Recognizing Situation Patterns from Self-Contained Stories*

Hiram Calvo, Alexander Gelbukh

Natural Language Processing Laboratory,
Center for Computing Research, National Polytechnic Institute,
Mexico City, 07738, Mexico
likufanele@likufanele.com, gelbukh@gelbukh.com; www.Gelbukh.com

Abstract. We propose extracting information about characters and actions from a self-contained story, such as news reports. This information is stored in structure patterns called situations. We show how these situation patterns can be constructed by unifying the constituents of sentence analysis with knowledge previously stored in Typed Feature Structures. These situations can be in turn used subsequently in the form of knowledge. The combination of situations constructs a supra-structure that represents the understanding of a factual report. The main pattern structure can be used to answer questions about facts and their participants.

1 Introduction

Extracting information patterns about characters and actions from a factual report, v. gr. news web pages, relies on the construction of a structure that expresses the relationship among the situations described in that text. To construct this structure there must be interaction with previously acquired knowledge. In turn, this knowledge must be expressed in a structured way so that inferences can be done by using it. This work focuses on obtaining such structures for factual reports.

Factual reports are texts in which facts are described in an ordered manner, and all the participants of these facts are circumscribed within the text. These characteristics allow us to apply knowledge techniques of practical complexity. By practical complexity, we mean mid-term feasible applications. Consider, for example, that understanding a story can be undertaken in several ways, being one of the elements to be taken into account the amount of the knowledge previously required for understanding a story. For example, Minsky includes in 1 an example of understanding with the following text:

"There was once a Wolf who saw a Lamb drinking at a river and wanted an excuse to eat it. For that purpose, even though he himself was upstream, he accused the Lamb of stirring up the water and keeping him from drinking"

Minsky argues that understanding this text is to realize the following situations:

* Work done under partical support of Mexican Government (SNI, CGPI-IPN, PIFI-IPN).

(C) Alexander Gelbukh, Manuel Montes y Gómez (Eds.)
Advances in Natural Language Understanding and Intelligent Access to Textual Information,
Proceedings of NLUIATI-2005 workshop in conjuction with MICAI-2005, p. 1-10.

1. The fact that the lamb is stirring the water produces mud,
2. If water has mud, it cannot be drunk,
3. If the wolf is upstream, the fact that the lamb stirs the water does not affect the wolf, and so,
4. The wolf is lying.

However, these inferences require a quite large structured knowledge system that would permit a machine to display common sense. The construction of such general knowledge systems is a major task that could take years or perhaps decades to complete. As a practical midterm solution, we propose solving the following tasks in order to understand a story:

1. To identify characters, places and objects of the story,
2. To identify the described actions,
3. To identify the actions that are not done, but are mentioned within the story,
4. To determine the arguments for each of these actions. These arguments can be seen as answers to wh questions: who, where, what, when, why and whom. Each action with its arguments is a structure that we call situation. And,
5. To establish a temporal sequence among situations that corresponds to the story flow.

Following this approach, for the passage of the wolf and the lamb we find that:

The characters are: the Wolf and the Lamb,
The places are: the River and Upstream,
The objects are: the Water.

The situations are:

Wolf sees Lamb
Lamb drinks
Lamb is *in* River
Wolf wants (Wolf eat Lamb)
Wolf is *in* Upstream
Wolf accuses Lamb *of* (Lamb stirs *the* Water) and
(Lamb *does* not let (Wolf drink Water))

Each one of the strings of words in parenthesis is also a situation. Note that these situations do not necessarily occur. In this case it does not occur (and we do not know yet if it will happen) that (Wolf eats Lamb). Neither it occurs that (Lamb stirs *the* Water) nor (Lamb *does* not let (Wolf drink Water)).

Capitalized words point to particular instances of the characters, places and objects of this particular story.

2. Representing Situations with Typed Feature Structures

In order to construct and represent the situations, we propose the use of Typed Feature Structures (TFS). This formalism permits us to cover every level of linguistic description 2: basic sentence types (POS) construction, intermediate type construction (e.g.

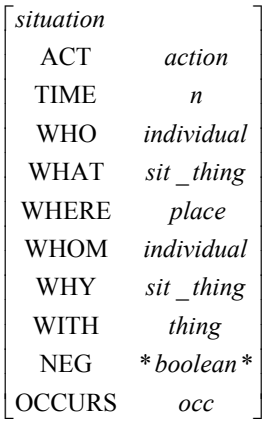

Figure 1: AVM for the type *situation*.

individuals), situations with complements construction, and finally, story structure construction.

We can represent a situation as a feature structure, as shown in Figure 1. This representation is an AVM (Attribute Value Matrix). We follow the convention of representing attributes in uppercase and values in lowercase.

sit_thing denotes that the value for WHAT or WHY can be a *situation* or a *thing*. TIME has a numeric value that corresponds to the sequence in which situations are mentioned. OCCURS is the feature that points out if a situation occurs or not within the story.

The fact that feature structures are typed, permits us to handle an object hierarchy, as stating that a man is a human, that humans are individuals, and therefore, they can be participants in an action as values of WHO and/or WHOM.

2.1 Interaction between Syntax and Knowledge

Before we proceed on the explanation about how situations are constructed, we will discuss briefly on the interaction of syntax and knowledge.

Traditionally, TFS have been used mostly for syntax analysis whereas frame-based systems are used for handling knowledge. Examples of the use of these formalisms are HPSG 3, a well known formalism that combines the use of generative grammars with the benefits of TFS, and NEOCLASSIC 4, a frame-based knowledge representation system.

We believe that in order to construct successfully a situation, these two traditionally separated stages need to be blended, so that their interaction is possible. The formalism we choose for representing syntax and knowledge as well, is TFS, as it shares characteristics with frame based systems. These characteristics are:

(1) Frames and TFS are organized in hierarchies,
(2) Frames are composed out of slots (equivalent to feature structures' attributes) for which fillers (equivalent to feature structures' values or references to other

frames—feature structures) must be specified or computed 5,
(3) Frames and TFS are declarative, and
(4) TFS's logic is similar to the logic used by frames: Description Logics.

Description logics are part from first order logics and are used by the frame-based systems for reasoning. In Description Logics it is possible to construct a hierarchy of concepts from atomic concepts and attributes, usually called roles 6. The only difference between the Feature Logics used by TFS and the description logics used by frame-based systems is that Feature Logics' attributes are single-valued, while Description Logics' attributes are multi-valued. This could seem a light difference, but it could be the difference between decidable and undecidable reasoning problems 7

3 Construction of Situations

The Linguistic Knowledge Building system (LKB) is a programming environment for Typed Feature Structures. LKB follows the formalism as it was introduced by 8.

Although LKB has been most extensively tested with grammars based on HPSG 3 as ERG from the LinGO project 9, LKB is intended to be framework independent 10.

LKB is used mostly for lexical parsing of sentences. However, LKB can be used not only for parsing sentences, but also for storing and interacting with knowledge.

Currently, to deal with a semantic representation, LKB makes use of a grammar that has special markers (e.g. LISZT and HANDEL) to build up an MRS representation. MRS stands for Minimal Recursion Semantics 11, and it produces a flat logical representation intended mainly for handling transfer and quantifier phenomena. The MRS output is mainly suited for translation, and it has been successfully used in the Verbmobil project 12. However, for our purposes MRS drops off specific syntactic information that would allow us to identify the grammatical role that each constituent plays, thus, making it difficult to determine if a constituent fills the WHO or the WHOM slot for example.

As we have proposed in Section 2.1, we will use typed feature structures to construct a situation. The construction of situations corresponds to each sentence.

To handle *situations* as TFS in LKB, we establish the types shown in Figure 2 with their corresponding hierarchy. We use standard LKB notation for lists, TFS and unification (&). We assume that types enclosed in asterisks are predefined, being *top* the most general type in the type hierarchy.

To illustrate how situations are constructed, we will use the example of the Wolf and the Lamb taken from 1. We reproduce that fragment of story for the reader's convenience.

"There was once a Wolf who saw a Lamb drinking at a river and wanted an excuse to eat it. For that purpose, even though he himself was upstream, he accused the Lamb of stirring up the water and keeping him from drinking"

```
n := *top*.

occ := *boolean*.
occ_if := occ &
  [IF situation].

whatsit := situation &
  [WHAT sit_thing].

whysit := situation &
  [WHY sit_thing].

sit_thing := situation & thing.

tpi := *top* &
   [ORTH string].

thing := tpi.
place := tpi.
individual := tpi.
action := tpi.
```

```
situation := *top* &
   [ACT action,
   TIME n,
   WHO individual,
   WHERE place,
   WITH thing,
   NEG *boolean*,
   WHOM individual,
   OCCURS occ].

story := *top* &
   [INDIVIDUALS *list*,
    PLACES *list*,
    OBJECTS *list*,
    ARGS *list*].
```

Figure 2: LKB types for representing situations and stories.

The example we present in this section illustrates the construction of situations. Because of this, syntax and other phenomena are not covered with their inherent complexity. Syntax analysis is addressed in this example as pattern matching. For larger scale systems, formalisms such as HPSG can be used within this approach since they handle TFS.

To construct a situation, we assume that the system previously has information about the possible roles that each entity can have. For example, River and Upstream are places, Wolf and Lamb are individuals, Water is an object. See Figure 3.

Entities can be formed by more than one word. We do not know *a priori* any of the possible properties of these entities (e.g. *big* Lamb, Lamb *named* Dolly, etc.). These properties will be filled as the story is analyzed.

The fact that situations occur or not, is important to understand the flow of the story. In the example of the Wolf and the Lamb, it does not really occur that the Lamb stirs the water keeping the Wolf from drinking. This is a situation mentioned as a consequence of the wolf wanting an excuse to do something. To define if a situation occurs or not, we consider then that when a situation is subordinated by other situation, the subordinated situation does not occur.

We will analyze the fragment of the story presented above word by word following a specific order. Note that feature logic is declarative, so that this analysis could be

```
#wolf [individual]
#lamb [individual]
#river [place]
#upstream [place]
#water [object]
```

Figure 3: Entities consulted from the lexicon

done in any order yielding the same results.

We will begin analyzing sentence (1):

```
There was once a wolf who saw a lamb drinking at a river and wanted an
excuse to eat it.                                                    (1)
```

The first words of (2) match with a pattern that introduces `wolf` as an individual. This pattern is `there + was + once + a +` *`individual`*, leading to the representation shown in (3).

```
there was once a Wolf                                                (2)
```

```
[individual
NAME wolf                                                            (3)
ORTH "wolf"]
```

This structure can unify with a corresponding structure in the knowledge base (implemented as TFS) to find the possible properties of wolves in general.

To avoid writing again the feature structures we identify in this analysis, we will write a reference to them, following the LKB notation – labels begin with #.

```
#wolf [individual
    NAME wolf                                                        (4)
     ORTH "wolf"].
```

then, we can write the sentence being analyzed as:

```
#wolf who saw a Lamb drinking at a river                             (5)
```

A feature structure of type *individual* followed by a lexeme *who*, makes *who* to absorb the individual. The rule that does this is (6).

```
individual_who := individual &
 [ NAME #1,
  ORTH #2,                                                           (6)
  ARGS < individual & [NAME #1,
      ORTH #2],
      lexeme_who]>].
```

The sentence becomes now:

```
#wolf saw a Lamb drinking at a river.                                (7)
```

We turn our attention then to *Lamb drinking at a river*. This is another situation, but first we must add the `lamb` individual to our story.

```
#lamb [individual
    NAME lamb                                                        (8)
    ORTH "lamb"].
```

`a Lamb drinking at a river` is converted then into:

```
#lamb drinking at a river.                                           (9)
```

The previously defined lexicon (see Figure 3) provides the information that `river` can be a place. `river` is not restricted to belong only to one category; in case of sev-

eral choices, unification will help to select the correct one(s). river is then considered as:

```
#river [place
    NAME river                                                    (10)
    ORTH "river"]
```

We do not show here details of a reference resolution mechanism that would establish the difference among "a river" and "the river" according to previously introduced entities. For this work we assume that each time that an entity is mentioned, its feature structure equivalent is introduced. When the supra-structure story is formed, two Feature Structures (FS) corresponding to the same entity will unify. If two FS of the same kind have conflicting particular characteristics (such as red river and blue river), unification will fail, and then, two different entities will be considered.

#river can be later unified with a knowledge base so that the system could infer that #river is made of #water. For simplicity, in this example we assume that this kind of information has not been implemented.

Returning to the analysis of (9), we can verify from our lexicon that drinking unifies with the type *action* (verb). We will call the feature structure for this particular action, #drink (12), obtaining (13).

```
#drink [action
    NAME drink                                                    (12)
    TENSE gerund
    ORTH "drinking"]
```

```
#lamb #drink at #river                                            (13)
```

Now we can apply the FS rule that creates a situation when the sequence: individual, action, "at", place is found:

```
[ situation
 ACT #2
 WHO #1                                                           (14)
 WHAT
 WHERE #3
 ARGS <#1, #2, lexeme_at, #3 >]
```

Exceptions to rule (14) can be handled as additional constraint rules. Applying this rule we have the situation #s2:

```
#s2 [ situation
   ACT drink
   who #lamb                                                      (15)
   WHAT
   WHERE #river ]
```

We return to the main sentence (7) substituting the last situation we have just found:

```
#wolf saw #s2.                                                    (16)
```

This forms another situation:

```
#s1 [ situation
   ACT see                                                    (17)
   WHAT #s2 ]
```

Finally, the first sentence is a situation

```
#s1                                                           (18)
```

`#s1` has a subordinated situation `#s2`. The rest of the story fragment of the wolf and the lamb can be analyzed in a similar way. The entities consulted from the lexicon are shown in Figure 3, and the story structure obtained after this analysis is shown in Figure 4.

4 Minsky's Frames and Situations

Minsky argues in 1 that frames are like a network of nodes and relationships; the top levels of a frame are fixed and represent things that are always true about a supposed situation. Lower levels have terminals (slots) that must be filled by specific instances or data. There are conditions specified by markers that require that a slot assignation is a person, an object, or a pointer to a sub-frame of certain type. A terminal that has acquired a *feminine person* marker will reject pronominal *masculine* assignations. In this sense, Minsky's frames are very similar to a Feature Structure. Each frame would be regarded a Feature Structure, with slots being the values of the attributes in the attribute-value structure. However, there is an important difference among Minsky's frames and our viewpoint of situation representation.

Minsky talks about frames as a data-structure to represent a stereotyped situation, like being in a special kind of room, or going to a child's birthday party. Minsky considers that frames must contain information about how they must be used, information about expectations, and information about what to do if expectations are not confirmed. In contrast, we consider a situation as a simple transitory unit of the state of things within a story. Consider the sentence: *The man wants to dance with Mary*. This sentence contains two situations: (Situation 1:) *wants*. Who? *The Man*, What? Refers to situation 2, Occurs? yes. (Situation 2:) *dance*. Who? *The Man* (the same man), (with) Whom? *Mary*, Occurs? no. In these situations we do not consider (in contrast with Minsky) information about how to use a frame, information about expectations nor what to do if expectations are not confirmed.

5 Conclusions and Future Work

The Typed Feature Structures formalism permits to address different levels for the understanding of a story. Typed feature structures are a well studied formalism that guarantees the computability of its logic. The groundwork we have presented allows extracting situations from a factual report so that it is possible to ask simple questions about the text such as who did something, or where she or he did it. This can be used in a web query system to find relevant results about events described in a factual report. Implementation of such system is currently under development.

```
story & [
          INDIVIDUALS   <#wolf & wolf1, #lamb & lamb1>,

          PLACES        <#river & river1, #upstream &
                            upstream1>,

          OBJECTS       <#water & water1>,

          SITUATIONS    <#S1 [situation      #S2 [situation
                           TIME 1               TIME 1
                           ACT see              ACT drink
                           WHO #wolf            WHO #lamb
                           WHAT #s2             WHAT (liquid)
                           OCC yes],            WHERE #c
                                                OCC yes],

                        #s3 [situation       #s4 [situation
                           TIME 2               TIME 2
                           ACT want             ACT eat
                           WHO #wolf            WHO #wolf
                           WHAT #s4             WHAT #lamb
                           OCC yes],            OCC no],

                        #s5 [situation       #s6 [situation
                          TIME 3                TIME 4
                          ACT is                ACT accuse
                          WHO #wolf             WHO #wolf
                          WHERE #upstream       WHAT #s8
                          OCC yes],             WHY #s3
                                                OCC yes],

                        #s7 [situation       #s8 [situation
                           TIME 4                TIME 4
                           ACT accuse            ACT stir
                           WHO #wolf             WHO #lamb
                           WHAT #s9              WHAT #water
                           WHY #s3               OCC no],
                           OCC yes],

                        #s9 [situation       #s10 [situation
                           TIME 4                TIME 4
                           ACT let               ACT drink
                           WHO #lamb             WHO #wolf
                           WHAT #s10             WHAT (liquid)
                           NEG true              OCC no] >]
                           OCC no],
```

Figure 4: Feature structure for the story fragment of the wolf and the lamb.

Reference resolution was addressed very lightly in this paper, but a strong mechanism for reference resolution is essential for a larger scale operation of the system. This has been left out of the scope of this paper as ongoing work.

With regard to Minsky's frames approach, as a future work, through the analysis of individuals through a story, characters' behaviour could be generalized in a model to predict their reactions and interactions, tending towards common sense acquisition and expectations in the sense of Minsky's frames.

References

1. Marvin Minsky, "A Framework for Representing Knowledge", in P. Winston (ed.): *The Psychology of Computer Vision*, McGraw Hill, New York, 1975, pp. 211- 277.
2. Bob Carpenter. *The Logic of Typed Feature Structures*, Cambridge University Press, 1992.
3. Ivan Sag and Thomas Wasow, *Syntactic Theory: A Formal Introduction*. CSLI Publications, 1999.
4. Peter F. Patel-Schneider, Merryll Abrahams, Lori Alperin Resnick, Deborah L. McGuinness, and Alex Borgida. *NeoClassic Reference Manual: Version 1.0* Artificial Intelligence Principles Research Department, AT&T Labs Research, 1996.
5. Bernhard Nebel, "Frame-Based Systems", in Robert A. Wilson and Frank Keil (eds.), *MIT Encyclopedia of the Cognitive Sciences*, MIT Press, Cambridge, MA, 1999, pp. 324-325.
6. Bernhard Nebel, "Logics for Knowledge Representation", in N. J. Smelser and P. B. Baltes (eds.), *International Encyclopedia of the Social and Behavioral Sciences*, Kluwer, Dordrecht, 2001.
7. Bernhard Nebel and Gert Smolka, "Attributive description formalisms... and the rest of the world", in O. Herzog and C. Rollinger, eds., *Text Understanding in LILOG*, Springer, Berlin, 1991, pp. 439-452.
8. Stuart Shieber, An Introduction to Unification-Based Approaches to Grammar, CSLI Publications, 1986
9. Ann Copestake and Dan Flickinger, "An open-source grammar development environment and broad-coverage English grammar using HPSG", in *Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece, 2000.
10. Ann Copestake, Implementing Typed Feature Structure Grammars, U. of Chicago Press, 2001.
11. Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann and Ivan Sag, "Translation using Minimal Recursion Semantics", in *Proceedings of The Sixth International Conference on Theoretical and Methodological Issues in Machine Translation* (TMI95), Leuven, Belgium,1995
12. Caroli, Folker, Rita Nübel, Bärbel Ripplinger & Jörg Schütz, **"**Transfer in VerbMobil", in IAI Saarbrücken *VerbMobil-Report 11,* May 1994