Coursework Master's Thesis Proposal

December 1999

University of South Australia
School of Computer and Information Science

Student: David Benn (9809422R)

Supervisor: Dan Corbett

Introduction

Sowa's [1984] Conceptual Structures Theory (CST) provides a means — via Conceptual Graphs (CGs) — to represent knowledge as concepts and relationships between concepts. CST's Canonical Formation Rules (CFRs) permit the modification of a Knowledge Base (KB) of CGs via such operations as copy, join and simplify. However, CST does not cater for the dynamic retraction and assertion of facts (CGs) from a KB. Mineau [1998] proposed the notion of *processes* to overcome this. Several of Mineau's papers consist of discovering some lack in CST, and providing a — usually formal, rather than ad hoc — solution. Mineau's intent seems to be in building upon CST to improve its expressive power and consistency, and his processes proposal is no exception.

Processes

[Mineau 1998] addresses one of this author's misgivings about CG collections namely that they are too static. Mineau points out that the representation of dynamic knowledge does not fall within the current scope of CST — but believes it should — and that the need for this arose from a major R&D project which aimed at capturing knowledge about corporate processes. The representation of such business processes using CGs is briefly discussed in [Gerbe´, Keller & Mineau 1998]. They give as an example, the steps involved in the fabrication of windows.

In [Mineau 1998] it was determined that dynamic knowledge could be represented by processes which Mineau defined in terms of transitions between states; previous and new states can be characterised by the pre and post conditions associated with them. The triggering of a transition to a new state depends upon the truth of these pre and post conditions. Specifically, Mineau describes these state changes in terms of the assertion and retraction of graphs in a CG system.

Mineau [1998] points out that actors and demons are specialisations of processes. Actors take concepts as input and compute individuals of a predetermined concept type. Demons also take concepts as inputs but assert or retract concepts as their result. Processes generalise this by taking CGs as input and asserting or retracting CGs as the result of their processing. The formally defined contexts of [Mineau & Gerbe´ 1997] are suggested as the basis for specifying input parameters, and for representing pre and post conditions, "…providing a formal environment for using contexts as state descriptions" [Mineau 1998 p. 67]. He does not spell out the details or consequences of using contexts in this way however. See Appendix A for a brief overview of contexts.

Mineau cites other work on process-related primitives and believes that a primary motivation in the CG community for processes being added to the CST is for the creation of an object-oriented architecture on top of a CG-based system. In [Mineau 1998] however, the emphasis is on simple primitives for modelling processes, which could be extended in arbitrary ways.

Mineau [1998] suggests that an algorithm — such as iterative factorial — can be automatically translated into a set of pre and post condition pairs. A context-free language like C or Horne Clauses (Prolog) could plausibly be, but what of more complex knowledge, e.g. corporate knowledge?

Mineau proposes a process statement as an extension to CST, and suggests a syntax for it. The statement consists of a *rule set* comprising pairs of pre and postconditions — $r_i = <pre_i; post_i>$ — such that each process has *n* transition rules (where $1 \leq i \leq n$), and it is this which Mineau suggests can be automatically generated from an analysis of the algorithm. To avoid ambiguity he permits only one parameter which is defined as a single CG independent of the rule set, and is incorporated into the precondition of the first transition rule when the process is triggered. This mechanism is described in some detail as is the invocation and clean up of a process [Mineau 1998].

There is a lengthy example which shows part of the iterative factorial algorithm being converted to a rule set for a process, the means by which that process would be triggered by an asserted CG — `[Line:#10]←(to_do)`, the pre and post conditions corresponding to each line of the algorithm, and briefly mentions the use of coreference within the process to capture the output parameter's value which will be asserted in the KB when the process exits. The example does *not* show the dynamic execution of the process, just its definition. Related to this is that the definition of a process shows only generic concepts (e.g. Line: *x) not individuals, in much the same way as a C program only shows variable definitions, not their run-time values. At run-time, this would correspond to a simple specialisation from generic concept to individual. Mineau indicates that the execution path (i.e. via transitions) is dynamically generated due to the assertion and retraction of CGs in the postconditions of each rule in the rule set. He also points out that — except when looping is required — rules don't refire once executed [Mineau 1998].

Mineau [1998] gives a synchronisation graph to illustrate how each part of the algorithm is dependent upon other parts, e.g. before the first line of the while loop's body can begin, certain variables must have been initialised and the loop condition must have been checked. The example definition shows how these dependencies are incorporated into the preconditions of rules. This representation is very similar to that used by graphical dataflow programming languages like Prograph CPX such that all inputs to a particular rule must be computed before that rule will fire. This likeness is strengthened by the fact that actors are used to represent such imperative constructs as relational and arithmetic operations (e.g. >=, !=, *).

In the conclusion of [Mineau 1998], Mineau suggests that a process should be considered in terms of its input and output rather than its internal structure, where the only side effects will be assertion or retraction of CGs in the KB. The notion of processes would seem to require minimal modification of CST while at the same time providing great expressive power.

Implementing Processes

In e-mail correspondence with Mineau [1999b] it was confirmed that there has been no implementation of the processes mechanism described in [Mineau 1998]. Mineau only showed an example of process definition, and discussed part of its execution. In his words, "…there is only a theory which needs to be refined, implemented and applied." It is proposed that the subject of this author's Master's thesis should be to provide such an implementation and apply this to a problem where dynamic knowledge is required (e.g. Sisyphus [Linster 1999] or the business processes of [Gerbe´, Keller & Mineau 1998]). In so doing, Mineau's theory will also be refined.

Here are some preliminary thoughts about implementing processes.

- Someone wishing to use processes should be able to do so by expressing them in a particular source language. Mineau [1998] provides the beginnings of such a language, but it is incomplete.
- Java could be used for platform independence (Mineau agrees with this).

- The Java based Notio framework could be used to take care of basic CG functionality (e.g. KBs, type lattices, CGs, CFRs, reading and writing of CGs in CG interchange format (CGIF) or linear form) to enable the author to focus upon the additional functionality required by processes [Southey & Linders 1999].
- A simple GUI could be provided to cater for platforms where a command line is not typical (e.g. on a Macintosh).
- Mineau and Gerbe´´s [1997] contexts could be used to implement rules such that preconditions are represented by the intention of a context and postconditions are represented by the extension of that context. The search operations defined in that paper could be used to find the matching precondition given the current state of the KB. See Appendix A for more about contexts.

Research Issues

Some questions to be answered as part of the thesis work are:

- How should processes be represented syntactically?
- How should processes be represented algorithmically?
- What is the behaviour of this implementation and does it accord with how Mineau originally thought processes should work (i.e. what was missed in [Mineau 1998]?).
- How will the implementation be complicated by permitting more than one parameter to a process? This is an issue raised in [Mineau 1998].
- How should the proposed processes implementation interact with other systems? For example, it would be desirable to be able to import CGs created by another tool (e.g. CharGer [Delugach 1999]) into the KB of the proposed tool, and to export CGs from it, as CGIF or linear form ([Sowa 1999]) in both cases.
- How will the process execution engine know when to stop: no more matching preconditions? Will it always run to completion or can its execution be stopped and restarted?
- What should be the mechanism for retraction? Mineau [1998] gives the example of `(to_do)` and `(done)` for the assertion and retraction of the lines of a factorial algorithm. Mineau [1999a] discusses the notion of explicit negation and assertion via contexts.
- The clean and efficient representation of actors and demons as specialisations of processes. Is more than one syntactic construct and/or internal representation required?
- To what extent is Mineau's process mechanism generally applicable? His original paper only gives an example of applying processes to a simple problem (factorial). It is hoped that demonstrating the process mechanism in one or more complex applications will help to answer this question.

Other interesting directions that could be pursued but for which time may not be available are:

- The automatic generation of process rule sets from arbitrary input. Mineau [1998] claims in his conclusion that the latter is possible.
- In [Mineau 1998]'s conclusion, the author points out that processes would have problems relating to soundness and completeness (e.g. do two postconditions conflict?; are they independent?). To what extent can these problems be solved? This could possibly form the basis of another thesis. [Mineau 1999a] goes some way to address this issue.
- In [Mineau 1998]'s conclusion, the author says that a goal is to develop a Conceptual Programming Environment which combines logic and imperative programming. This would be a large undertaking. How does the proposed thesis work fit into that goal?

- Mineau [1999a] proposes another use for contexts which involves imposing constraints on processes. This would require adding a step after precondition matching to ensure that applying the corresponding rule's postconditions would not lead to an invalidated KB.
- Does it make sense to think of processes producing other processes and/or taking other processes as inputs (i.e. processes as first class values)?
- [Gerbe´, Keller & Mineau 1998, p. 411] says that "Processes have not been extensively studied and only a few works are related to the representation of processes. John Sowa…presents some directions to represent processes. Dickson Lukose…and Guy Mineau have proposed executable conceptual structures." What is the relationship of Mineau's processes to other formalisms such as dataflow languages and executable CGs (e.g. see [Gerbe´, Keller & Mineau 1998])? This would require a wider literature review, not confined just to the CG literature. A more limited literature review of processes within the CG literature will presumably still be required for the current thesis however.

Appendix A — Contexts

[Mineau & Gerbe´ 1997] begins with the observation that [Sowa 1984] gave no formal definition of the notion of *context*, saying only that a context exists when a proposition is asserted. They suggest that a lack of consensus regarding the use of contexts prior to their paper also hindered any standardisation. They give a formal semantics for contexts when used for information packaging and give an application of these semantics to querying a KB.

A brief survey of the literature by the authors reveals the historical usage of contexts. They point out that while there is common agreement on Peirce's original definition of contexts as sheets of assertions, the same is not true for their semantics. [Sowa 1984] introduced a special concept type, PROPOSITION, an instance of which, *p*, has as a referent a set of CGs. The graphs are said to be true in the context of *p*. They cite an analysis of contexts by Sowa in 1995 which yielded 3 syntactic aspects: an information packaging mechanism, the contents (a set of assertions, i.e. CGs), permissible operations (e.g. import and export of assertions to and from a context). Mineau and Gerbe´ add semantics to Sowa's syntactic aspects in an attempt to bridge the gap between different viewpoints in the CG community.

Two main uses of contexts are identified. The first is the partitioning of a universe of discourse into distinct spaces (e.g. temporal) based upon the cognitive operations used by people to understand discourses. The second use is in representing texts by packaging sentences in contexts. The authors' view of contexts is expressed in terms of *worlds of assertions* that are created when the truth value of a set of assertions (CGs) is dependent upon certain conditions. Formally, a context Ci is defined to be a tuple consisting of an *intention* — I(Ci) — and an *extension* — E(Ci). The former typically consists of a single graph, while the latter is a set of graphs which are also conjunctively true in Ci. A context's intention acts to constrain the conditions under which that context exists and therefore whether the assertions it represents — its extension — hold in a given circumstance. They point out that a graph g in E(C1) could be an implicit member of E(C2) if g is a generalisation of another graph in E(C2) under the CST subsumption relation. Also, if the intention graphs of two contexts are related by subsumption, the extension of one will be a subset of the extension of another.

Since a graph can appear in more than one context, the set of all situations in which a particular graph is asserted is the *scope* of that graph. The authors give the example of Mary's thoughts forming a context in which one or more statements may be true, e.g. that Peter loves Mary. In this context, the intention is that Mary believes certain things to be true, and those things are in the extension — just one in this case: that Peter loves Mary. This may be true in the context of Mary's thoughts as well as Anne's (a friend of Mary's), but not Peter's.

The notion of a *formal context* builds upon the idea of simple contexts to yield, among other things, the scope of particular graphs. The set of all formal contexts forms a structure called a *context lattice* which can be computed automatically, providing an access structure which relates worlds of assertions. A KB can then be structured in terms of the semantics of the CGs contained within it. At the top of this lattice we find a formal context with an intention of Ø, and at the bottom, a context with an extension of Ø. This appears to be analogous to the universal and absurd types of a type lattice. The alternative is a single global context in which all graphs exist and no constraints apply.

Mineau & Gerbe´ [1997] provide a discussion about querying a KB whose contents are structured via formal contexts. They present two query equations for extracting graphs from the intention and extension of a formal context. In both cases, given a formal context and a single query graph, the set of all graphs that are the same as or a generalisation of the query graph is returned. They also give two equations for obtaining a context, given an extension or intention set. A key idea is that queries can be simplified by first identifying a target context, and that a KB's structure can be queried by searching intention and extension sets, optimising and reducing the complexity of KB querying.

References

[Delugach 1999] Delugach, H. *CharGer: A Conceptual Graph Editor*, [Accessed Online: December 1999], URL: http://www.cs.uah.edu/~delugach/CharGer/

[Gerbe´, Keller & Mineau 1998] Gerbe´, O., Keller, R., Mineau, G. Conceptual Graphs for Representing Business Processes in Corporate Memories. In *Proceedings of the 6th International Conference on Conceptual Structures*, pp. 401-415, 1998.

[Linster 1999] Linster, M. *Documentation for the Sisyphus-I Room Allocation Task*, [Accessed Online: December 1999], URL: http://ksi.cpsc.ucalgary.ca/KAW/Sisyphus/Sisyphus1/

[Mineau & Gerbe´ 1997] Mineau, G. & Gerbe´, O. Contexts: A Formal Definition of Worlds of Assertions. In *Proceedings of the 5th International Conference on Conceptual Structures*, pp. 80-94, 1997.

[Mineau 1998] Mineau, G. From Actors to Processes: The Representation of Dynamic Knowledge Using Conceptual Graphs. In *Proceedings of the 6th International Conference on Conceptual Structures*, pp. 65-79, 1998.

[Mineau 1999a] Mineau, G. Constraints on Processes: Essential Elements for the Validation and Execution of Processes. In *Proceedings of the 7th International Conference on Conceptual Structures*, pp. 66-82, 1999.

[Mineau 1999b] Mineau, G. *E-mail correspondence from Guy Mineau to David Benn*, 1st December, 1999.

[Southey & Linders 1999] Southey, F. & Linders, J.G., Notio — A Java API for developing CG tools. In *Proceedings of the 7th International Conference on Conceptual Structures*, 1999.

[Sowa 1984] Sowa, J., F. *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.

[Sowa 1999] Sowa, J.F. et al, *Conceptual Graph Standard*, draft proposed American National Standard (dpANS) NCITS.T2/98-003, 1999. [Accessed Online: November 1999], URL: http://www.bestweb.net/~sowa/cg/cgdpansw.htm