Searching for Narrative Structures

Henrik Schärfe

Department of Communication
Aalborg University
Kroghstraede 3, 9220 Aalborg East, Denmark
scharfe@hum.auc.dk

Abstract

Based on a knowledge base consisting of a formal representation of a narrative, some algorithms are discussed for the purpose of computer aided narrative analysis. The algorithms mentioned vary in complexity from the quite simple such as agent / patient relationship, to more advanced queries that chart deep narrative structures in order to identify actantial roles and functions. Examples and results discussed in this paper are derived from a formal representation of a full-scale narration. This representation and the algorithms mentioned are implemented in PROLOG+CG.

Introduction

This paper presents some results of an ongoing investigation in the area of Computer Aided Narrative Analysis [10], [11]. The basic idea is that rhetorical and narrative patterns can be formally described as scripts [1], [2], [3], [7], [8], [9], and that these scripts can be instantiated through iterative queries in a knowledge base containing a formal representation of a narrative. This approach to semi-automated text analysis rests on the assumption that narratives are manifestations of deeper lying structures, and that these structures can be computed if they can be adequately described in a formal way. This implies the notion of text as 'distributed representations of cognitive and cultural structures' [11]. The challenge is to analyze real full-scale narratives with natural language characteristics. Because the queries in this study are conducted on a formal representation, the methodology brought forth here cannot be immediately transferred to systems operating on 'raw' text material. In that context, this study should be regarded as a contribution to the development of a methodology for future work.

The case discussed here is the story of Gideon, from the Book of Judges in the Old Testament chap. 6-8, which has been formally represented using Conceptual Graphs. The representation along with a set of algorithms are implemented in PROLOG+CG, whereby the program reaches a size of approximately 2700 lines.

Formal representation

The knowledge base used for this analysis is hand encoded. In terms of web mining this of course is a serious drawback. In the other hand, it seems evident that techniques such as the ones suggested here, should be developed in a controlled environment with appropriate attention given to the characteristics of natural language narratives. This includes the use of metaphors and other tropes; the fact that not all what is said is true (characters may contradict each other), and in general the freedom of an author to express meaning through the unrestricted use of language as opposed to controlled versions of language.

The story of Gideon has been represented using Conceptual Graphs (CG), a formal language developed by John Sowa based on Charles Sanders Peirce's existential graphs, and the semantic networks of artificial intelligence [12], [13]. See also [6] for an introduction. The collection of graphs is implemented in a program called PROLOG+CG, written by Dr. Adil Kabbaj [4], [5]. This program combines the programming possibilities of PROLOG with the expressive power of CG, which means that semantic description can be used directly to manipulate the knowledge base [6]. In this way it becomes possible to employ relation-oriented as well as ontology-driven analyses at the same time.

In CG there are two kinds of nodes connected with arcs. Concepts are written in square brackets and conceptual relations are written in parentheses. A signature determines the directions of the arcs for each relation.

```
jud6(verse19c,
[act:put]-
   -agnt->[person: Gideon],
   -obj->[food: meat],
   -dest->[artifact: basket]).
```

This graph from chapter 6, verse 19c states that Gideon puts meat in a basket. Note that concepts contain a type as well as a referent. The graphs are accompanied by an ontology in form a lattice of concept types. This means not only that any textual element can by quite accurately

accounted for, but also that specific information can be added to the knowledge base independent of the actual representation. For example: it is commonly agreed that conflicts are of prime interest in narratives, but how can conflicts be identified? In the 'Gideon ontology' the concept type 'act' is divided into several types including one called 'conflict_act'. This category contains words such as 'encamp', 'destroy', and 'slay'. Because of the subtype relation in the ontology, this information in not required in the actual representation: [act: slay] is just as computable as [conflict_act: slay]. But having the information present in the ontology enables the analyst to search for words that belong to that particular category. This methodology has also been used to implement speech act theory in the knowledge base, but naturally there are some limitations to this approach; there may exist conflicts that are not made explicit through the use verbs indicating aggression.

Searching for structure

The algorithms range in complexity from very simple structures such as family relations to rather complicated matters such as deep actantial structures.

Analyzing talk

Following is a simple example of an algorithm that reveals two pieces of relation-oriented content in one query.

```
talks(c,v,s,r):-judge(c,v,
[universal:s]<-agnt-[uttr]-rcpt->[universal:r]).
```

Because the type of agent and patient are left open (universal is the top of the ontology), the results include individuals as well as societies. Among the results are:

```
{c = 6, v = verse22b, s = Gideon, r = himself}
{c = 6, v = verse26b, s = LORD, r = Gideon}
{c = 6, v = verse30, s = Ophrah, r = Joash},
```

stating that Gideon is talking to himself in 6:22b, that the LORD is talking to Gideon in 6:26b, and that the men of Ophrah are talking to Joash in 6:30. It follows that the algorithm can be altered to search for a specific agent or patient of an utterance.

Conflicts

A slightly more elaborated example involves conflicts. In this algorithm:

```
conflict(c,v,s1,a,s2):-
judge(c,v,g),
subsume([universal]<-ptnt-[conflict_act]-agnt-
>[universal],g),
branchOfCG([conflict_act]-agnt->[r : s1],g),
branchOfCG([conflict_act]-ptnt->[s : s2],g),
branchOfCG([a]-agnt->[r : s1],g),
isInstance(a,conflict_act).
```

'subsume' searches for graphs in the knowledgebase that are *less* general than the graph specified in the query. Note, that the first argument of subsume is a Conceptual Graph, containing conceptual relations as well concepts. In this particular case, the algorithm is constructed for maximum flexibility by allowing any concepts below 'universal' in the roles of agent and patient. In this way, persons, animals, or societies can be listed in one single query. All occurrences of graphs containing a 'conflict_act' linked to an agent and a patient are now listed under the name 'g'.
The second operation, 'branchOfCG' investigates the graphs identified above for more specific information. This is done three times, one time for each variable. BranchOfCG splits the entire collection of graphs 'g' into parts, listing all occurrent relations between any two concepts in these graphs. In order to find agent and patient, the concept in question is equipped with type as well as referent. The types are not reported as a result, but the referents are now defined as agent and patient respectively of some conflict_act. The act itself is than replaced by a variable so that it can be reported by the query. Because the graphs 'g' may contain references to other actions (concepts that have agents), the operation 'isInstance' is applied to the graphs. IsInstance is a type goal, which checks that the variable is in fact an instance of the type 'conflict_act'.

When executed, this algorithm returns chapter (c), verse (v), agent (s1), action (a), and patient (s2) of all conflicts reported in the text. Examples include:

```
{c = 6, v = verse3, s1 = Midian, a = attack, s2 =
Israel}
{c = 7, v = verse25c, s1 = Ephraim, a = kill, s2
= Zeeb}
{c = 8, v = verse21b, s1 = Gideon, a = slay, s2 =
{Zebah, Zalmunna}}
```

These results can be paraphrased as: 'Midian attacks Israel in chapter 6, verse 3', 'Ephraim kills Zeeb in chapter 7, verse 25c', and 'Gideon slays Zebah and Zalmunna in chapter 8, verse 21b'.

Deep Structures

In the works of narratologists such as Propp and Greimas, deep structures are of great importance [1], [2], [3]. The notion of deep structures implies that structures are distributed into different parts of the narration. This again calls for a method of analysis that can handle narrative succession and transformation by combining semantic information retrieved from different parts of the narration. I submit that at least some of these structures can be formally described as scripts, thereby becoming available for computer aided analysis. A good example of this is the perhaps most common instantiation of the semiotic square: the two-fold transportation of values of object [3].

Greimas argues that a common semiotic display of a narration can be articulated as a conflict between two societies, and that two movements manifest the conflict and its solution. The first movement is performed by the villain, who removes something (or someone) from the home of the hero; the second movement consists of the hero bringing the item in question (or something similar) back. This structure is commonly displayed as in figure 1.

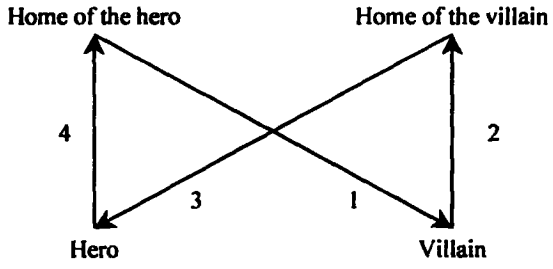

Figure -1 A Typical narrative scheme

The importance of this structure lies in the fact that this relatively simple arrangement holds an enormous amount of information embedded in it. I will refer to this structure as a 'double_transfer'. The double_transfer figure contains information about a fundamental plot structure, and reveals four out of six actants in the actantial model formulated by Greimas [2].

The double_transfer script

What really strikes me about the double_transfer pattern is its simplicity. The graphical display of Figure 1 can be formally described as a script by the following CG:

```
[Narrative_Script: double_transfer:
[Proposition:
[transfer: #1]-
  -(agnt)->[Actant: *2],
  -(ptnt)->[Society: *1],
  -(obj)->[Value_object: *1],
  -(benf)->[Society: *2]]-
-(succ)->
[Proposition:
[transfer: #2]-
  -(agnt)->[Actant: *1],
  -(ptnt)->[Society: *2],
  -(obj)->[Value_object: *2],
  -(benf)->[Society: *1]]].
```

The script is a closed structure in the sense, that if any of the elements are present in precisely this capacity, all of the others must by necessity be present in the text as well. Furthermore, if this structure is found in a text, the slot named 'Actant:*2' will always signify the villain, and the slot named 'Actant:*1' will always signify the hero. The detailed theoretical foundations for the double_transfer script are accounted for in [10].

I find it appropriate to suggest, that this structure is far more common that the fairy tales from which it was original derived. However, the script may be found in different variants, depending on the actual use. We might say, that the original script as suggested in [10] is a prototypical example of a number of scripts that share the same basic structure and purpose, but that some elements can be replaced by other elements. The basic structure is constituted by two successively analogous actions characterized by the switching of agent and patient, both of which are represented by a society and by an individual.
The objects of the script, on the other hand, may appear as physical, abstract, or symbolic manifestations of something important enough to be of significance to the entire narration. If vengeance or justice is a prevalent theme of a narrative, the object of the second action is not necessarily physical, but the basic structure remains the same. From this it follows that certain modifications can be made to the script without depriving it of its fundamental qualities.

Searching for a hero

Based on the double_transfer script an algorithm can be designed that starts by locating two societies in conflict. Secondly, individuals from the two societies must be identified. This is done by a separate algorithm that searches for appropriate relations between persons and societies. Thirdly, a temporal order must be established between the two parts of the script. Applying a predicate depending on the graph number does this. From this it follows that an example of a double_transfer algorithm can look like this:

```
double_transfer(A2,S2,b,O1,A1,S1,d):-
  judge(c,v,g),
  subsume([conflict_act:a]-agnt->[society],g),
  branchOfCG(
    [conflict_act:a]-agnt->[society : S2],g),
  branchOfCG(
    [conflict_act:a]-ptnt->[society : S1],g),
  home_of(A1,S1),
  home_of(A2,S2),
  judge(c1,v1,g1),
  subsume([conflict_act:b]-agnt->[society],g1),
  branchOfCG(
    [conflict_act:b]-agnt->[society : S2],g1),
  branchOfCG(
    [conflict_act:b]-R1->[s:O1],g1),
  member(R1,(ptnt,thme)),
  after(c2,v2,c1,v1),
  judge(c2,v2,g2),
  subsume([conflict_act:d]-agnt->[person],g2),
  branchOfCG([conflict_act:d]-agnt->[p:A1],g2),
  branchOfCG([conflict_act:d]-ptnt->[p:A2],g2).
```

The algorithm searches the knowledge base for three different sets of conflict-actions in different graphs, incorporates two different algorithms that checks the temporal order, and the societies to which the candidates belong. When tested on the Gideon story, the results are:

{A2 = Zebah, S2 = Midian, b = attack, O1 = Israel, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zebah, S2 = Midian, b = attack, O1 = Israel, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zebah, S2 = Midian, b = encamp, O1 = Israel, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zebah, S2 = Midian, b = encamp, O1 = Israel, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zebah, S2 = Midian, b = destroy, O1 = produce, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zebah, S2 = Midian, b = destroy, O1 = produce, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zebah, S2 = Midian, b = wasted, O1 = land, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zebah, S2 = Midian, b = wasted, O1 = land, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zalmunna, S2 = Midian, b = attack, O1 = Israel, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zalmunna, S2 = Midian, b = attack, O1 = Israel, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zalmunna, S2 = Midian, b = encamp, O1 = Israel, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zalmunna, S2 = Midian, b = encamp, O1 = Israel, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zalmunna, S2 = Midian, b = destroy, O1 = produce, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zalmunna, S2 = Midian, b = destroy, O1 = produce, A1 = Gideon, S1 = Israel, d = slay}
{A2 = Zalmunna, S2 = Midian, b = wasted, O1 = land, A1 = Gideon, S1 = Israel, d = pursue}
{A2 = Zalmunna, S2 = Midian, b = wasted, O1 = land, A1 = Gideon, S1 = Israel, d = slay}.

This result can be paraphrased as: "Zebah and Zalmunna from Midian attack and encamp Israel, destroy the produce (of the land), and waste the land, after which Gideon from Israel pursues and slays Zebah and Zalmunna".

Even though the script may seem quite abstract, this result is a remarkably close to the master plot of the narration. Thus the algorithm identifies Gideon as the hero of the story, Zebah and Zalmunna as the villains, the land and what is produces as the actantial object, and Israel as the actantial receiver.

Once these key elements of the deep structure of the narrative have been identified, several new algorithms can be designed by replacing a variable with a precise reference to an actant.

Searching for functions

In the work of Propp [7], the narration is seen as a sequence of functions performed by the key characters. Examples of such functions are 'villainy', 'struggle', and 'the hero's wedding'. Some of the very important functions are the 'interdiction' and the 'violation of the interdiction'. These functions are of particular interest because the theme of the narrative quite often is reflected in the nature of the interdiction. For computer aided analysis the identification of such a 'Law' is complicated by the fact that the Law may or may not be referred to directly. In many narratives the Law is simple presupposed, or referred to by some non-central character. To comprise the most common realizations of this narrative element a number of algorithms are needed. In the following I shall present one example of such an algorithm.

```
fc-violation(D,i,S1,e,h):-
double_transfer(A2,S2,b,O1,A1,S1,d)
judge(c1,v1,g1),
subsume([universal]-thme->[universal],g1),
branchOfCG([universal]-thme->[proposition=P],g1),
branchOfCG([perceive:i]-agnt->[j:D],g1),
subsume([act:e]-agnt->[society],P),
branchOfCG([act:e]-agnt->[society:S1],P),
branchOfCG([act:e]-attr->[negative:h],P),
after(c2,v2,c1,v1),
judge(c2,v2,g2),
subsume([conflict_act:f]-agnt->[society],g2),
branchOfCG([conflict_act:f]-
            -agnt->[society:S2],g2),
branchOfCG([conflict_act:f]-
            -ptnt->[society:S1],g2).
```

This algorithm searches for some act (e) with a negative attribute assigned to it (h). The act must be reported in the propositional content of some perception, it must be carried out by society (S1) from the double_transfer script – which is the home of the hero, furthermore the act must be committed *prior* to a conflict_act (f) committed by society (S2) from the double_transfer script against society (S1).

In the Gideon story this query produces the following result:

```
[D = LORD, i = see, S1 = Israel, e = action, h =
evil].
```

This result is consistent with the fact that Israel is the agent of some act perceived as evil by the LORD. In this manner the legislator of the narration and the function 'interdiction' have now been computed.

Conclusion

A few examples of algorithms for computer aided narrative analysis have been presented. The ability to identify deep narrative structures has been illustrated by use of iterated queries and compound algorithms. This includes the double_transfer algorithm that successfully identifies – among other things – the hero and the villain of a full-scale narrative. The fc_violation algorithm illustrates how an abstract theoretical phenomenon such as a proppian function can be identified. This algorithm also illustrates the possibility of limiting or extending a query to be conducted in different textual layers such as the propositional content of a perception or an utterance.

The queries mentioned here benefit from the ability of performing relation-oriented as well as ontology-driven analyses at the same time. This approach holds some promise in terms of mining answers from knowledge bases consisting of formalized versions of natural language texts.

Future work

As mentioned in the introduction, this is work in progress, and the results must be tested on other narrations, preferably from different genres. Currently more texts are being represented in CG for that purpose. Furthermore, other narratologies than the ones mentioned here are likely to benefit from this approach. Examples include Possible World narratology and the relation between rhetorical and narrative structures.

PROLOG+CG is implemented in Java, which should make it a doable enterprise to build a graphical front-end that allows users to explore the formal representation in a natural language environment.

References

1. Bremond, Claude (1980): **The Logic of Narrative Possibilities.** New Literary History 11 387-411.
2. Greimas, A. J. (1966) **Sémantique structurale.** Librairie Larousse, Paris
3. Greimas, A. J. (1987) **On Meaning.** University of Minnesota Press, Minneapolis
4. Kabbaj, Adil & Janta-Polczynski. (2000) **From PROPOG++ to PROLOG+CG: A CG Object-Oriented Logic Programming Language.** In B. Ganter & G. W. Mineau, eds., *Conceptual Structures: Logical, Linguistic, and Computational Issues*, Lecture Notes in AI #1867, Springer-Verlag, Berlin, 2000, pp. 540-554
5. Kabbaj, Adil. (2000) **PROLOG+CG User's Manual** http://www.insea.ac.ma/CGTools/PROLOG+CG.htm
6. Petersen, U. et alt. (2001) **Online Course in Knowledge Representation using Conceptual Graphs:** http://www.hum.auc.dk/cg
7. Propp V. J. (1968) **Morphology of the Folktale.** University of Texas Press, Austin, 1968
8. Schank R.C. & Abelson, R.P (1977) **Scripts, Plans, Goals and Understanding.** Hillsdale (NJ) Lawrence Erlbaum
9. Schank, R. C. (1990) Tell Me a Story: Narrative and intelligence. Northwestern University Press
10. Schärfe, Henrik & Øhrstrøm, Peter. (2000) **Computer Aided Narrative Analysis using Conceptual Graphs** in Gerd Stumme (Ed.) Working with Conceptual Structures – Contributions to ICCS 2000, p 16-29 Shaker Verlag, Aachen.
11. Schärfe, Henrik (2001) **Reasoning with Narratives.** Forthcoming. Available at: http://www.hum.auc.dk/~scharfe/reasoning.pdf
12. Sowa John F. (2000) **Knowledge Representation: Logical, Philosophical, and Computational Foundations,** Brooks/Cole Publishing Co., Pacific Grove, CA, 2000.
13. Sowa, John F. (2001b) **Existential Graphs** MS 514 by Charles Sanders Peirce – with commentary by John F. Sowa. http://www.bestweb.net/~sowa/peirce/ms514w.htm