Financial information extraction using pre-defined and user-definable templates in the LOLITA System

Marco Costantino, Richard G. Morgan, Russell J. Collingham
Department of Computer Science, University of Durham
Durham, DH1 3LE, U.K.
Tel +44 191 374 2549, Fax +44 191 374 2560
marco.costantino@durham.ac.uk

September 24, 1996

Abstract

This paper addresses the issue of information extraction in the financial domain within the framework of a large Natural Language Processing system: LOLITA. The LOLITA system (Large-scale Object-based Linguistic Interactor Translator and Analyser) is a general purpose natural language processing system. Different kinds of applications have been built around the system's core. One of these is the financial information extraction application, which has been designed in close contact with experts from the financial market in order to overcome the lack of usefulness of many other systems.

Three predefined groups of templates have been built according to the "financial activities approach": company related templates, company restructuring templates and general macroeconomic templates. In addition, the user-definable template interface allows the user to define new templates using natural language sentences.

After describing LOLITA as a general purpose base NLP system, the paper addresses the issue of how information extraction is performed within the system and how the user-definable template interface has been designed.

Keywords: Natural Language Engineering, Information Extraction, User-definable information extraction, Finance.

1 Introduction

Many natural language systems have been built to solve specific and limited tasks. LOLITA has been built with no particular application in mind. However, the flexibility of the system allows the realisation of different kinds of applications around the system's core. One of these is the LOLITA financial information extraction application, which is described in this paper. The templates have been defined according to the "financial activities" approach which identifies three main groups of relevant financial activities in the financial market each of which is associated to a specific template. The user can also define additional templates using sentences in natural language with the user-definable template interface. The main characteristic of how information extraction is performed within the LOLITA system is that deep natural language understanding is used in order to identify the relevant information in the source articles. The work is organised as follows: section 2 describes the generic information extraction tasks and the main systems previously. In section 3 we describe the LOLITA system as a general purpose natural language processing system. In section 4 we describe the way in which information extraction in LOLITA is performed. In section 5 we focus on the LOLITA financial information extraction system. Finally, in section 6 we describe the natural language user-definable template interface.

2 Information extraction

The goal of *information extraction* is to extract specific kinds of information from a source document [Riloff and Lehnert, 1994]. For example, the source article could consist of the following text:

FLORHAM PARK, N.J. (AP) - Generic drug maker Schein Pharmaceutical Inc. will acquire Marsam Pharmaceuticals Inc. with 240 million dollars, the two companies said.

The agreement calls for Schein to acquire all stock outstanding of Marsam at about 21 dollars a share. In May, Marsam, which makes injectable drug products, disclosed it had received unsolicited takeover dollars in the range of /dollar 19 a share.

The information extraction system will identify the most important information in the text and could produce, for example:

```
Template: Takeover
   Company Predator: Schein Pharmaceutical Inc.
   Company Target:   Marsam Pharmaceuticals Inc.
   Type of takeover: FRIENDLY
   Value:            240 million dollars.
```

This summary is called a *template*, which is a structure with a predefined number of slots.

Many of the actual systems, for example some of those developed for the MUC-5 [ARP, 1993] and MUC-6 [ARP, 1995] competition, are based on statistical, probabilistical and pattern-matching techniques (shallow techniques). These systems are normally optimised for specific tasks and constraint domains and, therefore, their portability is limited. The LOLITA System is instead based on *deep natural language understanding* and has been designed as a *general purpose* natural language processing system.

2.1 Information extraction in finance

The goal of information extraction applied to finance is, as within other domains, to extract relevant information from a text producing an output usually consisting of a template of the original text. The information extracted from the source texts can be used by the financial operators to support their decision making process and to analyse the effect of news on price behaviour [Costantino *et al.*, 1996b].

One of the outstanding properties of the texts that make up financial articles, as opposed to general free-text, is the presence of a considerable high number of metaphors.

A randomly selected text from the *Financial Times* of October, the 21th 1992 showed 11 metaphorical expressions (metonymies and conceptual metaphors [Lakoff *et al.*, 1995]) in the first sentence, which was 37 words long.

Financial Times 21 Oct 92 London Stock Exchange: Equity futures and options trading

The *German Bundesbank's* decision *to move to* a variable repo rate, *leading to strong speculation* of further *cuts* in UK base rates, *enlivened a dull derivatives sector* and *sent* Footsie futures *moving sharply ahead*, writes Joel Kibazo.

Very few financial information extraction systems have been realised in the past. One of the few systems is ATRANS [Lytinen and Gershman, 1986], a system based on the scripts-frames approach for extracting information from telex messages regarding money transfers between banks. The system has been successful mainly because of the extremely limited domain and the limited information to be extracted from highly standardised and structured source documents. ATRANS is important because it was the first attempt to apply information extraction in the financial field using templates rather than summaries.

The systems that competed in the MUC-5 competition [ARP, 1993] were also able to perform the extraction of information from financial articles. However, these systems were only able to extract information regarding *Joint*

Ventures and, thus, work in an extremely restricted subset of the financial domain.

3 The LOLITA system

LOLITA (Large-scale Object-based Linguistic Interactor Translator and Analyser) has been designed as a *general purpose* natural language processing system and has been under development at the University of Durham for the last eight years [Garigliano *et al.*, 1993].

The approach taken for designing and implementing the system follows the lines of natural language engineering rather than those of traditional computational linguistics. The NLE approach emphasises the following aspects of engineering that should be considered when building a NL system [Garigliano, 1995].

Scale: the size of NLE systems must be sufficient for realistic large-scale application. Properties such as as the vocabulary size, grammar coverage and the number of word senses are critical.

Feasibility: this aspect concerns ensuring that constraints on the running of the system are acceptable. For example. hardware requirements should not be too great and execution speed must be adequate. Feasibility incorporates making the system and its components efficient.

Robustness: robustness is a critical aspect of large-scale systems. Robustness concerns not only the linguistic scope of the system but how it deals with input which falls outside of this scope. At the very least, it should be able to carry on and try its best to cope with the conditions it is working under.

Maintainability: maintainability is a measure of how useful the system is over a long period of time. There are four different aspects that can be referred to the term: *corrective maintenance of software repair*, *enhancement*, *perfective maintenance*, *preventive maintenance.*

Usability: the system must satisfy a need, i.e. there must be a set of users in the market who can benefit from using the system [Garigliano, 1995].

The LOLITA system is written in the functional programming language Haskell (currently about 45,000 lines of code, corresponding to about 450,000 lines of code in an imperative language) and based on a large, WordNet-compatible semantic network, *SemNet*, (over 100,000 nodes), similar to a conceptual graph [Sowa, 1984]. Its core, being the main part of the system around which individual applications are built, consists of 8 main modules (figure 1).

The semantic network consists of a hierarchy of nodes (concepts) connected with arcs. The nodes represent entities (*the company*), events (*The company made losses*) and relations (*A company IS A business*). The knowledge is stored in the network by using control variables. Control variables are the essential information stored at each node, there are about 50 different control variables. Knowledge is represented in the Semantic Network according to the connectivity between the nodes and arcs. Some of the control variables are:

- **Rank.** This control gives the nodes quantification, i.e. individual, (*the loss Company XY made in the first quarter of '94*), universal (*every loss*), generic (*losses*, or *some lofsses*), existential, bounded existential etc.

- **Type.** This control values are very similar to grammatical qualification with few exceptions and additions: entity, relation, typeless, event, fact, greeting etc. The *relation* type mainly represents verbs, *attribute* represents adjectives and *entity* represents nouns [Garigliano *et al.*, 1993].

- **Family.** This control groups nodes into the semantic "families", eg. living, animal, human, man-made, abstract, location, organisation, human-organisation etc. [Garigliano *et al.*, 1993].

Concepts are linked with arcs such as **specialisation_** (and its inverse, **generalisation_**), or **instance_** (inverse **universal_**). Specialisation links a set to a possible subset. For example, the concept of "*company*" is a specialisation of the concept of "*business*" which is a specialisation of the concept of "*enterprise*". The **specialisation_** (**generalisation**) link can be therefore used to specify hierarchies of concepts.

The **instance_** link allows to connect a concept to an instance of that concept. For example, the node corresponding to the organisation "*ALPHA*" in the sentence "**ALPHA bought BETA**" will be connected with a **universal_** link to the set of all organisations, of which ALPHA is an instance.

These mechanisms allow the network to contain an elaborate "knowledge base" (i.e. encyclopedic "world" knowledge, linguistic knowledge) which can be expanded via the natural language interface that is part of the system.

Input natural language text is processed by various hierarchic modules and the result stored in the semantic network. The main processing phases are: *morphology*, *parsing*, *semantics* and *pragmatics* (figure 1).

- the **morphology** module is responsible for splitting the input text into words and smaller units and producing for each word a list of possible meanings of that word combined with their syntactic (noun, verb etc.) and semantic categories. The input is then passed to the parser;

- the **parser** determines the syntactic information contained in the source text. It performs a full grammatical analysis of the input text, recognising the role of each word in the sentence (e.g. subject, verb, adjective, object etc.). At this stage, the meaning of each of the words in the sentence can be still ambiguous and will be resolved by subsequent modules;

- the **semantic analysis** module associates the words with the appropriate meaning(s) and maps them onto the system's internal representation;

- finally, the **pragmatic analysis** module performs the disambiguation of the meaning of the words and type checking. Lexical ambiguities (e.g. different meanings of the same word) and anaphora are resolved using a series of preference heuristics, taking into account the *topic* which has been set for the current text and the information in the *context*.

At this stage, the new knowledge can be stored in the semantic network and can be subsequently retrieved by the various applications.

To generate natural language output, the relevant part of the semantic network is fed to the generator component, which is capable of generating natural language output from the internal representation stored in the network [Smith *et al.*, 1994]. The output from the generator can be varied according to a large set of parameters.

Various kinds of applications have been realised around the LOLITA core including: machine translation from Italian to English, English to Spanish, Language Tutoring [Wang and Garigliano, 1992], query application and contents scanning [Garigliano *et al.*, 1993].

4 Information extraction in the LOLITA system

The way in which the information extraction application works is quite straightforward. The articles are firstly processed by the LOLITA system and stored in the semantic network. The task of the template application is then to search for the information needed in the semantic network. The relevant concepts for each of the template slots are identified in the semantic network using the slot fill rules. These concepts are then converted into English text using the generator or fragments of the source text.

The template application, thus, interacts with the LOLITA system core retrieving data from the semantic network by using the inference system and producing output in English either by using the generator or by referring to the original text.

A template is defined in LOLITA as an Haskell data-type comprising a predefined set of slots with associated fill rules that direct the search for appropriate information in the semantic network.

Three different kinds of templates are currently available in the LOLITA system, the *Event-based templates*, the *Summary templates* and the *Hyper templates*.

4.1 Event-based Templates

Event-based templates are structures where it is possible to identify a clear underlying top-level event to which all the information of the template's slots can be referred to [Garigliano *et al.*, 1993].

For example, the *takeover* of a company by another company can be considered a suitable event for building an event-based *takeover* template. In fact, all the information regarding the takeover: *amount*, *type of takeover*, *company target*, *company predator* etc. can be associated to the "parent" *takeover event.*

More than one event-based template can be identified in a source document, according to the number of relevant top-level events that are generated by the semantic analysis of an article.

Once the template's *parent event* has been identified each slot is filled by searching in the semantic network for the relevant information according to the slots' fill rules and starting from the node of the *parent event.*

4.2 Summary-templates

Summary templates represent structures for which it is not possible to identify a *parent event.* A summary template is basically a collection of objects stored in different slots which may not directly refer to the same concept or relate to each other. For example, a summary template can be composed of the following slots *personal names, organisations, locations, temporal, acronyms, monetary values, descriptions, animates, inanimates* (figure 4).

Summary templates are built differently from event-based templates. In fact, the slots' fill rules are applied to the list of all the nodes generated by the semantic analysis of the source document since a *parent-event* doesn't exist.

4.3 Hyper-Templates

Hyper-templates are structures whose slots can refer to other templates, thus creating a linked *graph* of templates. For example, in the *takeover template* (described later in this section) the slots *company predator* and *company target* could potentially be linked to an *organisation* template which would include detailed information regarding the company (figure 5). The hyper-template mechanism is potentially usable for linking different kinds of information, not necessarily extracted from the source text, such as company databases or historical share prices. Hyper-templates have been used for implementing the MUC-6 scenario dependent templates (a complete description of the implementation of the templates can be found in [Morgan *et al.*, 1995]).

4.4 Template slots and slot fill rules

There are currently five different type of slots available, which differ depending on the kind of slot fill rule used to identify the relevant information and the way in which the final English text is produced.

Concept Slot. The information in this slot is represented as a set of concepts which have the appropriate references to the template as a whole. For example, in the *takeover* template shown in figure 7 the *COMPANY_PREDATOR* slot contains a reference to the concept of "*Drakes Office Systems*" (the company which performed the takeover). As a result, the fill rule associated with a concept slot will be a function which identifies the relevant concepts, leaving the generation of the final English text to the core system.

String Slot. The fill rule for this slot produces the slot contents directly as a string. It is mainly useful in situation where the possible values can be enumerated and do not necessarily occur in the input text, for example the slot **Type of Takeover** of the takeover template in figure 6 which can assume the values: *FRIENDLY / HOSTILE*.

User-Defined Slot. This slot is similar to concept slots in that the fill rule produces a set of concepts. However, it differs because the fill rule itself is represented as a concept instead of a function. For example, the fill rule for the slot *S=VALUE-OF-TAKEOVER* of the user-defined takeover template in figure 8 will be represented as the concept of the question "*what is the cost of the T=TAKEOVER?*".

Text Reference Slot. The output is produced using fragments of the original text. The slot is used when the exact copy of the original text is needed. This is possible by using the *TextRef* mechanism which has been introduced in the LOLITA system for the participation in the MUC-6 competition [Morgan *et al.*, 1995]. The *textref* system allocates new nodes in the semantic network which correspond to the components of the document (words, phrases, sentences, ...). These nodes will act as a reference into the original document and can be used to produce the output English text.

Template Reference Slot. This slot is used to create a link to another template. The output of the slot will consist of a pointer to the new template. The *template reference slots* provide the basic mechanism for handling *hyper-templates* in LOLITA (see section 4.3) which are widely used in the MUC-6 scenario templates [Morgan *et al.*, 1995]

The slot fill rules are used for locating the relevant information (*entities* or *events*) for a particular slot in the semantic network. The rules can be built by checking the control variables associated with the nodes or by using the inference functions available in the core system in case the information is not directly stored in the semantic network but has to be inferred.

For example, the identification of a organization is performed by searching in the semantic network for all the new nodes that belong to the families *organisation* or *human organisation.* The semantic network is in fact updated with all the new nodes created by the semantic analysis of the source documents during the core analysis.

If the information to be located is an *event*, the searching process will involve the identification, through inference functions, of the subjects, objects or target events of the event itself.

5 Financial information extraction in LOLITA

Four main aspects have to be defined for designing a financial information extraction system: the kind of source articles (from on-line services or from newspapers or magazines), the information to be extracted, the output of the system (summaries or templates) and the interface to the user. Assuming that the output of the system consists of templates, the most important decision is the kind of information to be extracted.

Financial articles represent an extremely wide domain, including different kinds of news: financial, economical, political etc (cf. section 2.1). Therefore, the identification of a unique template able to summarise all the possible financial articles is extremely difficult, if not impossible. The best solution is thus to design more than one template.

Financial information extraction in the LOLITA system is based on the *financial activities approach* [Costantino *et al.*, 1996a]. A *financial activity* is here defined as one potentially able to influence the decisions of the players in the market (brokers, investors, analysts etc.) regarding securities issued by companies. A finite number of relevant *financial activities* are identifiable in the financial market and can be grouped into three different categories.

- **Company related** activities which are those related to the "life" of the company, changes in its status, in the ownership of the company, the number and ownership of its shares etc. This group includes the following financial activities: merger, takeover, flotation, new issue (of shares, bonds etc.), privatisation, market movement, bankruptcy, broker's recommendations, taking a stake, dividend announcement, overseas listing, profits/sales forecasts, profits/sales results, directors' dealings, legal action, investigation.
- **Company restructuring** activities. This activities are related to changes in the productive structure of the company and include: new product, joint venture, staff changes, new factory.
- **General macroeconomics** activities, which include general macroeconomics news that can affect the prices of the shares quoted in the stock exchange and comprises: interest rate movements, currency movements, general macroeconomics data (inflation, unemployment, trade deficit).

A specific template is associated to each of the *financial activities* and comprises a predefined set of slots. In figure 6 the definition of the takeover template is shown.

The implementation of the financial application within the LOLITA system is straightforward. The documents are firstly processed by the system. The analysis takes into account *prototypical* information and the *topic* which has been set for the financial application. *Prototypical information* are used by the pragmatic analysis module to restrict the kind of entities that can be used within a particular event according to the appropriate action and for the disambiguation of word meanings. One of the relevant *prototypes* identified for the financial templates is the *acquisition* prototype. The subject of the acquisition can be either a *person* or a *company*, while the object can be a *company*. The action can be *acquire*, *buy* or any other verbs with appropriate meanings.

The *topic* of an article can be defined as the *theme* or the *subject of discourse* of the article. For example, it is likely that the meaning of the verb *to buy* in a financial article will be:

```
buy: To take., To purchase, To buy. -> To acquire;
```

rather than:

```
buy: To believe. -> To accept
```

The *topic* for the financial application includes the appropriate meanings of words and concepts one would expect in a financial context, for example *buy*, *takeover*, etc. The information available in the current *context* together with those stored in the *topic* influences the choice of the word meanings. The meanings preferred are those which are semantically closer to the meanings in the context or the topic. The semantic closeness is computed depending on the distance between the nodes in the semantic network. The choice of the appropriate meaning depends also on other factors, such as the system's knowledge of the concept and its frequency of use.

Once the analysis of the source article has been completed and the new knowledge stored in the semantic network, the financial application will look for any event which match the main-events corresponding to the financial activities. For example, a *takeover* is identified by the application looking for any *takeover* event in the semantic network. The *takeover* event is defined as:

- an event that can be generalised to the concept of *takeover*, *acquisition* or *purchase*. For example "The acquisition of X by Y", where X is an organisation;
- or an event that has a *takeover-action* (e.g. *to buy*, *to take-over*, *to purchase*, etc.) or any other action that can be generalised to these, and the object is a company. For example: "X has acquired Y for 100 million dollars", where Y is an organisation.

Similarly, the conditions of all the other financial activities (events) are checked and the full list of the financial activities found is shown to the user. Subsequently, the slots are filled using the associated slot-rules. For example, the *company predator* of the *takeover* template will be identified as the subject of the takeover event, while the *company target* is the object.

Four kinds of slots: concept slots, string slots, text reference slots and template reference slots (cf. section 4) can be used to define the financial templates. For example, the *company target* slot of the *takeover* template is defined as *concept slot*, while the slot *type of takeover* is defined as *string slot*, since it can only assume values *friendly* or *hostile* which may not be explicitly stated in the source document and, therefore, must be inferred by the system. The *hyper-template* mechanism can be used in a large number of cases. For example, in the *takeover* template, the slots *company predator*, *company target*, *bank adviser predator* and *bank adviser target* can potentially be linked to other templates or other sources of information, such as company databases. In the same way, the *market movement* template could potentially be linked to a historical shares price database.

At present, the financial application is only partially implemented within the LOLITA system, only the first group of financial activities (company related activities) has been partially coded in the system.

One of the disadvantages of the way in which information extraction is currently performed within the LOLITA system is the fact that only deep natural language techniques are used. Therefore, the performance in terms of speed can be, in particular situations, penalised in comparison to systems based on pure pattern-matching or fragment-parsing techniques.

6 The NL user-definable template interface

The templates defined according to the *financial activities* approach should, in our view, represent the information which is relevant for the financial operator's investment decision-making process. However, the financial application allows the user to define additional templates using the *user-definable* template interface. This allows the maximum degree of flexibility for the user.

The LOLITA user-definable template interface has been designed for the end-user. This differs from the approach taken in the design of the MUC-5 SRA system [Krupka, 1995], a system based on pattern-matching techniques. This system, in fact, allowed the user to enter the template definitions by identifying all possible patterns for each slot-rule. However, the interface was designed for the developers, rather than for the end-user of the system. A considerable amount of patterns had to be enter for defining a template (132 patterns for the MUC-6 management scenario template). Moreover, the patterns had to be entered using a specific notation which requires specific linguistic knowledge which the end-user would probably not have.

A template such as the takeover template shown in figure 7 is defined in the LOLITA system by the following four key-elements:

1. the **template-name** which *uniquely* identifies the template among the others in the collection;

2. the **main-events** of the template, which represent the conditions under which the template has to be instantiated by the system;

3. the **slot-names** which *uniquely* identify each of the slots in the template;

4. the **slot-rules** which are used by the system to identify the relevant information for each of the slots. The slot rules will normally refer to the template as a whole or to information which has already been defined in other slots.

Similarly, the user-definable template interface will need to define these elements using the natural language definitions supplied by the user. The definition of the LOLITA user-definable template interface has been done by analysing the results of an experiment carried out by potential users of the system. The test required the potential users to describe a generic takeover template using sentences in natural language. More specifically, the users were asked to describe the *condition* under which the template should have been filled (the template *main-condition*) and the specific slot rules. The target of the experiment was to identify two key-points:

- how easy is for the user to define the templates using unconstrained input natural language text;
- how easy would it be for the system to understand such unrestricted input definitions.

The ultimate target was to identify the optimum compromise between the two. The analysis of the results suggests that allowing complete freedom to the user can lead to a difficult situation for both the user and the system:

- the user can find it difficult to express the template definitions using unrestricted natural language text without the support of any formal element;
- the unrestricted natural language input can be rather difficult to process for the system and a relevant number of ambiguities can be found in the template definitions. These ambiguities mainly concern with the resolution of *anaphora*. In other words, how to resolve the relations between objects and events in the template-condition and in the slots (coreference resolution).

Four main styles in the template definitions have been identified in the experiments carried out:

1. **The use of questions.** e.g. *What was the cost of the takeover.*
2. **The use of Noun-Phrases.** e.g. *The cost of the takeover.*
3. **The use of statements.** e.g. *A company acquires another company.*

4. **The use of variables.** e.g. *Company X acquires company Y.*

The use of *questions* in the definition has not been chosen for the user-definable interface. In fact, the potential users were unable to use the question for defining all the elements of the templates, but they were able to express the same definitions using noun-phrases or statements in place of questions. Allowing the user of questions would have therefore meant introducing additional ambiguities which could have been avoided. *Noun-phrases* and *statements* have been chosen for entering the template definitions, depending on the template element to be defined. The most important characteristic of the user-interface is that it allows the use of *variables* in the template definitions. This drastically reduces the amount of ambiguities in the definitions. The definition of the slot "*COMPANY_PREDATOR*" in the takeover template will therefore be:

```
Template Condition: V=COMPANY1 acquired V=COMPANY2
COMPANY_PREDATOR:   V=COMPANY1
```

The contents of the slot are clearly defined and no ambiguities arise from the definition. The same definition without the use of variables could have been, for example:

```
Template Condition: A company acquired another company.
COMPANY_PREDATOR:   name of the company that is purchasing
```

Differently from the previous definition, this second definition presents difficult points for both the system and the user. Firstly, the system would have to identify the specific company to which the user is referring. This is not necessary in the processing of the slot definition "*V=COMPANY1*", where the system can immediately identify the specific company, which corresponds to the variable. Secondly, the user may find it difficult to express the concept of "*company predator*" using a natural language sentence, while the definition of the slot using the variable "*V=COMPANY1*" is immediate.

Three different kinds of variables, **formal elements**, have been introduced in the user-definable template interface. The formal elements have been designed to reduce the amount of possible ambiguities in the templates definitions but does not reduce the user's expression power and are the:

- the **name of the template**, which must begin with the letters "*T=*". The name of the template can be used in the definitions of the slots to refer to an event which is represented by the template as a whole. For example, the slot *S=VALUE=TAKEOVER* in the takeover template is defined as "*the cost of the T=TAKEOVER*" (figure 8).

- the **variables** which can be defined by the user (beginning with the letters "*V=*") to identify the elements of the main-events which will be later used in the definition of the slot-rules. For example, in the definition of the takeover template in figure 8, the user can define the variable "*V=COMPANY1 is a company*" which is used to identify the company predator and, therefore, appears in both the *main-event* ("*V=COMPANY1 acquired V=COMPANY2*") and the slot-definitions (*"S=COMPANY-PREDATOR: V=COMPANY1"*) (figure 8).

- the **slot-names** which can be used in the definition of other slot rules to refer to the information contained in the previous slot. For example, the slot *S=ATTRIBUTION* of the takeover template (figure 8) refers to the other slots.

Noun-phrases or *statements* can be used in the definition of the templates depending on the template element, *questions* are not allowed in the definition.

The main-condition of the template can be entered using either a *noun-phrase* or a *sentence*. For example, the definition of the takeover template main-condition in figure 8 includes:

```
The acquisition of V=COMPANY1 by V=COMPANY2.
```

which is a noun-phrase and

```
V=COMPANY1 acquired V=COMPANY2.
```

which is a statement.

The slot definition must instead be entered only using *noun-phrases*. For example, the slot "*S=VALUE-OF-TAKEOVER*" of the takeover template in figure 8 includes:

```
The cost of the T=TAKEOVER.
```

which is a noun-phrase.

6.1 Filling the templates using the inference system.

The way in which templates are filled by the user-definable interface is rather different from how the predefined financial templates are handled.

First of all, no code describing the templates rules are available in the system. The definitions and rules for the predefined financial templates are coded directly within the system, as part of the source code. User-defined templates, instead, are filled by the system using the *inference system* which matches the templates definitions against the knowledge contained in the semantic network and, in particular, the new knowledge acquired with the analysis of a source article. Therefore, the *inference engine* identifies entities and events which satisfy the template rules stored in the semantic network corresponding to the *variables*, the *main-conditions* and the *slot-rules* definitions.

The first step taken by the user-definable interface is to process the template definitions supplied by the user which are marked as questions. The *template-name*, the *variables* (figure 9), the *main-conditions* (figure 10) and the *slot rules definitions* are processed and stored in the semantic network.

The inference system will then try to match these questions against the new information acquired from the processing of a source article.

For example, for the main-event shown in figure 10:

```
V=COMPANY1 acquired V=COMPANY2
```

the inference engine will recognize that an event such as:

```
Fiat purchased Renault
```

is a relevant one, because of the fact that the action is compatible with "acquire" and the subject and object can be matched against the variables V=COMPANY1 and V=COMPANY2. In figure 11 the representation of the main-event and the candidate event "Fiat bought Renault" is shown. The inference system tries to match each of the components of the candidate event onto the main-event.

The inference engine will therefore look for an event which satisfies the following condition:

$$\exists\ V{=}COMPANY1,\ V{=}COMPANY2.\ Acquire(V{=}COMPANY1, V{=}COMPANY2)$$

Once the candidate events have been identified, these can be used by the inference engine for searching for concepts which match the slot rules.

Inference and the variables

The variables are filled in by the inference system as part of the processing of the main-events. Therefore, specific calls to the inference system for locating information which corresponds to the variables is not necessary.

Inference and the slots

The slot-rules definitions entered by the user can be subdivided into two different categories:

- rules which refer only to a specific variable used in the main-event, for example:

  ```
  S=VALUE-TAKEOVER:       V=VALUE
  ```

  This kind of slots is filled with the concepts which have already been identified for the specific variable.

- rules which refer to specific variables, the template-name or other slot-names but adding additional conditions, for example:

  ```
  S=VALUE-TAKEOVER:       the cost of the T=TAKEOVER
  ```

  In this case, the inference engine will be called again and will look for any event or entity which matches the slot-rules.

The template user-definable interface is currently under development. At present, the template definitions are correctly processed and stored in the semantic network, while the inference engine is connected to the application and simple templates can be filled from the templates definitions.

7 Conclusions

In this paper we have tried to give an overview of how financial information extraction is performed in the LOLITA system. Unlike many other information extraction systems, the emphasis is on full natural language processing of the text. By using the financial information extraction system the user can extract the most important information from the source articles and use them as a support for his investment decision-making process. The pre-defined templates based on the *financial activities* should represent the information which is relevant for the investors. However, the application allows the maximum degree of flexibility and the user can easily add new templates using the user-definable template interface.

References

[ARP, 1993] ARPA, *Proceedings of the Fifth Message Understanding Conference*, Morgan Kaufmann Publishers, August 1993.

[ARP, 1995] ARPA, *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, October 1995.

[Costantino *et al.*, 1996a] M. Costantino, R. J. Collingham, and R. G. Morgan, "Natural Language Processing in Finance", *The Magazine of Artificial Intelligence in Finance*, 2 No.4, 1996.

[Costantino *et al.*, 1996b] M. Costantino, R. J. Collingham, and R. G. Morgan, "Qualitative Information in Finance: Natural Language Processing and Information Extraction", *NeuroVe$t Journal*, November, 1996.

[Garigliano *et al.*, 1993] R. Garigliano, R. G. Morgan, and M. H. Smith, "The LOLITA System as a Contents Scanning Tool", in *Avignon '93*, 1993.

[Garigliano, 1995] R. Garigliano, "Editorial", *Natural Language Engineering*, 1, March 1995.

[Krupka, 1995] G. R. Krupka, "SRA: Description of the SRA System as Used for MUC-6", in *Proceedings of the Sixth Message Understanding Conference*, Morgan Kaufmann Publishers, October 1995.

[Lakoff *et al.*, 1995] G. Lakoff, J. Espenson, and A. Schwartz, *Master metaphor list*, Berkely, California, 1995, http://cogsci.berkely.edu/.

[Lytinen and Gershman, 1986] S. L. Lytinen and A. Gershman, "ATRANS: Automatic Processing of Money Transfer Messages", in M. Kaufmann, editor, *9th International Joint Concerence on Artificial Intelligence*, pages 821–825, 1986.

[Morgan *et al.*, 1995] R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, R. Collingham, M. Costantino, C. Cooper, and The LOLITA Group, "University of Durham: Description of the LOLITA System as used in MUC-6", in *Sixth Messages Understanding Conference (MUC-6)*, Morgan Kaufmann, October 1995.

[Riloff and Lehnert, 1994] E. Riloff and W. Lehnert, "Information Extraction as a Basis for High-Precision Text Classification", *ACM Transactions on Information Systems*, 12 No.3:296–333, 1994.

[Smith *et al.*, 1994] M. H. Smith, R. Garigliano, and R. G. Morgan, "Generation in the LOLITA system: An Engineering Approach", in *7th International NL Generation Workshop*, June 1994.

[Sowa, 1984] J. F. Sowa, *Conceptual Structures, information processing in mind and machine*, Addison-Wesley, 1984.

[Wang and Garigliano, 1992] Y. Wang and R. Garigliano, "Detection and Correction of Transfer by CAL", in *Second International Conference on Intelligent Tutoring Systems (ITS-92)*, Sprinter-Verlag, June 1992.

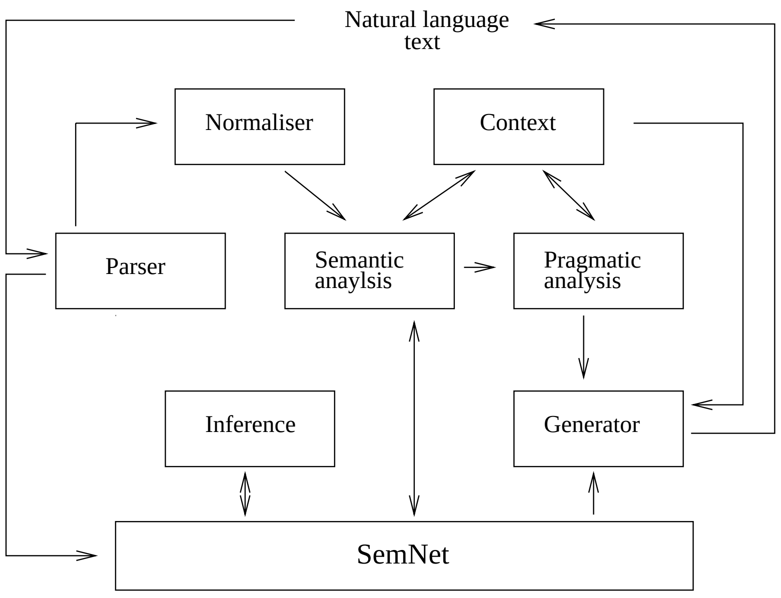

Figure 1: The LOLITA system core.

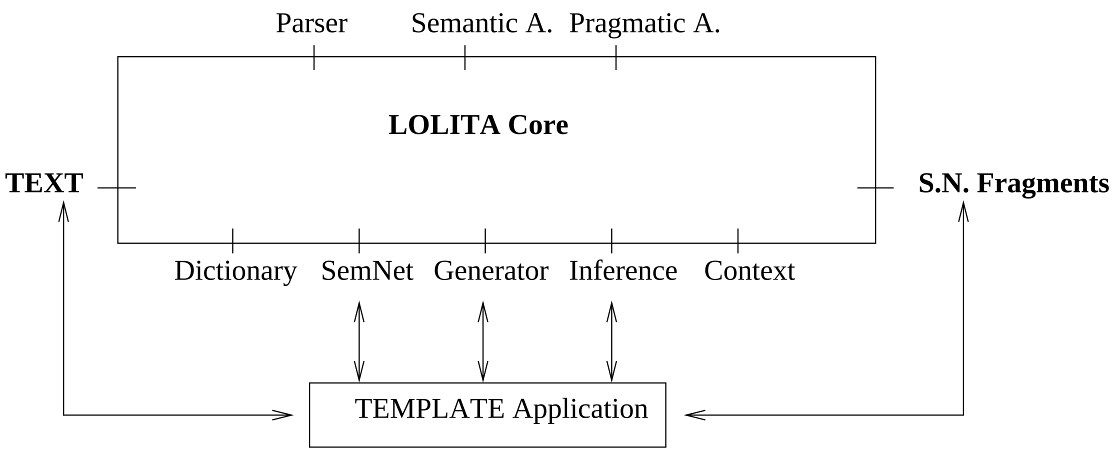

Figure 2: Information extraction using LOLITA.

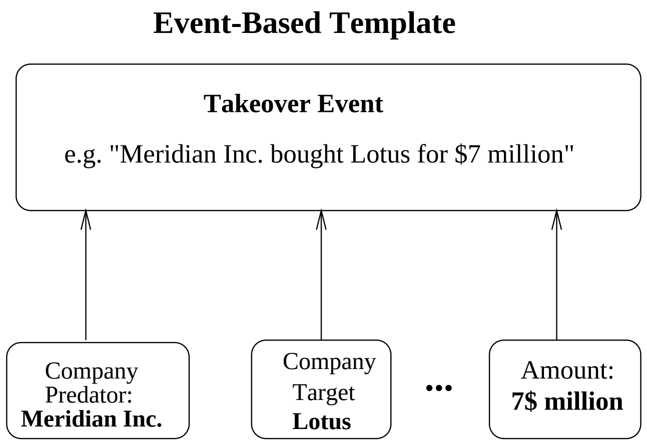
Event-Based Template
Takeover Event
e.g. "Meridian Inc. bought Lotus for $7 million"
Company Predator: Meridian Inc.
Company Target Lotus
...
Amount: 7$ million

Figure 3: Event-based templates.

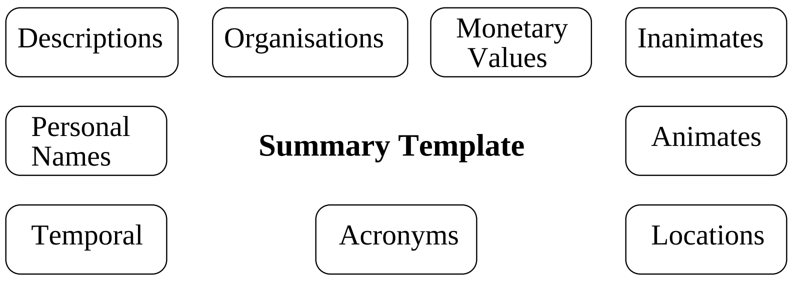

Figure 4: Summary Templates.

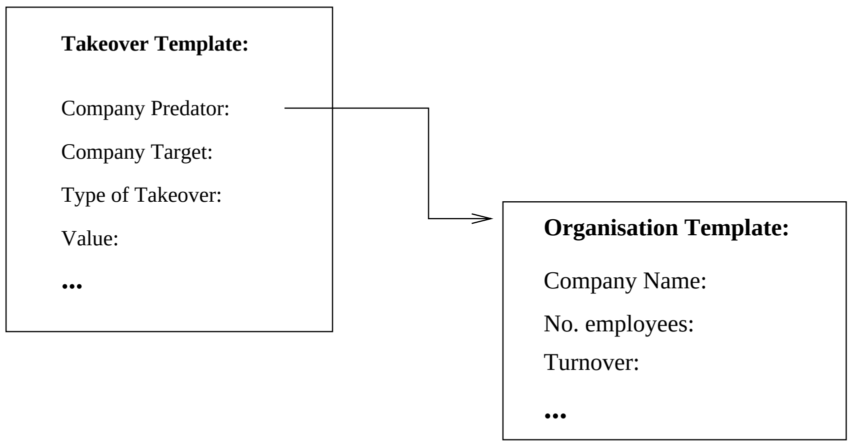

Figure 5: Hyper Templates.

```
Company Target:           COMPANY_NAME
Company Predator:         COMPANY_NAME
Type of takeover:         {FRIENDLY, HOSTILE}
Value:                    [NUMBER] & CURRENCY_UNIT
Bank adviser predator:    BANK_ADVISER
Bank adviser target:      BANK_ADVISER
Expiry date:              DATE
Attribution:              ATTRIBUTION:
Current Stake predator:   {PERCENTAGE, SHARES, AMOUNT} [NUMBER]
Denial:                   DENIAL

COMPANY_NAME, CURRENCY_UNIT, BANK_ADVISER, ATTRIBUTION, DENIAL: String
DATE: [[0-31][0-12][00-99]]
PERCENTAGE, AMOUNT: Value
```

Figure 6: The Takeover template

Source article from the Financial Times:
Filofax Group announced yesterday it will acquire Drakes Office Systems with 3 million dollars from its founder, Mr. Tom Drake. Initial consideration comprises a mixture of cash and the issue of 1m ordinary shares. Drake claims to be the UK market leader in Wire-O bound carbonless duplicate message books. Its Ring Back brand forms a range of telephone message and similar business forms with a dominant market share. In 1992, Drakes made gross profits of Pounds 727,000 on sales of Pounds 1.7m.

Template produced:

```
Template: TAKEOVER
    COMPANY_TARGET:   Drakes Office Systems.
    COMPANY_PREDATOR: Filofax Group.
    TYPE_TAKEOVER:    FRIENDLY
    VALUE:            3 million dollars.
    ATTRIBUTION:      Filofax Group.
```

Figure 7: A example of a LOLITA-produce takeover template

```
Template-name:          T=TAKEOVER
Variables:              V=COMPANY1 is a company.
                        V=COMPANY2 is a company.
                        V=VALUE is money.

Template main-event:    V=COMPANY1 acquired V=COMPANY2.
                        V=COMPANY1 acquired V=COMPANY2 with V=VALUE.
                        The acquisition of V=COMPANY2 by V=COMPANY1.
                        The V=VALUE acquisition of V=COMPANY2 by V=COMPANY1.
                        V=COMPANY1 paid V=VALUE for V=COMPANY2.
                        V=COMPANY1 acquired a majority stake in V=COMPANY2.
                        V=COMPANY1 took full control of V=COMPANY2.

Definition of slots:

S=COMPANY-PREDATOR:     V=COMPANY1

S=COMPANY-TARGET:       V=COMPANY2

S=TYPE-OF-TAKEOVER:
  String-fill: HOSTILE    T=TAKEOVER is hostile.
  String-fill: FRIENDLY   T=TAKEOVER is not hostile.

S=VALUE-OF-TAKEOVER:    The cost of T=TAKEOVER.
       V=VALUE

S=BANK-ADVISER-PRED:    The adviser of V=COMPANY1.

S=BANK-ADVISER-TARG:    The adviser of V=COMPANY2.

S=EXPIRY-DATE:          The date of expiry of T=TAKEOVER.

S=ATTRIBUTION:          The person or the company that announced T=TAKEOVER

                        The person or the company who said something about
                        T=TAKEOVER or said something about S=COMPANY-PREDATOR
                        or said something about S=COMPANY-TARGET or said
                        something about S=TYPE-OF-TAKEOVER or said something
                        about S=VALUE-OF-TAKEOVER or said something about
                        S=BANK-ADVISER-PRED or said something about
                        S=BANK-ADVISER-TARG or said something about EXPIRY-DATE

S=CURRENT-STAKE-PRED:   The stake that V=COMPANY1 owns of V=COMPANY2

S=DENIAL:               The person or company who denied T=TAKEOVER
                        or denied COMPANY-PREDATOR or denied the
                        COMPANY-TARGET or denied TYPE-OF-TAKEOVER or
                        denied S=BANK_ADVISER-PRED or denied
                        S=BANK-ADVISER-TARG or denied S=VALUE-TAKEOVER
                        or denied EXPIRY-DATE.
```

Figure 8: The takeover template as defined for the template user-interface.

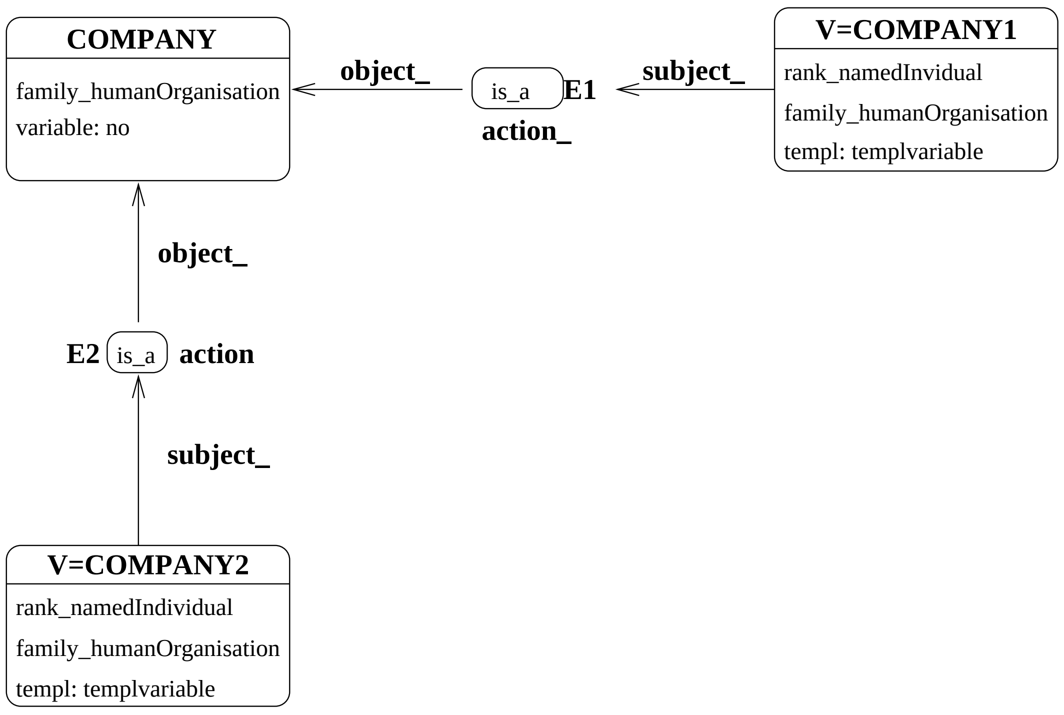

Figure 9: The processing of a the variable "V=COMPANY1 is a company."

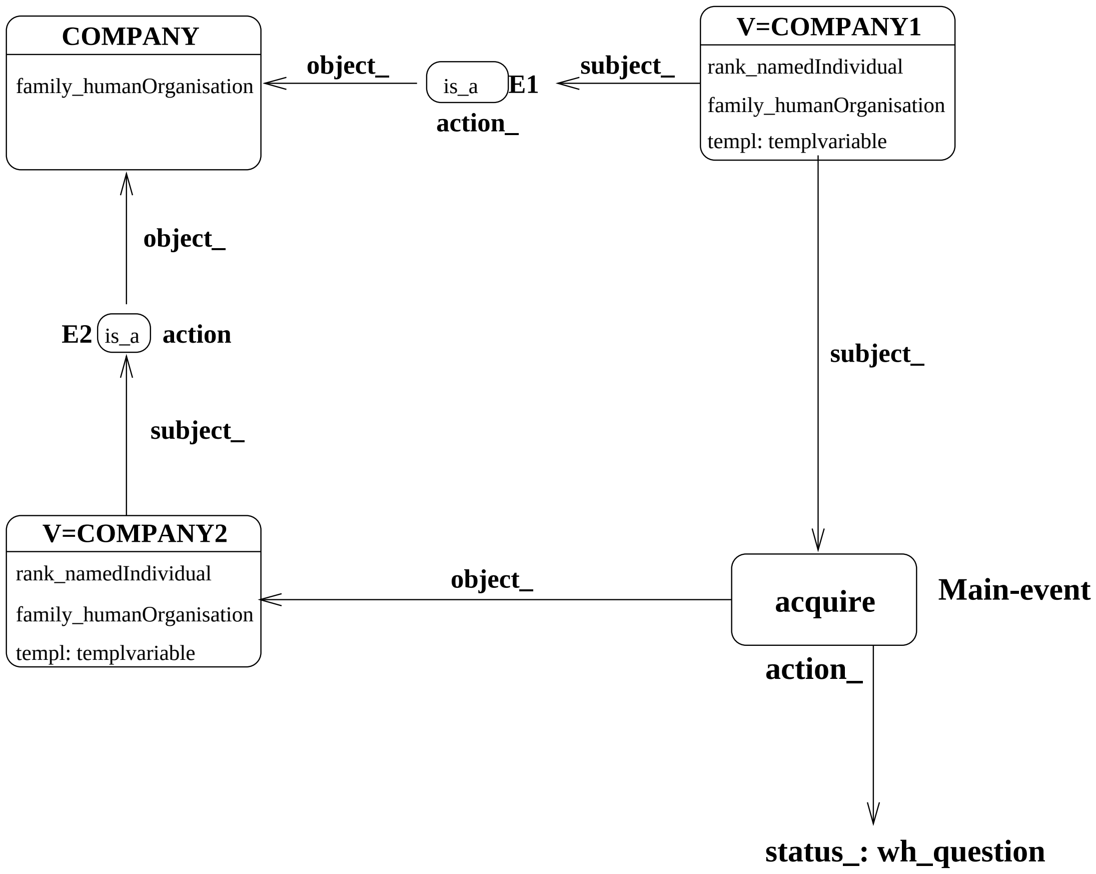

Figure 10: The processing of the main-event "V=COMPANY1 acquired V=COMPANY2.

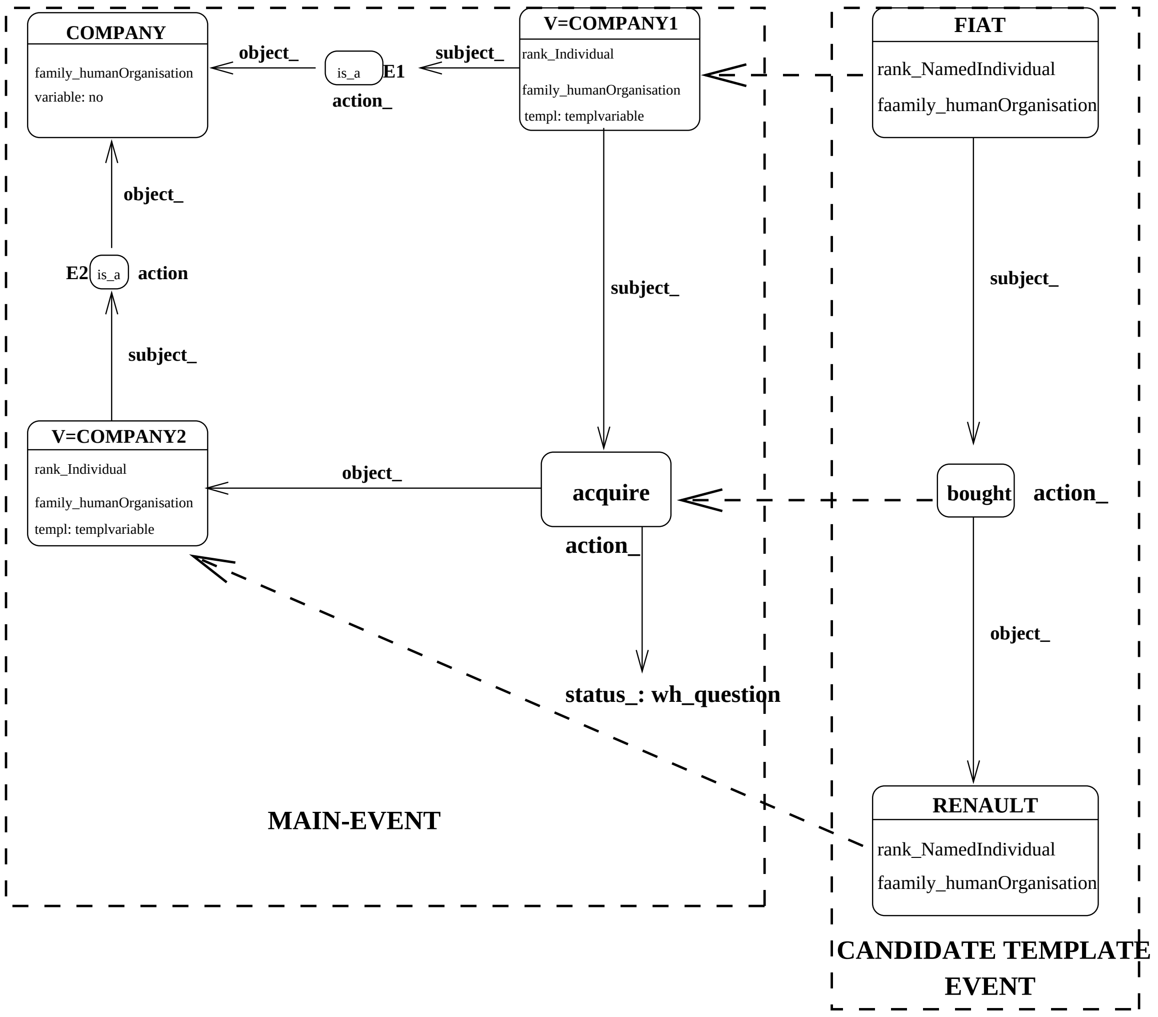

Figure 11: Identification of candidate main-events by the inference system.