confused with the roles of KL-ONE-like languages) in the different contexts in which it is being used: as a *rad_finding* it has different conceptual relations than as an *observation*. However, with the given type hierarchy the different roles of *pleural_effusion* cannot be differentiated.

A pragmatic solution to the problem is to change the type hierarchy to

```
pleural_effusion < effusion.
rad_finding_pleural_effusion < {rad_finding, pleural_effusion}.
observation_pleural_effusion < {observation, pleural_effusion}.
```

and to replace the types in the above CCG with the appropriate new subtypes. Intuitively, however, the role of a concept is a different property than its generic type, so that combining both in one hierarchy may appear awkward; another solution is to separate role and concept and maintain different taxonomies, as does the GALEN approach [Rector].

Problem #2: specialization of CCGs

The specialization of concepts specified in the type hierarchy is complemented by CCGs restricting the ranges of the subtype's related concepts. The CCG of such a specialized concept is given by

```
[pleural_effusion] -
      (has_observation) -> [effusion]
      (has_location) -> [pleural_space].
```

According to the two roles *pleural_effusion* can play (see above), two cases must be distinguished:

a) specialization of the CCG for *rad_finding*

In this case the problem is the same as #1, although it is questionable why the observation of the radiology finding *pleural_effusion* should be the (more general) observation *effusion* (allowing other subtypes of *effusion* to take the place) and not the observation *pleural_effusion*; formally, stating the fact that a pleural effusion is located in the pleura is not redundant (*cf*. Problem #4 below).

b) the CCG for the specialized observation *pleural_effusion*

Surely the pleural effusion is an effusion (the characteristic property of which is that it is located in the pleura); yet this *is-a* relationship is coded in the type hierarchy so that

```
[observation_pleural_effusion] -
      (has_location) -> [pleura].
```

suffices.

Problem #3: unconstrained recursion

Although the generative power of CCGs such as

```
[body_location] -
      (has_location) -> [body_location:{*}]
      (has_location_qualifier) -> [location_qualifier: {*}]
      ...
```

where

```
body_location < location_qualifier.
```

is intriguing, they are somewhat underconstrained, as they allow it to construct arbitrarily deeply nested structures without adapting the range restrictions on the related concepts with the depth. In reality, the qualification of body locations is much more constrained, and because these constraints are highly complex, it seems that they cannot be covered by a relatively small number of generic CCGs. In a limited domain such as radiology, explicit enumeration of all 'legal' body location/location qualifier combinations may be considered a viable alternative. For example, the *location_qualifier* (or *has_location*?) of *finger* could be restricted to *hand*.

Problem #4: intuitive vs. formal semantics

The intuitive meaning of the CCG

```
[cardiomegaly] -
      (has_observation) -> [heart]
      (has_property) -> [enlarged].
```

is something like "with cardiomegaly the heart is enlarged", where *enlarged* is a property of the *heart* under observation. However, the correct (formal) semantics is "cardiomegaly is a radiology finding (follows from the type hierarchy) that is observed of the *heart*, yielding that it (the *cardiomegaly*) is *enlarged*" [Sowa]. Friedman et al. maintain that in a CCG of the above kind a core concept (in this case the *heart*) is further qualified by the other relations of the CCG (here: *enlarged*) [Friedman]. Formally, however, it is unclear which relation should qualify which; rather, each relation qualifies the concept in the head of the CCG.

That cardiomegaly is the radiology finding associated with an enlarged heart is expressed by the nested CCG

```
[cardiomegaly] -
      (has_observation) ->
            [heart]-
                  (has_property) -> [enlarged].
```

and the following construct:

```
cardiomegaly < observation_of_heart.

[cardiomegaly] -
      (has_property) -> [enlarged].
```

2. Mines of Solutions

The object/relationship dichotomy is ubiquitous in modelling. Indeed, CCGs are similar to entity-relationship diagrams, a notation heavily employed in database and object-oriented software analysis and design. Both fields offer a wealth of know-how, some of which is also relevant to the design of a medical concept representation language; for example, the treatment of roles (*cf*. Problem #1) is addressed in [Embley, Wirfs-Brock].

There is a striking similarity between CCGs and what have been termed *feature structures*, *feature terms* and *feature types*. Feature structures stem from computational linguistics and are essentially collections of (possibly nested) label-value pairs. Feature structures are extensively employed in unification-based grammars [Shieber], chiefly to record syntactic and semantic information pertaining to the components of a sentence. They have been adopted by the logic programming community, generalizing first-order terms to feature terms the arguments of which are labeled, arbitrarily ordered, and variable in number [Knight, Carpenter].

The interpretation of feature terms as record-like data types (called feature types), their arrangement in an inheritance hierarchy, and the adding of set-theoretic semantics (presented in [Smolka]) provide the theoretical basis of the logic programming language LOGIN [Ait-Kaci], a feature-based extension of PROLOG. The mapping of the CCGs presented in [Friedman] to the feature types of LOGIN is nearly one-to-one; the implementation of CCGs as logic programs would give them an immediate operational semantics that should aid the creators of CCGs in detecting potential design flaws. Other logic programming efforts such as IBM's LILOG project [Herzog] also provide rich sources of experience.

Many of the problems encountered in designing the core merged model's CCGs are also encountered in designing grammar rules for radiology reports uttered in natural language. Among the many formalisms the natural language processing community has offered, the recent rediscovery of the lexicon in the form of dependency grammar [Fraser, Steimann] should be most influential: rather than striving to find the generic patterns behind complex expressions, the individual word (concept) and its syntacto-semantic dependencies are considered the root of all structure. For example, rather than covering the expression *finger of hand* with the generic CCG for the concept *body_location*, one could provide a CCG naming *hand* as a possible related *body_location* of *finger* (*cf*. Problem #3 above). Again, IBM's LILOG project [Herzog] should prove as a good reference.

3. Conclusion

Medical concept representation languages such as the one put forward by the Canon Group inherit from both formal and natural language. This duality is a blessing and a curse: while confronting us with the intricacies of human thought and its expression, it sets the focus on regularity and formalism, leaving the venture susceptible to experience from all areas of knowledge representation and modelling. It is a problem at the intersection of many disciplines; collaboration should thus be broad, as should be the interest in its outcome.

4. References

1. Evans DA, Cimino JJ, Hersh WR, Huff SM, Bell DG. Toward a medical-concept representation language. J Am Med Informatics Assoc. 1994;1:207–217.
2. Friedman C, Huff SM, Hersh WR, Pattison-Gordon E, Cimino JJ. The Canon Group's effort: Working toward a merged model. J Am Med Informatics Assoc. 1995;2:4–18.
3. Sowa JF. Conceptual Structures. Reading, MA: Addison-Wesley, 1984.
4. Rector AL, Glowinski AJ, Nowlan WA, Rossi-Mori, A. Medical-concept models and medical records: an approach based on GALEN and PEN&PAD. J Am Med Informatics Assoc. 1995;2:19–35.
5. Embley DW, Kurtz BD, Woodfield SN. Object-Oriented Systems Analysis: A Model-Driven Approach. Englewood Cliffs, NJ: Prentice-Hall, 1992
6. Wirfs-Brock RJ, Johnson RE. Surveying current research in object-oriented design. Comm. of the ACM 1990;33(9):104–124.
7. Shieber SM. An Introduction to Unification-Based Approaches to Grammar. Stanford University: CSLI Lecture Notes 4, 1986.
8. Knight K. Unification: a multidisciplinary survey. ACM Computing Surveys 1989;21:93–124
9. Carpenter B. The Logic of Typed Feature Structures. Cambridge: Cambridge University Press, 1992.
10. Ait-Kaci H, Nasr, R. Login: a logical programming language with built-in inheritance. Journal of Logic Programming 1986;3:187–215.
11. Smolka G, Ait-Kaci H. Inheritance hierarchies: semantics and unification. Journal of Symbolic Computation 1989;7:343–370
12. Herzog O, Rollinger CR (eds). Text Understanding in LILOG: Integrating Computational Linguistics and Artificial Intelligence. Lecture Notes in Artificial Intelligence 546. Berlin: Springer Verlag, 1991
13. Fraser N. Dependency Parsing. PhD Thesis. University College London, 1993
14. Steimann F, Brzoska C. Dependency unification grammar for Prolog. Computational Linguistics 1995;21:95–102