INTERNATIONAL WORKSHOP

The First Conceptual Structures – Learning, Teaching and Assessment Workshop

CS-LTA2010

held in conjunction with ICCS-2010
the 18th International Conference on Conceptual Structures

PROCEEDINGS

Edited by

Simon Polovina, Simon Andrews, Richard Hill,
Henrik Scharfe, Peter Øhrstrøm

Kuching, Sarawak, Malaysia
26 July 2010

Knowledge Technology Week 2010 Sponsors

TABLE OF CONTENTS
CS-LTA2010

PREFACE

The 18th International Conference on Conceptual Structures (ICCS 2010) is the latest in a series of annual conferences that have been held in Europe, Australia, and North America since 1993. Conceptual structures (CS) recognise that organisations work with concepts; machines like structures. The objective driving research and development on CS is to harmonise the uniquely human ways of apprehending the world, with the power of computational information management and reasoning. Thus CS harmonises the creativity of humans with the productivity of computers. ICCS brings together some of the world's best minds in information and computer sciences, humanities and social sciences to explore novel ways that information technologies can leverage tangible business or social benefits.

The activity of the field is witnessed by two recently published books ("Conceptual Structures in Practice", ed. by Hitzler and Schärfe and "Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs" by Chein and Mugnier) as well as by an ISO standard ("Common Logic", ISO/IEC 24707) which originated in this community. This activity is further demonstrated by the emergence of industrial applications in this area (e.g. Sonetto, and Vivomind).

CS has therefore featured in the curricula in educational institutions (e.g. universities) around the world. Thus the Learning, Teaching and Assessment (LTA) of CS has become a significant topic of interest in its own right. The First Conceptual Structures - Learning, Teaching and Assessment (CS-LTA) workshop at ICCS 2010 recognises this work, and provides a forum for it to be disseminated as yet another key activity in CS.

We are certain that you will find the papers contained in this ICCS 2010 CS-LTA workshop volume a most illuminating read. They evidence the rich diversity that Conceptual Graphs (CG), Formal Concept Analysis (FCA), Peirce, and CS bring to educational curricula.

Simon Polovina, on behalf of the Organisers of CS-LTA 2010

ACKNOWLEDGEMENTS

This workshop is kindly supported by:

- Centre for Excellence in Teaching and Learning (CETL), Sheffield Hallam University, UK
- Centre for Promotion of Learner Autonomy (CPLA), Sheffield Hallam University, UK
- Communication and Computing Research Centre (CCRC), Sheffield Hallam University, UK
- Distributed and Intelligent Systems Research Group, University of Derby, UK
- Department of Communication and Psychology, Aalborg University
- ICCS 2010
- MIMOS Berhad

WORKSHOP ORGANISERS

- Simon Polovina & Simon Andrews
  Department of Computing / Communications and Computing Research Centre (CCRC), Conceptual Structures Research Group
  Sheffield Hallam University, UK

- Richard Hill
  Distributed and Intelligent Systems Research Group, School of Computing
  University of Derby, UK

- Henrik Scharfe & Peter Øhrstrøm
  Department of Communication and Psychology
  Aalborg University, DK

Aligning the Topic of FCA with Existing Module Learning Outcomes

Simon Andrews

Centre For Excellence in Teaching and Learning
Centre For Promoting Learner Autonomy
Sheffield Hallam University, Sheffield, UK
s.andrews@shu.ac.uk

Abstract. Although Formal Concept Analysis is worthy of study on computing courses, it is not always possible or practical to dedicate a whole module to it. It may, however, fit into an existing module as a topic but require some careful design of teaching and assessment activities to properly align it to the intended learning outcomes of the module. This paper describes and evaluates a three year project to align the teaching and assessment of FCA with the learning outcomes of a final-year undergraduate *Smart Applications* module at Sheffield Hallam University. Biggs' constructive alignment was used, incorporating an adapted version of Yin's case study research method, in an iterative process; progressively modifying teaching and assessment activities to align them more closely with the prescribed learning outcomes. The process involved examining conclusions made by students, from carrying out FCA case study assignments, to draw cross-case conclusions about the learning outcomes achieved, and how they deviated from the prescribed ones. These cross-case conclusions were used to feed back into the design of learning and assessment activities for the next delivery of the module. After three cycles, the learning outcomes achieved closely matched the prescribed learning outcomes of the module.

1 Introduction

Formal Concept Analysis (FCA) is a valuable subject to study as part of many Degree courses; it has applications in biological sciences [6], music [11], linguistics [8], data mining [4], semantic searching [3] and in many other area. Its mathematical basis [10], visualisation [12] and wide scope for software development [9] make it suitable for study in a variety of disciplines. However, it may not be possible to devote a complete study module to FCA: there may be competition from other, similar, subjects; it may not be possible to create a new module because of administrative or procedural issues; or FCA may be suitable but only as a smaller aspect of a wider context. In such cases, FCA may still be appropriate as a topic of study in an existing module. Such a module will probably have a existing set of intended learning outcomes. Introducing FCA may require 'bedding in' before it can properly meet theses requirements and become integrated with the existing topics.

At Sheffield Hallam University, a project was undertaken to introduce FCA as a topic to an existing undergraduate computing module, called *Smart Applications*. It was felt that that the applications of FCA, particularly in the semantic search and knowledge organisation areas, made it an interesting topic for the module. To monitor the success of its introduction and make modifications where it was found to be not properly aligned to the existing learning outcomes, an iterative approach was used, applying the *constructive alignment* model of Biggs [1]

2 Biggs Constructive Alignment

The purpose of Biggs' constructive alignment is to design learning activities and assessment tasks so that they are aligned with the learning outcomes that are intended (figure 1). The method includes modification of learning activities based on the outcomes of assessment. It is essential that the learning outcomes are assessed and a proven way of doing this is by criteria-based assessment where grades are awarded according to how well students meet the intended learning outcomes [2]. The problem here, with the introduction of a new topic into an existing scheme, is that the intended learning outcomes of the module will have probably been devised with an idea of the curriculum to be delivered, and not with the new topic in mind. A means is required of testing the alignment of the new topic with the existing scheme. Central to Biggs' is the notion that students construct their own meaning from their learning activities; a means of accessing these constructed meanings could, therefore, be used to ascertain the extent to which particular learning outcomes have been met.

Familiarity with of research methods led to the idea of using such a method to identify the extent to which existing Smart Applications learning outcomes were being met by FCA. The common practice of using case studies as assignments for the Smart Applications module, combined with the idea that modifications would play a key role in the alignment process, suggested Yin's case study research method [13].

3 Yin's Case Study Research Method

Yin describes a method whereby theory is modified from conclusions drawn from a series of case studies (figure 2). The idea is that a theory is investigated through carrying out several distinct case studies. In this way, conclusions regarding the veracity of the theory are made stronger by the fact that a single experiment is not relied on; by putting the theory to the test in different ways, corroborating cross-case conclusions can be made.

In the approach used for aligning FCA with the Smart Applications module, the mapping between Yin and Biggs hinges on the modification of a current entity: theory in the case of Yin and learning activities in the case of Biggs. The case studies used to test and modify the theory are the assessment activities of the module. Yin's method adapted for use in Biggs becomes that in figure 3.

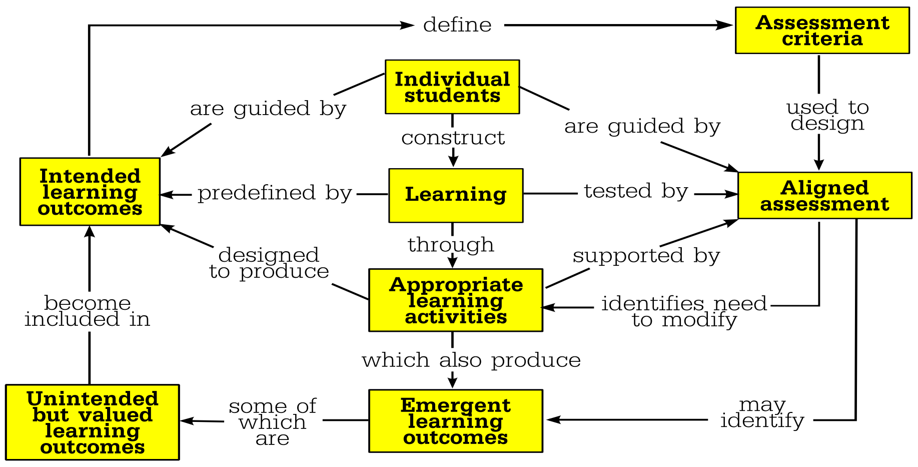


Fig. 1. Biggs constructive alignment: source HEA academy [5]

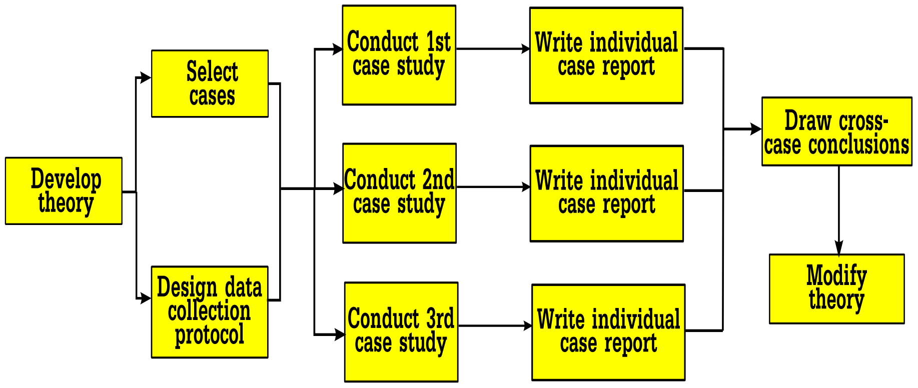

Develop theory
Select cases
Design data collection protocol
Conduct 1st case study
Conduct 2nd case study
Conduct 3rd case study
Write individual case report
Write individual case report
Write individual case report
Draw cross-case conclusions
Modify theory

Fig. 2. Yin's case study method

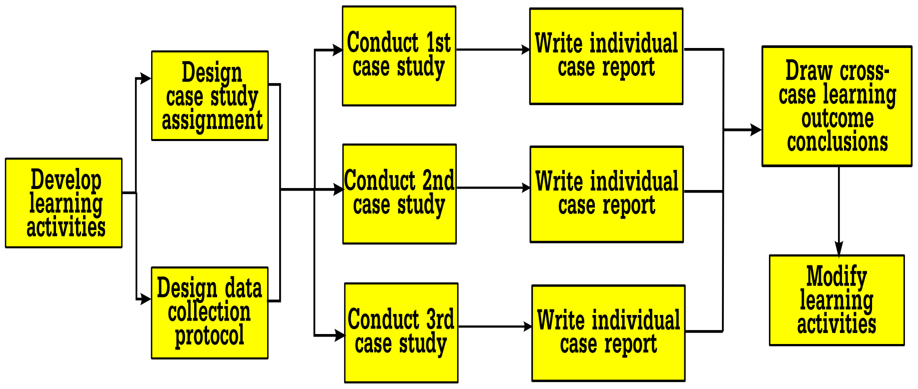

Develop learning activities
Design case study assignment
Design data collection protocol
Conduct 1st case study
Conduct 2nd case study
Conduct 3rd case study
Write individual case report
Write individual case report
Write individual case report
Draw cross-case learning outcome conclusions
Modify learning activities

Fig. 3. Yin's case study method adapted for Biggs

3.1 Data Collection

To conduct Yin, a data collection protocol is required. Data must be collected that allows the testing of the theory. For the purpose of using Biggs, a data collection protocol was required that would allow cross-case conclusions to be made regarding the outcomes of learning activities; how well have the students achieved the intended learning outcomes of the module? Four possible data sources were considered:

1. The marks obtained by the students.
2. Student feedback.
3. A marking scheme.
4. The contents of the coursework.

Student marks Marks taken alone would not indicate whether a particular leaning outcome had been achieved. However, marks will provide an indication as to the depth to which learning outcomes have been achieved. In conjunction with a measure of what has been learned, marks can tell us how well something has been learned.

Student feedback Student feedback is obtained as a matter of course for all modules taught at Sheffield Hallam University, usually through an on-line questionnaire. Likert scale-based questions are routinely used to measure a variety of module qualities and could be tailored to ask about the achievement of individual learning outcomes. However, the response rates are not always high and the Smart Applications module typically has between 12-16 students, so the quantity of data obtained in this way is likely to be small.

Marking scheme It would be possible to design a matrix marking scheme to directly measure the intended learning outcomes; a row in the matrix representing a learning outcome and columns to grade the extent to which it has been achieved. The problem with this approach has been discussed above; assessment criteria should reflect the assignment activities of the new topic. Explicitly basing the case study assessment criteria on the intended module learning outcomes would defeat the purpose of testing the current 'theory'.

Coursework content The content of coursework clearly provides qualitative evidence of learning. But taking the whole of the students' reports as data would be difficult to manage in terms of sheer quantity of written work. However, as part of a case study assignment, it is appropriate to require students to make conclusions. Requiring this as part of the individual case study report will provide a good source of data on which learning outcomes can be judged. And as a data collection protocol, this has the added benefit that students will be motivated to provide good quality data; they are being assessed on it.

As a by-product, this protocol also provides the mechanism in Biggs whereby emergent learning outcomes can be identified. The incorporation of the method into Biggs can be seen in figure 4.

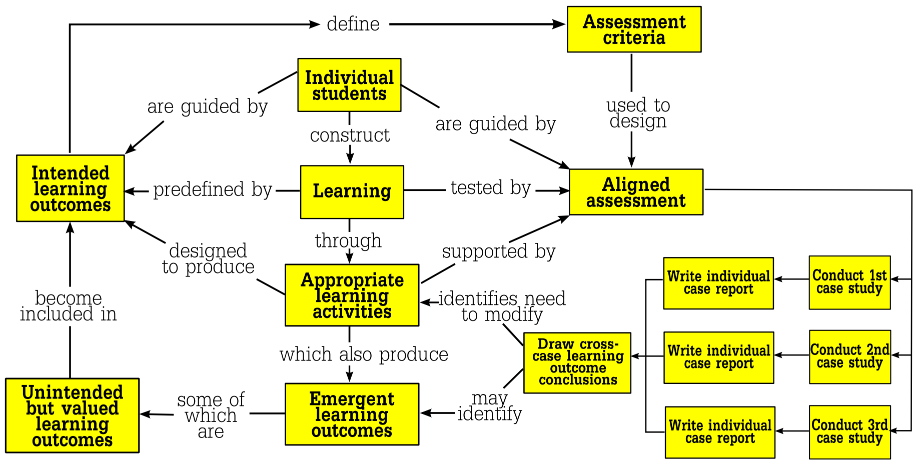


Fig. 4. Yin's adapted case study method incorporated into Biggs constructive alignment

4 Applying the Method: The Three Cycle Smart Applications Project

Smart Applications is a final year undergraduate computing module taught at Sheffield Hallam University. One of the aims of the module is to introduce students to frameworks and techniques for representing and reasoning with knowledge for smart applications. It was felt that FCA was an appropriate technique to study as a topic to help achieve this aim. However, the topic would have to be aligned with the existing intended learning outcomes of the module. It was decided to use this as an opportunity to develop the pedagogical method, described above, for incorporating new topics into existing modules. The process was carried out in three cycles, over the academic years 2007/8, 2008/9 and 2009/10.

FCA, as a new topic, was required to provide one or more of the intended learning outcomes (ILOs) of the Smart Applications module, which were:

ILO1 Describe the notion of representing and reasoning with knowledge for smart applications.
ILO2 Draw on one or more frameworks and techniques for representing and reasoning with knowledge for smart applications.
ILO3 Critically evaluate the key issues in knowledge representation and knowledge sharing for smart applications.
ILO4 Identify the practical use of software tools for developing smart applications.

4.1 Cycle 1: 2007/8

The learning activities associated with FCA in the 2007/8 delivery of the Smart Applications module were primarily concerned with the mathematical underpinnings of FCA. After an introduction to FCA, lectures and tutorials were based on the themes of *Knowledge Architectures for Smart Applications through Conceptual Structures* and *Specifying Smart Applications using FCA*. Because FCA was one topic of several, the main assessment of FCA had to be limited to a single assignment. To ensure that this would not conflict with Yin's requirement that case studies should to be distinct, the design of the assignment had to allow for as much variation as possible, whilst still being consistent in terms of assessment criteria. The assessment was based around a smart application case study for managing user profiles in a business information system. Variation was designed into the assignment by phrasing it as openly as possible; the students were asked to investigate possible solutions and make their own recommendations. The deliverable was a report that considered "How might the implementation of a 'user profile concept lattice' be accomplished, that supports the capture and reconfiguration of user profiles? The FCA approach should also consider how appropriate data might be assimilated, managed and presented to the user."

The following are quotes from the conclusions of the students' case study reports that gave an indication of the learning outcomes achieved:

1. "FCA lends itself well to mapping user profile details. This is best achieved using a series of different lattices covering the different aspects of user profiles. Trying to capture all of the relevant user information in one lattice would probably be a little ambitious and also result in a complex and impractical lattice."
2. "Through implementation of web mining techniques of Association rules or Conceptual Clustering Mining you can capture User Profiles."
3. "By utilising BI techniques and FCA modeling, user profiles would be smart as they would show relevant information to users across the company."
4. "I have found that using FCA for capturing User Profiles makes seeing the relationship between objects and attribute a lot easier, which we can then use to see relationships for that particular profile which helps gathering data for use in trends."
5. "Through user profiling with FCA I found that content and information only relevant to that particular individual within the company can be displayed, saving time and improving productivity."
6. "Although FCA seems to be a good way to analyse and create user profiles it can become slightly difficult if you have a model that constantly changes as it can be difficult to adapt a new role into the model."
7. "Once a Smart Google system can comprehend the context of a sentence, through the identification of relations between words in a search phrase, it will then deliver the user with answers rather than results. This I would consider as smart."
8. "Combining FCA and BI with the functions and facilities available in Web DynPro has the capacity to store a lot of information in a very well organised and formal structure, which can be added to or reduced very easily, with little adaptation of the data structure."

The marks for the assignment were slightly disappointing with a mean of 54%, although the first time pass rate was a reasonable 88%.

It was clear that some of the intended learning outcomes were being met, at least to some extent. Much of the learning seems to have been centered around the visualisation aspect of FCA, and thus relevant to ILO1, but only quote 1 suggests any depth to that learning. ILO2 appears to have been achieved in quotes 1 and 7 and perhaps in quote 2, but without depth. In terms of evaluating issues (ILO3), displaying relevant information through FCA appears to be the main message, but with some contradictory understanding of how changes in information can be managed in FCA; quote 6 suggesting this is a problem and quote 8 advocating FCA as being advantageous in this regard. Only quote 1 suggests that lattice complexity is a problem in FCA. For ILO4, only quote 8 mentions a software tool for developing smart applications (Web Dynpro is a web application user interface programming tool that was introduced to the Smart Applications students as part of another topic in the module). In general, the learning appears to be about FCA itself rather than the practical application of FCA as a framework or technique. Disappointingly little was learned about the issues involved in knowledge representation for smart applications or of the use of software tools for the development of smart applications.

It was decided, therefore, to modify the learning activities of the module to focus less on the mathematical theory of FCA and more on its practical application; perhaps by implementing FCA-based software, students' learning outcomes would be better aligned with the intended ones.

4.2 Cycle 2: 2008/9

The emphasis of the learning activities associated with FCA in the 2008/9 delivery of the Smart Applications module was on the development and implementation of FCA-based smart applications. Modifications were made to the learning activities of the previous cycle to focus less on the theoretical aspects of FCA and more on the engineering of FCA-based software. The themes were *Semantic Search: 'Sleuthing' with FCA*, *Data Structures for FCA* and *FCA-based Smart User Interfaces*. The assessment was a modified version of the 2007/8 assignment, with an element of application prototyping replacing some of the investigation of theory. The assignment was still open enough to allow sufficient variation for Yin.

The following are quotes from the conclusions of the students' case study reports that gave an indication of the learning outcomes achieved:

1. "Integrating FCA user profiles into an SUI such as Dynpro comes with issues. Dynpro is not very flexible in terms of changing things by the program. It requires the user profiles to already be configured to support an FCA ontology."
2. "Concept lattices were used to structure ... an FCA-based user profile. This data was ... integrated into a Smart User Interface (SUI) by being focused on the organising, sorting and searching of data, and finding matching concepts."
3. "Using the unique way that FCA stores and represents knowledge the Smart User Interface can give the user multiple ways of finding information."
4. "We defined user sessions as only the URLs that have been requested and visited by the user. This considerably lowered the amount of attributes and data that had to be analysed."
5. "We have presented how Web Dynpro implements the MVC framework which allows FCA of the data used within it, integrated with BI to support user profiles, customising a user's interface, manipulating it to suit the aims of enterprise and user. Using FCA to discover patterns and correlations [was a] way of determining relationships and discovering the implicit ones that other methods struggle to uncover."
6. "It is possible to integrate the FCA user profiles successfully; however, there are several difficulties when implementing them: firstly there's the issue with accuracy, as the larger the company or department the more complicated it is going to be to implement. There's also a concern with timing, as it's a long process computing the profiles."
7. "The use of FCA based user Profiles as the basis for e-commerce recommender systems does bring benefits, but the same level of functionality can be achieved using [off-the-shelf] alternatives."

The marks for the assignment were poor, with a mean of 47%; down significantly on the previous year's mean of 54%. The first-time pass rate was also poor, at 71%.

This time it was clear that most of the intended learning outcomes were being met, and also to higher degree. ILO1 was evidenced by many of the comments as was ILO2. Key issues were highlighted more this time (ILO3); some of the quotes indicate a good awareness of complexity and performance issues and some of them indicate that some of the key advantages of FCA have been better explored and understood. Some quotes, including the final one, indicate that some broader understanding of the context has taken place. Less in evidence is ILO4; very few of the students reflected on the use of software tools for developing smart applications.

Unfortunately, the low mean mark suggests that, whilst the learning outcomes may now be better aligned, the achievement in terms of results was not good. Although the quotes give an encouraging picture of the learning that has taken place, the technical aspects of the learning and assessment activities proved to be a struggle for some of the students. The programming skills, for example, of many of the students were not adequate for developing a useful prototype application. It was therefore decided to modify the learning activities to require less of these technical skills, by de-emphasising the construction of new software, and focus more on the investigation, use and development of existing FCA tools and applications.

4.3 Cycle 3: 2009/10

This time the learning activities were based on the themes of *FCA Tools*, *FCA 'Sleuth' Applications* and *Data Mining with FCA*. Practical sessions were designed to explore the capabilities and limitations of existing software. The assignment was changed by replacing the prototyponmg element with one that used existing tools and techniques to carry out FCA on real sets of user profile data.

The following are quotes from the conclusions of the students' case study reports that gave an indication of the learning outcomes achieved:

1. "FCA raises new questions regarding the actual warranty of 'hidden' information. How can we trust that a smart application is actually giving the right results?"
2. "To make this investigation even more interesting, we could use similar FCA techniques on the categorised attributes and incorporate the boolean attributes to discover if the top 4 factors are actually the factors of people from all age groups, for example."
3. "Filtering the formal concepts generated allowed for visualisation of the large data set and allowed for affective analysis."
4. "Although there may be some current issues with the interoperability of FCA in existing technologies ... FCA could be integral to the development of semantic knowledge architectures."

5. "While formal concept analysis and the enabling technologies described are now in a usable state, the applications to use it now have to catch up with them."
6. "Visualisation techniques are key to enabling Smart Applications. Information that would be hard to find otherwise was made clear to understand."
7. "There seems to be a lack of ability to be able to communicate and exchange data between FCA systems and tools with non-FCA applications. Although the data mining is 'smart', problems arise when changes are needed to be made to lattices."
8. "[It would be useful if] visualisation techniques can be brought together, allowing easy access to the produced images and concepts and so giving a broad and easily digestible view of the information contained within."

The marks for the assignment had a mean of 67%. The first-time pass rate was 93%.

The quotes indicate that the learning outcomes have been met, and to a good extent. Notions of representing and reasoning with knowledge for smart applications (ILO1) and the drawing on FCA as a framework/technique for smart applications (ILO2) are apparent in many of the comments. There is good evidence that an ability to critically evaluate key issues has been demonstrated (ILO3), particularly in quotes 1, 3, 4 and 7. And there is a strong sense that the students can identify the practical use of software tools for developing smart applications (ILO4).

5 Conclusion

The three-year Smart Applications project shows that FCA can be aligned with the intended learning outcomes of a suitable existing module using sound pedagogical practice. In Yin's adapted case-study method, students drew conclusions on the work they had undertaken, then, from these, teachers drew cross-case conclusions regarding the learning outcomes achieved and how far they were aligned with the intended ones. The students' concluding remarks gave a qualitative measure of their learning whilst their marks gave a quantitative measure of the depth of their learning. The three cycles of the project are summarised in figure 5 and the results are summarised in table 1.

cycle	mean mark %	pass rate %
1	54	88
2	47	71
3	67	93

Table 1. Summary of *Smart Applications* Results

There were problems, however. Firstly, the use of marks as an indicator of the depth to which learning had taken place did not take the quality of the student cohort into place. The small number of students involved meant that statistical anomalies were easily possible. A degree of additional judgment from observation and assessment was used, for example, in detecting the problems that the students had with programming, leading to poor performance in the case study assignment in the second cycle. Secondly, there was no clear metric for determining the nature of the modifications required to the learning activities. Again, a degree of additional judgment was required to fathom a sensible change. Notwithstanding the limited amount of student feedback one would be likely to get on a small module, an additional element of data gathering involving a module questionnaire might be useful in judging new directions for learning activities.

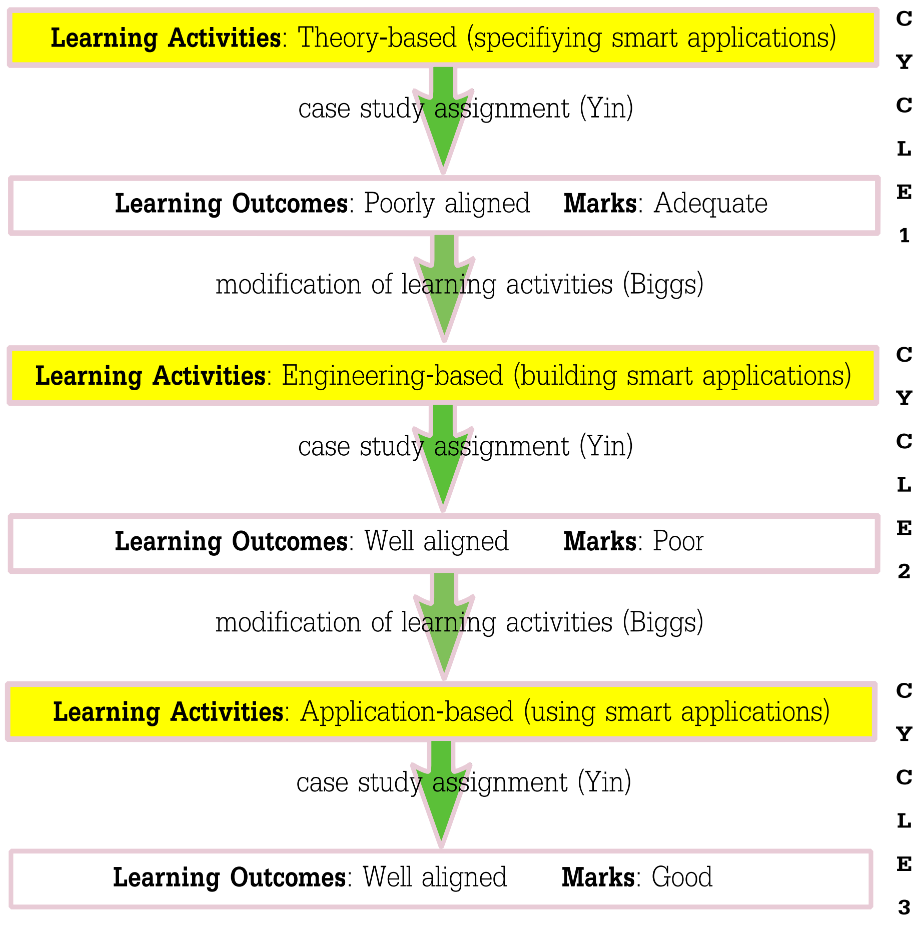

Learning Activities: Theory-based (specifiying smart applications)
case study assignment (Yin)
Learning Outcomes: Poorly aligned
Marks: Adequate
modification of learning activities (Biggs)
Learning Activities: Engineering-based (building smart applications)
case study assignment (Yin)
Learning Outcomes: Well aligned
Marks: Poor
modification of learning activities (Biggs)
Learning Activities: Application-based (using smart applications)
case study assignment (Yin)
Learning Outcomes: Well aligned
Marks: Good
CYCLE 1
CYCLE 2
CYCLE 3

Fig. 5. Three cycle alignment of FCA with the ILOs of Smart Applications

References

1. Biggs, J.: Teaching for Quality Learning at University, SRHE and Open University Press, Buckingham (1999)
2. Biggs, J.: Aligning Teaching and Assessment to Curriculum Objectives, (maginative Curriculum Project, LTSN Generic Centre (2003)
3. Dau, F., Ducrou, J., Eklund, P.: Concept Similarity and Related Categories in SearchSleuth. In: Eklund, P., Haemmerlè, O. (eds.) ICCS 2008, LNAI 5113, pp. 255-268. Springer-Verlag, Berlin/Heidelberg (2008)
4. Grigoriev, P., A., Yevtushenko, S.A.: QuDA: Applying Formal Concept Analysis in a Data Mining Environment. In: Eklund, P. (ed.) ICFCA 2004, LNAI 2961, pp. 386-393. Springer-Verlag, Berlin/Heidelberg (2004)
5. HEA Academy, Learning and Teaching Theory Guide: `http://www.engsc.ac.uk/learning-and-teaching-theory-guide/constructive-alignment`
6. Kaytoue-Uberall, M., Duplesssis, S., Napoli, A.: Using Formal Concept Analysis for the Extraction of Groups of Co-expressed Genes. In: Le Thi, H.A., Bouvry, P., Pham Dinh, T. (eds.) MCO 2008. CCIS vol. 14, pp. 439-449. Springer-Verlag, Berlin/Heidelberg (2008)
7. Kuznetsov, S.O., Obiedkov, S.A.: Comparing Performance of Algorithms for Generating Concept Lattices. In: *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 14, pp. 189-216 (2002).
8. Priss, U., Old, L., J.: Bilingual Word Association Networks. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007, LNAI 4604, pp. 310-320. Springer-Verlag, Berlin/Heidelberg (2007)
9. Priss, U. (ed.): FCA Home-page, `http://www.upriss.org.uk/fca/fcasoftware.html`
10. Wille, R.: Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies. In: Ganter, B., Stumme, G., Wille, R. (eds.), *Formal Concept Analysis: Foundations and Applications*, pages 1-33. Springer-Verlag, Germany (2005).
11. Wille, R., Wille-Henning, R.: Towards a Semantology of Music. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007, LNAI 4604, pp. 269-282. Springer-Verlag, Berlin/Heidelberg (2007)
12. Wollf, K., E.: A First Course in Formal Concept Analysis: How to Understand Line Diagrams. In: In: Faulbaum, F. (ed.) SoftStat'93, Advances in Statistical Software 4, 429-438 (1993)
13. Yin, R., K.: Case Study Research: Design and Methods (4th Ed.). Applied Social Research Methods Series, Vol. 5. SAGE (2009)

Building a Dependency Network for Teaching-Learning of Conceptual Structures

Nagarjuna G., Meena Kharatmal, Rajiv Nair
{nagarjuna,meena,rajiv}@hbcse.tifr.res.in

Homi Bhabha Centre for Science Education, TIFR, Mumbai, India

Abstract. We propose in this paper a simple method to construct a machine processable semantic network, called a dependency network, that gathers all the concepts and skills as nodes and the relation type "depends on", (and its inverse "required for"), as their edges. As a conceptual structure can be used to compute a roadmap of any learning-teaching objective. As a prelude we build and contribute a seed graph for demonstration and introduce a collaborative portal for contributing, publishing and dynamically building dependency networks. A possible generalization of this methodology for knowledge organization is discussed.

1 Introduction

Every good author of a text book informs the reader the prior knowledge before introducing any new topic. Most of the good text books therefore explicitly mention the prerequisites at the beginning of each chapter or teachers provide a refresher course before commencing a new topic. This is due to the obvious assumption that if the prerequisites are not satisfied adequately the learner may not be able to comprehend the new ideas introduced in the book. This has been the generic guiding principle of curriculum design. This principle also stands out as one of the consensus from the widely accepted constructivist philosophy of education[1,2]. Constantly helping and reinforcing the prior knowledge to learn something new is a time-tested ancient wisdom shared among most educationists. Based on this assumption we propose a simple method of processing prerequisites for conceptual structures by employing a conceptual structure itself.

Mastering conceptual structures is a skill that requires inter-disciplinary understanding involving certain topics from domains such as logic, linguistics, mathematics, AI, databases, philosophy, computer science etc. We propose in this paper a simple method to construct a machine processable semantic network, which may be called a *dependency network*, that gathers all the concepts and skills as *nodes* and the relation type "depends on", (and its inverse "required for"), as their *edges*.

There are several studies on the dependency relation in different contexts.[3,4,5,6] Semantics and logic of dependency are covered in [6]. Keller identified some kinds of dependencies in the context of requirements analysis[4]. Cox et.al. did a more general modeling of dependencies and an ontology of dependency relation[5]. In this paper we focus on a specific kind of dependency, which can be classified as a species of causal dependency in the context of semantics.

Arguably one of the largest operating system stack available today is the Debian GNU/Linux[1], which maintains its stable opearation by asserting dependency relations among them. Each package contains in its metadata all other packages that are required, and some of the required packages again may inturn depend on other base packages. This is possibly the largest working example where dependency network is used as a semantic structure, with more than 20,000 packages with mutual dependencies. We harvested all the asserted relations from their packages and created a dependency network, which is also available from the portal www.gnowledge.org/search_debmap?val=1, where one can query for any package and obtain the dependency graph. We further interpreted this semantic network as a complex system, because it exhibits same characters of any natural networks that have been studied.[7]. The most notable being the scale-free character and power law distribution[8].

Inspired by the way the Debian OS uses dependency relation, we think we can construct collaboratively a similar knowledge base that can give us the roadmap of any given learning objective. An artificially constructed operating system does not work if the dependencies are not met, similarly cognitive agents will not understand the meaning of a concept *explicitly* unless the prior meanings are already understood. This is therefore a kind of critical dependency[4]. While this being the case from a learner's point of view, we can make a corollary statement from a teacher's point of view: a teacher could not make a student learn unless she ensures that the prerequisites are already introduced to the student. Thus, teaching and learning requirements are linked, though the processes may not happen concurrently and to the same agents.

Based on the assumption, that prior-knowledge is essential for understanding new concepts, we undertook the program of collaboratively constructing a comprehensive propositional semantic network that uses dependency as a primary relation.

2 The Proposal

A prerequisite can be expressed as a dependency link between two concepts/skills. For example, to understand the meaning of the concept "semantic network", we expect the learner must already know the concepts "node", "links", "proposition" etc. And to understand what a proposition is we expect the learner must already know what a sentence is, a statement is, the distinction between a sentence and a proposition, to know "subject" and "predicate" the learner must already know what a sentence is etc. This prior knowledge can be mapped as a simple graph shown in Figure 1. The figure shows teaching-learning sequences for the learning objective semantic network. One may extend it by further asserting that semantic network is required for semnatic web, etc.

[1] The Debian GNU/Linux Operating System, http://www.debian.org/

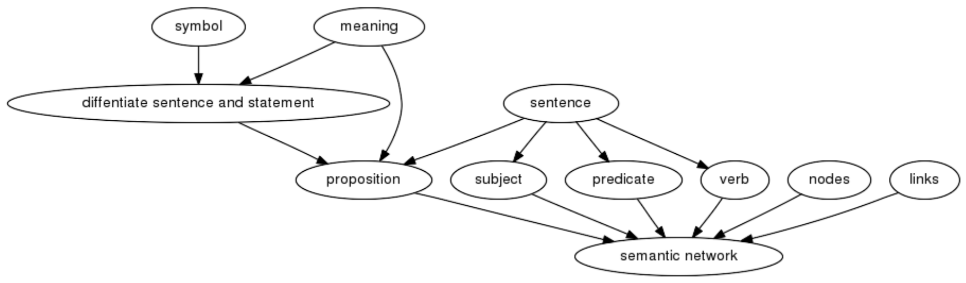


Fig. 1. A sample dependency network. The learning objective in this map is "semantic network".

All definitions will yeild a clear set of dependency assertions between *definiendum* and *definiens*. Any definiendum semantically depends on the definiens in a definition, or conversely the definiens are required for understanding the definiendum. Such dependencies may give us a conceptual network among entirely explicit and declarative knowledge. However we cannot ground all concepts in the world of concepts alone. Some concepts may need experience, an activity, a skill, solving a problem or such real experience situations that help us learn. Similarly some skill/activities may need conceptual skills, such as reading an algebraic expression. The resulting network may not contain exclusively all concepts, therefore the network may not be a strict conceptual structure. It is a conceptual/cognitive structure. Nevertheless, grammar of this network, and its computability properties, will not be different from that of a directed graph.

The resulting network, which is a directed graph, could provide the following for supporting teaching-learning and assessment process:

– **a road map** to reach a given learning objective;
– **a road-ahead map** informing what learning paths lie ahead of a given learning objective;
– **a navigational aid** providing immediate guide to inform what are next and previous steps to take direct result of being a road map);
– a dynamically generated **teaching-learning sequence** suggesting the order of presentation in, for example, a text book;
– **a relative depth measure** informing the number of levels to follow depending on the current position;
– **a surface map** of the knowledge with some special emerging properties (discussed below);
– **a directed acyclic graph**[2], which can be used in decision algorithms by an automated LMS (learning management system).

[2] Some of the nodes may have mutual dependencies. For example, concept required for class and class is required for concept. By merging all such mutual dependencies into a single node, we can construe the resulting network as a causal Bayesian kind.

For the subject of conceptual structures we collected some assertions based on the concepts introduced in the introductory chapters from [9,10,11,12]. A sample collection of assertions are as follows:

```
digraph G {
        subtype -> species;
        categories -> metatypes;
        classification -> taxonomy;
        types -> class;
        type -> supertype;
        type -> subtype;
        type -> universals;
        general -> "common nouns";
        ...
        "predicate calculus" -> "formal semantics";
        "knowledge organization" -> ontology;
        "semantic network" -> "semantic web";
        RDF -> "semantic web";
        ...
}
```

where "→" stands for the inverse of the dependency relation, *required for*. A comprehensive set of such assertions as an input file can be prepared and submitted to a graph processor (in our case Graphviz[13]) to generate the directed graph. To reduce the clutter of the large graphs as well as suggesting the roadmap we could use transitive reduction filter (also available with the Graphviz library), to eliminate redundant edges that could be deduced by applying transitivity rule.[14]

To facilitate collaborative contribution of the assertions, we have created a community portal www.gnowledge.org[3] which provides the following features:

– a facility to upload the input files that contain assertions in a specified format;
– an algorithm to check preexisting nodes and edges in the memory;
– a search interface;
– dynamic generation of three different directed graphs (road-map, road-ahead-map, and a combination of the two previous graphs);
– a basic facility to edit and delete links;
– a version control feature to record who did what and when;
– an option to save the generated graphs in several image formats including SVG (scalable vector graphics);

The portal, though requires more useful features and better user interface for convenient use, we introduce this effort here to obtain the feedback from the peer group to first of all comment on its projected use and implications.

[3] Gnowledge portal http://www.gnowledge.org/.

The proposed method is general, and not specific to mapping the knowledge of conceptual structures. However, since the method used is an application of conceptual structures, and can also serve the purpose of dynamically generate the teaching learning sequences, we think it could be useful for knowledge representation community. A general introductory paper written for non-formal audience, discussing the motivation, method and possible application and impact of this method are presented in [15].

3 Observations

The sample merged graph of dependencies of the seed content cannot fit in a paper format, therefore it is uploaded at the portal in a SVG format, and also in a PDF format[4] The network contains more than 500 nodes, therefore to read the details we need to zoom in 200-300% to read the text. Due to several edges the graph looks quite cluttered, and so not very convenient to read. To extract the specific dependencies one may query from the portal and see the generated graph in isolation for a given node. Though cluttered, the merged graph has some properties worth mentioning.

Most notable observation is the nodes on the top of the graph are to be learnt or taught first, and nodes at the bottom of the graph are to be learnt later. The nodes that are aligned at the top are: and, or, sentence, object, class, words, brackets, aspect, symbols, variable, concept, types, subtypes etc. The nodes that got aligned at the bottom are: canonical graphs, specialiation rule, conceptual graphs, ontology, OWL, semantic web, conceptual model, schema, syllogism, formal semantics, proof, etc. When advanced chapters from the text books are processed, other deeper and difficult concepts and methods could appear at the bottom. This clearly indicates that using this simple method, we can frame a specific curriculum sequence more objectively.

Nodes with a large number of outgoing links cannot be neglected in education. At any given level, the first priority can be given to the nodes with larger number of outgoing links. Since, outgoing links are required for learning several other objectives/sequences, these nodes have greater potential to serve in future learning and can also be marked as core concepts. In the seed graph the nodes "class" and "relation" have high number of outgoing links. After additional assertions we may find other core concepts emerging. This needs more work.

The directed graph algorithm automatically positions the nodes and draws the edges. When merged as a large graph, unlike the isolated dependency graphs obtained by querying for one of the learning objectives, we notice that those nodes which are at a same level (a sort of latitude of the graph) are placed horizontally. When new assertions are inserted that may have links with existing nodes, the graph *accommodates* itself by *assimilating* the new nodes automatically following the well known digraph construction rules. All the adjacent nodes

[4] The hyperlinks are http://www.gnowledge.org/cs-merged-graph.svg and http://www.gnowledge.org/cs-merged-graph.pdf respectively.

descending from a parent share similar learning sequence. When a child node has more than one parent node, it indicates as many sequences to cover.

The generated directed graph is a horizontal spread with only a depth of 13 levels. As the graph indicates, the prerequisites for graph based knowledge representation are rich and belong to different disciplines. We have barely scratched the surface of the required prerequisites since we have not inserted several other assertions. For example, when the skills and conceptual requirements from linguistics, logic, graph theory, algebra, predicate calculus, computer science and philosophy are comprehensively contributed the generated graph will appear richer than the shown map. We expect the depth to grow when we insert more assertions into the knowledge base.

This data cannot be finite but can only be judgeable as comprehensive enough by the peer group. The result obtained is far from satisfactory, but we hope it can take a good shape if the community finds it necessary and begins to contribute collaboratively.

4 Discussion

Since the method used is simple and general and could be used for all domains of knowledge[5], the network, we hope, may evolve as a general mapping and sequencing tool for learning and teaching. Using the scaffolding of dependency one can add some additional tags, say modalities to code the strength of the assertions as possible or necessary. For example, while adding an exercise as a node we may add the *possibility* tag to the node since there can be other possible exercises that may give similar experience, difficulty and understanding. When the relation is core semantic dependency, as in the case of definiendum and definiens relation, one may tag with *necessity*. Additional enrichment of this network can be done by also inserting assessment objects as nodes linked to some of the learning objectives, which can be used as pre/post-test resources. Such a knowledge base could be useful for automated assessment and delivery of lessons.

Such a map, if available as a easily queryable knowledge base, a teacher or a student in a classroom can bring on board explicitly the prerequisites, so that the focus of deliberations in the class room fall on satisfying all the requirements. However, if a map of this kind is available to normal teachers as well as slow learners could do better in teaching and learning respectively, based on the assumption that explicit knowledge helps in mastering any domain. Moreover, even an artificial tutoring system can guide the learner by presenting the requirements in the order suggested by the generated roadmap and can present the assessment items at appropriate places. However these are more or less obvious applications of the map. We think the map does more than the obvious.

The students and teachers of CS can study the properties of directed graphs using this map. Since there does not exist a comprehensive repository of depen-

[5] The gnowledge.org portal is open for contributions from all domains of knowledge.

dencies of various domains of knowledge, one can meanwhile study the software dependency network such as the Debian dependency network, which exists for six major releases[6] We conducted on such study where the structure and dynamics of software dependencies as a semantic system[7]. A software package as a node in dependency network is very gross. We can undertake to map dependencies at a granular level, such as dependencies between libraries within packages, and in turn at a much more granular level between the functions within the libraries. This is a massive task, but doable since every dependency is explicitly asserted in a software. The students could write programs/algorithms that will extract the dependencies at various levels of granularity. We think such a massive network can become a basis for conducting several other serious semantic and computing studies.

Returning to the natural knowledge network context, the CS students could write decision algorithms for an artificial tutoring system based on dependency networks agregating at gnowledge.org: to locate learning objectives that are core (higher number of outgoing links); locate cognitively interesting terminal destinations (nodes with higher number of incoming links); calculate the roadmap of a given learning objective from a relative starting node; generate pre-test and post-test sets for administering assessments; a method for generating clusters that have similar but not same dependencies; measuring the semantic distance between nodes using dependency; write web services for accessing this information from a server to a remote client; etc.

A computer processable dependency network can also serve as a scaffolding for *parking* ontologies and other conceptual structures. This is possible since every conceivable node of any ontology or any conceptual structure can also be part of the dependency network. No formal proof for this claim is offered here, but an argument can be as follows. Given the holistic assumption that there are no loosely hanging concepts in a knowledge network[16], every concept will have a prerequisite network of nodes. This also follows from the need of at least two concepts to explicitly individuate a concept.[17] Currently all ontologies are held together by an artificial *summum bonum*, the "thing" super-class. If we hold the scaffolding of dependency network as a surface map, which is constructed using semantic flow relationships, a *natural* way of holding together all other ontologies seems possible. Could this be the topography of knowledge?

Since each node in this network has only a set of incoming and outgoing nodes, each node must have a unique set of nieghbouring nodes when all relations are fully specified. Can this criteria of unique meaning be demonstrated practically? If so, this could become a criterion for disambiguation. If the above is possible, we can obtain a more realistic measure for semantic distance based on the obtained directed graph. Given any two nodes in the dependency network, we can compute the order of separation between the nodes.

Based on the reasons and speculated implications of the program, we hope the dependency network of any knowledge can be constructed. Since, the community of conceptual structures, semantic web, AI, expert systems, cognitive scientists

[6] Starting from 1994 Debian released six major releases.

have the interest to map knowledge in general, this may become a useful resource for both research and material development.

To sum up, we proposed a method of constructing a dependency network for the domain of conceptual structures, and argued that it can be a general strategy for any other domain.

References

1. Ausubel, D., Novak, J., Hanesian, H.: Cognitive Physchology: A Cognitive View. Holt, Rinehart and Winston, New York (1978)
2. Ausubel, D.: The Psychology of Meaningful Verbal Learning. Grune & Stratton, Oxford, England (1963)
3. Keller, A., Kar, G.: Dynamic dependencies in application service management. In: 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Citeseer (2000)
4. Keller, A., Blumenthal, U., Kar, G.: Classification and computation of dependencies for distributed management. In: Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), Citeseer (2000) 78
5. Cox, L., Delugach, H., Skipper, D.: Dependency analysis using conceptual graphs. In: Proceedings of the 9th International Conference on Conceptual Structures, ICCS, Citeseer (2001)
6. Pearl, J., Verma, T.: The logic of representing dependencies by directed graphs (1987)
7. Ray, A., Nair, R., Nagarjuna, G.: Saturation in the scale-free dependency networks of free and open-source software. Arxiv preprint arXiv:0901.4904 (2009)
8. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts (1949)
9. Sowa, J.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Publishing Company, USA (1984)
10. Sowa, J.: Knowledge Representation: Logical, Philosophical and Computational Foundations. Brooks/Cole, USA (2003)
11. Chein, M., Mugnier, M.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs. Springer-Verlag New York Inc (2008)
12. Sowa, J.F.: Semantic networks. In Shapiro, S.C., ed.: Encyclopedia of Artificial Intelligence. 2 edn. John Wiley & Sons, Inc., New York (1992)
13. Ellson, J., Gansner, E., Koutsofios, L., North, S., Woodhull, G.: Graphviz-open source graph drawing tools. Lecture Notes in Computer Science (2002) 483–484
14. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM Journal on Computing **1** (1972) 131–137
15. Nagarjuna G.: Collaborative creation of teaching learning sequences and an atlas of knowledgge. Mathematics Teaching-Research Journal Online **3** (2009) 23–40
16. Quine, W.: From A Logical Point Of View. Harward University Press, Massachusetts (1953)
17. Strawson, P.: Individuals: An essay in descriptive metaphysics. Methuen London (1971)

Culture, Critical Thinking and Computing

Richard Hill[1] and Dharmendra Shadija[2]

[1] School of Computing
University of Derby,
Derby, DE22 1GB, UK.
r.hill@derby.ac.uk
[2] Department of Computing
Sheffield Hallam University
Sheffield, S1 1WB, UK.
d.shadija@shu.ac.uk

Abstract. The internationalisation of Higher Education is a major thrust for EU Universities. This article examines how the curriculum can be designed to not only accommodate different and disparate cultures, but also enhance the cultural experience for all concerned. Using Peirce's writings on critical thinking, we describe how a research-informed curriculum can deliver an improved experience for postgraduate learners. Specifically we refer to Peirce's core concepts with respect to the management of doubt and proffer a structure for the facilitation of critical thinking behaviours in the context of web application design and modelling.

1 Introduction

Internationalisation of the UK Higher Education experience is an issue for many institutions[5],[9]. The decline in UK government funded undergraduate places and (apparently) unrestricted numbers of non-European Union students, makes a compelling argument to meet the needs of a different marketplace. However, this is not a trivial undertaking[4] and as some HE institutions have discovered already, a change in marketing approach cannot fully support the deep-seated differences of disparate learning cultures[5].

Discussions around culture are fraught with difficulties. Inevitably there is a situation whereby one might comment upon the apparent behaviours of another culture. This raises many questions, and to the philosopher at least there is the fundamental question of bias; how can the actions of one culture be judged through the eyes of another culture?

It is also apparent that the 'culture' word is used somewhat perjoratively in that it becomes a general container for all of the unexplained behaviours that students of a different culture can exhibit. Apparent plagiarism, or perceived academic dishonesty, is often described as one of the most visible behaviours from students from South East Asia and the Indian Sub Continent. This state of affairs has been described as a lack of understanding of what is meant by 'critical thinking'. The problem, therefore is one of 'culture'. In this article, the authors hope to proffer a more sophisticated response, based upon Peirce's philosophical writings. We start by examining what is meant by critical thinking.

2 Critical Thinking

Many academic teaching staff pride themselves on being part of the educational environment that is purported to 'teach' critical thinking. But what does it mean to think critically? Wells argues that critical thinking requires an ability to objectively reason[17]. The behaviours of critical thinking include an ability to scrutinise scenarios through the lens of another perspective. A critical thinker will not be unduly influenced by a hastily, or poorly constructed argument. Another dimension is that of the moral obligation of thinking critically[17];

"The critical thinker should bring a voice of reason to this discussion."

The moral position is important in that is usually the key justification for the teaching of critical thinking. Such a premise suggests that an individual will be 'improved' by undertaking the study of critical thinking, and is therefore, worthwhile study.

2.1 Cultural Perspectives

Critical thinking embodies the beliefs and approach to thinking of western societies. In cases of alleged plagiarism amongst international students, it is common to hear that the cause is that of 'cultural background'. If we adopt this premise, then it must be recognised that this view is being made about a particular culture, from the perspective of another, disparate culture. What might be judged a more intolerant society must therefore restrict any ability to develop critical thinking behaviours. Again, the authors challenge this belief.

After teaching hundreds of students from South East Asia and the Indian Sub Continent we have witnessed behaviours that demonstrate the presence of critical thinking. However, we have also witnessed the challenges faced by students when faced with an intrinsically western approach to thinking for the first time.

Peirce's writings offer many philosophical arguments that can be utilised to better understand critical thinking. 'The Fixation of Belief'[10] indicates an approach to understanding how critical thinking might come about. We now proceed to examine these ideas in order to inform our approach to delivering a more culturally-aware computing curriculum.

3 The Peircian Approach

Peirce offers a model by which we arrive at a belief in such a way that we are inclined to maintain that belief[10]. In other words, the conditions required for a belief to be acquired and held. Peirce summarises the four key scenarios as follows:

1. *Tenacity* - The holding of a belief without any consideration of arguments to the contrary. Any idea that counters what the individual already believes is immediately discounted. Many students exhibit this behaviour, irrespective of their cultural upbringing.
2. *Authority* - Here Peirce refers to the influence exerted by external agents to enforce the holding of a belief. If multiple agents are collectively *tenacious* then a considerable influence can be exerted over an individual. Peer pressure is an example of *authority.*
3. *A priority* - In some cases there may be a tendency to reason between alternatives, but those alternatives are actually based upon pre-ordained outcomes, thus there is no basis for critical thinking. This state can be misleading in that there may be a perception that critical thinking is taking place. One challenging aspect is that this potentially dishonest practice could be unwittingly fostered amongst students through the delivery of curricula if the conditions for critical thinking are ignored.
4. *The scientific* - This represents a condition whereby a conclusion is reached through a managed process of systematic, objective enquiry. There is a clear difference in the behaviour of an individual who operates in this state and students unanimously appear to aspire to be able to operate in such a way. The development of the personal characteristics required however, is challenging to facilitate.

Thus we have some conditions that serve to identify the characteristics of thinking critically, but perhaps more importantly serve to identify scenarios where pseudo-critical thinking is occurring.

3.1 Managing Doubt

One aspect that is common to all of the conditions above is the feeling of doubt. Peirce explains belief as something that opposes doubt; a belief is much stronger than an opinion[17]. A doubt is a state whereby an individual feels uncomfortable until a belief is achieved. This may provide some impetus to reach a belief, even prematurely, if only to achieve a degree of comfort. To think critically, it would seem, requires effort and tolerance.

The logical conclusion to be drawn from a feeling of doubt is to conduct enquiry, with the ultimate aim of resolving the doubt. Peirce describes this stage as 'managing doubt'. Essentially we need to create the conditions where students have doubts that they can manage towards a more enlightened, reasoned understanding. The engagement with this process may demonstrate the behaviours of critical thinking.

Unfortunately the realities and practicalities of the academic environment are such that the situation is far more convoluted. Firstly, a cohort of new students, or established students faced with a new topic, is riddled with doubts. There will be doubts that are readily predicted by the tutor, as well as doubts that are personal to an individual.

Secondly individuals will succumb to different conditions based upon their existing behaviours. So an individual may be *tenacious* and therefore have a reluctance to experience any doubtful feelings. Or the *authority* of the collective may come into play and create a compelling reaction that is difficult to resist. Thus it is the management of doubt that is important for the tutor to recognise, and ultimately the challenge for the educational setting to replicate.

4 A Case Study: Teaching Introductory Programming

Sheffield Hallam University has a range of postgraduate computing programmes that recruit significant numbers of students from South East Asia and the Indian Sub Continent. Whilst these programmes foster the development of advanced computing skills, the programmes are designed to accommodate individuals who are moving from one technical discipline to another also. As such there are some individual modules common to all of the programmes:

- Advanced Learning and Study Skills;
- Research Principles and Practice;
- Industrial Expertise;
- Dissertation

These modules provide the context for the higher order thinking that is demanded by Masters level study. However, the needs of emerging recrutiment patterns is that students are no longer constrained to an annual start date and there are multiple entry points throughout the year. This means that different students will have studied different modules at any one time, and the core, masters-level concepts therefore need to be distributed across all of the provision. Whilst these modules may appear to be quite generalist, there is one other computing module, *Web Application Design and Modelling*(WADM), that attempts to normalise students' prior experience of computing across the cohort.

WADM was originally intended to convey the essential software design and principles of application programming for web-based architectures, and is heavily biased towards the acquisition of foundation computing skills. For some time there was a noticeable benefit derived from the module, but there was an immediate set of issues presented once the demographics of the recruited cohort became mostly from South East Asia and the Indian Sub Continent.

An immediate effect with the increased numbers of international students was a sharp increase in the number of plagiarism cases. It appeared that international students could not differentiate between making distinct, individual contributions to a group activity, and then presenting their individual work for the purposes of assessment.

Secondly there was a pre-occupation with the desire to learn a particular programmimng language, without wanting to learn how to program.

Thirdly it was clear that the inevitable ambiguity experienced during requirements gathering and design stages was presenting insurmountable difficulties for students. For instance, students were unwilling to publicly disagree with peers when discussing functional requirements; clearly erroneous ideas were being left unchallenged and then creating subsequent design hurdles later on.

Lastly there was a great deal of anxiety surrounding any assessment that was not a time-constrained examination. Interestingly examination performance was generally good, but coursework and practical demonstrations were poor. This last point created the greatest academic concern, since potential employee's skills are appraised by their application of knowledge to practical scenarios. Good examination performance is only a small part of a holistic assessment package, and clearly a practical demonstration can be a much more effective indicator of how the student may perform in a work-based scenario. Since the practical demonstrations were disappointing, a re-think in terms of learning and teaching approach was required.

4.1 Learning Outcomes

Learning outcomes[14] are an established means of communicating the intentions of a course of study to students. Biggs has popularised an approach referred to as 'constructive alignment'[1],[3] which is based upon the premise that the students will take more responsibility for their own learning if they understand what the intended learning outcomes (ILO) are. To quote Shuell:

> "If students are to learn desired outcomes in a reasonably effective manner, then the teachers fundamental task is to get students to engage in learning activities that are likely to result in their achieving those outcomes" [15], p429.

Often it is evident that the assessment vehicle becomes the focus of the learning. Speculation as to how an individual will be assessed can adversely affect the behaviour of an individual in terms of their learning[2]. Gibbs argues that the assessment itself can be used to increase engagement[7]. An alternative is the traditional examination, but it is widely recognised that this particular mode is baised towards testing recall[6].

Learning outcomes can take many forms and here is a typical example:

– *Apply a modelling notation to construct a design representation of an existing business function.*

In many cases, at UK level 6 (undergraduate final year) and level 7 (post-graduate) the LO will make specific reference to being critical:

– *Critically appraise and select the most appropriate design pattern*

Biggs' argument is that the LO should align with both the learning and assessment activities to assist the student to construct their own learning. There can be a tendency to include the use of 'critical' since it is generally considered to be good practice. However, if critical thinking is to be fostered it is necessary to make this clear at the outset.

4.2 Module Organisation

Our approach has been to design a curriculum that takes the students through Peirce's four conditions to help students experience and manage their own doubt. This is achieved by placing less emphasis upon a scheduled delivery of material in favour of a curriculum informed by the following principles:

- *Opportunities for self assessment* - when the module commences each student is required to assess their own strengths, firstly with reference to a professional development framework (SFIA[16]) and then using their own criteria.
- *Reflection as a professional activity* - all in-class activity is recorded within a weekly discussion forum (in the University's Virtual Learning Environment), as well as a private online learning journal in the form of a blog.
- *Open debates* - questions that raise doubts are given to the class to openly debate. Since there is a significant effect from the collective authority, debates are managed in pairs, groups of four, then the whole class.
- *Frequent, prompted reflection in action*[13] - students are actively prompted to regularly reflect *in* action, and then post session *on* reflection.
- *Impromptu presentations* - opportunities to discuss, explore and present ideas are taken frequently to provide as much practice as possible.
- *Regular micro-teaching* - students are encouraged to teach each other challenging, but focused topics.
- *Large group tutorials to replace lecture presentations* - this was a controversial decision, but it was deemed necessary to maximise the amount of time spent facilitating critical behaviours and so all module materials were provided online as an alternative.
- *Feedback for learning* - all of the above principles are directly influenced by the notion that feedback be offered, exchanged, recorded and collated for the purposes of learning. In particular such feedback should be generated by student-to-student interaction rather than being limited to more formal, summative assessment feedback.

From the tutor's perspective the curriculum has been simplified in that there is an explicit focus upon the operation and development of behaviours that are recognised to be critically informed, instead of learning and repeating newly acquired skills.

5 Discussion

We have observed benefits merely from explaining the categories of conditions that can explain different behaviours. In many cases students are taught 'how to reference' as a means of being academically honest. In the absence of any additional contextual information, tentative links are made between referencing and critical thinking (erroneously), which severely hampers progress.

Additionally we have endeavoured to extend this limited understanding by providing a range of activities where critical thinking can be applied, that might not necessarily lead to a referenced piece of writing. The opportunity to engage with, and understand the processes of thought, discussion, evaluation and group concensus serves to communicate the true value of research activity as a means of developing new cognitive skills.

Furthermore the recognition that to think critically is inextricably entwined with an individual's cultural background is something that needs to be aired at the earliest opportunity. Using Peirce's classification we have witnessed more ideas based upon *authority* with students from South East Asia and the Indian Sub Continent. This is most prevalent when ideas are being explored in public, where one might expect the authority of the masses to be greatest. This might indicate a challenging environment for an academic who wishes to teach using a critical thinking approach. However, through the use of journalling and activities designed to promote the development of reflection we have also identified that Peirce's *scientific* also exists, albeit privately. The challenge therefore is to develop activities and the necessary fora to support the more public sharing of these ideas.

6 Conclusion

The key factor from a curriculum designer's perspective is that we are not intending to teach a module based upon 'critical thinking'. We are attempting to address the cultural differences exhibited within an increasing international student population with a view to improving the study of complex computing concepts.

Peirce's conditions have served to frame some of the scenarios and conditions that are exhibited by pseudo critical thinking behaviour. This has improved the tutor's understanding of the complexities of this teaching context, and also served to improve the students' comprehension, by providing a more informed set of learning experiences. Essentially the students are learning how to design and construct a web application, yet the actual technical content delivery has been mostly relegated to online documentation. The actual delivery concentrates upon the teaching of *process*, and is characterised by a set of principles that create the setting for critical thinking to flourish.

References

1. Biggs, J. (2003). Teaching for Quality Learning at University, The Society for Research into Higher Education, Open University Press, McGraw-Hill Education.
2. Biggs, J. (2002). Aligning the curriculum to promote good learning, Constructive Alignment in Action: Imaginative Curriculum Symposium, LTSN Generic Centre, November. Available at: http://www.palatine.ac.uk/files/1023.pdf
3. Biggs, J. (1996). Enhancing teaching through constructive alignment, Higher Education, 32, 347-364.
4. Carroll, J. & Ryan, J. (eds) (2005). Teaching International Students. Improving learning for all. Oxon: Routledge.

5. Caruana, V. & Spurling, N. (2007). The internationalisation of UK Higher Education: a review of selected material: project report. York England: Higher Education Academy, 147 pages. Available at: http://www.heacademy.ac.uk/ourwork/learning/international (accessed June 2010).
6. Elton, L., and Johnston, B. (2002). Assessment in universities: a critical review of research, LTSN Generic Centre. Available at: http://eprints.soton.ac.uk/59244/01/59244.pdf
7. Gibbs, G. (1999). Using assessment strategically to change the way students learn, in: S. Brown and A. Glasner (Eds), Assessment Matters in Higher Education, Buckingham, the Society for Research into Higher Education, Open University Press.
8. Hill, R. (2009). Why should I do this? Making the information systems curriculum relevant to strategic learners, ITALICS Journal, Higher Education Academy Information and Computer Science Subject Centre, June 2009.
9. Knight, J. (2008). Internationalization: A decade of changes and challenges. International Higher Education, 50, 6-7. Available at: http://www.bc.edu/bc_org/avp/soe/cihe/newsletter/Number50/p6_Knight.htm (accessed June 2010).
10. Peirce, C. S. (1877). The Fixation of Belief, *Popular Science Monthly*, 12, pp. 1-15. Reprinted in *Collected Papers of Charles Sanders Peirce, Vol. V*, pp. 358-87.
11. Peirce, C. S. (1878). How to Make or Ideas Clear, *Popular Science Monthly*, 12, pp. 286-302. Reprinted in *Collected Papers of Charles Sanders Peirce, Vol. V*, pp. 338-410.
12. Peirce, C. S. (1934). *Collected Papers of Charles Sanders Peirce, Vol. V, Pragmatism and Pragmaticism*, C. Hartshorne and P. Weiss, eds. (Cambridge, MA, Belknap Press, Harvard University) pp. 334-335.
13. Schon, D. (1983). The Reflective Practitioner, London: Temple Smith.
14. Scroggins, W. (2004). Student learning outcomes - a focus on results (Modesto Junior College), http://www.crconsortium.com/images/SLOfocuson.pdf
15. Shuell, T. J. (1986). Cognitive conceptions of learning, Review of Educational Research, 56, 411-36.
16. Skills Framework for the Information Age (SFIA). URL: http://www.sfia.org.uk/, last accessed 28th June 2010.
17. Wells, K. (2007) Learning and Teaching Critical Thinking: From a Peircean Perspective, Educational Philosophy and Theory, 41(2), pp.201-218, Philosophy of Education Society of Australasia. Available at: http://dx.doi.org/10.1111/j.1469-5812.2007.00376.x

Learning Perspectives of Enterprise Architecture through TrAM

Ivan Launders[1], Simon Polovina[2] and Babak Khazaei[2]

[1] BT Innovate & Design, PO Box 2000, London, United Kingdom
ivan.launders@bt.com
[2]Cultural, Communication & Computing Research Centre (CCRC)
Sheffield Hallam University, Sheffield, United Kingdom
s.polovina@shu.ac.uk, b.khazaei@shu.ac.uk

Abstract. Engaging with case studies and reasoning about complex business situations allows theoretical and practical skills to be practiced and developed by students as if they were industrial practitioners. There are often several solutions to enterprise architecture designs, and being able to abstract and conceptualise about enterprise architecture develops the students' design and analytical skills in preparation of industrial practice. The Transaction Agent Modelling (TrAM) framework enhances this practice in that it deepens the design thinking for transaction agent modelling and transactions within enterprise architecture through capturing its underlying business semantics. Core to TrAM are Conceptual Graphs (CG). We reflect on good examples produced by student design groups applying TrAM to case studies as simulation of industrial practice. The paper reveals the key lessons that can be learnt from these good examples and how its guidelines can be derived to improve the use of TrAM in industry, stimulated by the student experience.

1 Introduction

Case studies provide a life like experience of situations and events by describing them in all their complexity and uniqueness [1]. By experimenting and reasoning with life-like business case studies using Conceptual Graphs (CG), business computing students can be immersed in a rich enquiry based learning environment. In particular, these students can learn that there is often no straightforward answer to aligning an enterprise system with the enterprise's actual business needs. Just like enterprise architects in industrial practice, students can discover that there are a number of ways of tackling this fundamental problem that many enterprises increasingly face. Through this experience, the students are empowered to seek an understanding that goes deeper than these apparently ad-hoc choices. They begin to appreciate that there are overarching enterprise architectures that can be applied to an individual enterprise scenarios according to different levels of abstraction, thus providing a principled approach for all enterprises [11].

Architectures for Enterprise Applications (AEA) is a final year undergraduate (Level 6) module on the BSc Computing route as well a postgraduate (Level 7) module on the MSc Advanced Computing programmes in the Department of Computing at Sheffield Hallam University. These modules use the approach described above as a simulation of industrial practice since 2006, having been updated to reflect our deepening experience over the years. The students assemble themselves into design teams and apply Transaction Agent Modelling (TrAM) to these case studies. TrAM provides the vehicle by which the AEA students explore the core semantics underlying the complexities of enterprise architectures through the notion of transactions as if they were experienced industrial practitioners (enterprise architects) [9].

Case studies are effective in providing the focused narrative account on life-business transactions allowing student design groups to engage in applying reasoning through TrAM. This theoretical framework helps to organise the thinking about transactions in enterprise architecture and provides a description of the artefacts involved. It provides a structure within which the key components of the architecture and the relationship between these components are defined [11]. Case studies allow for significantly more design practice than is possible in an industrial setting in that a greater number of student design groups can apply modelling theory to a greater number of case studies, therefore allowing guidelines to be derived that improves the use of TrAM itself. They are thus not only learning about, but informing, industrial practice. To illustrate our arguments, we now explicate our approach that includes drawing from a sample of good discoveries made by some AEA student design teams.

2 Case Studies

Student design teams are first tasked with selecting a case study to model, outlining their reasoning from choosing a particular case study. These case studies included the following that we illustrate in this paper:

- *Sheffield Hallam Mortgage & Investment Company (SHAMIC)*: This is a (fictitious) bank which specialises in lending money to customers for the purposes of buying property, and also in investment products such as equities, property, and bonds. It also provides other standard financial services including current accounts and savings accounts;
- *Mobile NHS*: The NHS requires a mobile solution for clinicians to be able to access patient records during visits to patients. Clinicians currently have fixed computer access to the patient record system at their clinics. Fixed access to patient records in specific health centres can restrict the work patterns of a mobile clinical work force;
- *Scrupulous Chemicals Company*: This is a (fictitious) manufacturer. It uses a variety of raw materials to make its products, some of which have issues surrounding their safety and environmental effects, so they have to be managed carefully.

3 Design Samples

The student design teams then apply TrAM to their case study as follows [3, 7]:

- *Model fundamentals*: this consists of a) providing a Transactional Use Case (TUC) diagram capturing the transactional behaviour of the initial use case [6, 7]. Then b) producing a close mapping between TUC and CG, transforming that transactional behaviour and adding semantics through comprehensive analysis with CG, including co-referent passing and identifying syntactic and or semantic errors in CG. Next they c) map it into a generic Transaction Model (TM) in CG, producing a specialised TM CG for their case study including a specialised CG type hierarchy for that TM;
- *Model Visualisation*: The TM is then enhanced to account for the business rules in the case study. These business rules constrain and direct the business transactions that the case study enterprise ought to engage in. As these are conditional statements (e.g. 'if-then') they are visualised in the TM by adding the visual logic of Peirce's existential graphs [10]. Validation of the case study's business rules is performed essentially by the Peirce logic operations of deiteration and double negation, combined with the CG operations of projection and specialisation, refining the TM;
- *Model Automation*: So far, other than the use of a CG 'editing' environment such as CharGer (http://sourceforge.net/projects/charger/) the process has essentially been manual i.e. without software that tests the CG or executes the CG or Peirce logic operations. Given the formal nature of CG that the productivity of computers can be applied to, the Amine CG software (http://sourceforge.net/projects/amine-platform/) is used to check and infer the statements capture in the TM. As a further part of this automation, the TM is integrated with a generic conceptual catalogue (CC) of CG. For simplicity of understanding and use in our case we use Sowa's 1984 Conceptual Catalog (SCC) being Appendix B in [10]. SCC is used to test the wider applicability of the TM, as enterprises do not exist in isolation of the wider social world but influence, and are influenced, by it [6, 7];
- *Architecture Development*: The resulting, further refined TM is then mapped into existing enterprise architecture frameworks such as Zachman or TOGAF. This is another topic and is discussed elsewhere [7].

3.1 Model Fundamentals

The TrAM process starts by gathering use cases to form a business level (or 'kite' level) UML use case diagram [2]. The use case step provides the necessary process logic that informs the TM CG, identifying the stakeholders as actors (that become the agents in the TM). This use case step provides a design team with an informal model of the case study. It offers a rapid, initial review prior to working with the formal CG notation. Good practice use case diagrams would tend to capture a limited number of components (six or seven are norms). The use case diagram provided in Figure 1 provides an example 'use case' diagram from a 2006-2007 design team where the numbers of components are much larger.

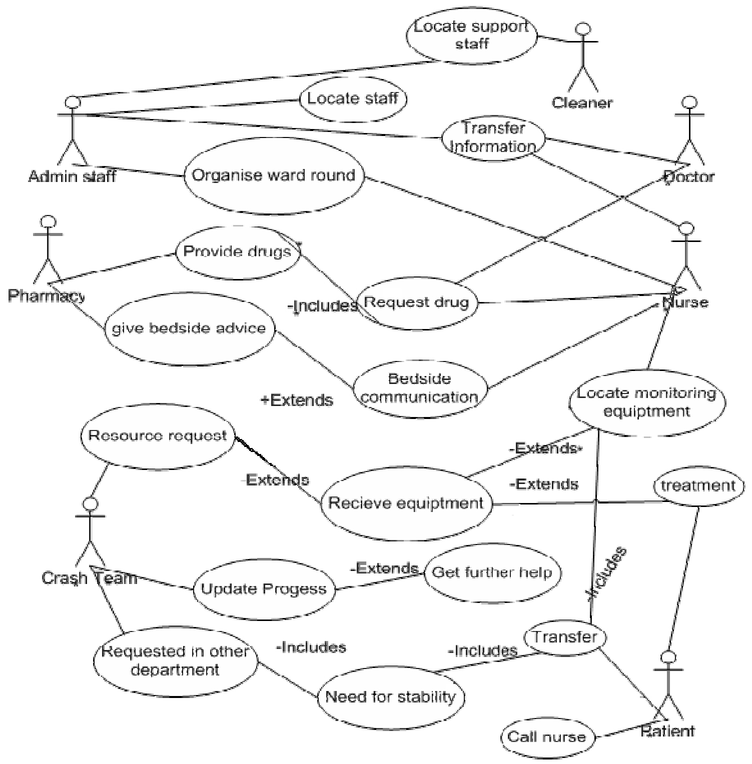


Fig. 1. Example #1, sample use case 06-07 (Mobile NHS)

Example #1 illustrates to what extent the use case diagrams may over capture (by a factor of three in this example). Example 1# captures detail beyond the key stakeholders and their transactional concepts and relationships in that it includes the 'cleaner' and 'admin staff' in the main transaction, these are supporting stakeholders which over complicate the initial capture. Example 1# is by no means a poor example, as after all it can be argued that the design team had captured the use case at a business level. To address this problem we refined the requirement in TrAM from a use case to a 'Transaction Use Case' (TUC). The TUC diagram focuses on the key transactions and their associated actors (agents), inherently reducing the number of nodes and providing a more seamless transition to the CG TM that will follow as the next step.

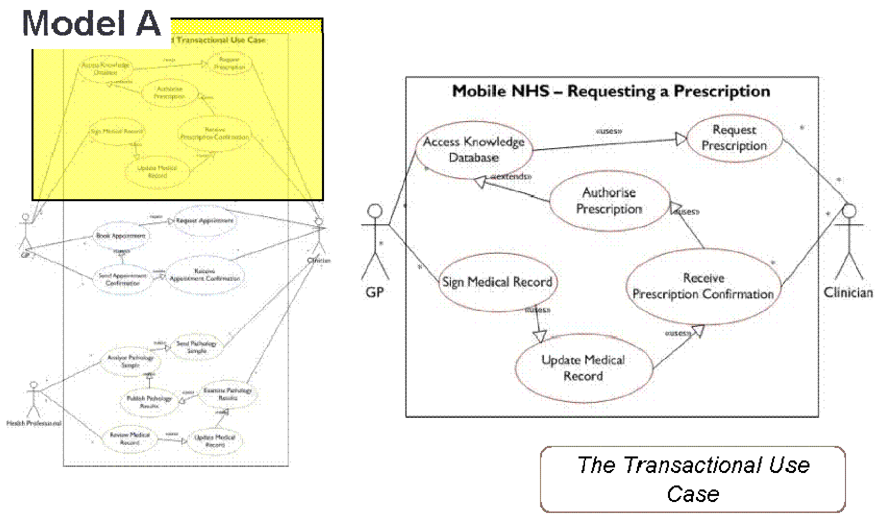


Fig. 2. Example #2 Sample transaction use case (TUC) (Mobile NHS, 2009)

Example #2 illustrated in Figure 2 thus provides more consistency and structure resulting in a good example of a TUC. It shows that the design team (here from 2008-09) have decomposed the initial 'use case' into separate 'use case' diagrams (here illustrated as 'Model A') and have clearly identified the enterprise 'system boundary' (Mobile NHS case study – Requesting a Prescription). This design team has attempted to balance the transaction and have clearly identified the stakeholders as agents. Placing the agents on each side of the system highlights the balancing of the transaction, together with links between the nodes to the agents thereby explicating the interrelationship of all the elements in the transaction to all the agents.

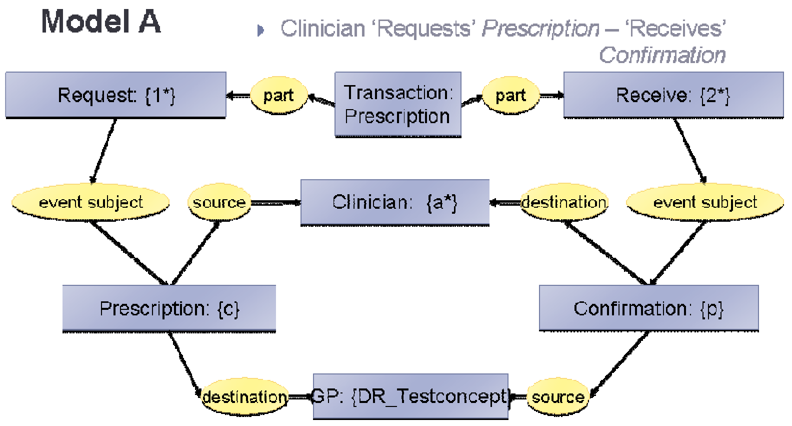
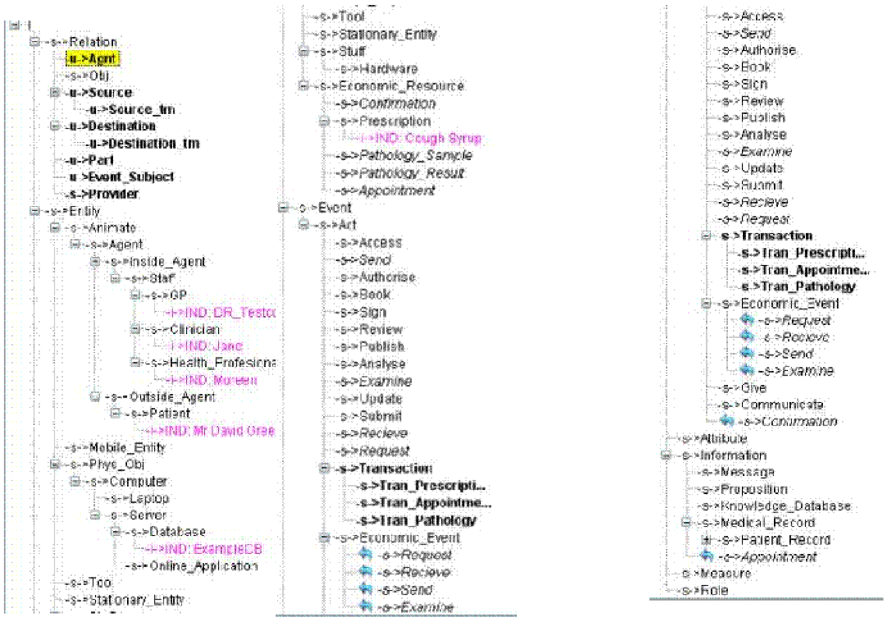


Fig. 3. Example #2 Sample integrated CG with generic TM (Mobile NHS, 2009)

This practice results in a better design understanding when translated into CG as illustrated in Figure 3 and then subsequently automated in Figures 4 and 10. Figure 4 illustrates how model automation can be applied at the outset, here in Amine (http://sourceforge.net/projects/amine-platform/). Figure 4 indicates the depth and refinement achieved showing both conceptual and relational type hierarchies.



```
I
  -s->Relation
      u->Agnt
      -s->Obj
      -u->Source
          -u->Source_tm
      -u->Destination
          -u->Destination_tm
      -u->Part
      u->Event_Subject
      -s->Provider
  -s->Entity
      -s->Animate
          -s->Agent
              -s->Inside_Agent
                  -s->Staf
                      -s->GP
                          -i->IND: DR_Testo
                      -s->Clinician
                          i->IND: Jane
                      -s->Health_Profesiona
                          -i->IND: Moreen
              -s->Outside_Agent
                  -s->Patient
                      -i->IND: Mr David Gree
      -s->Mobile_Entity
      -s->Phys_Obj
          -s->Computer
              -s->Laptop
              -s->Server
                  -s->Database
                      -i->IND: ExampleDB
                  -s->Online_Application
      -s->Tool
      -s->Stationary_Entity
      -s->Stuff
          -s->Hardware
      -s->Economic_Resource
          -s->Confirmation
          -s->Prescription
              -i->IND: Cough Syrup
          -s->Pathology_Sample
          -s->Pathology_Result
          -s->Appointment
  -s->Event
      -s->Act
          -s->Access
          -s->Send
          -s->Authorise
          -s->Book
          -s->Sign
          -s->Review
          -s->Publish
          -s->Analyse
          -s->Examine
          -s->Update
          -s->Submit
          -s->Recieve
          -s->Request
          -s->Transaction
              -s->Tran_Prescripti...
              -s->Tran_Appointme...
              -s->Tran_Pathology
          -s->Economic_Event
              -s->Request
              -s->Recieve
              -s->Send
              -s->Examine
          -s->Access
          -s->Send
          -s->Authorise
          -s->Book
          -s->Sign
          -s->Review
          -s->Publish
          -s->Analyse
          -s->Examine
          -s->Update
          -s->Submit
          -s->Recieve
          -s->Request
          -s->Transaction
              -s->Tran_Prescripti...
              -s->Tran_Appointme...
              -s->Tran_Pathology
          -s->Economic_Event
              -s->Request
              -s->Recieve
              -s->Send
              -s->Examine
          -s->Give
          -s->Communicate
          -s->Confirmation
  -s->Attribute
  -s->Information
      -s->Message
      -s->Proposition
      -s->Knowledge_Database
      -s->Medical_Record
          -s->Patient_Record
          -s->Appointment
  -s->Measure
  -s->Role
```

Fig. 4. Example #2 refined type hierarchy (Mobile NHS, 2009)

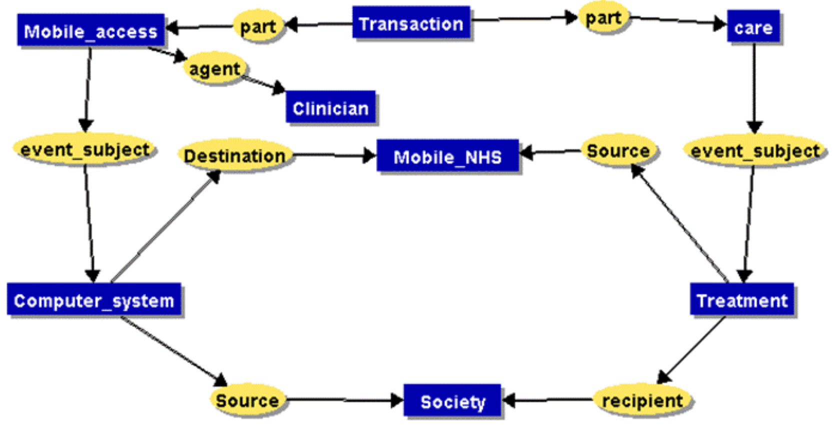


Fig. 5. Example #3 Transaction Model 08-09 (Mobile NHS)

The model fundamentals step concludes with the output of a case study specific Transaction Model (TM) based on the generic TM. Figure 5 illustrates one such example which has stayed at a sufficiently high (or business or 'kite' level [2]) to capture the key transactional concepts and relationships without over complicating the design. As we have already stated, over complicating the initial analysis steps can lead to extra work and confusion in subsequent steps. It is always possible however to make further iterations and to add in related concepts and relations once the key transactional concepts and relations are established.

3.2 Model Visualisation

Model Visualisation uses Peirce logic for visualising the inferences in CG and therefore TrAM. Hill [3] outlines the use of Peirce logic as a manual technique in TrAM. The TrAM theory from that work defines that once a TM is complete, queries are then created from use case scenarios and used to test by applying the case study's business rules, visualising them through Peirce Logic. The intention of the model visualisation step is to use Peirce logic to show contexts of knowledge elements visually dominating others. This visualises the constraints that define the transactions that this enterprise could engage in. Inference (e.g. arising from the 'if' part of a rule succeeding) is performed through deiteration and double negation in an attempt to reduce contexts (e.g. asserting the 'then' part of a business rule) enabling, together with the associated CG projection and specialisation operations, for the transaction to take place [8].

Whilst Peirce logic vividly shows the business rules integrated into the TM, because of it computational complexity the absence of software tools for automating Peirce logic makes it impossible to apply the added insight that this automation would bring. Thus model visualisation (Peirce logic) remains a manual step. Consequently the design case studies have essentially only shown examples of basic inferences such as those outlined in Figure 6 and as explained by Polovina [8].

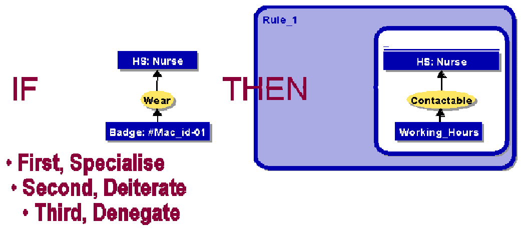


Fig. 6. Simple inferences due to 'manual' status of Peirce logic (Mobile NHS)

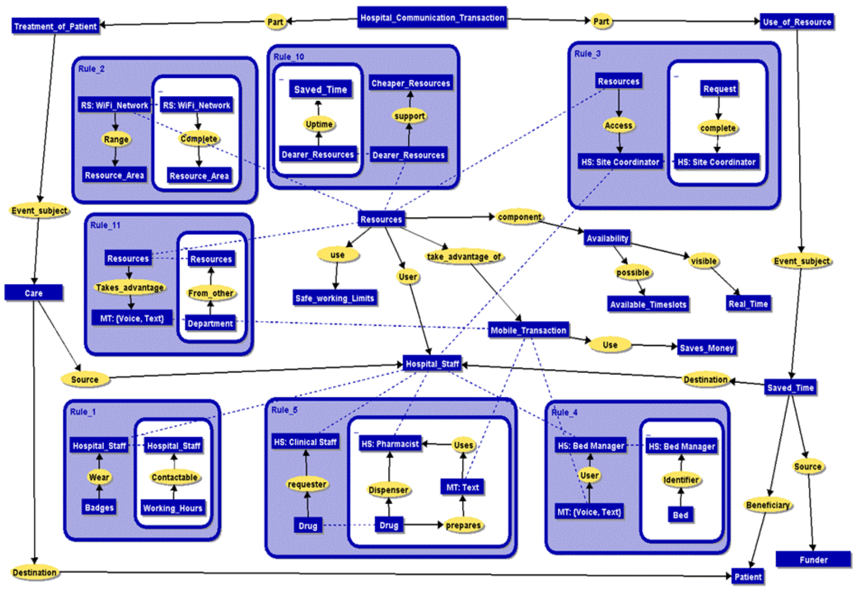


Fig. 7. Example #3 a refined TM visualised through Peirce (Mobile NHS, 2007)

Figure 7 illustrates a student design team's TM after model visualisation.

3.3 Model Automation

Model automation transfers the initial paper based transactional model analysis onto a computational model. A key learning point is that model automation using Amine enforces formalisation in terms of building the enterprise ontology, removing semantic and syntactic errors and building CG up through join operations, providing validation checks [4, 5]. Amine also tends to result in a more formal analysis than with CharGer in terms of the Type Hierarchy (TH) as demonstrated in Figure 4. The use of Amine in the automation tests the TH logic at the outset, thus it is less necessary to use CharGer in the process of creating the TH. Accordingly many design groups decided to go straight to Amine and not CharGer for their TH.

Figure 8 shows a rule extract from example 3#. In this extract a pathologist makes a report available through the communication system and a clinician accesses the system via a mobile device.

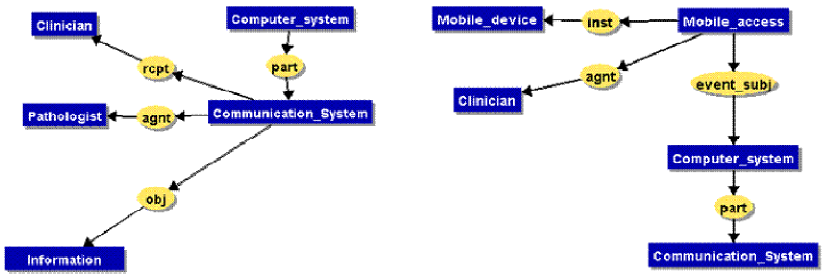
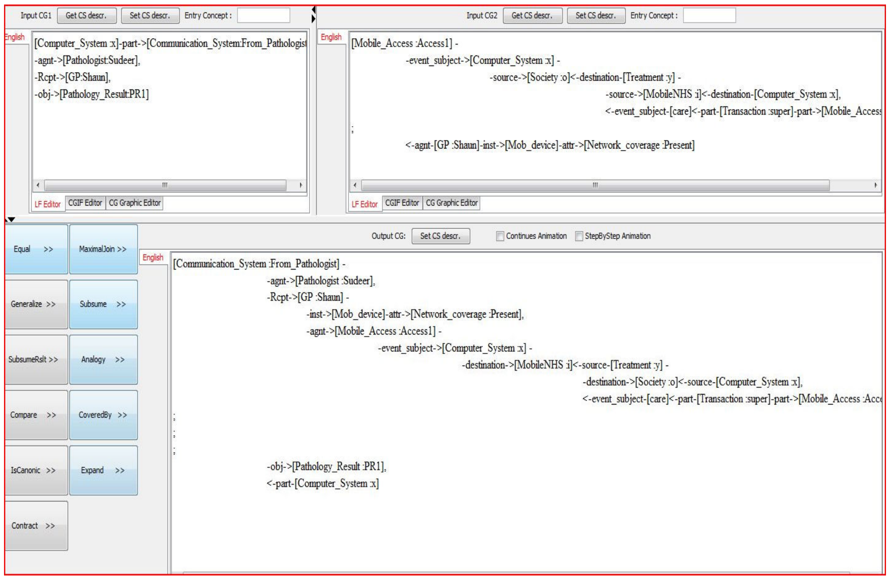


Fig. 8. Example #3 rules to automate in CharGer (Mobile NHS, 2009)

Figure 9 shows the automation of a Maximal Join operation on the CG illustrated in Figure 8 using Amine. Figure 9 illustrates how complex the resultant join can become in linear form with two relatively straight forward CG, and it could thus be argued that beyond a certain complexity automation is essential when working with CG operations. Amine allowed design teams to model CG operations and quickly observe the CG results in both linear and graphical forms.



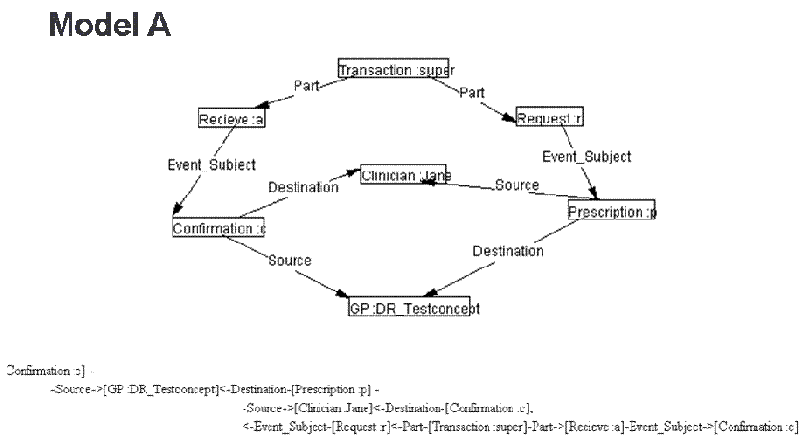
```
[Computer_System :x]-part->[Communication_System:From_Pathologist
-agnt->[Pathologist:Sudeer],
-Rcpt->[GP:Shaun],
-obj->[Pathology_Result:PR1]
```

```
[Mobile_Access :Access1] -
            -event_subject->[Computer_System :x] -
                        -source->[Society :o]<-destination-[Treatment :y] -
                                    -source->[MobileNHS :i]<-destination-[Computer_System :x],
                                    <-event_subject-[care]<-part-[Transaction :super]-part->[Mobile_Access
;
            <-agnt-[GP :Shaun]-inst->[Mob_device]-attr->[Network_coverage :Present]
```

```
[Communication_System :From_Pathologist] -
            -agnt->[Pathologist :Sudeer],
            -Rcpt->[GP :Shaun] -
                  -inst->[Mob_device]-attr->[Network_coverage :Present],
                  -agnt->[Mobile_Access :Access1] -
                          -event_subject->[Computer_System :x] -
                                  -destination->[MobileNHS :i]<-source-[Treatment :y] -
                                          -destination->[Society :o]<-source-[Computer_System :x],
                                          <-event_subject-[care]<-part-[Transaction :super]-part->[Mobile_Access :Acc
;
;
;
            -obj->[Pathology_Result :PR1],
            <-part-[Computer_System :x]
```

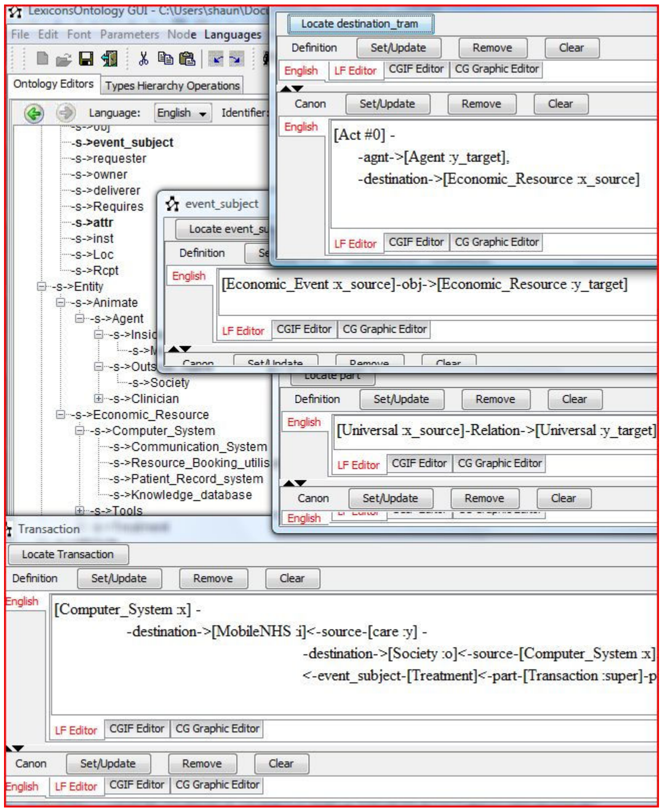
Fig. 9. Example #3 rule 4 automated in Amine (Mobile NHS, 2009)

Figure 10 shows the automation of Model A in Example 2# completing the sequences of steps from Figure 2 and Figure 3. The CG for 'Model A' is now in the form of a computational ontology and can subsequently be used to achieve projection with the inclusion of business rules.



```
Confirmation :c] -
     -Source->[GP :DR_Testconcept]<-Destination-[Prescription :p] -
                              -Source->[Clinician Jane]<-Destination-[Confirmation :c],
                              <-Event_Subject-[Request :r]<-Part-[Transaction :super]-Part->[Recieve :a]-Event_Subject->[Confirmation :c]
```

Fig. 10. Example #2 rule automation in Amine (Mobile NHS, 2009)



```
[Act #0] -
    -agnt->[Agent :y_target],
    -destination->[Economic_Resource :x_source]
```

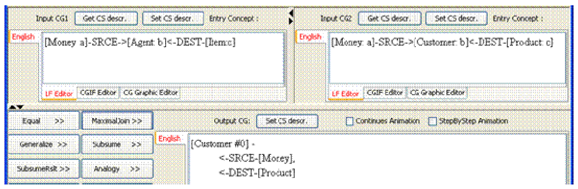
```
[Economic_Event :x_source]-obj->[Economic_Resource :y_target]
```

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

```
[Computer_System :x] -
          -destination->[MobileNHS :i]<-source-[care :y] -
                              -destination->[Society :o]<-source-[Computer_System :x]
                              <-event_subject-[Treatment]<-part-[Transaction :super]-p
```

Fig. 11. Example #3 SCC automation in Amine (Mobile NHS, 2009)

Automation provides the basis for practical re-use of the TM, adding definitions and canons to concepts in order to build the CG model (http://cg.huminf.aau.dk/Module_III/1152.html). This is demonstrated further in Figure 11 where the student design team had to create specific conceptual catalogue (CC) entries to cover the transaction model definitions which were not provided in Sowa's Conceptual Catalogue (SCC) [6, 10]. (We discussed the rationale for SCC in Section 3 above, before Section 3.1). This is a significant learning point in that the design team realised that in order to use a computational ontology they needed to develop a CC to describe the forms to apply to words and concepts therefore adding semantics to their models [6].



```
[Money a]-SRCE->[Agent b]<-DEST-[Item:c]
```

```
[Money: a]-SRCE->[Customer: b]<-DEST-[Product: c]
```

```
[Customer #0] -
        <-SRCE-[Money],
        <-DEST-[Product]
```

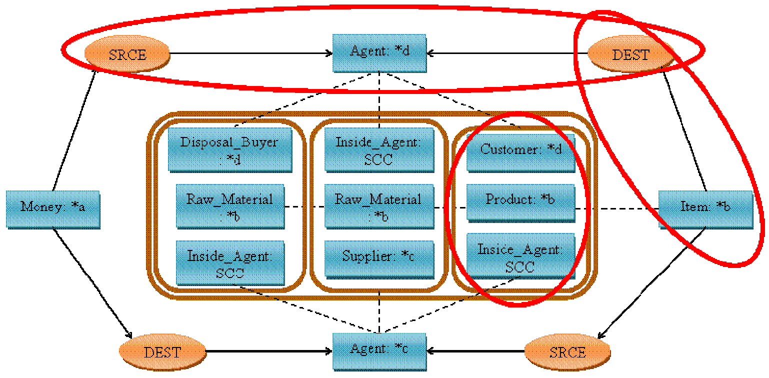


Fig. 12. Example #4 specialising the TM using Amine 2009 (Scrupulous Chemicals Company)

Automation allowed design teams to build, specialise, test and observe the results on parts of their models as they progressed. Figure 12 demonstrates a nice example of this showing both the TM and the automated model design artefacts together incorporating a variety of constructs at different levels of abstraction.

Concluding remarks

Based on a set of student design data (collected 2006-2010) we have observed that TrAM designs can exhibit attributes of a meaningful enterprise architecture development framework [7, 11]. These attributes include the following:
- consistency and structure;
- capturing the 'kite' or high level process;
- incorporating a variety of constructs at different levels of abstraction;
- a defined, enabling process for developing the architecture;
- a description of the artefacts that will be produced;
- a clearly described process.

In addition to these attributed the design data has shown evidence to inform research, for example Figure 11 illustrated a significant learning point in that this design team recognised the need to create specific Conceptual Catalogue (CC) entries to meet the transaction model definitions (adding semantics) which were not provided in Sowa's Conceptual Catalog (SCC) [7]. We have also observed the significance of students being able to view artifacts together, for example Figure 12 illustrates a good example of a design team building, testing and observing parts of their model in operation and therefore experimenting with how the model worked as a computation.

We have also identified areas where further clarity and practice is required, these include:

1. Producing the initial transactional use case that aims to identify the main transactional concepts without becoming overly complex, focusing on the 'what' (economic resources?), 'How' (Economic Events?), 'Who' (agents?), and the 'Why' (business goals).
2. The development of a re-usable and computable conceptual catalogue (for example in Amine) that goes beyond Sowa's initial Conceptual Catalogue.

It is worth noting that as a general rule student design teams who were more experimental and critical about the use of CG, TrAM and 'life like' case studies tended to produce better design results exploring to a greater depth the architectural design issues within enterprise transactions. Although we show only a small sample of the overall design effort produced by students it does illustrate that applying TrAM informs learning, teaching and assessment (LTA) and LTA informs industrial practice. From these experiences we can see how its how its guidelines can be derived to improve the use of TrAM in industry, stimulated by the student experience.

Acknowledgements

This work has been assisted by the efforts of AEA students in producing design data and by the CPLA organisation at Sheffield Hallam University.

References

[1] Elliot, J. (2009) "Case-base learning and the acquisition of practical and theoretical knowledge". Personal communication (available from the author) Encase lecture Cambridge 2009.

[2] Fowler. M. (2004) UML Distilled (Third Edition), Addison-Wesley, 103-104.

[3] Hill, R., Polovina, S., Beer, M. D. (2005) "From concepts to agents: Towards a framework for multi-agent system modelling", Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Utrecht University, Netherlands, ACM Press, 1155-1156.

[4] Kabbaj A., *Development of Intelligent Systems and Multi-Agents Systems with Amine Platform,* in 15th Int. Conf. on Conceptual Structures, ICCS'2006, Springer-Verlag, 2006.

[5] Kabbaj A., *An overview of Amine*, to appear in « Conceptual Structures in Practice: Inspiration and Applications », H. Schärfe, P. Hitzler (eds), Taylor and Francis Group, 2009.

[6] Launders, I., Polovina, S., Hill, R., (2009). "Exploring the Transaction through Automating TrAM", ICCS 2009: Supplementary Proceedings of the 17th International Workshops on Conceptual Structures; Moscow: Springer

[7] Launders, I., Polovina, S., Hill, R. (2010) "Semantics and Pragmatics in Enterprise Architecture through Transaction Agent Modelling", ICISO 2010 (http://www.orgsem.org/2010/), Aussino Academic Publishing, in press.

[8] Polovina, S., (2007). "An Introduction to Conceptual Graphs", Proceedings of the 15th International Conference on Conceptual Structures (ICCS 2007): Conceptual Structures: Knowledge Architectures for Smart Applications, July 2007, Sheffield, UK; Priss, Uta; Lecture Notes in Artificial Intelligence (LNAI 4604), Springer, 1-15.

[9] Shon, D. (1983) The reflective Practitioner, London: Temple Smith

[10] Sowa, J., (1984). 'Conceptual structures: information processing in mind and machine', Addison-Wesley.

[11] Sowa, J., Zachman, J.A. (1992) "Extending and formalizing the framework for information systems architectures", IBM systems journal VOL 31, No 3, 1992

Syllogistics with PrologPlusCG: Syllog - A Tool for Logic Teaching

(Draft July 18, 2010)

Peter Øhrstrøm, Ulrik Sandborg-Petersen and Thomas Ploug

Department of Communication and Psychology, Aalborg University
Kroghstraede 3, 9220 Aalborg East, Denmark

{poe,ulrikp,ploug}@hum.aau.dk

Abstract. This paper presents the tool, Syllog, which is a system designed to be used in support of teaching Aristotelian syllogistics as a part of an elementary logic course. Syllog is implemented using PrologPlusCG. The basic idea of the program is presented in section 1, and the complete Syllog program is included in an appendix. It is a basic idea in the program that the user activities are logged. The program is turned into an Applet. In this way the system can be made a part of an ordinary website, which may be used as a support in the context of an elementary logic course. This use of Syllog is presented and discussed in section 2. The idea is to develop various uses of Syllog in real life teaching contexts. One option is to use the program as a kind of game. It may also be possible to use Syllog for measuring the effect of a logic course introducing Aristotelian syllogistics. In addition, it may be possible to extend the system in such a way that it can be used for presenting Aristotelian syllogistics as a deductive system. These options and perspectives will be discussed in section 3.

Keywords: Syllogistics, logic teaching.

1. Logic Teaching and Aristotelian Syllogistics

The most famous part of classical logic is undoubtedly the Aristotelian syllogistics. Almost any introductory logic course includes a basic presentation of the basic ideas of argumentation and logical validity in terms of syllogisms. The aim is not only that the student should improve the quality of his own argumentation skills, but also that he should obtain a deeper understanding of basic logical and conceptual structures and ideas involved in syllogistic reasoning. The pedagogical challenge is to find a way to introduce these ideas to the individual student in a way which is as relevant and instructive as possible.

From a modern point of view classical syllogistics may be seen as a fragment of first order predicate calculus. A classical syllogism corresponds to an implication of the following kind:

(1) $(p \wedge q) \supset r$

where each of the propositions *p*, *q*, and *r* matches one of the following four forms

a(X, Y)	(read: "All X are Y")
e(X, Y)	(read: "No X are Y")
i(X, Y)	(read: "Some X are Y")
o(X, Y)	(read: "Some X are not Y")

The classical syllogisms occur in four different figures:

$(x(M, P) \wedge y(S, M)) \supset z(S, P)$ (1st figure)
$(x(P, M) \wedge y(S, M)) \supset z(S, P)$ (2nd figure)
$(x(M, P) \wedge y(M, S)) \supset z(S, P)$ (3rd figure)
$(x(P, M) \wedge y(M, S)) \supset z(S, P)$ (4th figure)

where $x, y, z \in \{a, e, i, o\}$ and where *M*, *S*, *P* are variables corresponding to "the middle term", "the subject" and "the predicate" (of the conclusion) . In this way 256 different syllogisms can be constructed. According to classical syllogistics, however, only 24 of them are valid.

The system Syllog has been implemented using PrologPlusCG (see [2,4,5,6,7]). The whole program is included in the appendix. The basic idea is to form a simple PROLOG database corresponding to the only 24 syllogisms which are valid according to classical (Aristotelian) syllogistics. This idea can be presented in the following manner, where the two first propositions (arguments) are the premises and the third proposition (argument) is the conclusion of the syllogism, cf. (1):

```
syllogism(a(M, P), a(S, M), a(S, P), "barbara, 1st figure").
syllogism(e(M, P), a(S, M), e(S, P), "celarent, 1st figure").
syllogism(a(M, P), i(S, M), i(S, P), "darii, 1st figure").
syllogism(e(M, P), i(S, M), o(S, P), "ferio, 1st figure").
syllogism(a(M, P), a(S, M), i(S, P), "barbarix, 1st figure").
syllogism(e(M, P), a(S, M), o(S, P), "feraxo, 1st figure").
syllogism(e(P, M), a(S, M), e(S, P), "cesare, 2nd figure").
syllogism(a(P, M), e(S, M), e(S, P), "camestres, 2nd figure").
syllogism(e(P, M), i(S, M), o(S, P), "festino, 2nd figure").
syllogism(a(P, M), o(S, M), o(S, P), "baroco, 2nd figure").
syllogism(a(P, M), e(S, M), o(S, P), "camestrop, 2nd figure").
syllogism(e(P, M), a(S, M), o(S, P), "cesarox, 2nd figure").
syllogism(a(M, P), a(M, S), i(S, P), "darapti, 3rd figure").
syllogism(i(M, P), a(M, S), i(S, P), "disamis, 3rd figure").
syllogism(a(M, P), i(M, S), i(S, P), "datisi, 3rd figure").
syllogism(e(M, P), a(M, S), o(S, P), "felapton, 3rd figure").
syllogism(o(M, P), a(M, S), o(S, P), "bocardo, 3rd figure").
syllogism(e(M, P), i(M, S), o(S, P), "ferison, 3rd figure").
syllogism(a(P, M), a(M, S), i(S, P), "bramantip, 4th figure").
syllogism(a(P, M), e(M, S), e(S, P), "camenes, 4th figure").
syllogism(i(P, M), a(M, S), i(S, P), "dimaris, 4th figure").
syllogism(e(P, M), a(M, S), o(S, P), "fesapo, 4th figure").
syllogism(e(P, M), i(M, S), o(S, P), "fresison, 4th figure").
syllogism(a(P, M), e(M, S), o(S, P), "camenop, 4th figure").
```

Whenever the Syllog user is confronted with a syllogism, the program checks whether it matches one of the 24 syllogisms listed about. If so, it is regarded as valid, and else it is regarded as invalid. It should be mentioned, however, that the match will also be accepted if it presupposes that the order of the premises is reversed. This means, for instance, that syllogism constructed by [e(M, S), a(P, M),o(S, P)] is regarded as the same syllogism as the one constructed from [a(P, M), e(M, S),o(S, P)]. By convention, the latter order of the premises is called *normal*. The convention is that the premise containing the grammatical subject of the conclusion (here: S) should be listed as the second in the order of the premises.

The last argument in the above database contains the classical name of the syllogism and the number of the figure. The names of the valid syllogisms were constructed by the medieval logicians. First of all, the order of the vowels in any of the names reflects the actual use of elements from {*a*, *e*, *i*, *o*} in the construction of the syllogism. In addition, the consonants in the names indicate the way the syllogisms can be demonstrated. This will be further discussed in Section 3.

The Syllog program presented above can be implemented as an Applet. In this way the system can be made part of an ordinary website to be used in support of the logic course. The Applet technique is presented and discussed in Section 2.

2. Syllog as an Applet

As discussed in [6], the PrologPlusCG 2.0 environment has been extended since [5] in various ways by the second author of the present paper. One such extension is the capability to embed any PrologPlusCG program in a normal HTML web page as a Java Applet. The PrologPlusCG 2.0 engine has been separated from the graphical user interface (GUI) into a stand-alone Java JAR file, which in turn is loaded by the Java Applet into the user's webbrowser. Based on the parameters given in the HTML web page, two additional files are loaded: First, a simple XML file which tells the PrologPlusCG Java Applet which buttons and text fields to present to the user. Second, the name of the PrologPlusCG program file which must be loaded and executed inside the user's browser. A button always executes a PrologPlusCG goal, specified in the XML file, and the values entered by the user in any input fields may be used in the arguments to these goals. The precise syntax can be found in the PrologPlusCG user's manual. All of this is readily accessible from within the

PrologPlusCG GUI, so that an applet can be easily deployed
.
For the purposes of the Syllog program, we have used no input fields and five buttons, as can be seen in Figure 1. The "New syllogism" button randomly generates a syllogism, using the write_syllogism/0 goal (see the source code of the program in the Appendix for details). The "Clear" button clears the console underneath the buttons by using the built-in clearConsole/0 predicate. The "Help" button executes the help/0 goal, which writes a help message to the user telling the user how to use the program.

The "Valid" and "Invalid" buttons execute the reaction_valid/0 and reaction_invalid/0 goals respectively. These buttons are meant for the user to tell the program whether the user thinks the syllogism generated is valid or invalid. Once one of these buttons is pressed, the program will tell the user whether the answer was right or wrong. In addition, the program will send a HTTP GET message to a server which, on the server side, logs the user's answer. The user is anonymously identified by a random number. across all answers given in a single session. In addition, the time taken to answer the question is logged, as well as what syllogism was presented. This happens in the log_answer/2 Prolog predicate.

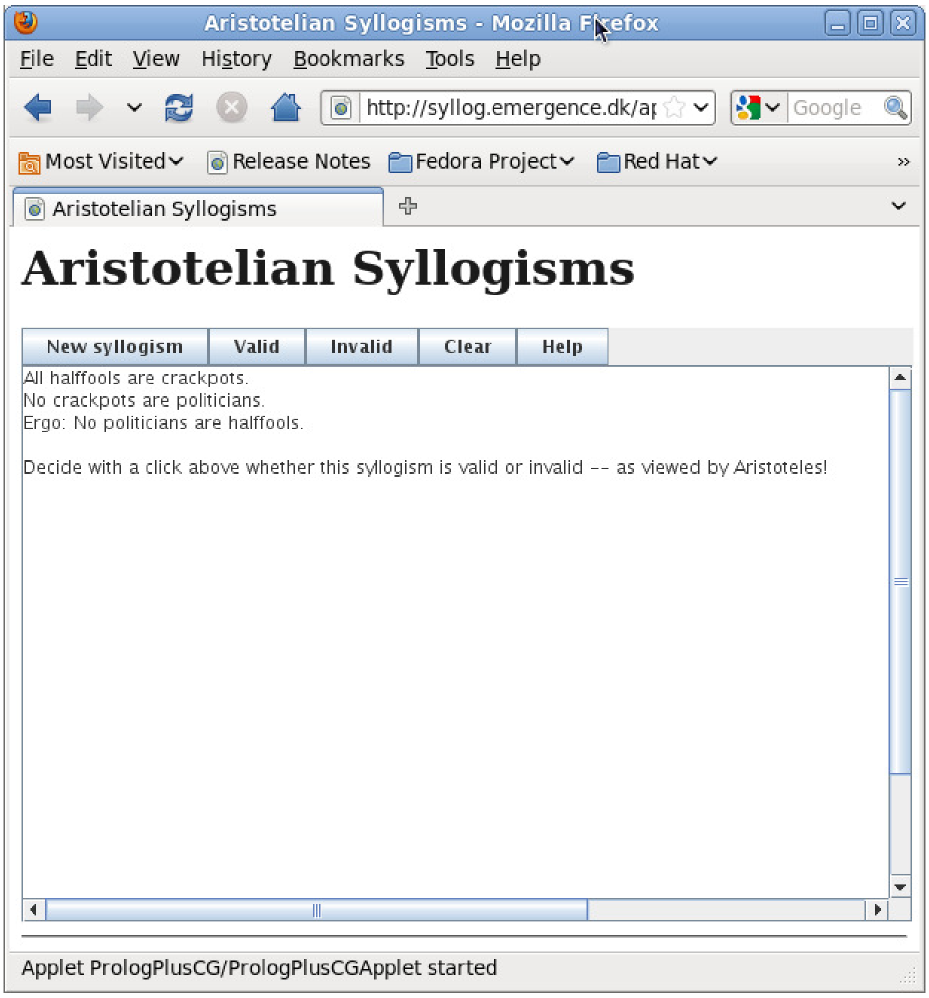
In order to follow the HTTP 1.1 specification to the letter, the HTTP GET message should, of course, have been a HTTP POST message, since something is actually changed on the server side, as required by the definitions of GET and POST in the HTTP 1.1 Protocol RFC [8]. This may be corrected in a future version of the Syllog program, but does not impact the functionality of the Syllog program.

Java applets are normally not allowed to send HTTP requests across the Internet, since this could compromise security. In order for the Java Applet to be allowed to send a HTTP GET request, the PrologPlusCG JAR must be signed with a cryptographic key, and the user must accept the certificate presented when the user opens the Java Applet. This security measure can therefore be easily overcome with the tools provided with the Java SDK. See the comments at the top of the Syllog program for how to proceed.

A preliminary version of the Syllog Java Applet can, at the time of writing, be found here:

http://syllog.emergence.dk/

The system is open for anyone to use, but the user should be aware of the fact that if the certificate presented when the user's browser loads the applet is accepted, the answers will be logged, even though the user's real identity is not known or stored.



All halffools are crackpots.
No crackpots are politicians.
Ergo: No politicians are halffools.

Decide with a click above whether this syllogism is valid or invalid -- as viewed by Aristoteles!

Applet PrologPlusCG/PrologPlusCGApplet started

Figure 1: Screenshot of the Syllog Java Applet.

3. The Use of Syllog in Logic Teaching

The idea is to use the system, Syllog, in various real life teaching contexts. The version of the system presented in the appendix may be used as a simple game. The system generates syllogisms at random and the user is in each case asked to decide whether the syllogism in question is valid or not. In the following we shall list some perspectives regarding the possible use of Syllog in logic teaching:

- Syllog (including its Prolog code) may be used to accustom the students to the basic ideas of argumentation. Thus they learn about the basic structure of an argument, i.e. the notions of premises, conclusion and inference and validity. The students' awareness of the crucial notion of a proposition following from other propositions is significantly increased.

- Syllog may be used to train the student in formalization and the recognition of formal patterns. Thus by incorporating and showing the formal syllogistic structure the student gains an increased ability to recognize the different syllogistic patterns - and when coupled with a growing awareness of when an argument is valid, the student acquires the ability to recognize valid syllogistic structures from their formal pattern. Moreover, from a formalization of arguments. In natural language the validity of arguments is typically difficult to decide, whereas simple formalization as may be incorporated in the prolog program significantly increases the students' ability to decide the validity of the arguments.
- Syllog allows for a variety of ways of learning. The students may use the program as a sort of game, and hence simply make quick guesses concerning the validity of the presented arguments. On the opposite end of the spectrum, the program may also be used to study the process of formalization step-by-step.
- Syllog may be used to stimulate interest for argumentation and the analysis of argumentation. Thus the program may incorporate simple features such as simple on-screen rewards and praise for correct evaluations of the validity of arguments.
- Syllog may be used in multiple choice testing of the students. For this purpose the program could be elaborated to include the possibility for the student to sketch counterexamples to the invalid arguments. This is an important creative element in understanding argumentation.
- It may also be possible to use Syllog for measuring the effect of a logic course introducing Aristotelian syllogistics. The students participating in such a course may simply run the program before and after the course, and since the user activity is logged by Syllog we may perhaps on this basis be able to say something about the effect of the course.
- If the program is extended, it may illustrate the system of Aristotelian syllogistic as an axiomatic system. In fact, it may serve as a basic illustration of what an axiomatic system is.

The last point deserves some further explanation. It is well known that the Aristotelian syllogistics may be seen as a deductive system (see [1]). It may even be regarded as the first axiomatic system in the history of logic. In a teaching context, it may be particularly attractive to use the system in its medieval version taking advantage of the teaching experiences through generations in medieval universities in Europe.

The last argument in each term in the database of the 24 valid syllogisms mentioned in section 1 contains the classical name of the syllogism and the number of the figure. The names of the valid syllogisms were constructed by the medieval logicians. First of all, the order of the vowels in any of the names reflects the actual use of elements from {*a*, *e*, *i*, *o*} in the construction of the syllogism. In addition, the consonants in the names indicate the way in which the syllogisms can be demonstrated from the following four axioms:

```
syllogism(a(M, P), a(S, M), a(S, P), "barbara, 1st figure").
syllogism(e(M, P), a(S, M), e(S, P), "celarent, 1st figure").
syllogism(a(M, P), i(S, M), i(S, P), "darii, 1st figure").
syllogism(e(M, P), i(S, M), o(S, P), "ferio, 1st figure").
```

It should be noted that the first letter in any of the names of the valid syllogisms belong to the set {*b*, *c*, *d*, *f*}, i.e. the set of the first letters in the names of these axioms. In the rest of the name it is indicated which rules we can use in order to prove the syllogism from the axiom beginning with first letter of the name.

The **x**-rule is based on the assumption that i(X,Y) follows from a(X,Y), and correspondingly that o(X,Y) follows from e(X,Y). Using the x-rule on the conclusion of “barbara”, we obtain:

```
syll(a(M, P), a(S, M), a(S, P))    "barbara, 1st figure"
syll(a(M, P), a(S, M), i(S, P))    Using the x-rule -> "barbarix, 1st figure"
```

A use of the **m**-rule causes the order of the premises to be reversed. The **s**-rule is a utilisation of the equivalences i(X,Y) ≡ i(Y,X) and e(X,Y) ≡ e(Y,X). Using the m-rule on “barbarix”, we obtain:

```
syll(a(M, P), a(S, M), i(S, P))    "barbarix, 1st figure"
syll(a(S, M), a(M, P), i(S, P))    Using the m-rule.
syll(a(S, M), a(M, P), i(P, S))    Using the s-rule ->  "bramantip, 4th figure".
```

The **p**-rule is the combination of the x-rule and the s-rule. The use of the rules is indicated in the name of the resulting syllogism: bra**m**anti**p.**

According to the **c**-rule we may exchange an a-proposition in the conclusion and an a-proposition in one of the premises if both are substituted by the corresponding o-propositions. Applying the c-rule on the first premise in barbara we may obtain bocardo in the 3rd figure:

syll(a(M, P), a(S, M), a(S, P))   "barbara, 1st figure"
syll(o(S, P), a(S, M), o(M, P))   c-rule used on pr.1 -> "bo**c**ardo, 3rd figure"

Similarly, we may obtain:
syll(a(M, P), a(S, M), a(S, P))   "barbara, 1st figure"
syll(a(M, P), o(M,P), o(S, M))   c-rule used on pr.2 -> "baro**c**o, 2nd figure"

Obviously, the concrete use of the c-rule is reflected in the location of the ‘c’ in the name of the resulting syllogism.

It should be noted that the c-rule is a special case of a more general rule according to which we may exchange the conclusion and one of the premises if both are negated.

Negation of the propositions in question may be carried out using the following relations: ~a(X,Y) ≡ o(X,Y) and ~i(X,Y) ≡ e(X,Y). This more general rule was not an explicit part of the medieval system although it seems that Aristotle was aware of it. Using the more general c-rule he indicated that we may in fact reduce to number of axiomatic syllogisms to just two: barbara and celarent (see [3: book 1, part 7]). In the following, however, we shall stick to the medieval system with the ordinary c-rule and the four axioms.

Using the rules on "celarent", we may obtain:

syll(e(M, P), a(S, M), e(S, P))	"celarent, 1st figure"
syll(e(P, M), a(S, M), e(S, P))	Using the s-rule -> "ce**s**are, 2nd figure"
syll(a(S, M), e(P, M), e(P, S))	Using the m+s-rules -> " ca**m**e**s**tre**s**, 2nd figure"
syll(a(S, M), e(P, M), o(P, S))	Using the m+p-rules -> " ca**m**e**s**tro**p**, 2nd figure"
syll(e(P, M), a(S, M), o(S, P))	Using the s+x-rules -> " ce**s**aro**x**, 2nd figure"
syll(a(S, M), e(M, P), e(P, S))	Using the m+s-rules -> " ca**m**ene**s**, 4th figure"
syll(a(S, M), e(M, P), o(P, S))	Using the m+p-rules -> " ca**m**ene**s**, 4th figure"

Using the rules on "darii", we may obtain:

syll(a(M, P), i(S, M), i(S, P))	"darii, 1st figure"
syll(a(M, P), a(M, S), i(S, P))	Using the p-rule -> "dara**p**ti, 3rd figure"
syll(i(M, S), a(M, P), i(P, S))	Using the s+m-rule -> "di**s**a**m**i**s**, 3rd figure"
syll(a(M, P), i(M, S), i(S, P))	Using the s-rule -> "datisi, 3rd figure"
syll(i(S, M), a(M, P), i(P, S))	Using the s+m-rule -> "di**m**ari**s**, 4th figure"

Using the rules on "ferio", we may obtain:

syll(e(M, P), i(S, M), o(S, P))	"ferio, 1st figure"
syll(e(M, P), a(S, M), o(S, P))	Using the x-rule -> "fera**x**o, 1st figure"
syll(e(P, M), i(S, M), o(S, P))	Using the s-rule -> "fe**s**tino, 2nd figure"
syll(e(M, P), a(M, S), o(S, P))	Using the p-rule -> "fela**p**ton, 1st figure"
syll(e(M, P), i(M, S), o(S, P))	Using the s-rule -> "feri**s**on, 3rd figure "
syll(e(P, M), a(M, S), o(S, P))	Using the s+p-rules -> "fe**s**a**p**o, 4th figure"
syll(e(P, M), i(M, S), o(S, P))	Using the s-rule -> "fre**s**i**s**on, 4th figure "

The above list of proofs is complete. This means that any Aristotelian syllogism may be demonstrated from the four axiomatic syllogisms, "barbara", "celarent", "darii", and "ferio" using the rules labelled as **s**, **m**, **c**, **x**, and **p**. Some syllogisms may even be proved in several different ways. It should also be mentioned that **p** is in fact just a combination of **s** and **x**.

Based on these ideas it will be possible to extend the Syllog system with a functionality for creating proofs of syllogisms. In this way, one may simply consider implementing an option allowing the student to work directly with the proofs of syllogisms from the four axiomatic syllogisms.

In modern logic the **x**-rule (and therefore also the **p**-rule) is not accepted in general. The reason is that we in modern logic normally accept that one can make true statements about all elements in an empty set. This would not be possible within

Aristotelian logic. As seen from the modern perspective (i.e. the modern understanding of the universal quantifier, "All") we have to operate with another notion of validity, according to which the syllogisms whose names include a 'x' or a 'p' will be invalid. Using this modern notion of syllogistic validity the number of valid syllogisms will be brought down to 15.

4. Conclusion

It has been shown how it is possible to build a PrologPlusCG system, Syllog, which may be used in a logic course in order to illustrate the basic structures of classical syllogistics. PrologPlusCG has the advantage that it is possible to integrate it in a web site using an Applet. Thereby, Syllog allows the student to work interactively with classical syllogistics. In addition, using the logging option of PrologPlusCG the system will allow the teacher to follow the student's learning process.

As mentioned in section 3, it seems obvious to extend the Syllog system in order to make tools presenting not only the straightforward validity of syllogisms but also syllogistics as a deductive system. However, the Syllog system may also be extended in other respects. One obvious option to implement is the distinction between the classical and the modern notion of syllogistic validity. Furthermore, various graphical tools for illustrating the logic of syllogisms (such as Euler circles, Venn diagrams, and Peircean existential graphs) may be implemented as an extension of the Syllog system.

References

1. Parry, W.T. & Hacker, E.A., Aristotelian Logic, State University of New York Press, 1991.
2. Sandborg-Petersen, U., Schärfe, H. and Øhrstrøm, P.: Online Course in Knowledge Representation using Conceptual Graphs, Aalborg University 2001-5, http://cg.huminf.aau.dk.
3. Aristotle, Prior Analytics, Translated by A. J. Jenkinson. 1994-2000. Provided by The Internet Classics Archive. Available online at http://classics.mit.edu//Aristotle/prior.html
4. Kabbaj, A., Janta-Polczynski, M., From PROLOG++ to PROLOG+CG : A CG object-oriented logic programming language. In Ganter, B., Mineau, G.W., eds.: Proceedings of ICCS 2000. Volume 1867 of Lecture Notes in Artificial Intelligence (LNAI)., Berlin, Springer Verlag (2000) pp. 540–554 .
5. Kabbaj, A., Moulin, B., Gancet, J., Nadeau, D., Rouleau, O., Uses, improvements, and extensions of Prolog+CG: Case studies.In Delugach, H., Stumme, G., eds.: Conceptual Structures: 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July/August 2001, Proceedings. Volume 2120 of Lecture Notes in Artificial Intelligence

(LNAI)., Berlin, Springer Verlag (2001) 346–359 .
6. Petersen, Ulrik, Prolog+CG: A Maintainer's Perspective. In: de Moor, Aldo, Polovina, Simon and Delugach, Harry (eds.): First Conceptual Structures Interoperability Workshop (CS-TIW 2006). Proceedings. Aalborg University Press, 2006.
7. Petersen, Ulrik, Using interoperating conceptual tools to improve searches in Kaj Munk. In: Pfeiffer, Heather D., Kabbaj, Adil and Benn, David (eds.): Second Conceptual Structures Tool Interoperability Workshop (CS-TIW 2007). Held on July 22, 2007 in Sheffield, UK, in conjunction with International Conference on Conceptual Structures (ICCS) 2007. Research Press International, Bristol, UK. Pages 45-55. ISBN: 1-897851-16-2, 2007.
8. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., Hypertext Transfer Protocol – HTTP/1.1. Request for Comments (RFC) 2616, The Internet Society, June 1999.

Appendix

The following is the full code of the Syllog program.

```
//
// In order for this to work across the Internet, the PPCGApplet.jar
// which accompanies this file must be signed with the following command:
//
// jarsigner PPCGApplet.jar YourKeyName
//
// The key can be generated (first!) with:
//
// keytool -genkey -keyalg rsa -alias YourKeyName
//

term(1, "swedes").
term(2, "politicians").
term(3, "dentists").
term(4, "halffools").
term(5, "halfminded").
term(6, "redheaded").
term(7, "thieves").
term(8, "crackpots").
term(9, "fools").

reset :- retract(reaction(_)), fail.
reset.

clearscreen :- clearConsole, reset.
```

```
write_syllogism :-
        generate_uid_if_not_there_already,
        clearConsole, reset,
        rnd(1, 4, S), term(S, _S),
        rnd(4, 7, P), term(P, _P),
        rnd(7, 10, M), term(M, _M),
        rnd(1, 5, _F), rnd(1, 13, N),
        syll(_S, _M, _P, N, _F, _U1, _U2, _U3, R),
        writeproposition(_U1), nl, writeproposition(_U2), nl,
        write("Ergo: "), writeproposition(_U3), nl,
        save_figure_and_syllnumber(N, _F),
        save_reaction(R), nl,
        save_time,
        write("Decide with a click above whether this syllogism is valid or invalid --
as viewed by Aristoteles!"),nl.

reaction_valid :- reaction(0), nl, write("Wrong answer!"), nl,
        write("This is an INvalid syllogism!"), nl, nl,
        log_answer(valid, false),
        /.
reaction_valid :- nl, reaction(R), nl, write("Correct answer!"), nl,
        write("This is a valid syllogism!"), nl,
        write("It is a "), write(R), nl,
        log_answer(valid, true),
        /.
reaction_valid.

help :- clearConsole, reset,
        write("Here you must decide, whether a syllogism, randomly generated by
        the system,"), nl,
        write("is valid or invalid."), nl,
        write("The syllogism will appear when you click the leftmost button named
        'New syllogism'."), nl,
        write("Then you must decide whether the syllogism is valid or invalid, and"),
        nl, write("indicate your decision by using the buttons named 'Valid' and
        'Invalid'."), nl,
write("The particular type of validity which must be used, is the Aristotelian one"), nl,
        write("(and not the one that follows from use of Venn diagrams)."), nl,
        write("This entails that all the concepts used in the syllogisms are assumed"),
        nl, write("to correspond to non-empty sets."), nl, nl,
        write("Click the button named 'Clear', if you wish to clear the screen."), nl, nl,
        /.

syll(S, M, P, 1, 1, au(M, P), au(S, M), au(S, P), "barbara, 1st figure").
syll(S, M, P, 2, 1, eu(M, P), au(S, M), eu(S, P), "celarent, 1st figure").
syll(S, M, P, 3, 1, au(M, P), iu(S, M), iu(S, P), "darii, 1st figure").
syll(S, M, P, 4, 1, eu(M, P), iu(S, M), ou(S, P), "ferio, 1st figure").
syll(S, M, P, 5, 1, au(M, P), au(S, M), iu(S, P), "barbarix, 1st figure").
syll(S, M, P, 6, 1, eu(M, P), au(S, M), ou(S, P), "feraxo, 1st figure").
syll(S, M, P, 7, 1, au(M, P), iu(S, M), au(S, P), 0).
syll(S, M, P, 8, 1, ou(M, P), au(S, M), eu(S, P), 0).
syll(S, M, P, 9, 1, iu(M, P), au(S, M), iu(S, P), 0).
syll(S, M, P, 10, 1, iu(M, P), eu(S, M), ou(S, P), 0).
syll(S, M, P, 11, 1, iu(M, P), iu(S, M), iu(S, P), 0).
syll(S, M, P, 12, 1, ou(M, P), au(S, M), ou(S, P), 0).
syll(S, M, P, 1, 2, eu(P, M), au(S, M), eu(S, P), "cesare, 2nd figure").
syll(S, M, P, 2, 2, au(P, M), eu(S, M), eu(S, P), "camestres, 2nd figure").
syll(S, M, P, 3, 2, eu(P, M), iu(S, M), ou(S, P), "festino, 2nd figure").
syll(S, M, P, 4, 2, au(P, M), ou(S, M), ou(S, P), "baroco, 2nd figure").
syll(S, M, P, 5, 2, au(P, M), eu(S, M), ou(S, P), "camestrop, 2nd figure").
syll(S, M, P, 6, 2, eu(P, M), au(S, M), ou(S, P), "cesarox, 2nd figure").
syll(S, M, P, 7, 2, ou(P, M), au(S, M), eu(S, P), 0).
syll(S, M, P, 8, 2, au(P, M), ou(S, M), eu(S, P), 0).
syll(S, M, P, 9, 2, ou(P, M), iu(S, M), ou(S, P), 0).
syll(S, M, P, 10, 2, iu(P, M), ou(S, M), ou(S, P), 0).
syll(S, M, P, 11, 2, iu(P, M), eu(S, M), ou(S, P), 0).
syll(S, M, P, 12, 2, ou(P, M), au(S, M), ou(S, P), 0).
syll(S, M, P, 1, 3, au(M, P), au(M, S), iu(S, P), "darapti, 3rd figure").
syll(S, M, P, 2, 3, iu(M, P), au(M, S), iu(S, P), "disamis, 3rd figure").
syll(S, M, P, 3, 3, au(M, P), iu(M, S), iu(S, P), "datisi, 3rd figure").
syll(S, M, P, 4, 3, eu(M, P), au(M, S), ou(S, P), "felapton, 3rd figure").
syll(S, M, P, 5, 3, ou(M, P), au(M, S), ou(S, P), "bocardo, 3rd figure").
syll(S, M, P, 6, 3, eu(M, P), iu(M, S), ou(S, P), "ferison, 3rd figure").
syll(S, M, P, 7, 3, au(M, P), au(M, S), au(S, P), 0).
syll(S, M, P, 8, 3, iu(M, P), au(M, S), au(S, P), 0).
syll(S, M, P, 9, 3, au(M, P), iu(M, S), au(S, P), 0).
syll(S, M, P, 10, 3, eu(M, P), au(M, S), eu(S, P), 0).
syll(S, M, P, 11, 3, ou(M, P), au(M, S), eu(S, P), 0).
syll(S, M, P, 12, 3, eu(M, P), iu(M, S), eu(S, P), 0).
syll(S, M, P, 1, 4, au(P, M), au(M, S), iu(S, P), "bramantip, 4th figure").
syll(S, M, P, 2, 4, au(P, M), eu(M, S), eu(S, P), "camenes, 4th figure").
syll(S, M, P, 3, 4, iu(P, M), au(M, S), iu(S, P), "dimaris, 4th figure").
syll(S, M, P, 4, 4, eu(P, M), au(M, S), ou(S, P), "fesapo, 4th figure").
syll(S, M, P, 5, 4, eu(P, M), iu(M, S), ou(S, P), "fresison, 4th figure").
syll(S, M, P, 6, 4, au(P, M), eu(M, S), ou(S, P), "camenop, 4th figure").
syll(S, M, P, 7, 4, au(P, M), au(M, S), au(S, P), 0).
syll(S, M, P, 8, 4, au(P, M), iu(M, S), eu(S, P), 0).
syll(S, M, P, 9, 4, iu(P, M), au(M, S), au(S, P), 0).
syll(S, M, P, 10, 4, eu(P, M), au(M, S), eu(S, P), 0).
syll(S, M, P, 11, 4, eu(P, M), iu(M, S), eu(S, P), 0).
syll(S, M, P, 12, 4, au(P, M), eu(M, S), eu(S, P), 0).

save_reaction(R) :- reset, assertz(reaction(R), ()).
```

```
// Way of saving current figure and syllogism number
syllnumber(0).
figure(0).
save_figure_and_syllnumber(_syllnumber, _figure) :-
        retract(syllnumber(X)),
        retract(figure(Y)),
        assertz(syllnumber(_syllnumber), ()),
        assertz(figure(_figure), ()),
        /.
save_figure_and_syllnumber.

writeproposition(au(x, y)) :- write("All "), write(x), write(" are "), write(y), write(".").
writeproposition(iu(x, y)) :-
        write("Some "), write(x), write(" are "), write(y), write(".").
writeproposition(eu(x, y)) :- write("No "), write(x), write(" are "), write(y), write(".").
writeproposition(ou(x, y)) :-
        write("Some "), write(x), write(" are not "), write(y), write(".").

reaction_invalid :- reaction(0), nl, write("Correct answer!"), nl,
        write("This is an INvalid syllogism!"), nl, nl,
        log_answer(invalid, true),
        /.
reaction_invalid :- reaction(R), nl, write("Wrong answer!"), nl,
        write("This is a valid syllogism!"), nl,
        write("It is a "), write(R), nl,
        log_answer(invalid, false),
        /.
reaction_invalid.


// This indicates that we have not generated a user id in this session yet.
// This will be retracted, and replaced with a randomly generated user id.
generated_uid(0).

// Generate a user id if we haven't done so already
generate_uid_if_not_there_already :- generated_uid(X), dif(0, X), /.
generate_uid_if_not_there_already :-
        generated_uid(X), eq(0, X),
        retract(generated_uid(0)),
        rnd(10000, 100000000, _newUID),
        assertz(generated_uid(_newUID), ()), /.
generate_uid_if_not_there_already.

// Get current time in Unix epoch milliseconds
get_time_now(_milliseconds_unix_epoch) :-
        // Get system time since January 1, 1970, midnight UTC.
        execMethod(_milliseconds_unix_epoch, "java.lang.System",
        "currentTimeMillis", ()).

// Save time of creation of syllogism
syllogism_creation_time(0).
save_time :- suppress(syllogism_creation_time, 1),
        get_time_now(_creation_time_in_milliseconds),
        assertz(syllogism_creation_time(_creation_time_in_milliseconds), ()), /.
save_time.

// Get time in milliseconds taken to answer question
get_time_to_answer(_milliseconds) :-
        get_time_now(_now_milliseconds),
        syllogism_creation_time(_start_milliseconds),
        val(_milliseconds, sub(_now_milliseconds, _start_milliseconds)).

log_answer(_validity, _correct) :-
        reaction(_R),
        syllnumber(_syllnumber),
        figure(_figure),
        get_time_to_answer(_timetaken),
        generated_uid(_uid),
        concat("uid=", _uid, _s1),
        concat(_s1, "&timelapse=", _s2),
        concat(_s2, _timetaken, _s3),
        concat(_s3, "&syllnumber=", _s4),
        concat(_s4, _syllnumber, _s5),
        concat(_s5, "&figure=", _s6),
        concat(_s6, _figure,_s7),
        concat(_s7, "&corr=", _s8),
        concat(_s8, _correct, _query),
        sendURL(_query).

myurlbase(_urlbase) :-
// This hack is necessary because http:// is read as http: followed by a comment!
        concat("http:/", "/syllog.emergence.dk/logsyllog/log?", _urlbase).

sendURL(_query) :-
        myurlbase(_urlstring1),
        concat(_urlstring1, _query, _urlstring),
        new(url, "java.net.URL", (_urlstring)),
        execMethod(_urlcontent, url, "getContent", ()),
        //writenl(_urlcontent), // For debugging
        destroy(url).
```