A Machine That Daydreams

Graham A. Mann

A Machine that Daydreams

Graham A. Mann[1]

School of Information Technology
Murdoch University
South St. Murdoch, Western Australia.
g.mann@murdoch.edu.au

Abstract. Some aspects of human cognitive processing across experiential cases in episodic memory seem quite different from conventional artificial reasoning by logical rules, such as that seen in CBR systems. One difference is that in humans, linkages between particular experiences can apparently be made in a number of qualitatively different ways, forming recollective chains of memories along different dimensions. Data-driven, creative, free-association from one memory to the next does not appear to be economically described by rules. Efforts to enable computers to deal with cultural content such as narratives in new ways could benefit from sequential indexing of this kind, provided that the conceptual representations are rich enough, and that a way can be found of modeling the emotional impact each elicits. A conceptual-graph-based FGP (Fetch, Generalise, Project) machine using a knowledgebase of archetypical narratives enhanced with affect is described, which demonstrates how such emotive "memory-walks" can be computed.

Keywords: cultural computing, narratives, conceptual graphs, affective computing.

1 Introduction

If computer scientists are to a stake any further claim on cultural content, it will be because i) their representations begin to include new forms, whether traditional expressions of culture that had never before been captured or entirely novel forms that had never before been conceived and ii) they discover ways to go beyond today's replay of cultural forms using linear, prescriptive representations for passive mass consumption, to entirely new paradigms of cultural expression, manipulation and engagement. Such a shift began with the manifestations of interactive computer-based artforms such as games, simulations and video performances. It can now progress to more sophisticated forms of interaction, particularly through very close attention to user's aesthetic experience, adding more sensory modalities, in-depth character depiction and very flexible user involvement in the generation of and interplay with storylines. Thought of this way, *cultural computing* involves not only the storage, manipulation and playback of cultural content such as personal histories, minority languages, myths, belief systems, poetical traditions etc. for human use, but also lays the foundations of enriched cultural processes which enable human and machine systems to create, experience and appreciate these forms in new ways.

Computer-based models of narrative processes, such as storytelling, have long been a staple of cognitive science. Even the early programs at the Yale University group lead by Roger Schank and his students such TALESPIN [1] and BORIS [2] showed that special purpose heuristic rules applied to properly-designed conceptual representations can achieve a remarkable degree of commonsense understanding of stories, in question-answering, summarization and translation tasks. More recently, Henrik Scharfe has demonstrated Prolog programs capable of identifying complex actantial roles and functions in the deep content of narratives such as Biblical texts, represented as conceptual graphs [3], while Elson and McKeown have developed SCHEHERAZADE, a promising platform for abstract conceptual processing in this domain [4]. Yet programs of this ilk seem to be missing a certain kind thinking familiar to humans. A good part of our understanding evidently draws on our store of experiences in episodic memory, through which our attention makes more of less controlled excursions that are sometimes called "memory-walks". Whether in well-organised, goal-oriented, rational trains of thought or in free associations with one idea recalling another almost at random, variations on these memory-walks are a familiar part of mental life. The mechanisms by which one experiential pattern may summon another, somehow similar pattern has also been of interest to the computing community since at least the origins of case-based reasoning (CBR) in the 1970s. “Reminding” is a recurring topic in memory-oriented views of conceptual processing [5]. In 1994, David Gelernter articulated a theory of “musing” to account for the memory walk phenomenon [6]. The theory conjectures that human minds link ideas together in different ways according to a continuum of mental focus, ranging from strictly constrained and quite logical operations on specifically selected ideas at the high end, through looser, more idiosyncratic and general connections between groups of related ideas in the middle range, to affect-linked or random "daydreaming" across the entire episodic memory at the low end.

To ratify part of the theory, a data-driven algorithm called the FGP (Fetch, Generalise, Project) machine was written to test the high-focus mode of reasoning [7]. In one experiment, the machine is tested on a series of room descriptions encoded as simple attribute pairs (e.g. ((oven yes) (computer no) (coffee-machine yes) (sink yes) (sofa no)). The program is to describe the kind of room which has a particular attribute or attributes, and is given an appropriate probe vector (e.g. (oven yes)). The FGP machine first fetches all the cases which closely match the probe attributes. It places these into a "memory sandwich" and examines this longitudinally for common attributes. If all cases in the sandwich have a common attribute, the program generalises, guessing that all cases with the probe attribute will also have the common attribute. If many of the cases have some attribute, the program "projects", or speculates that this attribute could be characteristic, and recursively uses this attribute as a probe. If the memory sandwich returned by this recursive probe is a good match to the characteristic pattern of attributes built up so far, the program accepts the putative attribute; if not, the attribute is discarded and the next attribute considered. The process continues this *fetch-generalise-project* cycle until all attributes have been accounted for, returning a composite room description.

An FGP machine may be described formally as follows: Let $\mathcal{T}$ be a feature tuple, $\mathcal{M}$ be an unordered database of feature tuples and $\mathcal{L}$ be a list of $\mathcal{T}$s, ordered by a suitably defined proximity metric to some arbitrary $\mathcal{T}$. Then let the following functions be defined:

- *fetch* ($\mathcal{T}, \mathcal{M}$) —> $\mathcal{L}$ Given a single pattern $\mathcal{T}$, returns an ordered list $\mathcal{L}$ of patterns from $\mathcal{M}$ which are closer than some threshold to $\mathcal{T}$ in the problem space.

- *generalise* ($\mathcal{L}$) —> $\mathcal{T}$ Given a list $\mathcal{L}$ of patterns, generate a new pattern $\mathcal{T}$, which captures general features of all patterns in $\mathcal{L}$. The contribution of each element of $\mathcal{L}$ to the new generalised pattern depends on its ordinal position in $\mathcal{L}$ and on the element's status as either a prototype or an ordinary case.

- *project* ($\mathcal{T}$) —> $\mathcal{T}'$ Given a single pattern $\mathcal{T}$, returns a new pattern $\mathcal{T}'$, which contains a subset of the most "evocative" features of $\mathcal{T}$, and which thereby shifts subsequent processing into new regions of the problem place.

The basic aim is to answer a query input by the user. Queries can be either a pair ($\mathcal{T}_0$, *a*?) consisting of a test feature tuple and a single attribute, or else a single tuple $\mathcal{T}_0$. An answer to the first query will be value of *a* for $\mathcal{T}_0$, while the second query examines the cases for a prototypical redescription of $\mathcal{T}_0$. In effect, it summons the memory walk process, asking "what does $\mathcal{T}_0$ bring to mind?". From the initial $\mathcal{T}_0$, the following two-step executive cycle calls the functions:

```
repeat ($\mathcal{T}_i$, $\mathcal{M}$)
    1) Extend: generalise (fetch ($\mathcal{T}_i$, $\mathcal{M}$)) —> $\mathcal{T}$
    2) Refocus: project ($\mathcal{T}$) —> $\mathcal{T}'$
            for each feature of $\mathcal{T}'$
                generalise (fetch ($\mathcal{T}'$, $\mathcal{M}$)) —> $\mathcal{T}_{i+1}$
```

The cycle iterates until either a) a value for *a* is discovered or b) an iteration produces no new conclusions. Gelernter and Fertig’s tests of the FGP machine on two databases show a predictive or diagnostic performance level close to or better than human domain experts in the same task (65% vs. a human expert's 53% correct predictions of the nationality of folkdances from a database of descriptive features, and 70% compared with a medical specialist's 60% correct differential diagnoses from descriptions of mammograms). Qualitatively, the behaviour of the program is interesting, as it interacts with the user, reporting its progress through the cycle of reminding, generalisation and speculation in a very lifelike fashion.

The ability to find logical relationships in data without any explicit rules is valuable enough, but Gelernter wanted to improve the FGP machine in two ways. First, he argued that more and better relationships could be found if the cases were better representations of real situations. This might also allow interactions with the machine to be carried out in natural language, making it easier to use. Second, he wanted a method by which affective states - emotions - can be represented in cases. These could be used to perform a kind linking at lower focus levels. In affect, the machine could make connections between situations which had the same emotional connotations. His own experiments apparently never pursued these in practice, but in Gelernter's hypothetical dialogues about such linking [6, p.145-146], a low-focussed FGP machine recognises the displeased state of a user, is reminded of a literary character by the plaintiff of a legal case study, and refuses to obey a request for financial help because the subject matter is boring!

The point of the current work is to pursue this line of development, and show how affective associations might be used to make computer reasoning more suited to cultural computing - more lifelike, more creative, and better grounded in emotions. Could a modern representation system, such as conceptual graphs [8], overcome the first problem? How can the design of the FGP machine be updated so that it functions with structured propositional forms instead of unstructured collections of features? And could the improved representations also incorporate emotional patterns, thus enabling affect-linking? In this paper, I answer these questions in the course of redesigning the FGP machine and making it work at lower focus levels.

2 Encoding of Aesop's Fables

For the current experiments, Townsend's translation of Aesop's fables [9] was chosen as an example of cultural content. Originating in approximately 600BC these are, of course, far from a new form, but here they will be encoded as conceptual representations so that they be dealt with and experienced in a new way – in a machine that meditates on their meanings and can mimic, at least in a simple form, some of the elements of human daydreaming. Practically speaking, the fables of Aesop were chosen for two reasons. First, it is difficult to overstate the impact of these narratives on the modern imagination of many societies. They form a fundamental wellspring of moral principles, animal archetypes and cautionary entertainment in both the Western and (since they were translated in Japan in the 16th century and China in 17th century) the Eastern traditions alike. Second, although the fables represent important cultural ideals, most are simple and short enough to make the task of encoding them into a consistent framework a practical proposition. For example, the following fable

The Wolf and the Crane

A Wolf who had a bone stuck in his throat hired a Crane, for a large sum, to put her head into his mouth and draw out the bone. When the Crane had extracted the bone from the throat of the Wolf and demanded the promised payment, the Wolf, grinning and grinding his teeth, exclaimed: "Why, you have surely already had a sufficient recompense, in having been permitted to draw out your head in safety from the mouth and jaws of a wolf." Moral: In serving the wicked, expect no reward, and be thankful if you escape injury for your pains. - Adapted from [9]

may be represented using a list of five conceptual graphs to encode the body of the story and one to represent the moral. The representations are composed of concept and relational instances from a catalogue of formal objects and are semantically "deep", in that they are invariant with respect to variations in surface expressive form; paraphrases, even in different languages, should map to identical graphs. For more detail on conceptual representations of this kind, see [10]. Building working conceptual structures at this level presents many difficulties. For one, a consistent ontology in the form of a conceptual and relational catalogues, containing all elements appearing in the fables is must be built, a substantial and ongoing task. Then once the elements are available for modeling, many questions remain about how best to compose logical representations of the actions, events, states, situations and juxtapositions that arise in the fables, simple as they are.

Such structural ambiguity is probably inevitable with any truly expressive conceptual language. Although the encodings of the meanings underlying the fables provided for this experiment are still crude in many ways, they are sufficient to illustrate the principles and explore some of the potential of this sort of play.

A proper demonstration of the daydreaming effect with the new FGP machine will require a good many fables to be codified. The easiest way to produce these would be to use a conceptual parser, such as SAVVY - part of the author's larger conceptual graph processing toolkit written in Common Lisp [11] - to automatically convert Townsend's natural language sentences into conceptual graph form. For example, the phrase

"…the crane extracted the bone from the throat of the Wolf…"

will, if input to the parser, generate the following graph in linear form (which is easiest for conventional computer input/output):

```
[PTRANS] –
            (AGNT) → [CRANE:#]
            (OBJ) → [BONE:#]
            (SRCE) → [THROAT] ← (PART) ← [WOLF:#]
            (DEST) → [PLACE]
             (INST) → [PHYSOBJ]
```

This is a case-role representation organised around the action primitive [PTRANS], which was retrieved from SAVVY's lexicon by the principal verb "extracted". In accordance with Schank's primitive action theory [12], PTRANS covers all instances involving the physical transfer of a object from one location to another, and carries with it a set of five thematic case roles as slots to be filled from other elements of the input expression during parsing, if possible. These are represented by the relations (AGNT), (OBJ), etc. each attached to a concept enforcing a type selection constraint for the give role e.g. the agentive role (AGNT) is attached to [ANIMATE], which can only be restricted to a more specific concept that can serve that role (of type PERSON or ANIMAL). As parsing continues, actors connected to the prototype [PTRANS] template search the input stream right and left for concepts capable of filling each role. When one is found, a restrict operation to replace the general concept is placed on the parser's recommendation list (the concept [CRANE:#], retrieved earlier from the noun phrase "the crane" replaces [ANIMATE] in the example). At the end of each textual unit, the recommendations are examined by a simple algorithm which resolves any competing claims on each role and executes a series of join operations to form the final conceptual graph. If a role's slot cannot be filled from the available information, it simply remains unspecified. Thus the ultimate destination of the bone is not specified by the above sentence, and so remains at the general but meaningful concept [PLACE], which might be read as "some unspecified place", pending further information.

The SAVVY parser can handle several kinds of phrasal structure and multi-sentence paragraph-length texts, but it cannot build very complex, nested graphical structures with subtle contexts and relationships of the kind needed for many of the sentences in Aesopian texts. Therefore, in creating the knowledgebase $\mathcal{M}$, most of the conceptual graphs were actually hand-crafted, a laborious and time-consuming process. The inadequacy of today's conceptual parsers is not especially problematic for the present experiments (except that it takes longer) though of course the problem is generally an important one; of more immediate concern is the need to convert conceptual graphs into English expressions, so that the results of processing them can be expressed in a lifelike way. The author's existing Lisp toolkit had no such generator, so one had to be built for these experiments. So far, the GRAVY English generator performs only a simple analysis on an input graph, recognises it as one or more stereotyped grammatical forms, usually organised around a principal verb or contextual form. Thematic role cases and their extension graphs are located, and a textual expression is opportunistically assembled, using words created from the conceptual type labels. This is done because although the program's lexicon contains definitions in conceptual graph form, indexing it in reverse to find suitable words for conceptual graphs is theoretically and practically troublesome. The problem with the shortcut of using type labels is that they are not really words, and choosing an apt verb phrase or noun phrase for complex aggregations is a non-trivial recognition task. Thus the above conceptual graph, when processed by the generator, produces the somewhat awkward

"The crane ptransed the bone from the throat of the wolf to a place."

Once again, although graphs of great subtlety cannot be properly expressed, GRAVY is still sufficient to add some realism to the CG-FGP machine's fragmentary day-dreams.

3 Rational Linking

What modifications need to be made to *fetch*, *generalise* and *project* to make them work in the CG-FGP machine? To begin with, they should be renamed *CG-fetch*, *CG-generalise* and *CG-project*, to distinguish them from the existing functions. We must deal with each function in turn to make it perform the equivalent function over the knowledgebase of fables, but this time at a medium or low focus. Before that, however, it is necessary to discuss some general obstacles to this development. One obstacle is that the original $\mathcal{T}$ feature tuples contained numerical weightings indicating their frequency of occurrence in generalisations, or their importance in prototypes, which information was needed by the *generalise* function. For example, if in a barrel of 100 apples half were red and the other half green, this would be represented as

((name apple 100) (type fruit 100) (colour (red 50) (green 50))).

How may this be done in a conceptual graph i.e. what is the equivalent of an attribute-value pair, and where should the weight be stored? While numerical values can easily be represented in conceptual graphs, the position advocated here is that since the numbers could mean relative importance as well as frequency counts, it is better not to try to explicitly represent them in the graph. Instead, the weights will be invisibly annotated to the concepts using an internal mechanism at implementation level, so that they do not appear explicitly the following graph, thus avoiding a commitment at the knowledge level:

```
[APPLE: {*}] -
                    (CHRC) -> [COLOUR:Red]
                    (CHRC) -> [COLOUR:Green]
```

With this kind of representation, the "scope" of the weights can be restricted to the concepts themselves, which seem to encompass both attribute and value. But with most representations, as above, relation-concept pairs are attached to a main stem as cases. To correspond strictly to attribute-value weighting, a weight would have to apply to the linkage of relation and concept only, but the picture is complicated by the realisation that the colours of apples should not be confused with the colours of other objects, though their relation-concept pairs appear the same. So perhaps the weights should cover concept-relation-concept triples. Yet this argument applies recursively, threatening to demand that weights must only apply to entire graphs, which is clearly going too far. Moderation here seems appropriate: triples seem to have enough structure to avoid confusion during weight summation, yet not so much as to blur the distinction between separate features.

The richer, more complex conceptual structures pose other problems for finding commonalities across cases, compared to the simple feature tuples in Gelernter's representations. For one, the fables are partitioned into four separate contexts. Should longitudinal comparisons across cases be made only through corresponding contexts, or should relationships to different contexts be sought? It could be argued that the value of reminding lies in its power to discover unexpected relationships across cases, so that restricting the possibilities is defeating the purpose of the exercise. On the other hand, the point of contexts is to establish boundaries of relevance, and restricting processing to only corresponding contexts greatly reduces computation during each cycle. So if only for feasibility's sake, the corresponding-context-only policy may be forced.

Furthermore, each context may contain more than one graph. How can corresponding contexts be directly compared if they have different numbers of graphs? A strict policy would forbid any such matching as potentially invalid. A more liberal approach seems preferable here. If total structure is not regarded as essential for a match, triples from any graph within a context should still probably be permitted to match pairs or triples from any graph in the corresponding context of another case. The more structure there is in a match probe, the less likely this is to lead to an invalid result, and not to permit within-context graph confusion would be to risk missing important relationships.

Better opportunities to find commonalities across cases could be had if their graphs were entered into the database in standardised forms. (Put more strongly, adequate methods have yet to be created to run all functions of the CG-FGP machine on non-standardised graphs). Though this creates an additional expressive constraint, that is not necessarily a bad tradeoff. The task might even be automated: Mineau [13] has suggested that excess variability in conceptual graphs - a range of ontological choices and structural compositions beyond that which is strictly necessary to encode different meanings - might be reduced by automatically *normalising* the graphs. He outlines eight basic methods for normalisation, including checking against a graph grammar, restriction of choice to "privileged" relations and the use of rewrite rules. If these were applied to a database immediately after input, runtime performance would not be compromised.

Consider now the modifications required for the three basic functions.

• *CG-fetch* must now take a conceptual graph $\mathcal{T}$ and return a short list of graphs $\mathcal{L}$ from $\mathcal{M}$ which are semantically close to $\mathcal{T}$ and to each other. The semantic distance measure used must take account of similarity in, but not depend on, graphical structure. Furthermore, in the high-focus mode, it must take into account the "evocativeness", or ability to add informative value, of the features within the graphs with respect to the query, which takes the form of a request to instantiate variables in $\mathcal{T}$ with one or more specific concepts.

Since this mapping is expected to be far more computationally expensive than that of *fetch,* an effort must be made to define an efficiently computable measure D of distance between conceptual graphs. Traditionally, this has been done by taking the sum over corresponding concepts in the two graphs of the distances on the type hierarchy between each concept's type and the most specific type which subsumes them both (minimal common generalisation). However, Wuwongse and Niyomthai [14] have that cogently argued that because concepts are not regarded as equally salient in human intuitions of similarity, so the orthodox semantic distance (SD) sum should be influenced by perceived importance values (IV) associated with the concepts:

D =□=*1*□IV□ - SD□ (1)

Normally the salience term IV presents a problem, because it requires a number to be assigned to each concept in the system manually, based on psychological data. Here we have the opportunity to automatically weight each distance measure by the "evocativeness" of $\mathcal{T}$ with respect to a query. This is a measure of how strongly and clearly $\mathcal{T}$ brings to mind a distinct value for the query. Fertig and Gelernter [7] used an information-theoretic formula in which the information gain (negative entropy) on the query from $\mathcal{T}$ is defined as:

□ = *1*ln □ −i□□ ln □ □ □+ln □ (2)

where

N is the total number of possible values for the goal in $\mathcal{M}$

n_i is the number of times goal value i appears in the top-cluster

T is the total number of goal values found in the top-cluster

S ranges from a 0 (maximum evocativeness) to 1 (minimum evocativeness). An equivalent computation can be made over conceptual graphs. Evocativeness E can then be included as a weight for each difference in our graph-matching measure with respect to the goal:

$$D = \square = 1 \square SD\square\ (1-S\square) \qquad (3)$$

Note that unlike Wuwongse and Niyomthai's importance values these measures of salience are automatically computed with respect to a given goal, then multiplied by the semantic distance measure. We might also require a way of ensuring that the graphs in $\mathcal{L}$ form a tight 'top-cluster', delineated from others in $\mathcal{M}$ by a discernable natural gap in semantic distances. Such a gap can be detected by performing a crude cluster analysis on the elements of the summed semantic distances. It results in a small number of close graphs being included in $\mathcal{L}$ each time *CG-fetch* is run.

• *CG-generalise* The requirement to collapse the ordered list $\mathcal{L}$ of $\mathcal{T}$s into one which captures essential elements of each presents a real problem if multiple graphs are allowed in case contexts. While it is possible to imagine techniques which find the correct correspondences for longitudinal coalescence across multi-graph contexts, the problem is formidable, and when taken together with the other problems this causes, a good case emerges for requiring contexts to be standardised (or normalised) single graphs as discussed above. With that constraint the problem simplifies to the extraction of high-frequency or weighty features and the elimination of contradictory features. These features are then instantiated into a template for this context's standard graph form. Since $\mathcal{T}$s which are ranked higher on $\mathcal{L}$ exert a greater influence than later $\mathcal{T}$s, this is best organised as an iterative process.

```
Begin $\mathcal{T}'$ as a new copy of the template for the current context
 For each $\mathcal{T}_i$ in $\mathcal{L}$
   For each concept j in $\mathcal{T}'$
     $\text{Weight}_j = \max \dfrac{\text{count}(j, \mathcal{T}_i) * \text{position}(\mathcal{T}_i, \mathcal{L}) * \varepsilon}{\text{length}(\mathcal{L})}$
Eliminate all features of $\mathcal{T}'$ with a weight $<\theta$
```

This algorithm adds new features to the evolving generalisation, but records agreements and disagreements in later graphs, allowing features with contradictory values to eliminated at step 4. ε and θ are arbitrary thresholds which are easily set to properly detect commonality or contradiction, respectively, across small lengths of $\mathcal{L}$.

• *CG-project* This function must take a single graph and return a subgraph containing its most evocative features. Evocation has been discussed already; the feature's attribute must be evocative with respect to the query For a given relation-concept pair, the possible different concept attachments could be simply counted. A feature can then be excluded from $\mathcal{T}'$ if

$$(1 - S) < \alpha \qquad (4)$$

where

α is arbitrary threshold selected to allow interesting features to prevail.

The structure of the returned subgraph must resemble something of the original. With stem-and-case structures, this is simply a matter of preserving the stem, and pruning some of the relation-concept cases away. Theoretically, any substructure may contain regularities for future processing, so it is not essential; it may simply help with higher level processing, such as natural language description.

4 Affective Linking

A key aspect of human cognition which must now be brought into the picture is emotional reaction, which are commonly experienced from (indeed, are much of the essence of) stories. By so doing, we can hope to provide for i) the evaluation of otherwise unclassifiable objects, events and situations ii) distinctive signaling of important bodily and mental states for behaviour control and

iii) positive and negative reinforcement for learning. Here, we focus on the value of affect as an important linkage between recalled memories. While any kind of content would serve, the basic idea is that at medium or low focus, the connections between one episode and the next could be essentially that they made the experiencer "feel the same", and thus allow them to form successive recollections as in a day-dream.

Emotional states could be simulated in our CG-FGP machine in two basic ways. Given a model of what (human) emotional states exist and how they relate to each other, a modeler could read each fable in the knowledgebase, and try to model his or her own reactions, creating a new conceptual graph and attaching it as special context within the episode. Given a good theory of emotions, this has the advantage of simplicity, and the machine's reactions should be at least recognisable to another human. But it requires extra human effort at the point of input of all new cases, and is essentially derivative. Alternatively, one could wish for a method of having the machine generate its own emotional graphs in response to cases. To do that, the model would need an attitudinal policy, which maps objects, events and situations onto a set of affective states. The mapping could be based on "survival-value" or other machine-oriented teleological principle. Though they could be difficult to create, and the resultant reactions are not guaranteed to be recognisable to a person, but it requires no extra effort at input.

The reader might wonder what justifies any effort at including affective states at all. Are these not simply more features of input for the CG-FGP machine to process? In a way, that is all they are. But emotions do not originate in the external state of affairs represented in the cases. They arise in a classificatory mechanism, whether in the human emoter or inherent in the system itself. They add important information about the relationship between a cognitive agent and a particular aspect of the world. If systematically assigned, affective commonalities might be discoverable between objects, events and situations which were otherwise unrelated, provided only that similar emotive representations were semantically close. For example, suppose a machine could capture and express reactions to situations in a fable where, despite a righteous act on the part of an actor he had not been well treated by others, or by the gods. Then one case of injustice could bring to mind another, even if the narratives had little else in common.

Here we experiment with two specific objects of each narrative, the protagonist and the antagonist. (Actually, not all of Aesop’s fables have an animate antagonist; in many the protagonist is a victim of circumstance, or learns a lesson through misadventure. In such cases, the general concept [SITUATION:#] has been used to fill the objective case role.) This will be a hand-crafted data structure expressing the author’s compassion, trust, aggressiveness, etc. toward each of them, and stored as two distinct contexts which can be longitudinally searched in the CG-FGP process.

Affective states are modeled by adapting Plutchik’s circumplex model of emotions [15]. Essentially, this holds that four primary dimensions, or scales, of affect can be recognised in humans. At any instant, an emotional reaction toward a target object can be described by four points: one on each scale. The extremities of these scales form bipolar opposites, so that naturally competing emotions may not be experienced simultaneously. The states can be arranged in a circle by similarity, like the colors on a color wheel (Figure 2). If integers are assigned to represent the

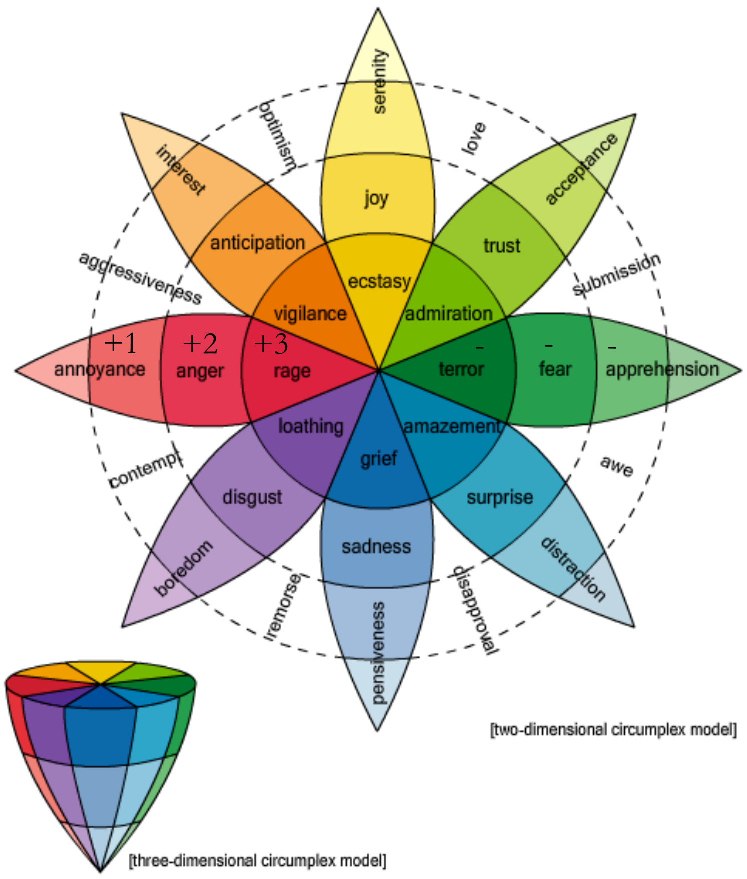

Fig. 2. Robert Pluchick's Wheel of Emotion (after Plutchick, 1980) represents affective states as four scales of polar opposites arranged around a circle based on the similarity of the states. The wheel can be folded into a 3D cone. The radial axis represents the intensity of arousal on each scale from maximum at the top of the cone to a minimum at the bottom, regardless of polarity (shown on one scale), which distinguishes the states on one side of each scale from those of the opposite.

intensity of each pair along the radial axis, the magnitude must represent intensity of the state, while sign must reflect which pole of the scale is chosen. Such a state might be:

```
[AFFECTIVE-STATE] –
                        (EXPR) ← [ROBOT:Self]
                          (OBJ) → [CRANE:#]
                        (CHRC) → [ANT/SUP:2]
                        (CHRC) → [JOY/SAD:-2]
                        (CHRC) → [TRUST/DISG:2]
                        (CHRC) → [ANG/FEAR:-1]
                        (CHRC) → [VALENCE:0]
```

In the adaption used here, an affective state consists of a stem characterised by number-type concepts, called [ANT/SUP], [JOY/SAD], [TRUST/DISG] and [ANG/FEAR]. The referent of each can be set to a signed integer in the range [-3,3]. The graph also has an object (here, the protagonist) and an experiencer (usually the machine's self-concept, [ROBOT:Self]), thus enabling the model to be used for the affective states of actors other than the system. Because most models of emotion (and Gelernter) call for way of including an overall hedonic tone (pleasantness/unpleasantness), an additional number, [VALENCE] in the range [-3,3] was added. The GRAVY generator can recognise an affective model of this kind, and produce an appropriate English expression based on the descriptive terms in Figure 2. The above graph produces the sentence:

"I feel a sense of anticipation for sad about trust toward and fear about the crane."

Since in Plutchik's model some combinations of primary emotions form recognisable, named compound emotions, it would be straightforward to refine such expressions by, for example, combining "anticipation for" and "fear about" into the secondary emotion "anxiety for" the crane. This refinement has not yet been added to GRAVY.

5 Results

A suitable modification of the basic FGP process as described in the Introduction was then developed for lower focus, free-association through (one context from) the database of fables:

```
repeat ($\mathcal{T}_i$, $\mathcal{M}$)
        1)  Extend: (cg-fetch ($\mathcal{T}_i$, $\mathcal{M}$)) —> $\mathcal{L}$
                    $\mathcal{T}$ <— min (sort($\mathcal{L}$))
        2   Refocus: cg-project ($\mathcal{T}$) —> $\mathcal{T}'$
        3   Query:
                 exit: break
                 why: compose response from $\mathcal{T}_i$ and $\mathcal{T}'$
                what: express affect ($\mathcal{T}'$)
                 next:  remove $\mathcal{T}_i$ from $\mathcal{M}$
                        $\mathcal{T}_i$ <— $\mathcal{T}'$
```

Because there is no goal in daydreaming, the role of evocativeness cannot be used in the semantic distance calculation of CG-fetch, but only to find features of interest in *CG-project* (where each feature in turn becomes the "goal"). Note also that according to theory, *CG-generalise* is not used here, since as Gelernter remarks, at lower focus the machine "does not build a memory sandwich" [6, p.132]. After Refocus, the algorithm breaks to allow some interaction with the user so as to demonstrate the value of accessing the fables in this way, as shown in the following fragment:

```
(daydream wolf&crane_ap)
  Free associating… It feels like The Cat and the Cock.
  Interesting…ROBOT
  Interesting…ANT/SUP
  Interesting…JOY/SAD
  Interesting…TRUST/LOATH
  Interesting…ANG/FEAR
  Interesting…VALENCE
>what do you feel most strongly? .
  I feel a sense of anticipation for a little pensive about acceptance of
  and apprehensive about the bird. Not good.
>next .
  Continue.
  Free associating… It feels like The Thirsty Pigeon.
  Interesting…ROBOT
  Interesting…ANT/SUP
  Interesting…JOY/SAD
  Interesting…TRUST/LOATH
  Interesting…ANG/FEAR
  Interesting…VALENCE
>why did you recall that one? .
  As with the cock, I felt sad about the pigeon. That's bad.
>next .
  Continue.
  …
```

The algorithm searches $\mathcal{M}$ for close matches to the affective-state relating to the protagonist of the fable – the crane. Finding 16 close matches that are tightly clustered, it examines the closest – "The Cat and the Cock" for interesting features which might explain the association. It discovers that all the features of the affective state except the object are sufficiently evocative to re-enter the cycle in this case. Rather than completely abandon the selectional constraint of the object ("the cock") in the first instance, however, the program relaxes the constraint on this concept by generalising to the next level ("a bird"). When asked what it feels most strongly, the program actually describes the emotional referents of $\mathcal{T}'$, so the optional exclamation expressing displeasure about the predicament of the crane is expressed as 'the bird'. Using $\mathcal{T}'$ as the next probe, the free association locates "The Thirsty Pigeon", another fable in which a bird protagonist comes to grief. When queried as to the reason for recalling that episode, the program compares the previous affective state with the current state and produces a comparative statement based on the closest features between the two, which will have been mostly responsible. Note that, although the program gains access to all the conceptual content describing the contents of the fable, and could potentially answer questions about it using well-proven conceptual reasoning methods, it could not as it stands find those particular events (the cheating of the crane, the capture of the pigeon) which lead to the emotions it is describing, because those were assigned by proxy and with respect to the whole fable, leaving no connection with particulars. (In the case of a policy-driven attitudinal system, such an explanation should be possible, though.)

What do we really gain from such a memory walk? Critics might object the above excursion is nothing more than a sequence of arbitrarily close matches through a database of cases based on an elaborate semantic distance measure. In such a small collection of cases, they might say, it is inevitable that apparently meaningful connections could easily be found. True, only a handful of cases are currently included in $\mathcal{M}$, though more are being added as quickly as time permits. But there are important differences between the current demonstration and, say, conventional reminding in the first part of on conventional CBR cycle. First, each narrative visited begins the entire refocus cycle anew, resulting in a chain of contextually evoked reminiscences, rather than just the top *n* matches with a single case on a simple distance metric. This is more like the operation of human memory. Second, the representation of the fables is unusually rich, episodic and able to be converted to and from natural language, making it possible to stop off and further ruminate over, or answer questions about, the currently indexed fable. Third, as mentioned in Section 4, affective-states are not inherent in the cases themselves, but represent reactions to them from a consistent cognitive agent (although this point is not as clear as it would be if the machine generated its own emotions to each fable as outlined in [16]).

Perhaps more importantly, the same basic FGP mechanism that can, without any domain-specific rules, perform logical reasoning at high focus has been demonstrated to function at low focus as a free-association system able produce emotion-like behaviour, just as Gelernter suggested it could. Therefore this simple mechanism may be of interest to anyone interested in more lifelike computing on ever richer representations of cultural episodic experiences. Imperfect as it is, the lower-focus CG-FGP process represents a subtle drifting indexation of narratives, which may have something in common with a human daydream.

References

1. Meehan, J.R.: TALE-SPIN, An Interactive Program that Writes Stories. Proceedings of IJCAI-77, pp. 91--98 (1977)
2. Dyer, M.G.: In-depth Understanding. MIT Press, Cambridge, Massachusetts (1983)
3. Scharf, H.: Searching for Narrative Structures. Proc. of the AAAI Spring Symposium on Mining Answers from Texts and Knowledgebases. AAAI Press, Technical Report SS-02-06 (2002)
4. Elson, D.K. McKeown, K.R.: A Platform for Symbolically Encoding Human Narratives. Proc. of the AAAI Fall Symposium on Intelligent Narrative Technologies. AAAI Press. Technical Report FS-07-05 (2007)
5. Kolodner, J.: Case-Based Reasoning. Ch. 4. Morgan Kaufmann, San Mateo, California (1993)
6. Gelernter, D.H.: The Muse in the Machine. Fourth Estate, London (1994)
7. Fertig, S., Gelernter, D.H.: The Design, Implementation and Performance of a Database-driven Expert System. Technical Report #851, Department of Computer Science, Yale University (1991)
8. Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Boston: Course Technology, Boston, Massachusetts (1999)
9. Townsend, G.F. (translator): Aesop's Fables (1867). Available at http://classics/mit.edu//Aesop/fab.html
10. Scharfe, H.: CANA A study in Computer Aided Narrative Analysis Ch.12. PhD dissertation, Dept. of Communication Aalborg University (2004). Available at http://www.hum.aau.dk/~scharfe/CANA.pdf

11. Mann, G.A.: Control of a Navigating Rational Agent by Natural Language. PhD Thesis, School of Computer Science & Engineering, University of New South Wales (1996). Available at http://wwwit.murdoch.edu.au/~S850124d/NL/Mann_PhD_Thesis.pdf
12. Schank, R.C.: The Primitive Acts of Conceptual Dependency. Proc. of the 1975 Workshop on Theoretical Issues in Natural Language Processing, pp.34-37. Association for Computational Lingusitics, Morristown, New Jersey (1975)
13. Mineau, G.W.: Normalizing Conceptual Graphs. In: T.E. Nagle et. al. (eds.), Conceptual Structures: Current Research and Practice, 339-348. Ellis Horwood, Chichester (1992)
14. Wuwongse, V. Niyomthai, S.: Conceptual Graphs as a Framework for Case-based Reasoning. Proc. of the 6th Annual Workshop on Conceptual Graphs, 119-133 (1990)
15. Plutchik, R. A General Psychoevolutionary Theory of Emotion. In: Plutchik, R., Kellerman, H. (eds.) Emotion: Theory, Research and Experience, Vol. 1, pp.3-33. Academic Press, New York (1980)
16. Mann, G.A.: Rational and Affective Linking Across Conceptual Cases – Without Rules. In D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (eds.), Conceptual Structures: Fulfilling Peirce's Dream. Lecture Notes in AI 1257, 460-473, Springer, Berlin (1997)

.