TOWARDS AUTOMATIC MODELLING OF REQUIREMENTS

Farid Meziane
Institute of Software Technology,
Universiti Malaysia Sarawak,
94300 Kota Samarahan,
Sarawak, Malaysia.
Tel: 082 670000 ext: 384
Fax: 082 672301
email: farid@fit.unimas.my

Sunil Vadera
Department of Maths and Computer Science
University of Salford.
Salford, M5 4WT,
United Kingdom
Tel: ++ 161745 5000 ext. 3622
Fax:++ 161 745 5559
email: S.Vadera@mcs.salford.ac.uk

ABSTRACT

The first phases of the FORSEN system that helps the analyst to use an informal specification as the basis of producing a formal specification and concerns the modelisation of the requirements into entity relationship models (ERM) is described. The modelisation is done from the logical form expressions obtained from the analysis of the natural language text. The ERM models are then used as a basis for the production of formal specification in the Vienna Development Method (VDM).

Keywords: ***Requirements Engineering, Requirements Modelisation, Software Engineering, Natural Language Processing***

1.0 INTRODUCTION

Requirements engineering is a very important stage in the software development life cycle. Having been ignored during the earlier years of the software revolution, requirements and specifications have attracted a lot of attention during the last decade. In the earlier years, the tools developed aimed to aid the control and management of requirements documents. Examples of such systems include the PSL/PSA system [1], the structured analysis language (SA) [2] and the SREM project [3,4]. Later, more formal and rigorous approaches were developed. These are known as formal methods or formal specification languages, which include VDM [5], Z [6] and RAISE [7].

One of the main aspects of requirements documents is informality. At an early stage of the requirements analysis, a document written in natural language will always exist and will be the starting point of the requirements analysis process. Many authors have stated that informality will always exist in the requirements analysis phase [8,9]. Adding to this statement, analysts find the use of formal methods very difficult because it requires a strong mathematical background. These two facts (existence of informality, and the difficulty to use formal methods) have encouraged some authors to investigate the possibility of developing tools to help analysts in the process of translating informal requirements to formal specifications [9,10,11, 12,13], or semi-formal specifications into formal specifications [14,15]. Some authors have used the later techniques of knowledge base systems to develop tools for the requirements and specifications phases. These include systems such as ARIES [16,17], the Requirements Apprentice (RA) [18,19,20], and the SPECIFIER [21].

The work presented in this paper is the earlier stages of the FORSEN system developed by Meziane and Vadera [13,22] which aims at producing formal specifications from informal (English) ones. These stages are mainly concerned with the analysis of natural language requirements documents. Their aim is to detect ambiguities and incompleteness present in the informal requirements. The result of the natural language analysis is the production of a unique interpretation for each English sentence. This interpretation is represented using the Logical Form Language. Once the logical form expressions are produced, the next step aims at producing an entity relationship model (ERM) for the proposed requirements. In the present paper, we start first by giving an overview of the logical form language in section 2. Section 3 summarises the different ambiguities that can be detected when using the logical form language. Section 4 shows how the entity relationships models are derived from the logical forms produced. Finally, in section 5 we show how the degrees of the relationships are identified. We conclude our work by applying our approach to a practical example that was developed independently of this work.

2.0 AN OVERVIEW OF THE LOGICAL FORM LANGUAGE

The Logical Form Language (LFL) is a Meaning Representation Language (MRL) [23] where the meanings of sentences are represented as logical forms. The LFL is used by the Modular Logic Grammars (MLG) formalism. For more details on the formalism, the reader is encouraged to consult McCord's works [24,25,26,27,28]. The main predicates in the LFL are word senses. Each predicate takes a fixed number of arguments. The arguments might be variables, constants or other logical forms. The formation rules for logical forms are as follows:

- If **P** is a predicate of LFL taking **n** arguments, and each of $\mathbf{x_1,...,x_n}$ is either a constant or a logical form or a variable then $\mathbf{P(x_1,...,x_n)}$ is a logical form.

- If **P** and **Q** are logical forms then **P** & **Q** is a logical form.

The predicates and arguments are obtained from the different parts of the sentence. Each syntactic group has a representation in LFL. The semantic interpreter will combine these syntactic groups to produce the complete interpretation of the sentence. In the next subsections, we will show how the different syntactic groups are represented in LFL.

2.1 Interpretation of Noun Phrases

In the LFL, a noun is generally represented as a 1-place predicate where the name of the predicate is obtained from the singular form of the noun. Examples:

stock is represented by *stock(X)*
company is represented by *company(X)*

where *X* is a variable and *stock* and *company* are predicates. In general, we adopt the Prolog convention that variables begin with a capital letter. There are two exceptions to the representation of nouns by 1-place predicates. Some nouns, called relational nouns, take two arguments. Example:

father is represented by *father(X,Y)*

which is interpreted as *X* is the father of *Y*. Depending on the context, some ordinary nouns may behave as relational nouns and therefore take two arguments as in the sentence:

"The company maintains a description for each item of stock."

The noun *item* is related to the noun *stock* and is represented by *item(X,stock)*. The other exception is the representation of proper nouns which correspond to constants in LFL.

In LFL, noun phrases do not have isolated meanings of their own but only contribute to the meaning of the sentence in which they appear. The head noun of a noun phrase is usually modified by premodifiers and postmodifiers. In the following subsections we analyse the classes of noun modifiers.

2.2 Determiners

Determiners are part of a large class of modifiers called focalizers. Focalizers are words that need a focus to determine the meaning of a sentence. In most cases, determiners' senses, as predicates in LFL, have two arguments which are filled by logical forms. The first argument is called the *base* of the determiner and the second is called the *focus.* In general a determiner is represented by:

determiner(Base,Focus)

Typically, the base comes from the remainder of the noun phrase in which the determiner appears, and the focus comes from some of the sisters of the noun phrase. An example of a sentence involving determiners and its logical forms is:

"The company maintains a system."
the(company(X),ex(system(Y), maintain(X,Y)))

We can read this logical form as follows. The quantifier *the* has two arguments: the base *company(X)* which comes from the noun phrase "The company" and the focus: *ex(system(Y), maintain(X,Y))* which comes from the verb phrase "maintains a system" (a sister of the previous noun phrase in the tree structure of Fig. 1). This denotes that there is a system *Y* that is maintained by the company *X.* The pair *(Base,Focus)* is called the *scope* of the determiner.

2.3 Adjectives

When interpreting adjectives, we distinguish between two categories: *extensional adjectives* and *intentional adjectives.* Intentional adjectives occur in a composed noun where it is not possible to dissociate the adjective from the other parts of the composed noun. Intentional adjectives take logical forms as arguments.

For example:

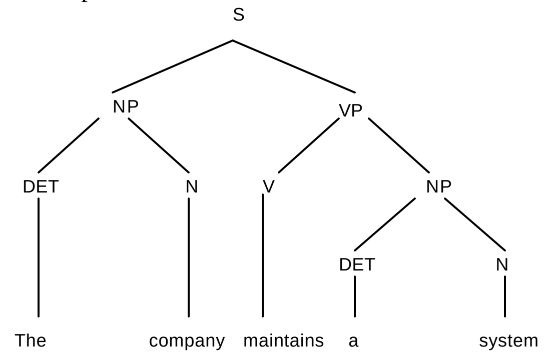


Fig. 1: A tree structure

"The pilot uses a moving map display."
is represented by:

the(pilot(X), the(moving(map(display(Y))),use(X,Y)))

which means that the *moving map display* is a single entity. Whereas if the sentence is interpreted as:

the(pilot(X), the(map(display(Y)) & moving(Y), use (X ,Y)))

it means that *map display* is an entity which is modified by the adjective *moving.* Extensional adjectives can be dissociated from the other parts of the composed noun and therefore behave as nouns in having one argument. For example:

*“**A complex aircraft uses a radar.”***
is represented by:

ex(aircraft(X) & complex(X), ex(radar(Y),use(X,Y)))

2.4 Interpretation of Verb Phrases

Depending on their category, verbs may be represented by predicates having nil, one, two or three arguments. The predicate's name is obtained from the infinitive form of the verb which is defined in the lexicon. Examples:

- ***“It snows.”***
  snow
- ***“The program crashed.”***
  the(program(X), ***crash(X))***
- ***“The student writes a program.”***
  the(student(X),ex(program(Y), ***write(X,Y)))***
- ***“The police gave a reward to John.”***
  the(police(X),the(reward(Y), ***give(X,Y,john)))***

One particular property of verbs is the voice. Verbs can have an active or a passive voice. For example the passive version of the last sentence is:

“The reward was given to John by the police.”

This sentence has exactly the same meaning as when the verb is in the active voice. Therefore it should have the same interpretation. MLGs have this ability and produce exactly the same logical form for both sentences.

2.5 Interpretation of Prepositional Phrases

The interpretation of prepositions is very difficult in the English language. In general, every preposition in a sentence can modify a noun phrase or a verb phrase. Hence, giving at least two different interpretations to the sentence. In the following subsections we analyse the different interpretation of different strings that are associated with prepositional phrases.

2.5.1 Interpretation of a String of the Form P-NP

Considering the general form PP $\rightarrow$ P-NP, a prepositional sense is a 2-place predicate. The first argument corresponds to the noun phrase associated with the preposition and the second corresponds to the phrase modified by the prepositional phrase. As mentioned earlier, a prepositional phrase may modify a verb or a noun phrase. An example where a prepositional phrase modifies a noun phrase is:

“The pilot uses an aircraft with a sophisticated system.”
which is represented by:

the(pilot(X), ex(aircraft(Y),
ex(system(Z)&sophisticated(Z)&with(Y,Z), use(X,Y))))

When a prepositional phrase modifies a verb, the second argument of the preposition predicate will be a logical form. For example the sentence:

“The pilot detects the obstacles with a radar.”
is represented by:

the(pilot(X), the(obstacle(Y), ex(radar(Z),
with(Z,detect(X,Y)))))

2.5.2 Interpretation of Intransitive Prepositions

For this category of prepositional phrases, the following interpretations are adopted:

1. If the preposition plays the role of an adverb, then it is treated like an adverb as in the following example:

   “John went downstairs.”
   downstairs(go(john))

The preposition takes a single argument that represents the rest of the sentence. In the above logical form, ***go*** is used since it is the infinitive of the verb ***went***.

2. If the preposition is a particle of a verb, then it is reacted as part of the verb as in:

   “John puts the clothes on.”
   the(cloth(X), put_on(john,X))

2.5.3 Interpretation of a String of the Form P-P-NP

Syntactically, this string can be parsed in two different ways. In the case where there is a preposition which has a prepositional phrase as a complement, the prepositions are combined as if they form a single one as in:

“The challenger ran out of time.”
the(challenger(X),out_of(time,run(X)))

out of is a constituent that acts as a unit and cannot be separated. The previous sentence can be reformulated as:

“Out of time ran the challenger.”
but we cannot split the two prepositions.

When an intransitive preposition is followed by a normal prepositional phrase, the first preposition is considered as a particle of the verb it modifies. For example:

“The man raced away in a red car.”
the(man(X), ex(car(Y) &red(Y), in(Y, race_away(X))))

Here the two prepositions can be split and we can say:

“In a red car the man raced away.”

2.5.4 Interpretation of a String of the Form P-NP-P-NP

Three different cases can be identified in the syntax analysis of such a string. In the first case where the whole prepositional phrase can behave as a single constituent, the prepositions successively modify the verb phrase. Example:

"The flight is planned from Blackpool to Doncaster."
from(blackpool,to(doncaster,the(flight(X),be(X,plan(X)))))

In the second case, only the noun phrase of the first prepositional phrase behaves as a single constituent. Therefore the noun phrase is used as a complement of the first preposition. The following example illustrates this:

"John went to the house in the woods."
the(wood(X),the(house(Y) & in(Y,X),to (Y,go(john))))

The third case is interpreted as two independent prepositional phrases modifying a verb phrase. Example:

"John went to the park with Mary."
the(park(X),to(X,with(mary,go(john))))

2.6 Interpretation of Adverbs

All adverbs take logical forms as arguments. Some adverbs take a single argument as in this example:

"John sold the car yesterday."
which is interpreted as:

yesterday(the(car(X), sell(john,X)))

Other adverbs take two arguments. This category of adverbs is part of the focalizer class. The adverbs have the same interpretation as determiners and need a base and a focus to determine their scope. They have the structure:

adverb(base,focus)

This case of adverbs is used by McCord to determine which part of the sentence is stressed. This aspect of McCord's work [27] is not relevant for the current research.

2.7 Interpretation of Conjunctions

A sentence can contain an infinite number of co-ordinating conjunctions. This makes the analysis of such sentences very difficult. An attempt to treat conjunctions by Dahl and McCord [29] has resulted in a system that analyses only very simple sentences containing a maximum of two conjuncts. For example the analysis of:

"Each man ate an apple and a pear."
can result in the following logical form:

each(man(X),ex(apple(Y),eat(X,Y))&ex(pear(Z),eat(X,Z)))

The embedding of co-ordinating conjunctions in a sentence makes the production of logical forms very difficult and its interpretation becomes ambiguous.

However, subordinating conjunctions are much easier to treat. In general, they involve only two clauses that are related by one conjunction as in the example:

"Each item of stock is assigned a unique identifier when it is introduced."
This is interpreted as:

all(item(X,stock), ex(identifier(Y)&unique(Y),
when(be(X,introduce(X)), be(X,assign(X,Y))))))

As we have hinted, handling co-ordinating conjunctions adequately in general remains a difficult research problem [29]. Hence we resolve co-ordinating conjunctions manually by splitting the conjuncts into simple sentences.

2.8 Interpretation of Pronouns

Pronouns are another class of words which are difficult to deal with in general [30]. An example of how some pronoun references are resolved is shown in the following:

"Bill owns a cat. He likes it."
This results in the following logical form:

ex(cat(X), own(bill,X) & like(bill,X))

Our current implementation also omits the resolution of pronoun references.

3.0 USING LFL TO DETECT AMBIGUITIES

The main disadvantage in using natural language to write specifications is the potential for ambiguities. In any specification document written in natural language, it is essential to remove any ambiguity before proceeding to any further analysis. This leads to two major problems. The first problem is the detection of the ambiguities that are present and the second is the resolution of these ambiguities.

It is claimed [9] that humans are bad at detecting ambiguities, but are very good at resolving them. In this section we show how ambiguities and incompleteness can be detected when using the logical form language. These ambiguities and incompleteness are categorised as:

- Lexicographic ambiguities.
- Grammatical ambiguities.
- Textual cohesion

We consider each of these in the following subsections.

3.1 Lexicographic Ambiguities

Many words in English have different meanings. The resolution of these ambiguities requires the selection of the exact definition for each word. The logical form language uses semantic types to resolve these ambiguities. These types can be used as a set of some general classes as defined by McCord[28], and Allen[31]. Another category of lexicographic ambiguities is when a word belongs to more then one syntactic category. For example *"flies"* may be the plural of the noun *"fly"* or the present, singular third person of the verb *"to fly"*. These ambiguities are easily identified when the syntactic category of the word is identified. For example, in the sentence:

"The pilot flies over the town."

Once the category of the word *"flies"* is identified as a verb, the ambiguity is resolved. When using LFL most of the lexicographic ambiguities can be resolved using this approach.

3.2 Grammatical Ambiguities

A grammatical ambiguity occurs when there is more than one possible parsing for a sentence or part of a sentence. The different parsings will lead to different interpretations and different meanings. The sentence:

"The pilot draws the tracks of the route on the map."

has three different parsings which lead to the following interpretations.

" The pilot draws (the tracks of the route) on the map."
That is, the information is drawn on the map.

"The pilot draws (the tracks of the route on the map)."
That is, the tracks are already on the map and the drawing is done somewhere else.

"The pilot draws the tracks of (the route on the map)."
That is, the route is given on the map and its tracks are drawn somewhere else.

The location where the drawing takes place is different according to which interpretation we choose. Finding the proper place to attach the preposition phrase needs prior knowledge of the event. In general, a natural language understanding system, cannot decide which interpretation is meant. It can, however, highlight the ambiguity and produce all possible interpretations and it is the role of the analyst to choose the desired interpretation.

Ambiguities may also occur within parts of a sentence. This ambiguity occurs mainly in noun phrases. For example the sentence:

"A complex aircraft uses a moving map display."

will have two different parsings, according to whether *moving* is interpreted as an adjective modifying the noun *map display* or as part of the noun *moving map display*.

This ambiguity may occur also in the interpretation of nominalisations -- where the **ing form** of a verb plays the role of a noun -- such as in the following sentence:

"The shooting of the hunters is disgraceful."
If *the shooting of the hunters* is interpreted as a subject then the shooting is done by the hunters, but if it is interpreted as an object then the hunters have been shot.

3.3 Textual Cohesion

When writing texts, many techniques are used to ensure that the different parts of the text are connected correctly and to ensure a smooth transition from one idea to another. These techniques are called *textual cohesion*. The connected sentences will define the context in which they have to be interpreted. Therefore, it will be difficult to take a sentence out of context and try to interpret it. Each sentence is linked to the others.

Unfortunately, textual cohesion is a source of ambiguities and incompleteness. In most specifications, the authors attempt to avoid the use of ambiguous textual cohesion. However, it is hard to find a realistic specification without ambiguities or incompleteness. A complete account of these problems can be found in Meziane's thesis [22]. These problems are very difficult to address and resolve if any ambiguity occurs. Our approach can help detect some lexical cohesion ambiguities such as synonyms, subtypes, or use of the same word in different sentences. After analysing the informal requirements and resolving ambiguities, the next step in our approach is to produce an entity relationship model.

4.0 IDENTIFYING THE ENTITY RELATIONSHIP MODELS

The first task in identifying an ERM is to obtain the list of entities in the specification and the relationships between the entities. There is no clear definition of what constitutes an entity. In SSADM [32], for example an entity is defined as something of importance to the system about which information can be held. The same definition is also used by Bowers [33], who further suggests that entities can be:

- **Objects**: Person, car.
- **Events**: Birth, scoring a goal.
- **Activities**: Production, playing.
- **Associations**: Marriage (X is married with Y).

Grammatically speaking, the above list gives types which define entities that are related. They all belong to the same grammar category of *nouns*. Several authors have reported that nouns denote entities [32,34].

Entities have relationships with other entities. In a bank system, for example, there is a relationship between the entities "Customer" and "Account". These two entities are related by an ownership relation which will probably be described in the various documents by the verb "have". We, therefore, believe that relations are described by verbs. This view is also supported by others [32,34].

We base our identification process on the view that entities are denoted by nouns and relationships by verbs. For example, in the following sentences:

1. ***"The pilot chooses the waypoints from the air."***
2. ***"A complex aircraft uses a radar."***

The nouns suggest the entities:

{pilot, waypoint, complex aircraft, radar,air }

and the verbs the relations:

{choose, use}

However, as we mentioned in the previous section, noise exists in specifications and this can lead to the presence of irrelevant entities and relations in such lists. For example, in the following sentences:

1. ***"An example route is planned for a flight from Blackpool to Doncaster."***
2. ***"The pilot may have unnecessarily flown through a storm."***

In the first sentence, the verb *planned* does not describe any process but just a statement. In the second sentence, the entity *storm* may be not important to the specification of a flight database.

Bearing in mind that we aim to develop an interactive tool, this deficiency can be circumvented by requiring the analyst to remove those nouns and verbs that are not important when they are detected. We, therefore, expect that an analyst should filter such a list of entities and relations before proceeding.

We can, of course, obtain a list of nouns and verbs by simply scanning the text. However, this approach has several deficiencies and does not help one to:

1. Find the relationships between entities. For example, in the sentence:
"A company maintains a description for each item of stock."
Although we can list the nouns as *company, description*, and *stock*, we do not obtain any relationships between these entities.
2. Identify the degree of the relationships between the identified entities.
3. Extract compound nouns without ambiguity. For example, consider the two sentences:

- ***"A computer-assisted flight planning system is used by a complex aircraft."***
- ***"A pilot planning a risky flight needs special training."***

In the first sentence, *planning* is a part of the compound noun *computer-assisted flight planning system* whose form must be preserved. In the second sentence, planning is part of the participial verb phrase "planning a risky flight" and is not part of a compound noun.

The following subsections show how the use of logical form helps us to overcome these problems.

4.1 Identifying Entities

The nouns form the basic list of entities. Below, we see how entities can be extracted from sentences that contain simple and compound nouns.

4.1.1 Simple Nouns

Simple nouns are extracted from noun phrases containing just one noun. The sentence:

"The aircraft may hit an obstacle."

contains two noun phrases: *The aircraft* and *an obstacle*. Each noun phrase is composed of a unique noun and each noun is extracted as an entity with its associated quantifier. Relational nouns are also extracted in a similar fashion. For example the sentence:

"The system of an aircraft comprises the plan of the pilot."

results in the entities:

{ system, aircraft, plan, pilot }.

Proper nouns identify particular objects and therefore do not normally constitute entities. Hence in a sentence like:

"An example route is planned for a flight from Blackpool to Doncaster."

Blackpool and *Doncaster* do not constitute entities.

4.1.2 Compound Nouns

Compound nouns are nouns which are composed of two or more nouns or a combination of nouns and adjectives. For example, the following sentences:

1. ***"A complex aircraft uses a computer-assisted flight planning."***
2. ***"The flight planning software package calculates the route tracks."***

will have respectively the following logical forms:

1. *ex(aircraft(X) & complex(X),*
   ex(computer-assisted(flight(planning(Y))),use(X,Y)))
2 *the(flight(planning(software(package(X)))),*
   the(route(track(Y)), calculate(X,Y)))

In the first sentence, the noun phrase, *a computer-assisted flight planning* is composed of four nouns. In the second, the noun phrase *the flight planning software package* is composed of four nouns. As we can see from the above logical forms we can easily extract the entities. The entities identified are: *computer-assisted flight planning* and *flight planning software package.*

Identifying entities by using just the head noun may, of course, lead to confusion. For example, in a specification of an aircraft system (see the case study in appendix A), both the description of a simple aircraft and a complex aircraft may occur.

4.2 Identifying Relations

As we have mentioned in section 4, a natural way of identifying relationships is to use verbs and relational nouns.

4.2.1 Identifying Relationships within Relational Nouns

Relational nouns always define relationships between nouns. The sentences:

1. ***"The company maintains a description for each item of stock. "***
2. ***"The system of a simple aircraft comprises the plan of the pilot"***

will produce respectively the following logical forms:

1. *all(item(X,stock) ,the(company(Y),ex(description(Z), for(X,maintain(Y,Z)))))*
2. *ex(aircraft(X) & simple(X), the(system(Y,X), the(pilot(Z), the(plan(T,Z), (comprise(Y,T)))))*

The logical forms show clearly the relations defined by relational nouns. In the first sentence there is a relation between *item* and *stock* and this is shown by *item(X,stock)*. In the second, there are two relations. The first relation is between *simple aircraft* and *system* and the second relation is between *plan* and *pilot*. These two relations are respectively shown by *system(Y,Z)* and *plan(T,Z)*. The direction of the relation is from the second entity to the first and we use *of* to name the relationship. The relations extracted for the above examples are shown in Fig. 2.

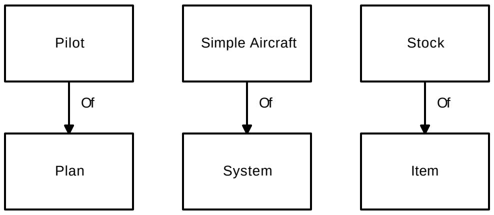


Fig. 2: Relationships extracted

4.2.2 Identifying Relationships within Verb Phrases

Verbs generally refer to actions, events and processes [35]. In the case of transitive verbs, the verb defines a relation between two entities. Let us consider the sentences:

1. ***"The pilot chooses the waypoints from the air."***
2. ***"The system of a simple aircraft is considered to comprise the plan of the pilot."***

In the first sentence, the verb *chooses* relates the entities *pilot* and *waypoints*. This information is readily available from the logical form of the first sentence:

the(pilot(X), the(waypoint(Y), the(air(Z), from(Z,choose(X,Y)))))

where *choose* relates the variables X and Y which are defined in the logical form to be of type *pilot* and *waypoint*.

The second sentence has two verbs, making it a little more complex to analyse. The logical form produced for this example is:
ex(aircraft(X) & simple(X),the(system(Y,X), the(pilot(Z), the(plan(T,Z), be(Y,consider(Y,comprise(Y,T))))))

The verb *comprise* introduces the main action (which is represented in natural language as an infinite complement of the verb *consider*), and is therefore extracted as the relationship between *the system of a simple aircraft* and the *plan of the pilot*. The verb *consider* plays a subsidiary role and does not relate any entities. Again this information is ready to extract from the logical form. In cases where the logical form contains more then one verb, the inner verb phrase identifies the relationships between the entities. The relations extracted for the above sentences are given in Fig. 3.

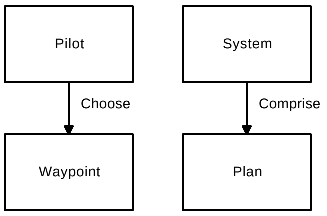


Fig. 3: Verb relation extracted

The next step is to identify the degree of the relationships. The next subsection shows how the degree of a relation can sometimes be determined from the logical form of a sentence containing the relationship.

5.0 QUANTIFICATION AND THE DETERMINATION OF THE DEGREE

Early attempts at natural language analysis assumed that quantifiers occurred explicitly in the text. Thus, it was assumed that the presence of a universal quantifier was always indicated by words like "every" and "all" and the presence of an existential quantifier was indicated by words like “some” [36]. However, many sentences are implicitly quantified by articles.

In this section we first examine how such implicit quantifiers can be identified and then show how quantified LFL statements can sometimes aid the identification of the degree.

5.1 Identifying Implicit Quantifiers

Most studies of quantification identify quantifiers from the articles present in the sentences [31,36]. Initial studies of quantification regarded both definite and indefinite articles as existential quantifiers, with some additional information in the case of definite articles [36]. More recent studies have shown various problems with this assumption and have shown how the indefinite article and the definite article can also lead to a universal quantifier. Below, we show the problems of identifying quantifiers from the articles “the” and “a” and our approach to these problems.

5.1.1 The Definite Article “the”

Russell [37] gives the following example to illustrate the meaning of the definite article “the”.

“The president of France is bald.”

Russell argues that this should be interpreted as:
\$X. president_of_france(X) ***L***
Ø(\$Y. president_of_france(Y) ***L*** *X ¹ Y)* ***L*** *bald(X)*

The “*additional information*” given for the interpretation of the definite article is given in the statement that **Y** is the one and only president of France.

McCord [28] recognises that this interpretation is inadequate in general but suggests that in some applications it is correct to translate “the” into the unique existence. However, he does not give any guidance as to when the usual existential quantifier should be used instead of the unique existential quantifier. In the case of obtaining the meaning of sentences in a requirements document, we cannot assume that one of these holds throughout the application. For instance, consider the sentences:

1. ***“The students passed the exams.”***
2. ***“The student passed the exams.”***

The first sentence does not suggest the unique existential quantifier, while the second does not suggest the normal existential quantifier. As we will see later, obtaining appropriate quantifiers is an important prerequisite for our approach to identifying the degree of relationships between entities. Hence, we have attempted to improve upon McCord's approach to this problem [28]. In our approach, we do not simply translate ``the" into the unique existence -- instead we use the singularity or plurality of the noun to determine if it should be translated to the unique existence or normal existence. That is, if the quantified noun is singular, we adopt the unique existence, otherwise we interpret it as a normal existential quantifier. Again this information is available from the LF of the sentence.

We concede that there remain sentences for which these approaches remain inadequate. For instance, the following examples given by Hess [36] are not covered:

1. ***“The unicorn is a mythical creature.”***
2. ***“The lion is a dangerous animal.”***

In the first sentence we do not presuppose the existence of unicorns, but the sentence nevertheless makes perfect sense. This kind of sentences are unlikely to appear in requirements documents because specification of systems are normally about concepts and objects that exist. The second sentence shows that depending on the context, “the" could be interpreted as a universal quantifier, or a unique existential quantifier.

5.1.2 The Indefinite Article “a”

The use of the indefinite article as a quantifier is always a source of ambiguity [31]. The indefinite article can sometimes be translated to the existential quantifier and sometimes to a universal quantifier. In this subsection, we investigate when the indefinite article is interpreted as an existential quantifier and when it is interpreted as a universal quantifier. According to Hess [36] the most important way to determine the quantification of a sentence is through the choice of the verb. For example, consider the following sentences:

1. ***“A text editor makes modifications to a text file”.***
2. ***“A text editor is making modifications to a text file”.***
3. ***“A text editor made modifications to a text file”.***
4. ***“A text editor has made modifications to a text file”.***

The present tense is used in example (1) to say that a text editor makes modifications to a text file in general. The main use of the present tense is to express habitual actions. In examples (2) to (4) we say that there is, or was, a case of a text editor making modifications to a text file. Therefore, Hess suggested that because the present tense is used in the first sentence, *text editor* must be universally quantified. Likewise, because of the tenses used in the other sentences, *text editor* must be existentially quantified in the remaining sentences.

In some cases the future is preferred over the present tense for general statements as in the following example:

“A man who loves a woman will stroke her”

Dynamic verbs, such as *to stroke*, seem to call for the future tense, whereas static verbs such as *to respect* seem to go better with the present tense. Hence, Hess formulated the following rules:

Rule 1: The subject of a sentence is existentially quantified if the VP is:
1. in the past tense.
2. in the progressive aspect, or
3. in the perfective aspect.

Rule 2: Otherwise the subject is universally quantified, in particular, if it is:
1. in the present tense or
2. in the future tense.

Once we have determined the quantification of the subject of the sentence, we have to do the same thing to the other components of the sentence. Let us consider the following examples:

1. ***"A man who loves a woman is happy."***
2. ***"A man that loves a woman respects her."***

Intuitively, we can see that *woman* should be existentially quantified in the first sentence and universally quantified in the second sentence. To observe the difference, let us consider the logical forms of these sentences:

1. *all(man(X),ex(woman(Y)&love(X,Y),happy(X)))*
2. *all(man(X),all(woman(Y)&love(X,Y),respect(X,Y)))*

The main verb of the first sentence is *happy* and does not refer to the noun phrase *woman*. In the second sentence the main verb *respects* refers to the noun phrase *woman*. This is the reason why the noun phrase *woman* should be existentially quantified in the first sentence and universally quantified in the second. Hence, Hess suggested a third rule which is:

Rule 3: In a restrictive noun phrase those arguments that are referred to by the main verb are universally quantified and those that are not referred to by the main verb are existentially quantified.

This rule now enables the correct interpretation of the above sentences. However, it does not hold for non-restrictive noun phrases. In particular, when a noun phrase appears at the right of a verb, the kind of sentences we have encountered suggest that the indefinite article should be interpreted as an existential quantifier. For example, in the sentence:

"A complex aircraft uses a radar."

The second indefinite article is interpreted as the existential quantifier and not as the universal quantifier. There are two exceptions to the above rules which are analysed in the following cases:

- As an exception to rule 2, the past tense can express a universally quantified assertion, as in the following example:

"A student read books when I was young."

This universal quantification is possible because the main verb (read) requires a spatial or temporal postmodifier(when).

- As an exception to rule 1, the progressive aspect can express universal quantification as in:

  "John is always coming late"

This is only possible when the verb is modified by expressions such as "always", "in general", "regularly". To cover these exceptions, we can suggest the following fourth rule which takes precedence over rules 1 and 2.

Rule 4:
1. The past tense can express a universally quantified assertion if the main verb requires a spatial or a temporal postmodifier.
2. The progressive aspect can express a universal quantification if the verb is modified by expressions such as "always", "in general" and "regularly".

5.2 Obtaining the Degree from the Quantifiers

In the last subsection we saw how we could obtain quantified logical forms. In this subsection we show how the quantifiers associated with each entity can be used to determine the degree of a relationship between the entities. It is not always possible to determine the degree of a relation from the quantifiers. However, we describe how our system gives a default degree for some cases. Of course, the user is allowed to override the system determined degrees.

5.3 Identifying Many-to-One Relationships

Consider the following examples and their logical forms:

1. ***"A complex aircraft uses the radar. "***
   all(aircraft(X) & complex(X), the(radar(Y), use(X,Y)))
2. ***"A student passed the exam."***
   ex(student(X), the(exam(Y), pass(X,Y)))
3. ***"The students passed the exam."***
   the(student(X), the(exam(Y), pass(X,Y)))
4. ***"The company maintains a description for each item of stock."***
   all1(item(X,stock), the(company(Y), ex(description(Z), for(X,maintain(Y,Z)))))

In the first example, the first entity in the relation is quantified by the universal quantifier and the second by the unique existential quantifier (the definite article quantifying a singular noun). Then, by definition, we have a many-to-one relationship from the first entity to the second. In the second and third examples, the first entity is quantified by the normal existential quantifier and the second by the unique existential quantifier. Based on our current experience, in such cases we interpret the existential quantifier as referring to more than one occurrence of the first entity. That is, many occurrences of the first variable are related to only one occurrence of the second variable. Then by definition, we have a many-to-one relation between the entities *student* and *exam.*

In the fourth example, we consider the relation between *item* and *description*. The entity *item* is quantified by the universal quantifier and the entity *description* is quantified by the existential quantifier. Based on the different examples we have analysed, we infer a many-to-one relation between *item* and *description*.

Notice that the degrees given by default by this approach to the last three examples are not as strong as the one given to the first example. The analyst, is of course, allowed to override the degrees identified by the system.

Some one-to-many relationships can also be detected by the system. For example, consider the following sentences and their logical forms:

1. ***"The company maintains a description for each of stock."***
   all1(item(X,stock), the(company(Y), ex(description(Z), for(X,maintain(Y,Z)))))
2. ***"The student passed all exams."***
   the(student(X), all2(exam(Y), pass(X,Y)))

Logical form language distinguishes between its different quantifiers by associating different predicates. For example, LFL associates the predicate *all1* for the determiner *each*, *all2* for the determiner *all* and the predicate *all* for the interpretation of the indefinite article as a universal quantifier. These different predicates are used to determine the priorities between the quantifiers. These differences also help in the interpretation of the sentences. Hence, in the first sentence, the phrase *each item of stock* suggests that we are talking about one stock system that contains many items (i.e., a one-to-many relation between the entities *stock* and *item*).

Sentences where the first entity is singular and quantified by the definite article define one-to-many relationships. The second sentence is a typical example. An exception to this rule occurs when the second entity is also quantified by ``the" and is singular. In this case, we infer a one-to-one relationship between the entities.

We have now given several cases in which we can identify the degree of a relationship. In other cases, when it is difficult to predict the degree of a relation, we let the user decide it. At this stage we should have a list of entities, relations and the degrees of the relations and therefore the entity relationship model.

6.0 CONCLUSION

We conclude this paper by presenting the results of a realistic case study provided by the Department of Systems Computing of British Aerospace Ltd [38]. It concerns the planning of a route for a flight from one point to another. The original (English) text needs some pre-processing because the current implementation of the natural language processor does not handle conjunctions and pronoun references, we resolve these two problems manually (see [22] for a complete description). The English text (given in appendix A) is translated into a set of logical forms after the first stage. Given the logical forms, the system produces a list of entities. For the current case study, the system identifies 55 entities. In general, this list may include "noisy" entities. Such noisy entities are detected and removed by the analyst by using problem dependent knowledge. The next stage is the identification of the relations between the entities. In this case study, 52 relations are identified. These relations are binary relations. Some degrees are identified automatically by the system. For the present case study, the system successfully identifies the degrees of 37 relations. The remaining degrees are identified by the analyst. In general, during the identification of an entity relationship model, an entity may be related to many other entities. For example, in this case study the entity "Complex Aircraft" is related to the entities: "Inertial Navigation System", "Moving Map Display", "Autopilot" , "Radar" ...etc. By combining these relations, we obtain the entity relationship model given in appendix B. For the whole set of the models produced for the case study, the reader is referred to papers [13] and [22]. The last step of the FORSEN system translates the entity relationship model to formal data type (in VDM [5]). This process is based on previous work on obtaining formal datatypes from semi-formal specifications [14,15]. In conclusion, the results obtained by this research are very encouraging and suggest that the use of natural language understanding techniques can lead to a useful tool for aiding the development of formal specifications from informal specifications.

REFERENCES

[1] D. Teichroew and E. A. Hershey, "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing System", *IEEE Transactions on Software Engineering*, Vol. 3, No. 1, 1977, pp. 41-48.

[2] D. T. Ross and JR K. E. Schoman, "Structured Analysis for Requirements definition", *IEEE Transaction on Software Engineering*, Vol. 3, No. 1, 1977, pp. 6-15.

[3] M. W. Alford, "A Requirements Engineering Methodology For Real-Time Processing Requirements". *IEEE Transactions in Software Engineering*, Vol. SE-3, No 1, 1977, pp. 60-69.

[4] T. E. Bell, D. C. Bixler and M. E. Dyer, "An Extendible Approach To Computer-Aided Software

Requirements", *IEEE Transactions in Software Engineering,* Vol. SE-3, No 1, 1977, pp. 49-60.

[5] C. B. Jones, *Systematic Software Development Using VDM.* Prentice Hall International, 1990.

[6] J. M. Spivey, *The Z Notation a Reference Manual*, Prentice Hall London, 1989.

[7] The RAISE Language Group. The RAISE Specification Language. Prentice Hall International, 1992.

[8] R. Balzer, N. Goldman and D. Wile, "Informality In Program Specification", *IEEE Transactions in Software Engineering,* Vol. SE-4, No 2, 1978, pp. 94-103.

[9] S. G. Presland, *The analysis of Natural Language Requirements Documents*, Ph.D Thesis, University of Liverpool, 1986.

[10] R. Balzer, "A 15 Year Perspective On Automatic Programming", *IEEE Transactions in Software Engineering,* Vol. SE-1, No 11, 1985, pp. 1257-1268.

[11] J. R. Comer, *An Experimental Natural Language Processor For Generating Data Type Specifications*. PhD thesis, Texas A & M University, 1979.

[12] A. Fantechi et al., "Assisting Requirements Formalization By Means Of Natural Language Translation". *Formal Methods in System Design* No. 4, 1994, pp. 243-263.

[13] S. Vadera and F. Meziane, "From English to Formal Specifications", *The Computer Journal,* Vol. 37. No. 9, 1994, pp. 753.

[14] J. Dick and J. Loubersac, "Integrating Structured And Formal Methods: A Visual Approach To VDM". In *Third European Software Engineering Conference, LNCS 550,* 1991, pp. 37-59.

[15] F. Polack, "Integrating Formal Notations and System Analysis: Using Entity Relationship Diagrams", *Software Engineering Journal,* Vol. 7, No. 5, September 1992, pp. 363-371.

[16] W. L. Johnson, K. M. Benner and D. R. Harris, "Developing Formal Specifications From Informal Requirements". *IEEE Expert*, August 1993, pp. 82-90.

[17] W. L. Johnson, M. S. Feather and D. R. Harris, "Representation and Presentation of Requirements Knowledge". *IEEE Transactions on Software Engineering,* Vol. 18, No 10, 1992, pp. 853-869

[18] H. B. Reubenstein, *Automatic Acquisition of Evolving Informal Description,* PhD Thesis, MIT Artificial Intelligence Laboratory, 1990.

[19] H. B. Reubenstein and R. C. Walters " The Requirements Apprentice: Automated Assistance for Requirements Acquisition", *IEEE Transaction on Software Engineering,* Vol. 17, No. 3, 1991, pp. 226-240.

[20] H. B. Reubenstein and R. C. Walters " The Requirements Apprentice: An Initial Scenario", *In Proceedings of the 5th International Workshop on Software Specification and Design,* IEEE Computer Society Press, Washington, DC, May 1989.

[21] K. Miriyala and M. T. Harandi, "Automatic Derivation of Formal Software Specifications from Informal Description", *IEEE Transactions on Software Engineering,* Vol. 17, No. 10, 1991, pp. 1126-1142.

[22] F. Meziane, *From English to Formal Specifications,* Ph.D Thesis, University of Salford, 1994.

[23] G. Gazdar and C. Mellish. "Natural Language Processing In PROLOG". *An Introduction To Computational Linguistics*. Addison-Wesley Company, 1989.

[24] McCord, "Slot Grammars". *American Journal of Computational Linguistics,* Vol. 6, No 1, 1980, pp. 31-43.

[25] M. McCord, "Using Slots and Modifiers in Logic Grammars". *Artificial Intelligence,* No 18, 1982. pp. 327-367.

[26] M. McCord, " Modular Logic Grammars", *in Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago 1985, pp. 104-117.

[27] M. McCord, "Focalizers, the Scoping Problem, and Semantic Interpretation Rules in Logic Grammars", in Michael Van Canengham and David H. D. Warren, editors, *Logic Programming and its Applications,* Alex Publishing Corporation Norwood, New Jersey, 1986, pp. 391-402.

[28] M. McCord, "Natural Language Processing in Prolog", in Walker Adrian, editor, Developing Expert, Database, and Natural Language Systems, Knowledge Systems and Prolog, Addison-Wesley Publishing Company, 1990, pp. 391-402.

[29] V. Dahl and M. McCord, "Treating Co-ordination In Logic Grammars". *American Journal of*

Computational Linguistics, Vol. 9 No. 3, 1983, pp. 69-91.

[30] J. Hobbs. "Resolving Pronoun References". *LINGUA*, Vol. 44, No 4, 1978, pp. 311-338.

[31] J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Inc., 1987.

[32] C. Ashworth and M. Goodland, *SSADM: A Practical Approach*, McGraw-Hill Book Company, 1990.

[33] D. S. Bowers, *From Data To Data Base*, Van Nostrand Reinhold (U.K.) Co. Ltd., 1988.

[34] C. Gane and T. Sarson, *Structured System Analysis*, Prentice-Hall Software Series, 1979.

[35] H. Jackson. *Analysing English*. Pergamon Press, 1982.

[36] M. Hess. "How Does Natural Language Quantify?", in Second Conference of the European Chapter of the Association for Computational Linguistics, February 85, pp. 8-15.

[37] B. Russel, "On Denoting", In *Mind*, NS, 14, 1905, pp. 479-493.

[38] B. Hepworth. *An Introduction to Z.* Technical report BAe-WIT-RP-GEN-SWE-152, Systems Computing Department, British Aerospace Ltd, February 1988.

APPENDICES

A. The Aircraft Specifications

In this appendix, we present the input text to the natural language analyser. A key is associated to some sentences to show how they are obtained from the original text. The explanation of these keys is:

Ci	All the sentences referred by Ci are the result of the split of the same conjunct.
T	The sentence is added to replace a table.
PR	A pronoun reference is encountered and resolved.

The text is:

[1] An example route is planned for a flight from blackpool to doncaster.

[2] The route is planned as a number of the discrete tracks between the intermediate waypoints.

[3] The planned tracks will assure the safe arrival of the aircraft over doncaster when they are flown in correct order by the aircraft.

[4] The pilot may have unnecessarily flown through a storm (C1).

[5] The pilot may have unnecessarily flown through a controlled airspace (C1).

[6] The aircraft may hit an obstacle (C2).

[7] The aircraft may hit another aircraft (C2).

[8] The pilot of a simple aircraft without a sophisticated electronic navigation system would be cleared to undertake a risky flight.

[9] The pilot chooses the visible waypoints from the air.

[10] The pilot draws the tracks of the route on the map.

[11] The pilot steers a heading giving the required tracks along the ground.

[12] The pilot scans the ground for the visible features (PR).

[13] The pilot verifies the visible features against the map.

[14] The system of a simple aircraft can be considered to comprise the map of the pilot (C3).

[15] The system of a simple aircraft can be considered to comprise the plan of the pilot (C3).

[16] The system of a simple aircraft can be considered to comprise a heading indicator (C3).

[17] The system of a simple aircraft can be considered to comprise the visual sense of the pilot (C3).

[18] The system of a simple aircraft contrasts with the system of a complex aircraft.

[19] A complex aircraft uses a whole range of the electronic equipment to support the navigation.

[20] A complex aircraft uses a computer-assisted flight planning (C4).

[21] A complex aircraft uses an inertial navigation system (C4).

[22] A complex aircraft uses a radar (C4).

[23] A complex aircraft uses a moving map display (C4).

[24] A complex aircraft uses a route display (C4).

[25] A complex aircraft uses a waypoint display (C4).

[26] A complex aircraft uses an autopilot (C4).

[27] The pilot chooses the waypoints from blackpool to doncaster in a complex aircraft.

[28] The pilot identifies each waypoint with a number (C5).

[29] The pilot identifies each waypoint with a grid reference (C5).

[30] The grid reference contains the latitude (T).

[31] The grid reference contains the longitude (T).

[32] The information is used as input to a flight planning software package.

[33] The flight planning software package calculates the route tracks (C6).

[34] The flight planning software package calculates the distance between waypoints (C6).

[35] The flight planning software package calculates the heading for the wind conditions (C6).

[36] The flight planning software package calculates the non-violation of a controlled airspace (C6).

[37] The derived information may be listed for the pilot to record on the map (C7).

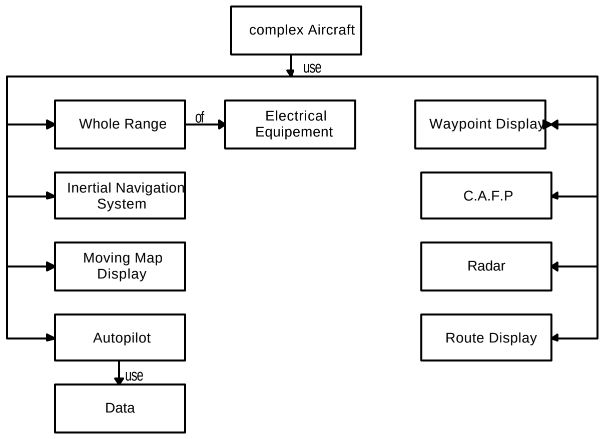
[38] The derived information may be transferred to a cassette tape (C7).

[39] The cassette tape is used to load the navigation database on the aircraft.

[40] The autopilot uses the data to fly the aircraft according to the plan of the pilot.

[41] The route is composed of waypoints (T).

B. The ER Diagram of a complex aircraft



BIOGRAPHY

Farid Meziane is a lecturer in Computer Science at the University Malaysia Sarawak (UNIMAS), Malaysia. He holds an engineer degree in computer science from the National Institute of Computer Science, Algiers, Algeria and a Ph.D in Computer Science from the University of Salford, UK. His research interests include requirements engineering, formal methods, natural language processing and knowledge based systems.

Sunil Vadera is a lecturer in Computer Science at the University of Salford, UK. He holds a Ph.D in computer Science from the University of Manchester. He is a member of the British Computer Society (BCS) and on the BCS membership accreditation panel.

His research interests cover formal methods, theorem proving, expert systems, machine learning, and case base reasoning. His research papers have been published in journals that include Formal A`spects of Computing, The Computer Journal, The Software Engineering Journal, and The Expert Systems Journal.