Dynamic Assertion and Retraction of Conceptual Graphs

Harry S. Delugach
Department of Computer Science
Computer Science Building Rm 109
University of Alabama in Huntsville
Huntsville, AL 35899 U. S. A.
Telephone: (205) 895-6614
Electronic mail: delugach@cs.uah.edu

I. Introduction

This paper proposes an extension to conceptual graphs that captures dynamic aspects of knowledge. This extension is necessary, because some kinds of temporal notions are not effectively represented by conceptual graphs containing only concepts, relations and actors [**Sowa84**]. The following work is based on the belief that a fundamental extension to conceptual graphs is needed that captures the temporal idea of a process or transformation. A new conceptual graph node type called a *demon* is proposed. This new node was briefly introduced in a previous workshop; the current work expands on the ideas presented there [**Delugach90**].

A description of the demon's semantics is presented, and its relationship to other temporal logics, both from within the conceptual graph community and from others. The introduction of demons was suggested by a feature in the Prolog language whereby facts may be dynamically asserted or retracted. In fact, Prolog's dynamic assertions are not limited merely to facts; one can also assert or retract functions and inference rules. Since these features are often employed in practical Prolog systems, it is reasonable to believe that such semantics are a natural part of a usable logic system. The demon structure fulfills this purposes.

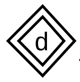
II. Expressing Temporal Knowledge

There is clearly a need to define semantics for temporal logic in relation to conceptual graphs. Several papers in a previous workshop discussed how to express temporal knowledge using conceptual graphs [**Esch90**], [**Moulin90**], and [**Zablit90**]. The work herein began as an effort to translate state transition diagrams into conceptual graph notation (in [**Delugach91**], Appendix B). While the relation (transition) could theoretically represent one state being transformed into another, there was a fundamental lack in conceptual graphs' ability to represent the change of state. This paper explains how demons capture such temporal notions.

Most current work on the structure of temporal knowledge in conceptual graphs is based on Allen [**Allen83**]. Esch and Nagle [**Esch90**] develop a notation and semantics for temporal intervals using conceptual graphs, based on Allen's taxonomy of intervals and Matuszek's temporal relations [**Matuszek88**]. Both the Moulin/Côté and Zablit papers are primarily concerned with the meaning of various tenses used in natural language. Moulin and Côté [**Moulin90**] develop an interval labeling notation to capture Reichenbach's ideas for determining tense and aspect in sentences [**Reichenbach47**]. Zablit [**Zablit90**] discusses the many shades of temporal semantics (both implied and explicit) that affect how sentences are to be conveyed by conceptual graphs.

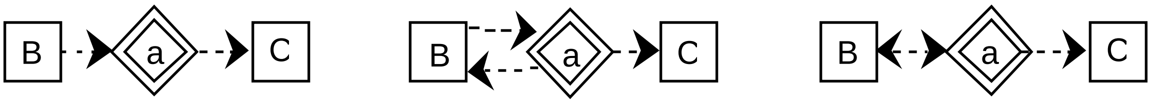
This paper will focus on how changes over time can be represented by conceptual graphs; i.e., how a graph denoting one assertion can be transformed into a new graph forming a new assertion. This paper is not concerned with temporal intervals *per se*, although they do play a part in the discussion. We are more interested in concise ways to show dynamic assertion and retraction of conceptual graphs.

III. Demon Syntax

Syntactically, a demon is similiar to an actor. A demon is represented in graphical form as a double diamond with a label, as 

. In the linear form, it is shown as a label in double brackets, as **<< d >>**. Links to and from a demon are shown as dashed arrows, similar to an actor's links. A demon is only linked to concepts; its input concepts and output concepts are denoted as with an actor. We use the term *ordinary conceptual graph* for a graph that does not contain any demons, and the term *temporal conceptual graph* for a graph containing at least one demon.

The basic syntax is shown in Figure 1. In Figure 1(a), a single concept [B] is an *input concept* to demon **<< a>>**, while the single concept [C] is an *output concept* from demon **<< a>>**. In general, there must be at least one input concept to every demon (with one exception; see initiator below), although there need not be an output concept.



(a) (b) (c)

Figure 1. Demons In Graph Form.

As illustrated in Figure 1(b), a demon may be connected to a concept that is both an input and output concept to the demon; in that case, two opposing links are shown. As a shorthand, the two links will be shown as a single two-headed arrow, as in Figure 1(c).

In the linear form, the links are shown just as the links for actors are shown.

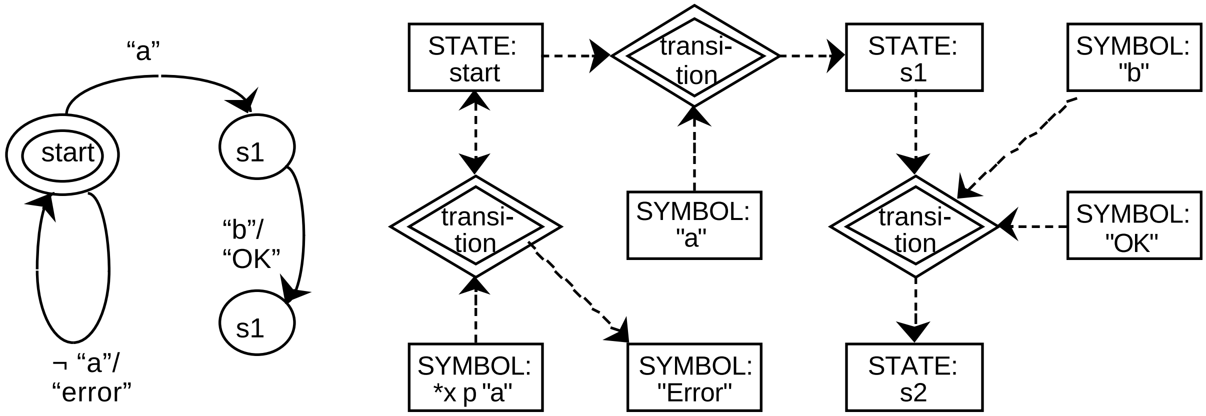
IV. Demon Semantics

The difference between a demon and an actor is that while an actor's algorithm changes only its output concepts' referents based on its input concepts (and their referents), a demon's algorithm causes each of its actual output concepts with referents to be asserted, while each of its actual input concepts is to be retracted. In a logical sense, a demon produces a non-monotonic logic [**Turner84**], whereby facts that are true at some point in time may no longer be true at some later time, when some new facts (whose truth was not known previously) are asserted. Facts that once existed on the sheet of assertion are retracted and new facts are asserted. In effect, a demon consumes all of its input concepts and creates all of its output concepts.

The meaning of Figure 1(a) above is that concept/context [B] is retracted, and the concept/context [C] is newly asserted. We do not yet make any statement about the duration of the assertion/retraction process. For the purposes of temporal reasoning, the assertion and retraction occur as an atomic action. We can say that Figure 1(a) represents the statement: *If B is ever true, then C will be true at some future time.* If there is more than one input concept, no demon action occurs until all of its input concepts have been asserted (usually by other demons being enabled).

There are many cases where we want the existence of a concept to enable the assertion of a new concept without retracting the first concept. In Figure 1(b), the existence of concept/context [B] enables demon **<< a >>** causing concept/context [C] to be asserted; the return arrow implies the "re-assertion" of concept/context [B] so that after demon **<< a >>** is enabled, both concepts exist. Although Figure 1(b) and Figure 1(c) are equivalent, we will use the latter for convenience.

As the term is used by Moulin and Côté, there is no explicit *present* in a temporal graph. An entire graph represents a series of states over time; each state forms a *present* when it appears on the sheet of assertion. Figure 2(b) shows the succession of different states in the state transition diagram of Figure 2(a).



(a) (b)

Figure 2. State Transitions Using Demons.

The demon’s semantics thus ensure that only one state exists at any given time. Note that the output symbols are permanently asserted — i.e., they are not input to any demon — so they accumulate on the sheet of assertion.

Initiator/Terminator Demons

In order to be compatible with existing conceptual graph theory, two primitive demons are proposed: an *initiator demon* with no input concepts to be automatically enabled when an ordinary graph is asserted, and a *terminator demon* with no output concepts that automatically retracts each of its input concepts (whether all of them are asserted or not).

The initiator demon is represented by the label **T**, since it represents the “top” of the temporal “hierarchy”. The terminator demon is represented by the label ⊥, to represent the “bottom” of the temporal “hierarchy”. An ordinary conceptual graph [G] is therefore considered as **<< T >>** —> [G], to indicate that it springs into being as an asserted fact, and never goes away — i.e., remains a fact until the end of time.

The terminator demon does not play a role in ordinary conceptual graphs, since there is currently no provision for retracting an ordinary graph once it has been asserted. The terminator is introduced here for the sake of completeness.

How Time Flies

Since the conceptual graph type label TIME already exists, we can establish a simple model for the passage of time. In this model, each distinct time value exists for a particular instant and then is no more, as shown in Figure 3. The interval between the time values is assumed to be some ε, where ε is sufficiently small to make the possible time values appear continuous. This corresponds to our intuitive notion of time’s passage. We call Figure 3 the *time line* or *time continuum*.

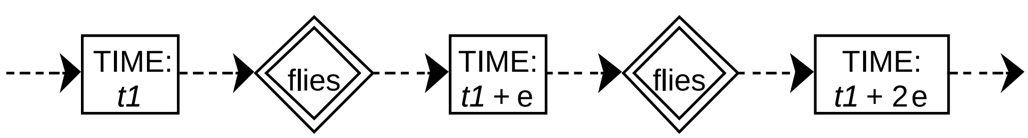


Figure 3. Time Continuum As Represented By Demons.

Wherever a [TIME] concept appears in a conceptual graph, we can consider that it appears somewhere along the time continuum. We can draw lines of identity from [TIME] concepts in the time line to any [TIME] concept in a graph. We can also define the relations (PAST) and (FUTR) as relations between two [TIME] concepts in appropriate relationship to each other on the time line. Section V discusses this further.

The continuum is shown as open-ended, since as others have noted, time has no beginning and time has no end. We thus beg the question of how to draw (even in theory) the entire time continuum. The initiator demon does not correspond to the beginning of time; its output concepts are not necessarily considered true for all times in the past. Likewise the terminator demon does not correspond to the end of time; its retractions may occur at any time along the continuum.

V. Canonical Demon Definitions

It is presumed that each demon will possess a canonical definition describing what the demon's input concepts are and what its output concepts will be. This is likely to pose additional problems, due to the large number of possible input concepts. For instance, consider the demon << burns >>. Many different materials are capable of burning. Suppose we choose just one and define << burns >> as in Figure 4:

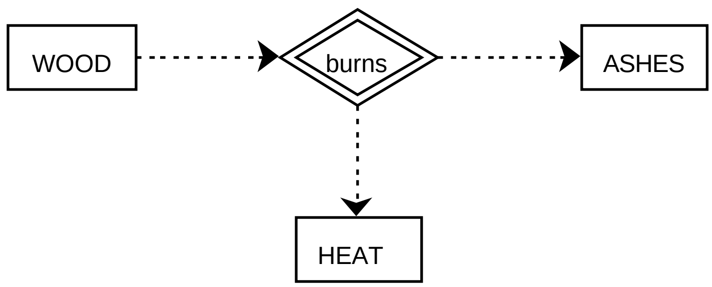


Figure 4. Demon Representing Combustion.

If the concept [WOOD] has attached relations, the question arises: what should be done about them? For instance, if the wood is brown, what color should the ashes be? One possibility is shown in Figure 5. One might make the claim that the wood's color does not matter in this definition; the resulting ashes will be black in any case. Is the attribute therefore unaffected by the process? Should the colors be part of the canonical definitions of WOOD and ASHES respectively, rather than included in the demon's definition? An actor may be inserted that alters the ASHES's COLOR's referent based on the WOOD's COLOR's referent.

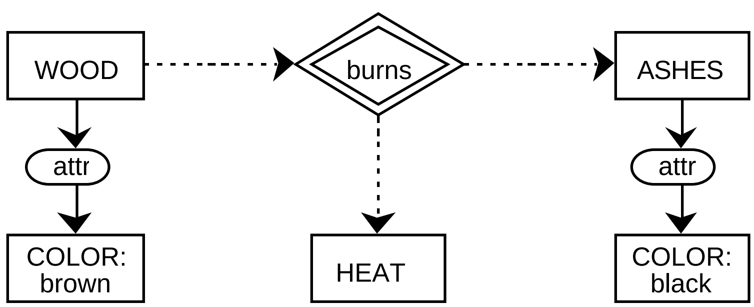


Figure 5. Relations In A Demon Definition.

There are certainly relations that would be unaffected by this process. For example, the wood's location before burning may be the same as the ashes' location after burning (unless the wind blows!). We adopt the rule that we preserve any relations that are not explicitly shown in the definition.

Relationship To Time Line

Since transformations take a finite amount of time, previous work has developed ways to represent time intervals in relation to events. The demon captures Moulin and Côté's notion of *interval* by linking [TIME] concepts as input and output concepts to a demon. For example, linking [TIME] to the previous definition results in Figure 6:

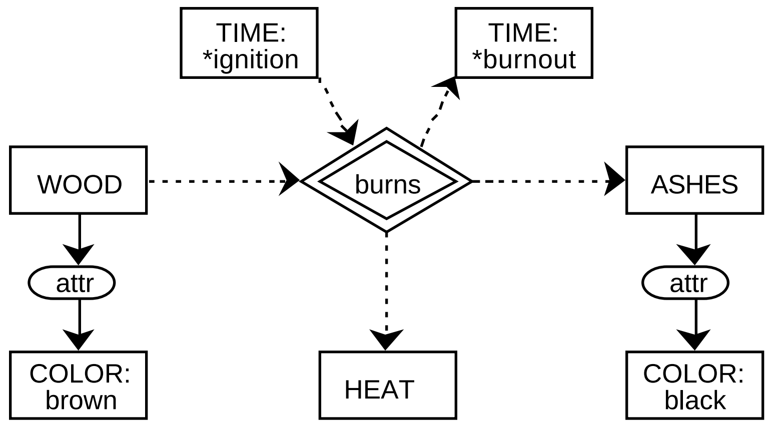


Figure 6. Time Intervals Using Demons.

The time interval is therefore simply the difference between the ignition time and the burnout time. For processes that we wish to consider instantaneous (i.e., their start and end times are indistinguishable), only one of the [TIME] concepts need be shown.

Since [TIME: *ignition] is an input concept and [TIME: *burnout] is an output concept, their temporal ordering is preserved: ignition must occur before burnout. Both of these concepts could be connected via lines of identity to the time line, in Figure 7.

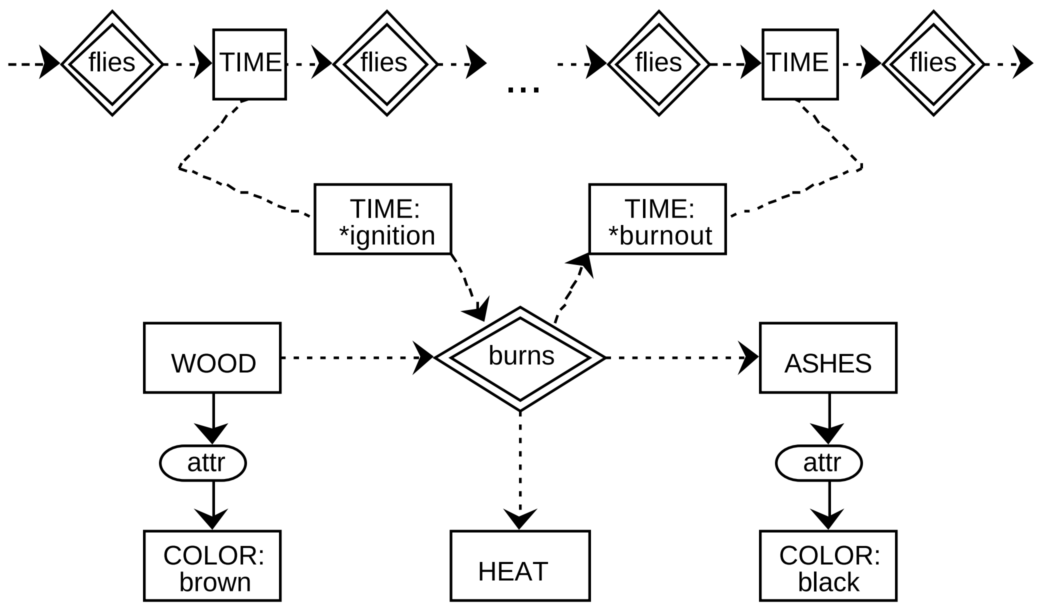


Figure 7. Demon Time Intervals On the Time Line.

In practice, the time line would not ordinarily be shown; it is included here to show explictly the relationship between the time line and the demon's input and output [TIME] concepts. Using this notion of a time interval, we need not know the exact time of each instant: we can still state that the time of ignition precedes the time of burnout.

VI. Discussion

Handling Relations and Actors

If demons are linked only to concepts, then rules must be developed to handle any relations to which the input and output concepts are linked. Clearly we can retract a relation if every one of its linked concepts is retracted by one demon; similarly we can assert a relation if all of its linked concepts are asserted at once. We must also deal with cases where only some linked concepts are asserted or retracted.

There is also the case to consider where a relation itself needs to be retracted or asserted without regard to the concepts it links. This presents problems that are both syntactic (e.g., we do not connect demons directly to relations) and semantic (e.g., what happens to the concepts that were connected to a retracted relation?)

Any rules dealing with relations must also be adapted to deal with actors, since both actors and relations are linked to concepts whose existence may be affected by the operation of a demon.

Coherence

Moulin refers to the problem of *coherence*, namely deciding whether the temporal events described by a graph conflict with our intuitive perception of time. The same problem persists when dealing with demons, although it should be pointed out that certain seemingly incoherent situations are in fact permitted when considering cyclical processes, such as in Figure 8:

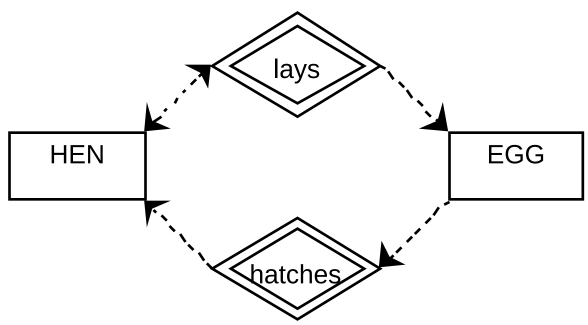


Figure 8. An Incoherent Temporal Graph?

Other Temporal Logics

In the work summarized by Karp [**Karp84**], two primitive operators are provided in order to prove temporal properties of programs. The operators are □ representing the notion of *henceforth*, and ◇ representing the notion of *eventually*. Demons represent these two notions as follows. *Henceforth* is represented by a concept [A] which is not an input concept to any demon — i.e., once asserted it cannot be retracted. *Eventually* is represented by a concept [A] that is input to some demon that has not yet been enabled.

Turner provides two additional primitive operators [**Turner84**]. In addition to *henceforth* and *eventually* (which he calls *G* and *F* respectively), he provides *P*, to denote that a proposition was true at some time in the past, and *H* to denote that a proposition has always been true in the past. *P* is represented by a concept [A] that is input to a demon that has already been enabled. *H* is represented by a concept [A] that is not output from any demon. Figure 9 summarizes these correspondences:

Turner	**Karp** (Pneuli)	**Meaning**	**As Interpreted In Temporal Graphs**
F *A*	◇ *A*	*A* is true at some future time; "Eventually"	*A* is input to a demon that has not been enabled.
P *A*		*A* was true at some past time.	*A* is input to a demon that has already been enabled.
G *A*	□ *A*	*A* will be true at all future times; "Henceforth".	*A* is not an input concept to any demon.

H *A*		*A* has always been true in the past.	*A* is not an output concept to any demon.

Figure 9. Temporal Logics As Captured By Demons.

Are Demons Necessary?

It has been suggested that demons are simply a very powerful form of an actor, since a concept (e.g., of type PROPOSITION) can have an entire sub-graph as its referent. It may be possible to thereby define a demon in terms of an actor's semantics; however, the need for a new dynamic node still exists. A demon captures the notion of an entire sub-graph being changed to another entire sub-graph. To show this with actors, an input concept's referent would become null while an output concept's previously null referent would become some non-null graph. Such semantics are stretching the limits of the transformation that an actor is supposed to indicate, and therefore the new semantics deserve straightforward representation by a new node that captures assertion/retraction.

Other Issues

Tense and Aspect. Neither tense nor aspect are handled explicitly by demons. Relationships along the time line can serve the purpose of identifying certain tenses and aspects, but this needs to be explored more fully. It is possible that we must explicitly represent the speaker of a sentence in order to effectively translate demons into natural language.

Negation and Nesting present problems for demons. Negation of a concept (or context) is not the same as the retraction of the concept (or context). Explicit negation means that an assertion is false; mere non-appearance says only that we no longer assert the fact — its truth or falsity is not known.

If a concept in a negated context is linked to a demon, we must determine the semantics of what should be the result. We must also consider what is to happen when nested contexts are linked to demons.

VII. Conclusion

The demon structure is a useful extension for representing temporal knowledge, allowing dynamic assertion and retraction of conceptual graphs. Demons allow reasoning about specific time intervals during which events occur; they also support reasoning that is independent of particular time intervals, if no specific times are involved. Temporal notions such as *past* and *future*, and other temporal logic operators are captured by the demon structure, when the interpretation of the time line is included. Many issues remain, however, in order to incorporate demons into existing conceptual graph theory.

VIII. References

[Allen83] Allen, James F., "Maintaining Knowledge About Temporal Intervals," *Comm. ACM*, Vol. 26, no. 11, pp. 832-843, Nov. 1983.

[Delugach90] Delugach, Harry, "Using Conceptual Graphs to Analyze Multiple Views of Software Requirements," *Proc. of the 5th Annual Workshop on Conceptual Structures*, Eklund, Peter and Gerholz, Laurie, eds., pp. 17-22, Linköping University, Boston & Stockholm, 1990.

[Delugach91] Delugach, Harry S., "A Multiple-Viewed Approach to Software Requirements," Ph.D. thesis, University of Virginia, Charlottesville, VA U.S.A., May, 1991.

[Esch90] Esch, John and Nagle, Timothy, "Representing Temporal Intervals Using Conceptual Graphs," *Proc. of the 5th Annual Workshop on Conceptual Structures*, Eklund, Peter and Gerholz, Laurie, eds., pp. 43-52, Linköping University, Boston & Stockholm, 1990.

[Karp84] Karp, Richard Alan, "Proving Failure-Free Properties of Concurrent Systems Using Temporal Logic," *ACM Trans. Programming Lang. and Sys.*, Vol. 6, no. 2, pp. 239-253, Apr. 1984.

[Matuszek88] Matuszek, D., Finin, T., Fritzson, R., and Overton, C., "Endpoint Relations on Temporal Intervals," Unisys Paoli Res. Ctr., Tech. Rpt. PRC_LBS_8810, June, 1988.

[Moulin90] Moulin, Bernard and Côté, Daniel, "Extending the Conceptual Graph Model for Differentiating Temporal and Non-Temporal Knowledge," *Proc. of the 5th Annual Workshop on Conceptual Structures*, Eklund, Peter and Gerholz, Laurie, eds., pp. 105-116, Linköping University, Boston & Stockholm, 1990.

[Reichenbach47] Reichenbach, H., *Elements of Symbolic Logic*, MacMillan, New York, 1947.

[Sowa84] Sowa, John F., *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley Publ. Co., Reading, Mass. U.S.A., 1984.

[Turner84] Turner, Raymond, *Logics for Artificial Intelligence*, John Wiley & Sons, New York, 1984.

[Zablit90] Zablit, P., "Conceptual Representation for Time Reference Interpretation: A Focus on Tense and Aspect," *Proc. of the 5th Annual Workshop on Conceptual Structures*, Eklund, Peter and Gerholz, Laurie, eds., pp. 323-336, Linköping University, Boston & Stockholm, 1990.