An Automated Conceptual Catalogue for the Enterprise

Richard Hill and Simon Polovina

Communication & Computing Research Centre
Faculty of Arts, Computing, Engineering & Sciences
Sheffield Hallam University, S1 1WB, UK
{r.hill, s.polovina}@shu.ac.uk

Abstract. This work furthers the work in Transaction Agent Modelling (TrAM) by merging its conceptual catalogue based on the REA (Resources-Events-Agents) accounting model with Sowa's 1984 conceptual catalogue. The merged catalogue features in a preliminary implementation of TrAM using the Amine software tool, which also offers the model-checking support that is core to TrAM. This automated process demonstrates how Conceptual Graphs (CG) might lucidly interrelate the divergent conceptual catalogues of the myriad domains in which contemporary enterprise systems operate.

1 Introduction

The Transaction Agent Modelling (TrAM) approach has been developed to demonstrate the advantages and wide applicability of Conceptual Graphs (CG) as a tool for capturing and representing the complex facets of enterprise systems [5], [6]. TrAM exploits the formal underpinnings of CG notation and the use of Polovina's Economic Accounting, a transactions-oriented approach based upon Geerts [3] and McCarthy's [8], [7] respected REA (Resources-Events-Agents) accounting model. We have merged TrAM's conceptual catalogue that is based specifically on transactions with a generic, non-transactions oriented catalogue to establish the true scope of TrAM's potential contribution. To achieve these aims we selected Sowa's conceptual catalogue in Appendix B of his original text [10]. This catalogue, produced as CG, had no evident basis in transactions. Sowa's catalogue through its simple but expressive canonical examples thus provides a far-reaching test of the generality of TrAM's conceptual basis. We further mandated that this merging operation is achieved through an automated software tool rather than as a 'pen and paper' exercise. Use of such a tool provides the automated checking that can be easily overlooked by a manual process, whilst additionally paving the way for TrAM's implementation as an integrated software component in contemporary enterprise applications.

2 Developing a Catalogue

Sowa published a 'Conceptual Catalog' [10](pp405-424), and together with Polovina's work it was deemed apt to investigate the extent to which TrAM could tolerate a catalogue that has no obvious orientation towards event accounting nor indeed, transactions. Of course it would be anticipated that those types and relations that exist at the highest levels of a hierarchy will accommodate most domains, but the extent of the commonality between Sowa's and Polovina's catalogue was notable. For brevity only some of these relations will be described further below.

2.1 Conceptual Relations

To begin with, Sowa (1984) [10] relation `part(x,y)` is:

```
[Entity:x_source]-Relation->[Entity:y_target]
```

We can simply extend this to relation `part(x,y)` is:

```
[Universal:x_source]-Relation->[Universal:y_target]
```

This is because it is possible for a part to relate Universal types (e.g. an act can be a part of another act as evidenced by a Transaction which can be commonly part of a bigger Transaction for instance). Indeed Sowa recognises that[10](pp405):

> "For any particular application, these lists can serve as a starter set that the reader may extend or modify as appropriate."

Moving on, in Sowa, relation `source(x,y)` is:

```
[Act:x_source]-Relation->[Entity:y_target]
```

In TrAM, relation `source(x,y)` is:

```
[Economic_Resource:x_source]<-source-[Act]-agnt->[Agent:y_target]
```

(where `Economic_Resource < Entity`)
Continuing, in Sowa, relation `destination(x,y)` is:

```
[Act:x_source]-Relation->[Entity:y_target]
```

In TrAM relation `destination(x,y)` is:

```
[Economic_Resource:x_source]<-destination
   -[Act]-agnt->[Agent:y_target]
```

In passing we have used synonyms for:

1. type `Economic_Resource` is `Economic_Entity`,
2. relation `source` is `srce`
3. relation `destination` is `dest`

This is purely for convenience (e.g. Sowa refers to 'source' as 'srce' and 'destination' as 'dest'; in TrAM, 'Economic_Resource' is a sub-type of entity). In Sowa, relation `agnt(x,y)` is:

```
[Act:x_source]<-Relation-[Agent]-Relation->[Animate:y_target]
```

In TrAM relation `event_subject(x,y)` is:

```
[Economic_Event:x_source]-obj->[Economic_Resource:y_target]
```

3 An Exemplar Case Study

Using the modified conceptual catalogue described in Section 2.1, we shall now explicate the process of developing models and rules of inference for a case study in the community healthcare domain. All of the graphs were produced within Amine[1] and therefore the notation used conforms to the relevant syntax. From prior work[9] we can represent the healthcare scenario as follows:

```
[Care #0] -
       -requester->[Elderly_Person],
       -deliverer->[Care_Provider],
       -manager->[Local_Authority]
```

For convenience the generic TM graph is described below:

```
[Act:super]-
 -part->[Economic_Event:a]-
  -event_subject->[Economic_Resource:x]-
         -source->[Inside_Agent:i],
          -destination->[Outside_Agent:o];;
 -part->[Economic_Event:b]-
  -event_subject->[Economic_Resource:y]-
         -source->[Outside_Agent:o],
         -destination->[Inside_Agent:i]
```

Specialising the generic TM graph with the community healthcare scenario we derive the `[ComCare_Transaction]` graph:

```
[Transaction:super]-
        -part->[Raise_Debtor:a]-
                -event_subject->[Money:x]-
                        -source->[Purchase_Agent:i],
                        -destination->[Care_Provider:o];;
        -part->[Sale:b]-
                -event_subject->[Care:y]-
```

```
                         -source->[Care_Provider:o],
                         -destination->[Purchase_Agent:i]
```

The graph above is now specialised further to account for `requester`, `provider` and `manager` relations from the original use cases[9]:

```
[Transaction:super]-
  -part->[Raise_Debtor:a]-
   -event_subject->[Money:x]-
    -source->[Purchase_Agent:i],
     -destination->[Care_Provider:o],
        -requester->[Elderly_Person:e]-characteristic->[Asset]-
        -total_value->[UKP:less_than_threshold],
    -manager->[Local_Authority:l];;
    -part->[Sale:b]-
     -event_subject->[Care:y]-
      -source->[Care_Provider:o],
      -destination->[Purchase_Agent:i],
      -provider->[Care_Provider:o]
```

3.1 Building the Rules

Prior to this, the models which had been developed exploited the expressivity of Peirce cuts for graph visualisation. We have elected to pursue the development of an implementation, and as such we shall now consider the construction of rules without Peirce logic. In each case we describe the `Antecedant` and `Consequence` for each rule. Rule 1 represents an aspect of the payment scenario whereby there is a liability relationship between the `[Local_Authority]` and the `[Purchase_Agent]`, as assets of the `[Elderly_Person]` are deemed to be less than a particular threshold (set by UK Government policy). Therefore, Rule 1: '*less_than_threshold*' comprises:

```
Antecedent
[Care:y]-
  -requester->[Elderly_Person:e]-characteristic->[Asset]-
    -total_value->[UKP:less_than_threshold],
  -manager->[Local_Authority:l],
  -destination->[Purchase_Agent:i]
```

```
Consequent
[Local_Authority:l]-liability->[Purchase_Agent:i]
```

For the alternate case, the `[Elderly_Person]` is judged to possess assets that are above a particular threshold, thus has the liability to the `[Purchase_Agent]`. Rule 2: '*above_threshold*' is thus:

```
Antecedent
[Care:y]-
        -requester->[Elderly_Person:e]-characteristic->[Asset]
    -total_value->[UKP:above_threshold],
        -manager->[Local_Authority:l],
        -destination->[Purchase_Agent:i]
```

```
Consequent
[Elderly_Person:e]-liability->[Purchase_Agent:i]
```

This leaves a rather clumsy third case whereby the assets are 'at `threshold`' (i.e. actually at the threshold itself). Really there should only be two ranges, namely below or at or above the threshold. In TrAM the thresholds can be shown as ranges in the form of measures i.e. using the @<*referent*>[9]. The 'hard-codings' of the threshold calculation in Amine is a workaround as there is no 'CG Actor'[4] representation within this tool, unlike CharGer[2] which does feature the CG Actor as its core means of processing CG. The inclusion of CG Actors would be particularly useful since the calculation of apportioning the extent of the payment liability can then be calculated or determined from data look-ups to provide a value within these ranges. Hence we have identified an immediately valuable area of interoperability between CG tools.

4 Results

Noting our comments above we now consider the outcomes of this processing, beginning with Rule 1: The `Elderly Person` possesses assets that are judged to be '`less_than_threshold`':

```
CG1
[Transaction:super]-
        -part->[Raise_Debtor:a]-
                -event_subject->[Money:x]-
                        -source->[Purchase_Agent:i],
                        -destination->[Care_Provider:o],
                        -requester->[Elderly_Person:e]
        -characteristic ->[Asset]-total_value->
          [UKP:less_than_threshold],
                        -manager->[Local_Authority:l];;
        -part->[Sale:b]-
                -event_subject->[Care:y]-
                        -source->[Care_Provider:o],
                        -destination->[Purchase_Agent:i],
                        -provider->[Care_Provider:o]
```

The second CG:

```
CG2
[Care:y]-
        -requester->[Elderly_Person:e]-characteristic-
    ->[Asset]-total_value->
[UKP:less_than_threshold],
        -manager->[Local_Authority:l],
        -destination->[Purchase_Agent:i]
```

If we project CG2 into CG1, the following graph, CG3 is asserted:

```
CG3
[Local_Authority:l]-liability->[Purchase_Agent:i]
```

The Maximal Join Result is in Amine output:

```
[Care #1] -
       -source->[Care_Provider :o]<-destination-[Money :x]-
                -source->[Purchase_Agent #0]-
                <-destination-[Care #1],
                <-liability-[Local_Authority :l]
       //the added consequent
      <-manager-[Care #1];
       <-event_subject-[Raise_Debtor :a]<-part
         -[Transaction: super]-part->[Sale :b]
           -event_subject-> [Care #1];
       -requester->[Elderly_Person :e]-characteristic->
         [Asset]-total_value->[UKP:less_than_threshold]
```

Let us now consider another rule, Rule 2: The `Elderly Person` possesses assets that are judged to be 'above_threshold':
`CG1`: Except for `[UKP:less_than_threshold]` which would be `[UKP:above_threshold]` instead, `CG1` will be the same as the previous `CG1`

```
CG2
[Care :y] -
       -requester->[Elderly_Person :a]-characteristic->[Asset]
         -total_value->[UKP :above_threshold],
       -manager->[Local_Authority :b],
       -destination->[Purchase_Agent :c]
```

Again, if we project CG2 into CG1, the following graph, CG3 is asserted:

```
CG3
[Elderly_Person :a]-liability->[Purchase_Agent :c]
```

The Maximal Join Result is in Amine output:

```
[Care #1] -
       -source->[Care_Provider :o]<-destination-[Money :x] -
                -source->[Purchase_Agent #0] -
                        <-destination-[Care #1],
                <-liability-[Elderly_Person :a] -
       //the added consequent
                        -characteristic->[Asset]-total_value->
        [UKP :above_threshold],
                        <-requester-[Care #1];;
        <-event_subject-[Raise_Debtor :a]<-part
          -[Transaction :super]
           -part->[Sale :b]-event_subject->[Care #1];
       -manager->[Local_Authority :b]
```

5 Discussion

The use of Sowa's 1984 catalogue[10] has proved straightforward, and appears to have been a sound base upon which we can enrich the process with a more transaction-focused vocabulary. Whilst the TrAM approach has been tested in a variety of domains, the work in the community healthcare domain has illustrated three specific points:

1. the case study requires CG Actors in order to represent the inherent calculations and data lookups in real-world scenarios more accurately;
2. if the visual expressivity of Peirce logic is desired then it will be necessary to translate Peirce cuts into a form that enables graph-joining and projection to take place;
3. a single tool does not yet exist to support this process. Efforts to improve the interoperability between tools would assist in this respect, and would be a valuable contribution to the conceptual structures community.

In the absence of a suitable Peirce logic theorem prover we have elected to move forward with tools that support specialisation and projection. This is the most practical way forward if an implementable system is to be realised. It should be noted that this does not compromise the TrAM approach unduly; the TM is proven to be based upon principled foundations and we have established the necessary proofs using Peirce logic. It is evident that we need to assess the impact of converting Peirce logic for requirements capture, into graphs without cuts, and to evaluate any adverse affects upon the process as a whole.

6 Acknowledgements

This work has been assisted by the generous efforts of Ulrik Petersen and Professor Adil Kabbaj. The project is also in receipt of an AgentCities Deployment Grant from the European Union AgentCities.rtd Project (IST-2000-28385).

References

1. Amine Platform, http://amine-platform.sourceforge.net/
2. Delugach, H., (2006). CharGer - Conceptual Graph Editor, Accessed 30th November 2007, http://sourceforge.net/projects/CharGer/
3. Geerts, G. L. and McCarthy. W. E. (1991). "Database Accounting Systems", in Information Technology Perspectives in Accounting: an Integrated Approach, Chapman and Hall, 159-183.
4. Harper, Lois W., and Delugach, Harry S. (2004). "Using Conceptual Graphs to Represent Agent Semantic Constituents, in Conceptual Structures at Work: Proc. 12th Intl. Conf. on Conceptual Structures (ICCS 2004), Lecture Notes in Artificial Intelligence, LNAI vol. 3127, Springer-Verlag, Heidelberg, K. E. Wolff, H. D. Pfeiffer and H.S. Delugach, eds., July 2004, pp. 325-338.
5. Hill, R., Polovina, S., Shadija, D., (2006) "Transaction Agent Modelling: From Experts to Concepts to Multi-Agent Systems", Proceedings of 14th International Conference on Conceptual Structures (ICCS '06): Conceptual Structures: Inspiration and Application, July 16-21, 2006, Aalborg, Denmark. Schärfe, Henrik Hitzler, Pascal, Øhrstrøm, Peter (Eds.), Lecture Notes in Artificial Intelligence (LNAI 4068), Springer (ISBN 978-3-540-35893-0, ISSN 0302-9743), 247-259.
6. Hill, R., (2006). "Capturing and Specifying Multi-Agent Systems for the Management of Community Healthcare" in Yoshida, H., Jain, A., Ichalkaranje, A., Jain, L.C., Ichalkaranje, N., editors, "Advanced Computational Intelligence Paradigms in Healthcare - 1", Chapter 6, 127-164, Studies in Computational Intelligence, 48, Springer, Berlin, ISBN: 978-3-540-47523-1.
7. McCarthy, W. E., (1982). "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment", The Accounting Review, 554-578.
8. McCarthy, W. E., (1979). "An Entity-Relationship View of Accounting Models", The Accounting Review, 667-686.
9. Polovina, S., Hill, R., (2005). "Enhancing the Initial Requirements Capture of Multi-Agent Systems through Conceptual Graph", Proceedings of 13th International Conference on Conceptual Structures (ICCS '05): Conceptual Structures: Common Semantics for Sharing Knowledge, July 18-22, 2005, Kassel, Germany; Dau, Frithjof; Mugnier, Marie-Laure; Stumme, Gerd (Eds.); Lecture Notes in Artificial Intelligence (LNAI 3596), Springer (ISBN 978-3-540-27783-5, ISSN 0302-9743), 439-452.
10. Sowa, J. F., (1984). "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley.