Journal of Theoretical and Applied Information Technology
20th October 2013. Vol. 56 No.2
© 2005 - 2013 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INTEGRATION OF A CONTROLLED NATURAL LANGUAGE IN AN INTELLIGENT SYSTEMS PLATFORM

[1]MOHAMMED NASRI, [2]ADIL KABBAJ, [3]KARIM BOUZOUBAA
[1, 3] Mohammed Vth University-Agdal, Mohammadia School of Engineers, Department of Computer Science, Rabat, Morocco
[2] Mohammed Vth University-Agdal, INSEA, Department of Computer Science, Rabat, Morocco
E-mail: [1]mohammed.nasri@gmail.com, [2]akabbaj@insea.ac.ma, [3]karim.bouzoubaa@emi.ac.ma

ABSTRACT

Amine Platform is an Integrated Development Environment (IDE) that allows developing many kinds of intelligent systems and agents. The knowledge in Amine is represented using the Conceptual Graph (CG) formalism, which is a powerful formalism for expressing knowledge and writing specifications. The drawback is that CGs are difficult and not obvious for human reading, especially for users that are not familiar with this formalism or for complex situations. Integrating a controlled natural language makes the platform more comfortable (for users), more powerful and helps so to eliminate this weakness. In this paper we present our work on the integration of the controlled natural language ACE in the Amine Platform and we outline the contribution of this integration in the improvement of the interaction between the user and the platform.

Keywords: *Knowledge Representation, Conceptual Graph, Conceptual Structures, Controlled Natural Language, ACE, Natural Language Processing, Amine Platform.*

1. INTRODUCTION

Amine Platform [1-2] is a multi-layer integrated development environment (IDE) suited for symbolic programming, intelligent system programming and intelligent agents programming. It allows implementing many kinds of ontology-based intelligent systems such as Knowledge Based Systems, Ontology based applications, natural language processing applications, problem solving applications, planning applications, reasoning based applications, learning based applications and multi-agents applications.

Amine, adopts the Conceptual Graph (CG) [3] as its main knowledge representation formalism, Amine uses CG because it is a powerful formalism for expressing knowledge and writing specifications, it allows expressing meaning in a form that is logically precise, humanly readable and computationally tractable. CG formalism has been adopted in many domains such as natural language processing, knowledge based systems, information systems, etc.

Many platforms/tools adopt CG as their main knowledge formalism (Cogitant, CharGer, CPE, etc.) but Amine remains the most promising one. In fact, as described by Kabbaj [2], Amine provides, in addition to the creation, manipulation and execution of CGs, i) a rich ontology layer (an API facilitating the development, the manipulation of ontologies / knowledge bases and various ontology-based processes), ii) two CG based programming languages (Prolog+CG and Synergy) and iii) allows the construction and execution of intelligent agents and multi-agents systems.

The use of CG presents, however, some difficulties and limitations since it remains a specific formalism that requires a degree of expertise, especially for complex situations. This makes the task difficult for users to construct ontologies for information systems (especially when formulating semantic information such as definitions, situations or rules), to formulate queries in question answering systems and each time the user has to express knowledge to be processed in an intelligent system. This makes Amine a bit complicated and restricts its use to a small population (CG experts only).

It would be interesting for humans to avoid having to express knowledge using such specific formalism.

With the aim of eliminating this weakness and making the platform more powerful and accessible for a wider population, we have integrated the controlled natural language “ACE”

[4-5] in Amine. ACE is a controlled natural language that covers a well-defined subset of English that can be translated unambiguously into first-order logic via discourse representation structures [6] and then be used for many intelligent systems as high level interface for expressing knowledge and writing specifications. With the integration of ACE, Amine provides a high level interface for a wider population, allowing exploiting Amine functionalities without being limited by CG formalism.

In this paper we describe our work on the integration of a linguistic component (that includes ACE) into the Amine Platform. The paper is organized as follows: Section 2 briefly presents Amine platform and its knowledge representation formalism CG, section 3 introduces the ACE controlled language, sections 4 and 5 respectively describe our work about the integration of the ACE in Amine Platform and outline some applications. This paper ends with a conclusion and outlines some future works.

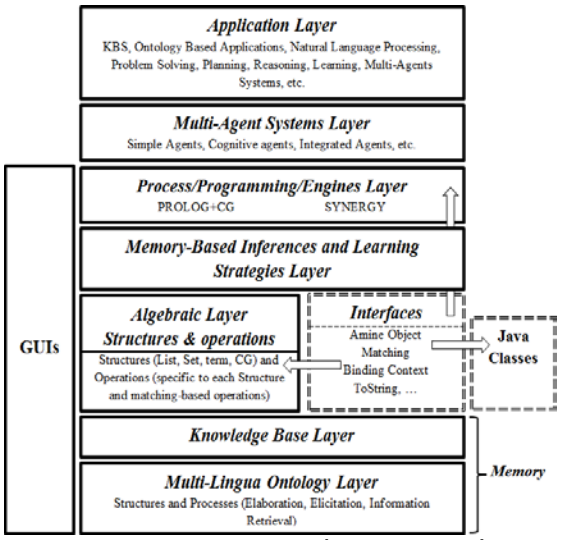
2. AMINE PLATFORM

2.1. An overview of Amine platform

Amine is a Java Open Source Platform suited for the development of different types of intelligent systems/agents[1]. The main aim/goal of Amine is to provide an integrated architecture to build Intelligent Systems and Intelligent Agents of interest not only for Artificial Intelligence and Cognitive Science, but also for Semantic Web.

As shown is the figure below, Amine is composed of seven layers:

- Multi-Lingua Ontology layer: Tackles the creation, edition and manipulation of multi-lingua ontology which provides the “conceptual vocabulary” and the semantic of a domain.
- Knowledge Base layer: Covers the creation, edition and manipulation of Knowledge Bases (KB).
- Algebraic layer: this layer provides several types of structures and operations useful for the other layers. Part of this layer is CG structure and CG operations, which is an API that can be used by any Java application.
- Memory-Based Inference and Learning Strategies Layer: Ontology and KB constitute the memory of an intelligent system/agent. Amine provides some basic inference and learning strategies that are defined as memory-based processes.


Application Layer
KBS, Ontology Based Applications, Natural Language Processing, Problem Solving, Planning, Reasoning, Learning, Multi-Agents Systems, etc.
Multi-Agent Systems Layer
Simple Agents, Cognitive agents, Integrated Agents, etc.
Process/Programming/Engines Layer
PROLOG+CG
SYNERGY
Memory-Based Inferences and Learning Strategies Layer
Algebraic Layer
Structures & operations
Structures (List, Set, term, CG) and Operations (specific to each Structure and matching-based operations)
Interfaces
Amine Object Matching Binding Context ToString, ...
Java Classes
Knowledge Base Layer
Multi-Lingua Ontology Layer
Structures and Processes (Elaboration, Elicitation, Information Retrieval)
Memory
GUIs
Figure 1: The architecture of the Amine Platform

- Programming layer: Three complementary programming paradigms are provided by Amine: i) pattern-matching and rule-based programming paradigm, ii) activation and propagation-based programming paradigm and iii) ontology or memory-based programming paradigm.
- Agents and Multi-Agent Systems layer: Amine can be used in conjunction with a Java Agent Development Environment to develop multi-agent systems.
- Application layer: various kinds of applications can be developed, using Amine (or a subset of Amine): Knowledge Based Systems, Ontology based applications, natural language processing applications, problem solving applications, planning applications, reasoning based applications, learning based applications, multi-agent applications, etc.

The most important thing in Amine is that it provides APIs that can be used directly in any Java application, thanks to the high level of modularity and independence of its layers. In addition to this, Amine provides several friendly graphical user interfaces with the aim of facilitating its functionalities.

Since its launch 10 years ago, Amine is today considered among the best platforms for manipulating CGs and developing intelligent systems[2]. In addition, it has been adopted by dozens

[1] Please refer to Amine website (http://amine-platform.sourceforge.net/) for further information

[2] http://www.jfsowa.com/cg/index.htm

of researchers in many Artificial Intelligence fields such as decision support in intensive medicine [7], decision support in spatio-temporal environment [8], Arabic Question Answering system [9], 3D interaction assistance in virtual reality [10], in Geosimulation [11], in Semantic Enterprise Architectures [12] etc.

2.2. Conceptual Graph (CG) formalism

In his first book [13], John Sowa proposed Conceptual Graph (CG) Theory as a foundation for "high cognition". CG theory is a synthesis from several fields: semantic networks and related topics in Artificial Intelligence, logic (Sowa showed the equivalence of CG with first-order logic and used Existential Graph Logic of S. C. Peirce as a logical foundation for CG theory), linguistic (like various semantic networks versions, CG can be used to represent meaning and background knowledge), relational database semantic, cognitive psychology and philosophy (John showed how CG theory can represent various conceptual structures that were proposed and developed in cognitive psychology and philosophy).

CG formalism has been used in many areas (data base semantics, natural language processing, knowledge based systems, information systems, multi-agent systems, etc.) by many groups and researchers from different countries around the world. Annual International Conferences on Conceptual Structures (ICCS) are organized since 1993[3].

CG is a powerful formalism that expresses meaning in a form that is logically precise, humanly readable, and computationally treatable.

Figure 2 shows two examples of CG (CG1 for “A patient takes the food carefully” and CG2 for “If a person coughs then the person is sick”).

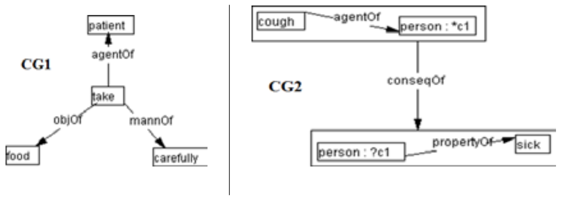


Figure 2: Examples of conceptual graphs.

CG1 can be read as: The “agent of” the action “take” is the “patient”, the “object” of the action is “the food” and the way this action is accomplished is “carefully”. CG2 can be read as: The subCG2 (the “person” is “sick”) is the consequence of the subCG1 (the “person” “coughs”).

[3] http://conceptualstructures.org/confs.htm

2.3. Knowledge representation in Amine platform

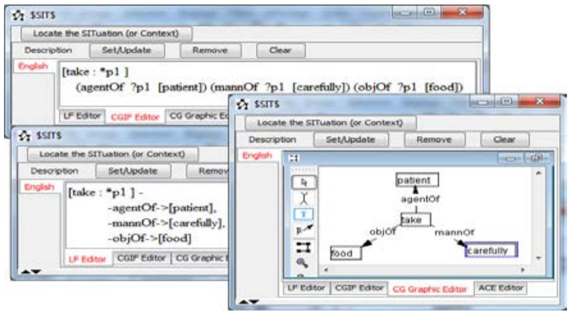
Amine uses the CG formalism to represent both the knowledge inserted into the ontology of an Intelligent System and the knowledge inserted into a Knowledge Base.

The ontology in Amine is basically made of a type hierarchy (concept types, relations and individuals), the meaning of these types is expressed in terms of various kinds of “conceptual structures” like Definition (definition of a concept type and of a relation type), Canon (canon for a concept type and for a relation type), Schematic Cluster (a set of schemas for a concept type) and Individual (description of an individual). Amine uses CGs to describe the content of these Conceptual Structures.

CG formalism is also used in the construction of the Knowledge Base (KB) (which has an ontology as a support and corresponds to a generalization graph composed mainly of situations and/or rules).

Amine provides several CG editors (Linear form editor, CGIF editor and Graphic Editor) and several CG operations. The semantical support of CG is the ontology of the domain, composed of a type hierarchy and conceptual structures.

Figure 3 shows the CG1 of the figure 2 printed in linear form, graphic and CGIF editors.



```
[take : *p1 ]
   (agentOf ?p1  [patient]) (mannOf ?p1  [carefully]) (objOf ?p1  [food])
```

```
[take : *p1 ] -
         -agentOf->[patient],
         -mannOf->[carefully],
         -objOf->[food]
```

Figure 3: Example of a conceptual graph printed in different editors using the Amine Platform.

3. ATTEMPTO CONTROLLED ENGLISH

3.1. The usefulness of a controlled natural language in an intelligent systems platform

Even if CG is a powerful knowledge representation formalism, it remains difficult and not obvious, especially for users that are not familiar with this formalism or for complex situations.

This is why, it would be interesting to avoid having to express knowledge using such specific formalism that requires a degree of expertise; it is more convenient for humans to use their natural language.

Given the numerous complexities and ambiguities of natural language, many researchers have chosen to focus on one subset of natural language[4], conventionally called "controlled language".

The purpose behind the development and implementation of controlled natural languages is typically to aid non-native speakers of a natural language to learn it, speak it and understanding it, or to ease computer processing of a natural language[5]. Thus, a controlled natural language can be used as high-level interface language to various kinds of knowledge systems [14], in particular Amine platform.

3.2. Attempto Controlled English

Attempto Controlled English (ACE) [4-5] is a well-defined subset of English with a domain specific vocabulary and a restricted grammar in the form of a small set of construction and interpretation principles. ACE allows representing specifications in simple or composite sentences.

A set of optional elements called "modifiers" can be added to a sentence in order to give additional information; ACE accepts noun modifiers (adjectives, relative sentences, propositional phrases, possessive nouns and appositions) and verb modifiers (adverbs and prepositional phrases).

In ACE, users can build composite sentences from simple sentences, or composite phrases from simpler phrases, with the help of four different types of constructors: coordinators, subordinators, quantifiers and negators.

ACE supports questions and command sentences too and is able to process with modality sentences, like: admissibility, possibility, recommendation, provability and necessity.

Here are some examples of ACE sentences:

- Is John sick?
- John does not take the medicine.
- Every person who coughs is sick.
- No customer is sick.
- The patient who takes the medicine heals.
- If the patient is sick then he should take the drug.

ACE can be translated unambiguously into first-order logic via Discourse Representation Structures (DRS) [5, 6] and then be used for automated reasoning.

ACE is supported by various tools, among them a text editor that helps users construct correct ACE sentences with the help of hints and error messages, a parser that translates ACE texts into DRS, a paraphrase that reflects the interpretation of the machine in Controlled Natural Language, and a Satchmo-style reasoning engine that can be used for consistency and redundancy checking as well as for question answering.

4. INTEGRATION OF THE ATTEMPTO CONTROLLED ENGLISH INTO THE AMINE PLATFORM

4.1. ACE as an alternative to CG in Amine Platform

Attempto group provides an ACE Parser Engine (APE) that parses and translates an ACE text unambiguously into a Discourse Representation Structure (DRS) [5-15].

DRS consists of a set of semantic elements: “object”, object “properties”, possession “relations”, “predicate”, “modifier_adv” / “modifier_pp” to give more information to the predicate, “has_part” to specify that an object belongs to a group and “query”.

APE provides many outputs formats[6], the most important (for us) is the “xml” one. Here are two examples of the ACE/DRS translation (please refer to the DRS documentation [15] for more information about the fields meaning):

DRS for: A person is sick.

```
<DRS domain="A B C">
  <object
      ref="A"
      noun="person"
      struct="countable"
      unit="na"
      numrel="eq"
      num="1"
      sentid="1"
      tokid="2"/>
  <property ref="B" adj="sick" degree="pos"
      sentid="1" tokid="4"/>
  <predicate ref="C" verb="be" subj="A"
      obj="B" sentid="1" tokid="3"/>
</DRS>
```

DRS for: The patient takes the food carefully.

```
<DRS domain="A B C">
  <object
      ref="C"
      noun="patient"
```

[4] http://sites.google.com/site/controllednaturallanguage/
[5] http://en.wikipedia.org/wiki/Natural_language

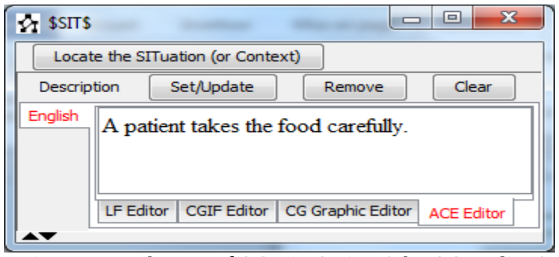
[6] Please refer to the attempto website for more information. http://attempto.ifi.uzh.ch/site/

```
        struct="countable"
        unit="na"
        numrel="eq" num="1"
        sentid="1" tokid="2"/>
    <object
        ref="B"
        noun="food"
        struct="countable"
        unit="na"
        numrel="eq"
        num="1"
        sentid="1"
        tokid="5"/>
    <predicate ref="A" verb="take" subj="C"
        obj="B" sentid="1" tokid="3"/>
    <modifier_adv
        ref="A"
        adverb="carefully"
        degree="pos"
        sentid="1"  tokid="6"/>
</DRS>
```

APE is based on ACE lexicon that defines all ACE words and their syntactic forms (verb, noun, adjective, etc.), so to successfully analyze a sentence by APE, words (of this sentence) need to be defined in this lexicon, otherwise, the translation fails, APE returns an empty DRS and a list of error messages telling the user that the text or the sentence is incorrect.

Henceforth, in order to integrate ACE in Amine, the DRS2CG module has been developed and integrated in Amine [16]. That module uses the DRS structure, extracts all the semantic elements and generates the corresponding CG.

On the other hand, with the addition of the APE engine, Amine provides an ACE2CG module allowing users to directly translate ACE to CG without handling DRS structures. It is possible also for users to alternatively use CG or ACE to represent and specify knowledge in Amine. They can formulate a proposition or a statement in ACE and Amine produces the equivalent CG. The following two figures (4.a and 4.b) show the use of ACE as alternative to CG and the generation of CG by Amine.



A patient takes the food carefully.

Figure 4.a: The use of ACE in Amine (The ACE editor)

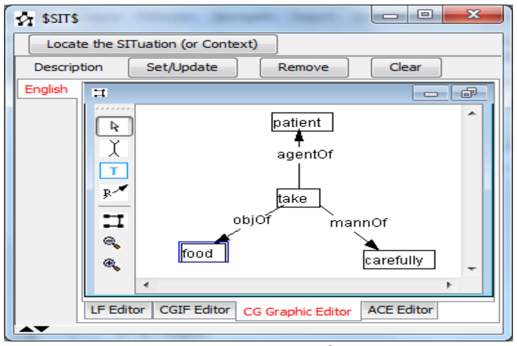
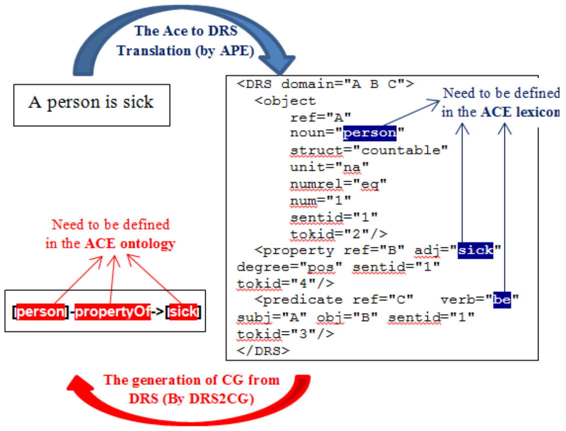


Figure 4.b: The generated CG of the previous ACE sentence (The CG graphic editor)

4.2. The ACE ontology

In the same way that APE needs the ACE lexicon (that contains ACE words), the DRS2CG module needs an ontology that contains types (of concepts) required for the building of the corresponding CG. Hence, the creation of an ACE related ontology (so called ACE ontology) is a necessity. This ontology should constitute the ontological support for the DRS2CG module. The following figure illustrates this principle:


The Ace to DRS Translation (by APE)
A person is sick
Need to be defined in the ACE lexicon
Need to be defined in the ACE ontology
[person]-propertyOf->[sick]
The generation of CG from DRS (By DRS2CG)

```
<DRS domain="A B C">
  <object
      ref="A"
      noun="person"
      struct="countable"
      unit="na"
      numrel="eq"
      num="1"
      sentid="1"
      tokid="2"/>
  <property ref="B" adj="sick"
degree="pos" sentid="1"
tokid="4"/>
  <predicate ref="C"   verb="be"
subj="A" obj="B" sentid="1"
tokid="3"/>
</DRS>
```

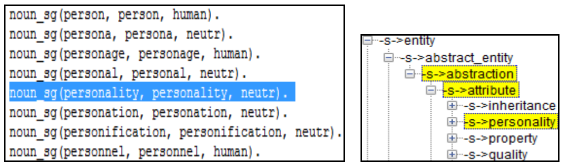
Figure 5: The usefulness of the ACE lexicon and the ACE ontology.

While translating ACE to DRS, APE needs to find words in the ACE lexicon. The same principle applies during generating the CG; the DRS2CG needs to find the concepts in the ACE ontology. Otherwise even if the CG is correct, it cannot be handled within Amine.

To create this ontology, the use of ACE lexicon alone was not enough, since it doesn't provide semantic information about entries (lack of supertype/subtype relationship); this is why we have used WordNet (WN) [17] as a complementary resource thanks to its structure which is similar to an ontology, since its entries are connected through semantic relations particularly the hypernymy/ hyponymy relations. Hence a combination of the two complementary aspects of WN and ACE lexicon presents an opportunity for us to construct the needed ontology.

We adopted, for this purpose, a two-step approach: For each ACE term, in the first step we get its syntactic form from the ACE lexicon (verb, noun, state, property, preposition, etc.) and in the second one, we look for the semantic information about super types via WN. This is how we managed the development of this ontology.

Figure 6 shows a part of both ACE lexicon and the ACE ontology (showing as an example the location of the word “personality” in the two resources).



```
noun_sg(person, person, human).
noun_sg(persona, persona, neutr).
noun_sg(personage, personage, human).
noun_sg(personal, personal, neutr).
noun_sg(personality, personality, neutr).
noun_sg(personation, personation, neutr).
noun_sg(personification, personification, neutr).
noun_sg(personnel, personnel, human).
```

```
⊟-s->entity
   ⊟-s->abstract_entity
      ⊟-s->abstraction
         ⊟-s->attribute
            ⊞-s->inheritance
            ⊞-s->personality
            ⊞-s->property
            ⊞-s->quality
```

Figure 6: A part of the ACE lexicon and the ACE ontology.

The Ace ontology does not exclusively contain terms coming from ACE lexicon and WN, but contains others extra concept types (like cg_proposition, cg_ant, cg_conseq, cg_modality, md_possibility, md_necessity, md_admissibility, etc.) and relations (like agentOf, objOf, propertyOf, in, etc.). These concepts and relations are added to the ontology (under the specific two nodes ace2cg_elt and relation) because of their usefulness for the construction of the CG. For example, the mapping of the ACE text: “The customer may be sick.” gives the following CG:

```
[cg_prop : [customer]-propertyOf->[sick]
        ]-propertyOf->[md_admissibility]
```

The two concepts “cg_prop” and “md_admissibility” and the relation “propertyOf” were extracted neither from ACE nor from WN, but were added to the ontology because of their usefulness for CG building.

4.3. Extension of ACE lexicon/ontology

Faced with the continuous extension of languages, the need for a technical language for some specific fields (such as medicine, pharmacy, etc.), the possibility to enrich linguistic resources such as ACE (Lexicon and ontology) is necessary.

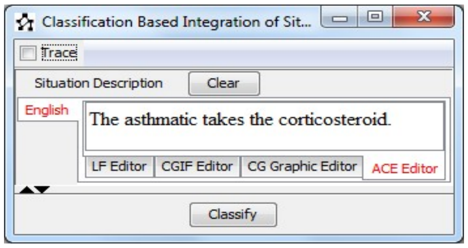
To enrich ACE in Amine, two modes are possible: The first mode is to **Manually** edit both the ACE lexicon and the ACE ontology files, this way is not appreciated since i) the edition of the lexicon file requires a good expertise of its structure and the required attributes for each syntactic form (verbs, nouns, adjectives ...) and ii) this extension cannot be done during the analysis, the analysis has to be stopped and launched again once the extension succeeded. The second mode is via the more interesting **Dynamic Extension** where the user is asked during the analysis via custom graphical interfaces to provide the required information for the word to extend and, once the extension succeeds, the analysis continues without being interrupted. This second mode has been implemented and integrated into Amine platform. Each dynamic extension consists of two steps:

1) The first step focuses on the extension of the ACE lexicon. Each ACE term is characterized by its syntactic form (noun, verb, adjective, adverb, preposition, pronoun or measurement noun), so while adding a new term to ACE, the user is requested to provide its syntactic form and then (depending on this syntactic form) indicates specific details:

- In the case of a new verb, the user must tell whether this verb is “Intransitive”, “Transitive” or “ditransitive”, specify its past form and for the ditransitive verbs, the preposition that goes with it (give to, take from, …).
- In the case of a new noun, the user must tell whether this noun is countable, mass or both (dom), then specify its singular and plural form, tell whether this noun is (or not) human and, in the case of human noun, is (s)he male or female ?
- In the case of an adjective (such as “tall”, “good”), the user must specify the comparison form of the adjective (like “taller”, “better”) and the superior form (like “tallest”, “best”).
- The same principle applies to the other syntactic forms (adverbs, pronoun …).

2) The second Step focuses on the extension of the ACE ontology that concerns basically the addition of a new type (that represents the new term) to the ACE ontology, this is why a window listing the ontology types tree appears asking the user to choose the super-type (or the location) of the new type.

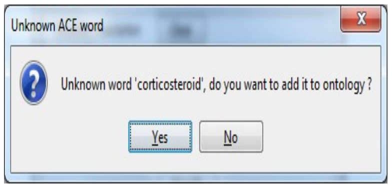
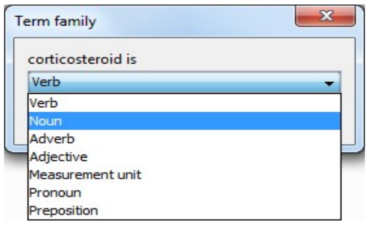
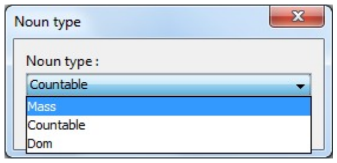
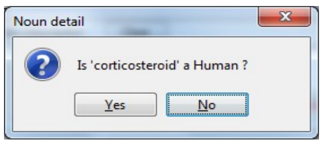
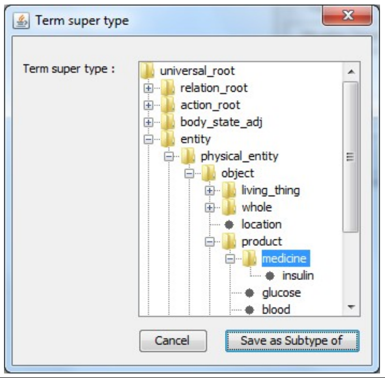
Let’s illustrate this extension with the analysis of the situation: “The asthmatic takes the corticosteroid.”. Initially, the ACE lexicon/ontology dont contain the “corticosteroid” term, so the dynamic extension process will be launched during the analysis. Figure 7.a, table 1 and figure 7.b respectively print the situation that contains the new word, the different steps for the dynamic extension process and the CG result of the analysis (after the extension).



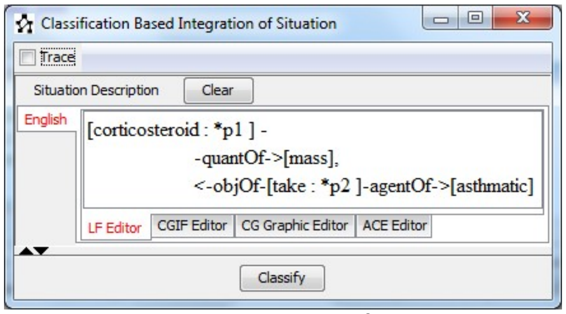
The asthmatic takes the corticosteroid.

Figure 7.a: The situation (to analyze) that contains the unknown word.

Table 1: The different steps for the dynamic extension process.

Steps	Graphical frames/Dialogs
Step 1 Amine Checks that the word is really new.	 Unknown ACE word — Unknown word 'corticosteroid', do you want to add it to ontology ? [Yes] [No]
Step 2 Amine Asks for the family of the new word.	 Term family — corticosteroid is: Verb ▾ — Verb, Noun, Adverb, Adjective, Measurement unit, Pronoun, Preposition
Step 3 Is the new word Mass, countable or dom (both)?	 Noun type — Noun type : Countable ▾ — Mass, Countable, Dom
Step 4 Is the word used for human?	 Noun detail — Is 'corticosteroid' a Human ? [Yes] [No]
Step 5 Amine asks for the location of the new type (related to the new word) in the ACE ontology.	 Term super type — Term super type : universal_root, relation_root, action_root, body_state_adj, entity, physical_entity, object, living_thing, whole, location, product, medicine, insulin, glucose, blood — [Cancel] [Save as Subtype of]

Now, the new term "corticosteroid" belongs permanently to user Ace lexicon/ontology files (not tentatively for this example) and will be automatically recognized during the next analyzes as can be seen in the following figure (figure 7.b).



```
[corticosteroid : *p1 ] -
        -quantOf->[mass],
        <-objOf-[take : *p2 ]-agentOf->[asthmatic]
```

Figure 7.b: The generated CG after the dynamic extension process

5. APPLICATIONS

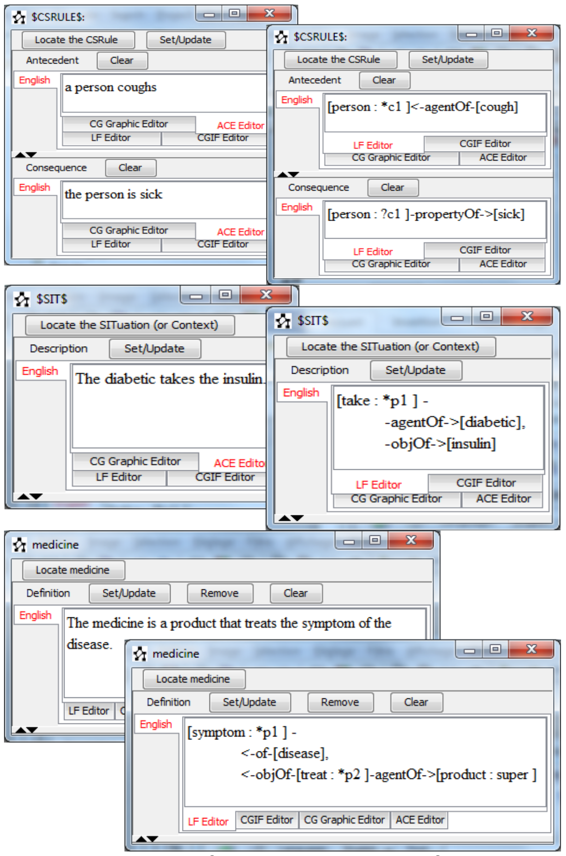
Towards a full use of ACE within Amine, this controlled language has been integrated as an alternative to CG in all the functionalities provided within Amine. Recall that CGs in Amine are used in the formulation of knowledge i) to be inserted in ontology (or Knowledge Base) as conceptual structures (definition, rule, canon, situation) with the aim to enrich it with extra semantic information or ii) to be processed with Amine ontology-based processes such as knowledge classification, reasoning, information retrieval, natural language processing, question answering, etc.

Next subsections show three examples of the use of ACE (alternatively to CGs) in Amine, the first one in the construction of the ontology and the two others in the two important processes: Information Retrieval and Question Answering.

Note that these two processes have been selected for illustration purposes only and the same principle applies to all other processes provided within Amine.

5.1. ACE in the construction of an Ontology and a Knowledge Base (KB)

While editing an ontology in Amine, the user is optionally asked to formulate the conceptual structures of a specific type. For this purpose and with the integration of ACE, the user has the choice between CG formalism and ACE controlled language. Figure 8 shows some examples of conceptual structures (the first one is for a rule, the second is for a situation and the third is for a definition) formulated using ACE and the translation into CGs proposed by Amine.



a person coughs

the person is sick

```
[person : *c1 ]<-agentOf-[cough]
[person : ?c1 ]-propertyOf->[sick]
```

The diabetic takes the insulin.

```
[take : *p1 ] -
      -agentOf->[diabetic],
      -objOf->[insulin]
```

The medicine is a product that treats the symptom of the disease.

```
[symptom : *p1 ] -
      <-of-[disease],
      <-objOf-[treat : *p2 ]-agentOf->[product : super ]
```

Figure 8: Example of conceptual structures formulated in ACE and their translations into CGs.

ACE has also been integrated in the construction of the Knowledge Base (KB) and can be used instead of CG in the formulation of situations and rules of the KB. The enrichment of the KB with situations and rules can be done in the same way as for the ontologies. This makes the task easier and does not require any expertise on CGs for creating ontologies/KB and therefore makes Amine adoptable for a wider population.

5.2. ACE in high level processes

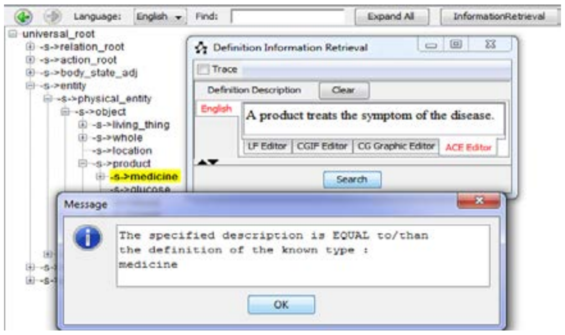
5.2.1 Case of Information Retrieval

Information Retrieval (IR) is a process that stores and manages information on documents. It assists users in finding the information they need. This process does not explicitly return information or answer questions, but informs on the existence and location of documents that might contain the needed information. Within Amine (which handled Ontologies/KBs and conceptual structures instead of documents and texts), IR assists users in knowing if a specified description is contained or not in the ontology/KB. It provides the user with the option to search for a type with a specific definition, search for a situation with a specific description and search for a rule with a specific description[7].

Let's look for a type with its definition. For example, the definition of "Medicine" is "A product treats the symptom of the disease.", to find this type (node) in the ontology, the user had to type the following text using the CG formalism:

```
[symptom : *p1 ] -
     <-of-[disease],
     <-objOf-[treat:*p2]-agentOf->[product]
```

Instead, the user can now type the definition in the controlled natural language ACE: "A product treats the symptom of the disease.", Amine translates it into a CG and runs the IR process. Figure below shows this example using the Ontology Editor.

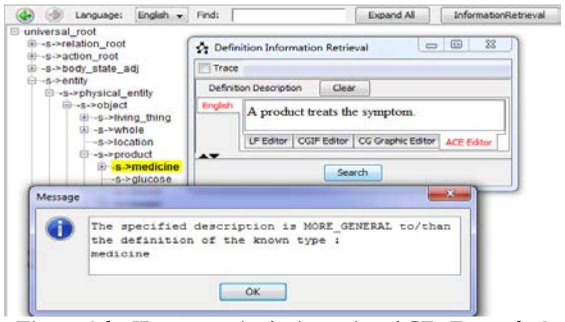


A product treats the symptom of the disease.

```
The specified description is EQUAL to/than
the definition of the known type :
medicine
```

Figure 9.a: IR process in Amine using ACE: Example 1.

As can be seen in the figure 9.a, the definition typed by the user is exactly (equals to) the definition of the type "Medicine". IR process can also recognize that the specified definition doesn't exist in the ontology but it is more general (or more specific) than a definition of a specific type in the ontology. Figure 9.b illustrates this process with a more general definition of the type medicine.



A product treats the symptom.

```
The specified description is MORE_GENERAL to/than
the definition of the known type :
medicine
```

Figure 9.b: IR process in Amine using ACE: Example 2

[7] For more details on IR process within Amine, please refer to the URL: http://amine-platform.sourceforge.net/component/kernel/ InformationRetrieval.htm

Please note that the ACE can be used instead of CG in all the other ontology-based processes provided within Amine like dynamic knowledge integration, ontology-based inferences, etc.

5.2.2 Case of a Basic semantic Question Answering System

The CG formalism has been widely adopted not only for its expressivity power, but also for processes that could be done with. Indeed, graph theory provides many operations that can be used for advanced knowledge management such as reasoning [18], information fusion [19], information retrieval [20], etc

Another interesting idea has been developed by Salloum [21] in the field of question answering, which is the use of the projection operation between the question CG and the text (the information text extracted from a document) CG. Projection operation is a syntactic mechanism that allows for a comparison of the knowledge contained in two CGs. It extracts the exact answer from the part of the sentence CG that has been projected under the question target concept.

This operation (and many others) is (are) provided by Amine (in the algebraic layer). Here is an example of the use of this operation:

CG1
[patient]-propertyOf->[property]
CG2 :
[patient: *p1] - -propertyOf->[sick], <-agentOf-[action_take:*p2]-objOf->[medicine]
The result of the projection operation
[patient]-propertyOf->[sick]

Figure 10: Example of the projection operation applied to two CGs.

With this operation and the integration of ACE, it became possible to develop an ACE-based question answering system. Since the principle and the operation have already been developed by others, we developed, only for illustration, a basic ACE-based question answering system that analyzes the question and the text, translates them to CGs and then applies projection operation.

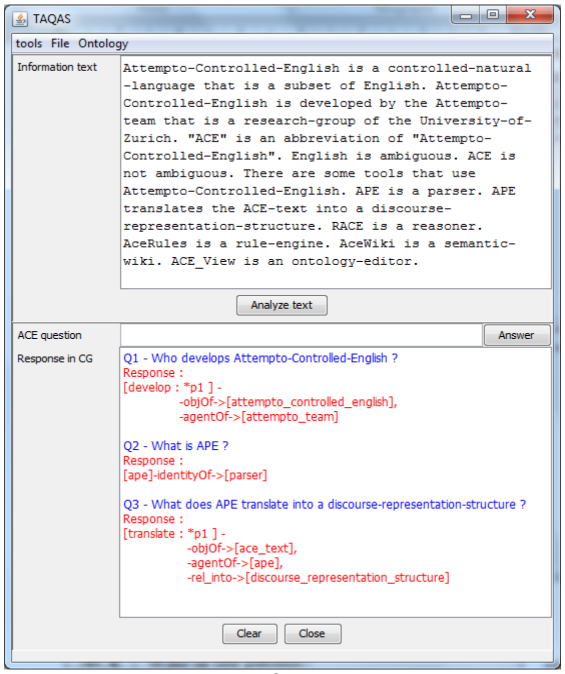
Let's briefly recall that Question Answering systems try to obtain a simple answer to a specific question (with both the question and the answer being formulated in natural language).

Before applying projection operation, the question had to be mapped in a way allowing projection operation to process with. The mapping of questions has been done as if they are affirmative sentences and then the searched information is replaced with variable (The question "where is the patient?" is seen as "The patient is in a location X"),. The following table displays some examples of ACE questions and the generated CGs:

Table 2: Examples of the mapping of the ACE questions into CGs.

ACE	Where is the patient?
CG	[cg_prop : [patient]-locOf->[**location : V1**]]-propertyOf->[cg_question : where]
ACE	Who is the patient?
CG	[cg_prop: [**universal_root:V2**]-identityOf->[patient]]-propertyOf->[cg_question : who]
ACE	What does the doctor give to the patient?
CG	[cg_prop : [action_give : *p1] - -agentOf->[object_doctor], -objOf->[**universal_root:V3**], -destOf->[object_patient]]-propertyOf->[cg_question : what]
ACE	How does the patient take the medicine?
CG	[cg_prop : [action_take : *p1] - -agentOf->[object_patient], -objOf->[medicine], -mannOf->[**manner_adv : V4**]]-propertyOf->[cg_question : how]
ACE	Is the patient well?
CG	[cg_prop:[object_patient]-propertyOf->[state_well]]-propertyOf->[cg_question : yes_no]
ACE	When does the patient take the medicine?
CG	[cg_prop : [action_take : *p1] - -agentOf->[object_patient], -objOf->[medicine], -timeOf->[**period : 5**]]-propertyOf->[cg_question : when]

The figure 11 illustrates this system with a complete (and real) example.



```
Attempto-Controlled-English is a controlled-natural
-language that is a subset of English. Attempto-
Controlled-English is developed by the Attempto-
team that is a research-group of the University-of-
Zurich. "ACE" is an abbreviation of "Attempto-
Controlled-English". English is ambiguous. ACE is
not ambiguous. There are some tools that use
Attempto-Controlled-English. APE is a parser. APE
translates the ACE-text into a discourse-
representation-structure. RACE is a reasoner.
AceRules is a rule-engine. AceWiki is a semantic-
wiki. ACE_View is an ontology-editor.
```

```
Q1 - Who develops Attempto-Controlled-English ?
Response :
[develop : *p1 ] -
          -objOf->[attempto_controlled_english],
          -agentOf->[attempto_team]

Q2 - What is APE ?
Response :
[ape]-identityOf->[parser]

Q3 - What does APE translate into a discourse-representation-structure ?
Response :
[translate : *p1 ] -
           -objOf->[ace_text],
           -agentOf->[ape],
           -rel_into->[discourse_representation_structure]
```

Figure 11: A snapshot of the developed ACE-based question answering system.

The user should load his ontology first (from the menu ontology/load), (s)he enters the text (the document content or the text supposed to contain the requested information) in the “information text” field and then activates the analysis (via the “analyze text” button), once the text is analyzed (and the CG is generated), the user can then ask the system about the meaning conveyed by the text by entering the questions in the “ACE question” field and pressing the “Answer” button. Amine analyzes the question, generates the CG and extracts the response CG from the information text CG using the projection operation (introduced above). This response CG is printed (in linear form) in the “Response in CG” field.

For instance, as we can see in the figure 11, when the user asks the question “Who develops Attempto-Controlled-English?” Amine generates the CG corresponding to this question[8]:

```
[develop : *p1 ] -
      -agentOf->[universal_root : V1 ],
      -objOf->[attempto_controlled_english]
```

Amine then extracts the response CG by applying the projection operation to the question CG and the information text CG[9], which gives the following result CG:

```
[develop : *p1 ] -
      -agentOf->[attempto_team],
      -objOf->[attempto_controlled_english]
```

6. CONCLUSION AND FUTURE WORKS

In this paper, we briefly introduced Amine as an open source platform, we explained their different components, its knowledge representation, some of its features (processes) and we outlined the benefits users can gain by adopting it.

We subsequently introduced the controlled natural languages, in particular ACE, and we emphasized the benefits it brings in the Human / Machine communication.

In the latest two parts, we presented our work on integrating ACE into Amine with examples illustrating the advantages gained by this integration.

Through this work, we can say that a high level interface has been added to Amine allowing humans to express knowledge using their natural language instead of a specific formalism.

The limitation with ACE is the fact that it is a controlled language rather than being a natural language. Therefore, its syntax and grammar are limited which makes it difficult for human users.

As future works, we are concerned by at least four research directions:

- The development of a real text-based question answering system.
- The (syntactic and grammatical) extension of ACE to support more sentence forms or the use of a natural language instead.
- The Generation of ACE from CG. This can be used in Amine CG editor (generates an ACE formulation for a given CG), in translation systems and in many other applications.
- The development of a "controlled Arabic language", that would allow the development of an Arabic-English translation system.

REFERENCES

[1] KABBAJ, Adil. An overview of Amine. In P. Hitzler and H. Schärfe (Eds.), *Conceptual Structures in practice*, CRC Press, Chapman & Hall Book. 2009.

[8] The ACE lexicon/ontology do not cover this example, much of the used words aren’t recognized and should (manually or dynamically) be added to them before running the example.

[9] The information text CG cannot be print here due to space limitation

[2] KABBAJ, Adil. Development of intelligent systems and multi-agents systems with amine platform. In: *Conceptual Structures: Inspiration and Application*. Springer Berlin Heidelberg, 2006. p. 286-299.

[3] SOWA, John F. Conceptual Graphs, in: *Handbook of Knowledge Representation*. 2008. p. 213-237.

[4] FUCHS, Norbert E., HÖFLER, Stefan, KALJURAND, Kaarel, and al. Attempto controlled english: A knowledge representation language readable by humans and machines. In: *Reasoning Web*. Springer Berlin Heidelberg, 2005. p. 213-250.

[5] FUCHS, Norbert E., KALJURAND, Kaarel, and KUHN, Tobias. Attempto controlled english for knowledge representation. In: *Reasoning Web*. Springer Berlin Heidelberg, 2008. p. 104-124.

[6] KAMP, Hans and REYLE, Uwe. *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*. Springer, 1993.

[7] SANTOS, Manuel Filipe, PORTELA, Filipe, and VILAS-BOAS, Marta. INTCARE: multi-agent approach for real-time intelligent decision support in intensive medicine. 2011.

[8] HADDAD, Hedi. Une approche pour supporter l'analyse qualitative des suites d'actions dans un environnement géographique virtuel et dynamique. 2009. *Thèse de doctorat*. Université Laval.

[9] ABOUENOUR, Lahsen, BOUZOUBAA, Karim, and ROSSO, Paolo. Improving Q/A Using Arabic Wordnet. In: *Proc. The 2008 International Arab Conference on Information Technology (ACIT'2008)*, Tunisia, December. 2008.

[10] DENNEMONT, Yannick, BOUYER, Guillaume, OTMANE, Samir, and al. 3D interaction assistance in virtual reality: a semantic reasoning engine for context-awareness. In: *Context-Aware Systems and Applications*. Springer Berlin Heidelberg, 2013. p. 30-40.

[11] MEKNI, Mehdi and MOULIN, Bernard. Informed Virtual Geographic Environments for Knowledge Representation and Reasoning in Multiagent Geosimulations. In: *COGNITIVE 2011, the Third International Conference on Advanced Cognitive Technologies and Applications*. 2011. p. 84-92.

[12] LAUNDERS, Ivan. The Transaction Graph for Requirements Capture in Semantic Enterprise Architectures. 2011. *Ph. D. Thesis*. Sheffield Hallam University

[13] SOWA, John F. *Conceptual structures: information processing in mind and machine.* 1983.

[14] SCHWITTER, Rolf. A layered controlled natural language for knowledge representation. In: *Machine Translation, Controlled Languages and Specialised Languages: Special Issue of Linguisticae Investigationes*. 2005. Vol. 28, No. 1, p. 85-106

[15] FUCHS, Norbert E., KALJURAND, Kaarel, and KUHN, Tobias. Discourse Representation Structures for ACE 6.6, *Technical Report ifi-2010.0010,* Department of Informatics, University of Zurich. 2010.

[16] NASRI, Mohammed, KABBAJ, Adil, and BOUZOUBAA, Karim. Integration of the controlled language ACE to the amine platform. In: *Conceptual Structures for Discovering Knowledge*. Springer Berlin Heidelberg, 2011. p. 159-172.

[17] MILLER, George A., BECKWITH, Richard, FELLBAUM, Christiane, and al. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography,* 1990, vol. 3, no 4, p. 235-244.

[18] CROITORU, Madalina. Conceptual Graphs at Work: *Efficient Reasoning and Applications.* 2006. PhD Thesis. University of Aberdeen.

[19] LAUDY, Claire, GANASCIA, J.-G., and SEDOGBO, Célestin. High-level fusion based on conceptual graphs. In: *Information Fusion, 2007 10th International Conference on*. IEEE, 2007. p. 1-8.

[20] MONTES-Y-GÓMEZ, Manuel, LÓPEZ-LÓPEZ, Aurelio, and GELBUKH, Alexander. Information retrieval with conceptual graph matching. In: *Database and Expert Systems Applications.* Springer Berlin Heidelberg, 2000. p. 312-321.

[21] SALLOUM, Wael. A question answering system based on conceptual graph formalism. In: *Knowledge Acquisition and Modeling*, 2009. KAM'09. Second International Symposium on. IEEE, 2009. p. 383-386.