Answering Why-Questions in Closed Domains from a Discourse Model

Rodolfo Delmonte
University of Venice "Ca' Foscari" (Italy)
email: `delmont@unive.it`

Emanuele Pianta
Fondazione Bruno Kessler – FBK (Italy)
email: `pianta@fbk.eu`

Abstract

In this paper we will present a system for Question Answering called GETARUNS, in its deep version applicable to closed domains, that is to say domains for which the lexical semantics is fully specified and does not have to be induced. In addition, no ontology is needed: semantic relations are derived from linguistic relations encoded in the syntax. The main tenet of the system is that it is possible to produce consistent semantic representations using a strict linguistic approach without resorting to extralinguistic knowledge sources. The paper will briefly present the low level component which is responsible for pronominal binding, quantifier raising and temporal interpretation. Then it will discuss in more detail the high level component where a Discourse Model is created from text. The system has been evaluated on a wide variety of texts from closed domains, producing full and accurate parsing, semantics and anaphora resolution for all sentences.

1 Introduction

In this paper we will present the system for Question Answering called GETARUNS, in its deep version applicable to closed domains, that is to say domains for which the lexical semantic is fully specified and does not have to be induced. GETARUNS is a GEneral multilingual Text And Reference UNderstander which follows a linguistically based approach to text understanding and embodies a number of general strategies on how to implement linguistic principles in a running system. The system addresses one main issue: how to restrict access to extralinguistic knowledge of the world by contextual reasoning, i.e. reasoning from linguistically available cues.

Another important issue addressed by the system is multilinguality. In GETARUNS the user may switch from one language to another by simply unloading the current lexicon and uploading the lexicon for the new language: at present Italian, German and English are implemented. Multilinguality has been implemented to support the theoretical linguistic subdivision of Universal Grammar into a Core and a Peripheral set of rules. The system is organized around another fundamental assumption: the architecture of such a system must be modular thus requiring a pipeline of sequential feeding processes of information, each module providing one chunk of knowledge, backtracking being allowed only within each single module. The architecture of the system is organized in such a way as to allow for feedback into the parser from Anaphoric Binding: however, when pronominals have been finally bound or left free, no more changes are allowed on the f-structure output of the parser.

Thus, we can think of the system as being subdivided into two main meta-modules or levels: Low Level System, containing all modules that operate at Sentence Level; High Level System, containing all the modules that operate at Discourse and Text Level by updating the Discourse Model. The deep and complete version of the system that we present here can be used with strictly closed domains and does not need any supporting ontology. However, it has also been used in one such context with a different architecture, which had OWL and RDFs as final external knowledge representation formalisms. Ontologies and Knowledge Sources should be used as Word Sense Disambiguation tools (we have not produced results on this however).

Texts belonging to what we define as closed domains are characterized by the fact that the system has all the semantic information which is needed process then; and most importantly, sentences making up the texts can be fully parsed without failures. In practice, these texts are relatively short and the length of sentences is below a certain threshold, typically 25 words. They are used for text understanding practice in a language learning environment. In this context, question answering is used to validate the appropriateness of the user's answer. Some such texts will be presented below. One will be the text used by Mitre in 2000 to organize the Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems (Brill et al., 2000). The system has been evaluated on a wide variety of such texts and has parsed fully and accurately all sentences with the appropriate Semantics and Anaphora Resolution (Delmonte, 2007).

2 The Low Level System

Even though we assume that the output of the Low Level System is mandatory for the creation of the semantic representation needed to create a consistent Discourse Model we will not be able comment it in depth for lack of space. We will simply show the internal components or modules it encompasses and add a few comments. However we stress the paramount importance of a deep linguistic analysis of the input text.

When each sentence is parsed, tense, aspect and temporal adjuncts are used to build the basic temporal interpretation to be used by the temporal reasoner. Every constituent is checked for semantic consistency and semantic features are added to each semantic head in the form of generic concepts taken from WordNet and other similar semantic lexical resources.

Eventually two important modules are implemented: Quantifier Raising and Pronominal Binding. Quantifier Raising is computed on f-structure which is represented internally as a DAG (Direct Acyclic Graph). It may introduce a pair of functional components: an operator where the quantifier can be raised, and a pool containing the associated variable where the quantifier is actually placed in the f-structure representation. This information may then be used by the following Higher System to inspect quantifier scope.

Pronominal Binding is carried out at first at sentence internal level. DAGs will be searched for binding domains and antecedents matched to the pronouns if any to produce a list of possible bindings. Best candidates will then be chosen.

3 The Discourse Model

Informally, a Discourse Model (DM) may be described as the set of entities "specified" in a discourse, linked together by the relations they participate in. They are called discourse entities, but may also be regarded as discourse referents or cognitive elements. A discourse entity (DE) inhabits a speaker's discourse model and represents something the speaker has referred to. A speaker refers to something by utterances that either evoke (if first reference) or access (if subsequent reference) its corresponding discourse entity.

As soon as a DE is evoked, it gets a description. The initial description ID that tags a newly evoked DE might have a special status, because it is the only information about an entity that can be assumed to be shared (though not necessarily believed) by both speaker and listener alike. However certain types of DE must be derived from other ones inferentially.

Definite descriptions can be used like definite pronouns to access entities which are presumably in the listener's DM, or they can be used to evoke new entities into that model.

Building a DM is clearly only a part of the overall process of understanding which makes heavy use of background mutual knowledge on the side of the addressee in order to carry out the complex inferences required. In order to build an adequate Discourse Model we rely on a version of Situation Semantics which takes perspectives or point of view as the higher node in a hierarchical scheme in which there is a bifurcation between factual and non-factual situations. Partially following Burke (1991) we assume that the notion of perspectives is significant in situation theory insofar as the very same situations can be viewed by an agent (or by different agents) from different perspectives, hence situations may support different and perhaps conflicting kinds of information (ibid., p.134). Situations are characterized in terms of infons, or better the infons that they support. In turn we distinguish between **facts** and **concepts** where the former have to do with concrete ostensive entities which yield information that is referential, in that they explicitly involve objects in the world relative to a given perspective. On the contrary **concepts** constitutes a piece of general information about the world relative to a given perspective, which does not directly refer to any particular entity or object, nor is it specific to particular ostensive entities.

Infons are built according to situation theory: a basic infon consists of a relation, its argument roles , a polarity, and a couple of indices anchoring the event/state/processe to a given spatiotemporal location.

In our system, **facts** may describe information relative to a **subjective** or an **objective** discourse domain: subjective facts are thus computable as situations viewed from the perspective of a given agent's mind, in our case also corresponding to the Main Topic of discourse. On the contrary, objective facts are reported from the perspective of the text's author. However, to highlight the difference existing between subjective and objective information in the model, we decided to call **facts** only objective infons; subjective infons are called **sit**. Also generic facts are treated as **sits**.

These main constituents of situations are further described by taking as primitives individuals, relations and locations and by using as logical notation set theory. Thus, individuals and inferences on individuals are wrought out in set theory notation: we use **ind** for a unique individual, **set** for a collection of individuals which can be individuated by means of membership, **card** for the cardinality of any set with a numerical or indefinite quantified value, **in** to indicate membership, **class** for generic sets which can be made up of an indefinite quantity however big enough to encompass sets, subsets, classes or individuals. Each entity is assigned a constant value or **id** and an **infon** which are uniquely individuated by a number.

Infons may express or contain a main relation: relations may be properties, social or relational roles, events or states, locational modifiers or specifiers — that is attributes, etc.. Simplex properties predicate some property of a semantic identifier; complex properties take individuals and propositions as their arguments and in this case individuals are usually associated to a semantic role. Semantic roles are inherited from the lexical form associated to a given predicate in the lexicon and transferred into the f-structure of the utterance under analysis. Semantic roles are paramount in the choice and construction of questions and answers.

Inferences are produced every time a given property is reintroduced in the story in order to ascertain whether the same property was already present in the model and should not be reasserted, or whether it should be added to it. Properties may be anchored to a given location or be universally anchored: a name, is a rigid designator in that it is computed as a property associated to a given individual and has a universal locational anchoring, meaning that the same individual will always be individuated by that name in the story. The same would apply to permanent properties like the substance or matter constituting an object, like a house, or other such properties. Persistence may then be computed both for entities, properties, relations and locations; also, a **Relevance Score** is computed by a separate module that analyzes information structure for each simplex utterance.

4 Semantic Rules

After collecting all modifier heads, if any, of the current predicate, the rule for the creation of semantic individuals separates previously resolved pronouns/nouns from non resolved ones. In both cases it uses some sort of equational reasoning in order to ascribe properties to already asserted semantic identifiers, by taking advantage of linguistic information encoded in Function/Role, according to a linguistically well-defined hierarchy which treats arguments and adjuncts as semantically different. New semantic individuals are added when needed.

The module handling semantic individuals treats new individuals to be asserted in the DM separately from already asserted ones — in which case, the semantic index should be inherited from properties belonging to previously asserted individuals. In addition, quantified expressions should be treated differently from individuals or sets, be they singleton sets, or sets with a given cardinality.

Semantic attributes are collected in the f-structure representation and come from the SPEC subsidiary function. We use the following attributes to separate semantic types: definiteness, partitivity and class. *Definiteness* applies to nominal expressions: these may be definite (`+def`), indefinite (`-def`), or zero definite (`def0`), which applies both to bare NPs and to proper nouns; *partitivity* is an attribute which gets a value only in case of quantified NPs. Finally the *class* attribute is used to differentiate proper nouns (`-class`) from common nouns (`+class`) which may undergo quantification, and quantified pronouns (`+me`).

5 Question Answering from a Discourse Model

In order to show how the system behaves we report and focus only on one small text. New texts are usually fully parsed: some intervention may be required to introduce contextual classes for tag disambiguation purposes. Here below is the text and the questions proposed for the Workshop on Text Understanding quoted above:

> **How Maple Syrup is Made**
> Maple syrup comes from sugar maple trees. At one time, maple syrup was used to make sugar. This is why the tree is called a "sugar" maple tree. Sugar maple trees make sap. Farmers collect the sap. The best time to collect sap is in February and March. The nights must be cold and the days warm. The farmer drills a few small holes in each tree. He puts a spout in each hole. Then he hangs a bucket on the end of each spout. The bucket has a cover to keep rain and snow out. The sap drips into the bucket. About 10 gallons of sap come from each hole.
> 1. Who collects maple sap? (Farmers)
> 2. What does the farmer hang from a spout? (A bucket)
> 3. When is sap collected? (February and March)
> 4. Where does the maple sap come from? (Sugar maple trees)
> 5. Why is the bucket covered? (to keep rain and snow out)

As far as we gathered from the proceedings of the conference, none of the participants was able to answer all the questions (Brill et al., 2000).

This is how we organize the system. We first compute the DM of the target question (hereafter QDM), the whole process is carried out on the basis of the facts contained in the question ad text DMs. Questions are classified into three types: partial or wh-questions, why questions and complete or yes/no questions.

Recovering the answer from the DM is essentially done in four steps:

- extracting question word or question type for yes/no questions
- extracting the main predicates from the question, which are then used to
- search for identical/similar predicates in the text DM
- extraction of the argument matching the answer

As commented in the sections above, the semantic representation contained in a DM can be basically defined as Predicate-Argument Structures or PAS, with a polarity and pair of spatiotemporal indices. Given a short text and a question about the text, the QA system will build a semantic model of the text where each distinct entity is assigned a unique semantic identifier, and is represented as a pool of properties, relations and attributes. Whenever possible, the system will also draw the necessary inferences to assign relation and attributes of sets to the individuals composing those sets.

Then it will completely parse the input question and produce a QDM for it, where facts are represented as **q_fact** terms. Afterwards, the first move consists in recovering the question word in the QDM by the following conjunction of queries

```
q_fact(K,focus,[arg:Id],1,_,_),
q_fact(_,isa,[_:Id,_:Focus],1,A,B)
```

where the variable Id, associated to the property "focus", is used to recover the actual Focus in the associated "isa" fact. This Focus is constituted by the question word used to formulate the query. This is used by the system to activate specialized procedures that will address specific semantic structures. As said above, *why* questions are processed separately from other *wh-* questions. The next query fired is

```
get_focus_arg(Focus,Pred,Args,Answer,True-NewFact),
```

which will give back the contents of the answer in the variable *Answer* and the governing predicate in *Pred*. These are then used to generate the actual surface form of the answer. *Args* and *True-NewFact* are used in case the question is a complete or yes/no question. In order to generate the answer, tense and mood are searched in the DM; then a logical form is build as required by the generator, and the build_reply procedure is fired:

```
get_focus_tense(T,M), Form=[Pred,T,M,P,[D]],
build_reply(Out,Focus,Form), !.
```

We will present general *wh-* questions at first. They include all types of factoid questions and also *how* questions. The main predicate looks for an appropriate linguistic description to substitute the wh- word argument position in the appropriate PAS. Here follows the full definition of the get_focus_arg procedure for the “who” case.

```
get_focus_arg(who,Pred,Ind,D1,NewP):-
   q_getevents(A,Pred),
   q_fact(X,Pred,Args,1,_,L),
   q_role(Y,X,Z,Role),
   answer_buildarg(Role,Pred,[Idx:Z],D,Facts),
   select_from_pred1(Pred,Role,Facts,NewP,D1), !.
```

We use a different procedure in case the question governing predicate is a copulative verb, because we have to search for the associated property in the QDM, as follows:

```
copulative(Pred),
q_fact(X,Pred,[prop:Y],1,_,_),
q_fact(Y,Prop,[_:K,Role:Type],1,_,_)
q_fact(_,inst_of,[_:K,_:Z],P,T,S),
q_get_ind_des(K,Propp,Ty),
```

Copulative predicates have a proposition as their argument and the verb itself is not useful, being semantically empty. The predicate corresponding to the proposition is searched through the infon *Y* identifying the fact. When we have recovered the *Role* and the linguistic description of the property *Propp* indicated by the wh- question, we pass them to the following predicate and search the associated individual in the DM:

```
answer_buildarg(Role,Pred,[Idx:Propp],Answer,Facts)
```

Suppose the wh-question is a *where* question with a copulative verb; then the role will be a location and the *Propp* will be "in". *How* copulative questions will search for **class** properties, i.e. not for names or individuals:

```
q_fact(X,how,[_:Y],1,_,_),
q_fact(Q,isa,[_:K,class:Pred],1,_,_),
q_fact(_,inst_of,[_:K,_:Z],P,T,S)
```

Semantic roles are irrelevant in this latter case: the only indication we use for the search is a dummy *prop* role. On the contrary, when lexical verbs are governing predicates, we need to use the PAS and the semantic role associated to the missing argument to recover the appropriate answer in all other cases. Here we should also use a different semantic strategy in case an argument is questioned and there is another argument expressed in the question — what, whom, who. Or else an adjunct is questioned — where, when, how, etc. — or the predicate is intransitive, an argument is questioned and there is no additional information available.

Now consider a typical search for the answer argument:

```
answer_buildarg(Role,Pred,Tops,Answer,Facts):-
   on(Ind:Prop,Tops),
   entity(Type,Id,Score,facts(Facts)),
   extract_properties(Type,Ind,Facts,Def,Num,NProp,Cat),
   select_allrole_facts(Role,Ind,Facts,Pred,PropLoc),
   Answer=[Def,nil,Num,NProp,Cat,PropLoc], !.
```

Here, *extract_properties* checks for the appropriate semantic type and property by picking one entity and its properties at the time. When it succeeds, the choice is further checked and completed by the call to *select_allrole_facts*. This is what *extract_properties* does:

```
extract_properties(Type, Ind, Facts, Def, Num, NProp, Gend):-
   ( Sclass=prop,
     extrfacts(Facts,Ind,Gend,Sclass,Prop), Num=sing
   ; Sclass=class,
     extrfacts(Facts,Ind,CGend,Sclass,Prop),
     select_gend(Prop,CGend,Gend) ),
   topichood_stack(Prop,Def),
   ( Type=ind, Num=sing
   ; Type=set, Num=plur ),
   set_def(Sclass, Ind, Prop, Role, Def),
   confirm_head(Def, Gend, Prop, NProp), !.
```

The procedure searches for individuals or sets filling a given semantic role in the predicate-argument structure associated to the governing predicate. In addition, it has the important task of setting functional and semantic features for the generator, like gender and number. This is paramount when a pronoun has to be generated instead of the actual basic linguistic description associated to a given semantic identifier. In particular, gender may be already explicitly associated in the DM to the linguistic description of a given entity or it may be derived from WordNet or other linguistic resources handling derivational morphology. The call *topichood_stack* looks for static definiteness information associated to the linguistic description in the DM. Proper names are always "definite". On the contrary, common nouns may be used in definite or indefinite ways. This information may be modified by the dialogue intervening between user and system and be recorded in the user model. The decision is ultimately taken by the *set_def* procedure which looks into the question-answering user model knowledge base where previous mentions of the same entity might have been recorded. Or else it does it — by means of *update_user_model* — to be used in further user-system interactions. If the entity semantic identifier is already present Def will be set to "definite", otherwise it will remain as it has been originally set in the DM.

```
set_def(Def,Id,Prop,Role,Def1):-
   ( tknow(Id,Role1), swap_def(Def,Def1)
   ; tknow(Prop,Role1), swap_def(Def,Def1)
   ; update_user_model(Id,Role), assign_def(Def,Def1) ).
```

6 Computing Answers to WHY questions

Why question are usually answered by events, i.e. complete propositions. They would in general constitute cases of rhetorical clause pairs labelled either as a Motivation-Effect or a Cause-Result. In Delmonte et al. (2007), causal relations are further decomposed into the following finer-grained subprocesses:

- Cause-Result
- Rationale-Effect

- Purpose-Outcome
- Circumstance-Outcome
- Means-Outcome

Furthermore, rationale clauses have been shown to be constituted structurally by untensed Infinitival Adjuncts: on the contrary, Cause-Result pairs are usually constituted by tensed propositions.

Consider now the pieces of knowledge needed to build the appropriate answer to the question "Why is the tree called sugar maple tree?". Sentences involved to reconstruct the answer are:

> Maple syrup comes from sugar maple trees.
> At one time, maple syrup was used to make sugar.
> This is why the tree is called a "sugar" maple tree.

In other words, in order to build the appropriate answer, the system should be able to build an adequate semantic representation for the discourse anaphora "This", which is used to essentially relate the current sentence to the event chain of the previous sentence. This is a fairly common way of expressing this kind of causal relation, that we then would like to assume as a paradigmatic one. Eventually, the correct answer would be:

> Because maple syrup was used to make sugar

which as can be easily gathered is the content of the previous complex sentence. Here below is the portion of the DM representation needed to reconstruct the answer:

```
ind(infon19, id8)
fact(infon20,inst_of,[ind:id8,class:edible_animal],1,univ, univ)
fact(infon21, isa,[ind:id8,class:[maple_syrup]],1, id1, id7)
set(infon23, id9)
card(infon24, id9, 5)
fact(infon25, sugar_maple, [ind:id10], 1, id1, id7)
fact(infon26, of, [arg:id10, specif:id9], 1, univ, univ)
fact(infon27,inst_of,[ind:id9,class:plant_life],1,univ, univ)
fact(infon28, isa, [ind:id9, class:tree], 1, id1, id7)

class(infon43, id13)
fact(infon44,inst_of,[ind:id13,class:substance],1,univ, univ)
fact(infon45, isa, [ind:id13, class:sugar], 1, id1, id7)
fact(id14,make,[agent:id8,theme_aff:id13],1,tes(finf_m3), id7)
fact(infon48,isa,[arg:id14,arg:ev],1,tes(finf_m3), id7)
fact(infon49, isa, [arg:id15, arg:tloc], 1, tes(finf_m3), id7)
fact(infon50, pres, [arg:id15], 1, tes(finf_m3), id7)
fact(infon51,time,[arg:id14,arg:id15], 1, tes(finf_m3), id7)
fact(id16,use,[theme_unaff:id8,prop:id14], 1, tes(sn5_m3), id7)

fact(id21,call,[actor:id9, theme_bound:id9], 1, tes(f1_m4), id7)
ent(infon61, id18)
fact(infon62,prop,[arg:id18,
             disc_set:[id16:use:[theme_unaff:id8, prop:id14]]],
              1, id1, id7)
ind(infon63, id19)
fact(infon66, inst_of, [ind:id19, class:abstract], 1, univ, univ)
fact(infon67, isa, [ind:id19, class:reason], 1, id1, id7)
fact(infon81, in, [arg:id21, nil:id19], 1, tes(f1_m4), id7)
fact(infon83, reason, [nil:id18, arg:id19], 1, id1, id7)
fact(id23, be, [prop:infon83], 1, tes(sn10_m4), id7)
```

These three pieces of knowledge representation are built respectively when the three sentences above are processed. When the second sentence is processed, the semantic identifier *id8* is simply inherited. Also, notice that it is transferred from USE predicate to MAKE by means of controlling equations which are part of LFG syntactic representations.

The system will at first look for a REASON semantic predicate associated to a CALL predicate, as derived from the question semantic representation, which we report here below:

```
q_loc(infon3, id1, [arg:main_tloc, arg:tr(f2_q6)])
q_ind(infon4, id2)
q_fact(infon5, tree, [nil:id2], 1, id1, univ)
q_fact(infon6, maple, [ind:id2], 1, id1, univ)
q_fact(infon7, sugar, [ind:id2], 1, id1, univ)
q_fact(infon8, of, [arg:id2, specif:id2], 1, univ, univ)
q_fact(infon9, why, [ind:id2], 1, id1, univ)
q_fact(infon10, inst_of, [ind:id2, class:plant_life], 1, univ, univ)
q_fact(infon11, isa, [ind:id2, class:tree], 1, id1, univ)
q_class(infon12, id3)
q_fact(infon13, inst_of, [ind:id3, class:substance], 1, univ, univ)
q_fact(infon14, isa, [ind:id3, class:sugar], 1, id1, univ)
q_class(infon15, id4)
q_fact(infon16, inst_of, [ind:id4, class:plant_life], 1, univ, univ)
q_fact(infon17, isa, [ind:id4, class:maple], 1, id1, univ)
q_fact(infon22, tree, [nil:id2, arg:id2], 1, id1, univ)
q_fact(id5, call, [prop:infon22], 1, tes(f2_q6), univ)
q_fact(infon23, isa, [arg:id5, arg:ev], 1, tes(f2_q6), univ)
q_fact(infon24, isa, [arg:id6, arg:tloc], 1, tes(f2_q6), univ)
q_fact(infon25, pres, [arg:id6], 1, tes(f2_q6), univ)
q_fact(infon26, time, [arg:id5, arg:id6], 1, tes(f2_q6), univ)
q_fact(infon27, focus, [arg:id7], 1, tes(f2_q6), univ)
q_fact(infon28, isa, [arg:id7, arg:why], 1, tes(f2_q6), univ)
q_fact(infon29, for, [arg:id5, motiv:id7], 1, tes(f2_q6), univ)
q_fact(infon35, perf, [arg:id8, ask:id5], 1, id1, univ)
```

The final part of the answer building process is constituted by the search of the actual linguistic description to associate to the original predicate. This is done in the pool of facts associated to the current entity which has been chosen from the inventory of entities of the world associated to the original text.

```
answer_buildarg(Role,Pred,Tops,Answer,Facts,[]):-
   on(Ind:Prop,Tops),
   entity(Type,Id,Score,facts(Facts)),
   extract_properties(Type,Ind,Facts,Def,Num,NProp,Cat),
   select_allrole_facts(Role,Ind,Facts,Pred,PropLoc),
   Answer=[Def,nil,Num,NProp,Cat,PropLoc],!.

select_allrole_facts(Role,Ind,Facts,Pred,PropLoc):-
   selarf(Pred,Fact,Args,Pol,Id),
   on(Fact,Facts),
   isa_role_fatto(Args),
   ind_role(Args,Inds),
   on(Prop-Role1,Inds),
   belongsrole(Role,Role1), !.
```

For instance, when searching the answer to the question "who collects the sap?", the answer is searched in the following pool associated to the entity FARMER:

```
entity(set,id32,28,facts([
   card(infon117,id32,5),
   fact(infon118,inst_of,[ind:id32,class:man],1,univ,univ),
   fact(infon119,isa,[ind:id32,class:farmer],1,id31,id8),
   fact(id33,collect,[agent:id32,theme_aff:id28],1,tes(f1_m6),id8),
   fact(id58,drill,[agent:id32,theme_aff:id56],1,tes(f1_m9),id8),
   fact(id63,put,[agent:id32,theme_aff:id61,loc_direct:id56],1,tes(f1_m10),id8),
   fact(id69,hang,[agent:id32,theme_aff:id66,loc_direct:id67],1,tes(f1_m11),id8)])).
```

This is reached from the COLLECT and SAP entities pools, which are cross-checked to verify that the same predicates are available.

```
entity(class,id28,7,facts([
  fact(infon102,inst_of,[ind:id28,class:substance],1,univ,univ),
  fact(infon103,isa,[ind:id28,class:sap],1,id27,id8),
  fact(id29,make,[actor:id9,theme_aff:id28],1,tes(f1_m5),id8),
  fact(id33,collect,[agent:id32,theme_aff:id28],1,tes(f1_m6),id8),
  fact(id41,collect,[agent:id39,theme_aff:id28],1,tes(finf_m7),id8),
  fact(id82,drip,[agent:id28, modal:id66],1,tes(f1_m13),id8),
  fact(infon343,has,[arg:id88,theme:id28],1,id84,id85)])).
```

Then *belongsrole* checks to verify that the *Role* belongs to the appropriate set of roles adequate for that slot in the PAS. In the “why” case it has to search recursively for events. This is the case represented by discourse anaphora of the type "this is why/that is why", where the reason is a complex event structure:

```
extract_properties(Role,Ind,Facts,NewProp):-
   Fact =.. [fact,Id,Pred,Args,Pol,Time,Place],
   on(Fact,Fa),
   on(_:Ind,Args),
   on(disc_set:Disc,Args),
   Disc=[Ind1:Pre:[Ro1:Id1, Ro2:Id2]],
   buildarg2(Ro2,NewP,[Id1:Prop],FirstProp,Facts,MDs),
   FirstProp=[Def1,nil,Num,NProp,Cat,PropLoc],
   Fact1 =.. [fact,Id2,NewP,Args1,Pol1,Time1,Place1],
   on(Fact1,Facts),
   on(Ro3:Ind2,Args1),
   Ind2$\backslash$=Id1,
   buildarg2(Ro3,What,[Ind2:Prop],SecProp,Facts1,MDs),
   SecondProp=[Def2,nil,Num2, NProp2,Cat2,PropLoc2],
   Prop_Why=[to,What,NProp2],
   NewProp=[Pre,[Def1,nil,Num,NProp,Cat,Prop_Why]], !.
```

Here below is the relevant DM representation of the other WHY question, the one requesting for a Motivation through Rational clauses: “why is the bucket covered?”

```
loc(infon288, id73, [arg:main_tloc, arg:tes(sn7_m11)])
ind(infon289, id74)
fact(infon290, inst_of, [ind:id74, class:event], 1, univ, univ)
fact(infon291, isa, [ind:id74, class:rain], 1, univ, univ)
ind(infon292, id75)
fact(infon293, inst_of, [ind:id75, class:event], 1, univ, univ)
fact(infon294, isa, [ind:id75, class:snow], 1, univ, univ)
ind(infon295, id76)
fact(infon296, isa, [ind:id76, class:cover], 1, id73, id8)
fact(infon297, inst_of, [ind:id76, class:legal], 1, univ, univ)
fact(infon301, cover, [nil:id69, arg:id76], 1, id73, id8)
fact(id77,have,[owner:id69,prop:infon301],1,tes(sn10_m12),id8)
fact(infon302, isa, [arg:id77, arg:st], 1, tes(sn10_m12), id8)
fact(infon303, isa, [arg:id78, arg:tloc], 1, tes(sn10_m12), id8)
fact(infon304, pres, [arg:id78], 1, tes(sn10_m12), id8)
fact(infon305, time, [arg:id77, arg:id78], 1, tes(sn10_m12), id8)
in(infon322, id74, id79)
in(infon323, id75, id79)
fact(id80,keep_out,[actor:id69,theme_aff:id79],1,tes(finf1_m12), id8)
fact(infon308, isa, [arg:id80, arg:pr], 1, tes(finf1_m12), id8)
fact(infon309, isa, [arg:id81, arg:tloc], 1, tes(finf1_m12), id8)
fact(infon310, pres, [arg:id81], 1, tes(finf1_m12), id8)
fact(infon311, time, [arg:id80, arg:id81], 1, tes(finf1_m12), id8)
fact(infon312, result, [arg:id77, arg:id80], 1, tes(sn10_m12), id8)
during(tes(sn10_m12), tes(sn7_m11))
includes(tr(sn10_m12), id73)
```

The relevant information is expressed as a semantic role RESULT, and is the one connecting the two predicates, HAVE/KEEP_OUT. This is the piece of information that will be used to answer the question.

7 Conclusions

In the paper we have shows that one can actually implement systems using deep linguistic and semantic analysis to answer hard questions. Our systems employs representations derived from Situation Semantics paradigm (Burke, 1991) and LFG syntactic theory (Bresnan, 2001). We have exemplified its performance on a series of factoid questions and we also added "why" questions. GETARUNS has been able to answer all questions proposed in the Mitre Workshop and also the additional semantically and syntactically hard discourse bound Why question based on the recurrent formulaic copulative expression "this/that is why". For a complete presentation of the system please refer to Delmonte (2007).

References

Bresnan, J. (2001). *Lexical-Functional Syntax*. Oxford: Blackwell.

Brill, E., E. Charniak, M. Harper, M. Light, E. Riloff, and E. Voorhees (Eds.) (2000, May). *Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Sytems*, Seattle, Washington. ANLP-NAACL.

Burke, T. (1991). Peirce on truth and partiality. In J. Barwise, J. M. Gawron, G. Plotkin, and S. Tutiya (Eds.), *Situation Theory and its Applications*. Stanford: CSLI Publications.

Delmonte, R. (2007). *Computational Linguistic Text Processing: Logical Form, Semantic Interpretation, Discourse Relations and Question Answering*. New York: Nova Science Publishers.

Delmonte, R., G. Nicolae, S. Harabagiu, and C. Nicolae (2007). A linguistically-based approach to discourse relations recognition. In B. Sharp and M. Zock (Eds.), *Natural Language Processing and Cognitive Science: Proc. of 4th NLPCS (Funchal, Portugal)*, pp. 81–91. INSTICC PRESS.