REASONING WITH A DISCOURSE MODEL AND CONCEPTUAL REPRESENTATIONS

Dario Bianchi, Rodolfo Delmonte

Section of Linguistic Studies - DSAO

Università Ca' Foscari - Ca' Garzoni-Moro
San Marco, 3417 - 30124 Venezia (Italy)
Tel.:041-2578464/52/19 E-mail:delmont@unive.it
WebSite:byron.cgm.unive.it

ABSTRACT

GETARUNS, the system for text and reference understanding which is currently used for summarization and text generation has a highly sophisticated linguistically based semantic module which is used to build up the Discourse Model. Semantic processing is strongly modularized and distributed amongst a number of different submodules which take care of Spatio-Temporal Reasoning, Discourse Level Anaphora Resolution, and other subsidiary processes like Topic Hierarchy which cooperate when creating semantic individuals. These are then asserted in the Discourse Model (hence the DM), which is then the sole knowledge structure for the reasoning processes to be independently carried out in BACK 5.2, the knowledge representation language which is used to do linguistically advanced queries. Some of the inference on events and their participants are elaborated on the basis of a subsidiary knowledge source constituted by Conceptual Representations for main event related predicated, already grafted onto the logical syntax of BACK programming language.

1. Introduction

Getaruns, the system for text and reference understanding which is currently used for summarization and text generation has a highly sophisticated linguistically based semantic module which is used to build up the Discourse Model. Semantic processing is strongly modularized and distributed amongst a number of different submodules which take care of Spatio-Temporal Reasoning, Discourse Level Anaphora Resolution, and other subsidiary processes like Topic Hierarchy which will impinge on Relevance Scoring when creating semantic individuals. The Knowledge Representation Language we use is called BACK - Berlin Advanced Computational Knowledge Representation System. We are currently using Version 5.2 released in September 1993. Functionality of the system as reported in the Readme file: Reasoning based on terminological logics (KL-ONE), supporting inheritance, consistency checking, cycle detection, classification, completion of partial descriptions, role inferences, abox revision, extended query answering.

Main system components: TBox (KB scheme), ABox (KB assertions), IBox (extensional implications). System interfaces: uniform access language.

We shall be working through the creation of the knowledge base for BACK on the basis of the following text from Sanford and Garrod:

At the Restaurant

John went into a restaurant. There was a table in the corner. The waiter took the order. The air was nice and clean. He began to read his book.

Here below the System Pipeline is shown with all its actual Prolog calls, where the starting call in_sentence receives as input the index of the DAG associated to the current sentence. We can divide up the Discourse Model processes into three parts or levels: Level one takes care of Topic Hierarchy and Anaphora Resolution :

System pipeline – LEVEL 1

```
in_sentence(Net, NoFr, NoCl, Ln) :-
 extract_ref_exprs(Net, RefList),
 write_ref_exprs('Tops & Sts', Net, RefList),
 resolve_externals(NoFr, RefList, Args),
 topic_hierarchy(NoFr, Args),
 print_states('Tops & Sts', NoFr), !.
```

Level 2 takes care of temporal reasoning and the creatiion of semantic individuals:

System pipeline – LEVEL 2

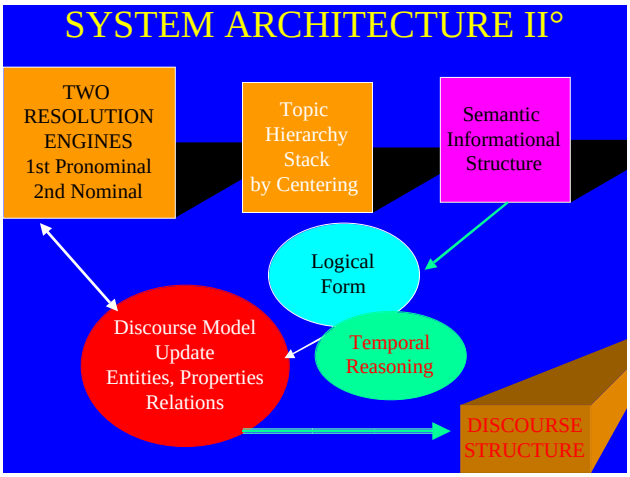
```
 allen_temp_reasoning,
 write_temp_relations('Temp-Relations'),
main_location(NoFr, time, temporal, RefList, Temp),
main_location(NoFr, place, locative, RefList, Loc),
semantic_information(NoFr),
semantic_inds(NoFr, Loc, Temp, Args),!.
```

Finally, Level three, takes care of rhetorical structure information, relevance score and builds the Discourse Model. In a final process, Discourse Structure is built:

System pipeline – LEVEL 3

```
write_rhet_structure('Info Evaluation', NoFr),
memorize_rels(NoFr,Lfs,Rels),
(write_ls_structure('L-Structure', Lfs, Rels);
write_ls_structure('S-Structure', Lfs, Rels) ),
write_sentence_infons('Model', NoFr),
connect_main_temp_loc(NoFr, Temp),
disc_structure(NoFr, Node, NoCl, Ln, NoClOut, At),
write_disc_structure('Discourse Structure', NoFr, Node,
NoClOut, Ln, At).
```

Table 1. Discourse Level Ssemantic Parser



Rather than describing Level 1 processes which have already been extensively reported in previous papers (1994, 1999), we shall concern ourselves with the intermediate Level 2 of the System Pipeline and shall describe in detail what semantic rules for the creation of individuals consist of. In particular, we shall be concerned with the use of linguistic information made availabe by f-structure representation in the semantic processing. LFG makes naturally available a semantic mapping from Grammatical Functions and Initial Descriptions into Semantic Individuals which is made simpler in our case by the discharge of Semantic Roles already during the parsing process (see Delmonte, Dibattista,Pianta, this volume).

As will be commented in detail below, it is just because of the principled subdivision of Predicative or non-semantically independent GFs from Non-Predicative ones that a Semantic Mapping is realized by rules in a completely general way. Predicative Functions in LFG are built in such a way to make directly available in the f-structure representation the syntactic index of their controller which is then automatically inherited by the semantics to assign a property.

2. The Discourse Model or DM

Informally, a DM may be described as the set of entities "naturally evoked" (or in Sidner's(1983) terms, "specified") by 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. We want to keep referring to what people do with language; evoking and accessing discourse entities are what texts/discourses do. A discourse entity 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. It is how the information is realized that determines what types of discourse entities are available when.

Now, a speaker is usually not able to communicate all at once the relevant properties and relations he may want to ascribe to the referent of a discourse entity. To do that he may have to direct the listener's attention to that referent (via its corresponding discourse entity) several times in succession. When the speaker wants to re-access an entity already in his DM (or another one directly inferable from it), he may do so with a definite anaphor (pronoun or NP). In so doing, the speaker assumes (1) that on the basis of the discourse thus far, a similar entity will be in (or directly inferable from) the listener's growing DM and (2) that the listener will be able to re-access (or infer) that entity on the basis of the speaker's cues. (For example, pronouns are less of a cue that anaphoric NPs). The problem then, at least for definite anaphora, is identifying what discourse entities a text naturally evokes.

What characterizes a discourse entity? Webber's view is that a discourse entity is a "conceptual coathook" (a term coined by William Woods) on which to hang descriptions of the entity's real world or hypothetical world correspondent. As soon as a DE is evoked, it gets a description. Over the course of the text, the descriptions it receives are derived from both the content of the speaker's utterances and their position within the discourse, as well as whatever general or specific information about the discourse entity the listener can bring to bear.

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, from the first and without question, be assumed to be shared (though not necessarily believed) by both speaker and listener alike. Even though certain types of DE must be derived from other ones inferentially, and that is the simplest way of account for anaphoric access to "generic set" DE.

The problem we set out to solve is transformed into (1) identifying the discourse entities a text evokes and (2) ascribing to them appropriate IDs.

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

Generally speaking, building a Discourse Model is a precondition for any reader or generic addressee of the contents of a legal text to enable reference to entities and events contained in the text. 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 adressee in order to carry out the complex inferences required by this genre.In order to build an adequate Discourse Model we rely on a version of Situation Semantics which takes perspectives 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.,134). We want to accomodate the fact that the theory is concerned with finite agents with limited information-handling capabilities: different agents carve the world up differently into objects, properties and relations according to their respective experience. 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.

A basic infon consists of a relation together with a polarity; which in turn consists of an assignment of appropriate arguments to the argument roles of that relation.

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, in our case 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**; finally, generic assertions are called **conc**.

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, which is very straightforward when using Prolog as programming language: we use **ind** for a unique individual, **set** for a collection of individuals which can be individuated by means of membership, **card** for every set with a numerical or indefinite quantified value, in to indicate membership, **class** for generic sets which can be made up by an indefinite quantity however big enough to encompass sets, subsets, classes or individuals, and **ensemble** to indicate mass nouns whose internal structure is made up only of subsets. Each entity is assigned a constant value or **id** and an **infon** which are uniquely individuated by a number. As we said, infons contain a main relation: relations may be properties, social or relational roles, events or states, locational modifiers or specifiers, etc.. Simplex properties predicate some property of a given identifier; complex properties take individuals and propositions as their arguments and in this case individuals may be assigned 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 conceptual representations.

Infons are built according to situation theory: they have a location which is made up of a couple of indices anchoring the event/state/processes to a given spatiotemporal location and a polarity.

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 world 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.

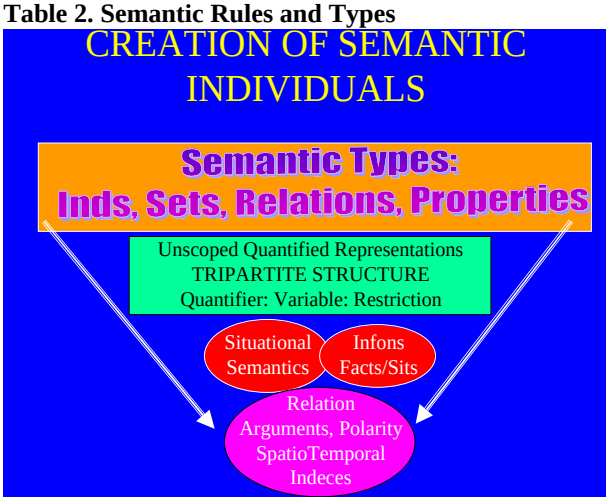
Persistence may then be computed both for entities, properties, relations and locations and from a separate module that computes information structure for each simplex utterance we get a **Relevance Score**. In this way complex situation and courses of events or eventualities may be constructed and checked.

3. 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 creation of semantic individuals has as its fundamental task that of treating separately new individuals to be asserted in the DM 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 clearly individuated individuals, be they singleton sets, or sets with a given cardinality.

Table 2. Semantic Rules and Types




```
create_sem_individuals(SnX/NoFr,Head,Def,Part,Card,
Class,Num,Cat,F/Role,Mods,Id,Temp, Loc)  :-
   (1<NoFr,
   resolved(NoFr, SnX, Anaf, Head),
resolv_inds(SnX/NoFr,Anaf,Head,Def,Part,
Card,Class,Num,Cat,F/Role,Mods,Id,Temp, Loc)
   ;
exist_inds(SnX/NoFr,Head,Def,Part,Card,Class,
         Num,Cat,F/Role,Mods,Id,Temp, Loc)  ).
```

3.1 Semantic Types

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

```
exist_inds(SnX/NoFr,Head,Def,nil,Card,Class,
Num,Cat,F/Role,Mods,Id,Temp,Loc)  :-
(Def= def0,
proper_noun(SnX/NoFr,Head,def0,nil,Quant,
                Card,Class,Num,Cat,
             F/Role,Mods,Id,Temp,Loc),!
;    Def \= -def,
old_indiv(SnX/NoFr,Head,-def,nil, Card,Class,
         Num,Cat,F/Role,Mods,Id,Temp,Loc),!
; (Role\=prop, Role\=compar),
    Part\=nil,
    (Class = +me; Class = +class),
 quantified_indiv(SnX/NoFr,Head,Def,Quant,Card0,
         Class,Num,Cat,F/Role,Mods,Id,Temp, Loc),!
; Def = -def,
  new_indiv(SnX/NoFr,Head,Def,Part,Card,Class,
         Num,Cat,F/Role,Mods,Id,Temp,Loc)),!.
```

In the following section we shall be discussing some of the most interesting semantic rules for the creation of semantic individuals. We shall start by commenting on the Rule for Proper Noun Association, which is computed as a property of an individual, usually a "name" property. As will become clear, text features such as genre or register, may determine the way in which varies the linguistic expression used to associate a proper name.

3.2 Rule for Proper Nouns Association

The first and more canonical method is the one that introduces the character by using a proper name, as in,

- John went into a restaurant.

```
proper_noun(SnX/NoFr,Head,Def,Part,Quant,_,Card,
-class, Num,Cat,F/Role,Mods,Id,Temp, Loc) :-
creatert(NoFr,name,Mods,SnX,Head,Card,Quant,
         Num,Id,Temp, Loc),
  class_props(NoFr, SnX, Cat, Id, fact, name, Head,
         1, univ, univ), !.
```

Another very common method, at least in children stories and novels, is the one that is represented by a copulative construction, as in,

- This is the story of three little pigs who went around the world seeking their fortune. Their names were Timmy, flute player, Tommy, violinist, Jimmy, great worker.

A more subtle and indirect way of obtaining the same result is shown in the following example from another of our texts,

- The three friends went all outside. As they were walking in the garden, John said to himself "Sara will marry that man", without any resentment. Richard would marry her.

In the latter example, the association is implicitly achieved by the semantic import of the structural organisation of the utterances or by discourse structure.

A pronoun is used in the subordinate clause to corefer to the property "friends" in the previous text; then, the subordinator indicates coincidence of temporal relation between the main and the subordinate clause. The main clause, in turn makes the pronoun explicit and introduces proper names as prominent characters. The inference we are naturally drawing at this point is that the Subject pronoun "they" and the Subject of the main clause point at the same individual.

Proper Noun Association in that case will require both Anaphora Resolution processes and an inferential rule that will search the previous Discourse Model (hence DM) for a suitable semantic description, a plural set whose cardinality is big enough to accomodate the characters appearing on the scene by means of their proper name.

3.3 Inferences and Triggers

Some inferences will be presented here below starting from Case 1, which we defined Predicate-argument equality match process.

Case 1 - Predicate-argument equality match

1. John gave Mary a rose.
2. She took it and put it in her hair.
3. She knew that she had been given a present, something precious.

TRIGGER: external pronoun “she” and the dummy logical predicate “exist” added by the grammar to account for the missing agent of the passive sentence. The search in the Discourse Model is also activated by the indefinite NP "a present" in presence of a governing presuppositional predicate “know”, in addition to tense Pluperfect in which the passive is expressed.
INFERENCES TO BE DRAWN: to recover the complete propositional content of the passive sentence a search in the Discourse Model is performed to look for a similar relation, “give” in a previous portion of text in which the same participant, “Mary” was involved with the same semantic role, that of “Recipient”. In addition the indefinite NP "a present" is computed as a property cospecifying "a rose" and not as a new property in the DM. And here below are the main rules in prolog code, starting from the indefinite expression “a rose”:

```
new_indiv(SnX/NoFr,Head,Def,Part,Card0,Class,Num,
        Cat,F/Role,Mods,Id,Temp, Loc):-
      nuclear(F),
sentential_feats(NoFr, F/Role, SnX,
         sem_cat([presuppositional,subjective]), Pred),
 ( resolved_argument(NoFr, SnX, Head1, R3)
  ;
 find_argument(fcomp/Role1, F/Role, SnX, Head1,
R3)),
 nth_place_pred(3,Pred,R3,Head1,Class),
creater(NoFr,Mods,SnX,Head,Card,Quant,Num,Id,T,
L),
class_props(NoFr, SnX, Cat, Id, sit, isa, Head, 1, T, L),
 assert_one(NoFr,SnX,fact(isa,[Class,Id],1,T, L)), !.
```

A second example shows the importance of building a Discourse Structure (see Delmonte & Pianta, 1994), which may then be used to corefer event names when used with a discourse deictic pronoun.
Case 2 - Selectional restrictions membership match
Mary picked up the phone and called Jason. Her husband, she thought, would have considered such a move as untruthful and utterly base.
TRIGGER: the deictic NP “such a move” realized linguistically as an Object semantically dependent on the predication.
INFERENCES TO BE DRAWN: The grammatical nature of the NP is such that it does not allow it to be computed as an independently referring entity, so the previous discourse is searched for. In particular complements of an ECM verb like consider are treated as semantically dependent arguments: i.e. they don’t receive a semantic role from the main predicate “consider” but act as controller of the predicative function governed by “as”. The indefinite NP is taken to be deictic by the presence of “such” and is computed as a case of textual deixis: the entity being referred to in this case is a relation, that is the “calling” event. This is due to the semantical restriction associated with the noun "move", which is classified as an "event". All the preceding clauses would have constituted a suitable discourse segment in that case. That is, the move would be understood as both the “pick_up” and the “calling” events.
Here below we report the DM for the text “At the Restaurant”, which will then be used to reason in Back.

DISCOURSE MODEL FOR TEXT UNDER ANALYSIS

```
sentence(r01.new,[john, went, into, a, restaurant])
loc(infon3, id1, [arg:main_tloc, arg:tr(f5_r01)])
loc(infon4, id2, [arg:main_sloc, arg:restaurant])
ind(infon5, id3)
fact(infon6, inst_of, [ind:id3, class:man], 1, univ, univ)
fact(infon7, name, [john, id3], 1, univ, univ)
fact(infon9, isa, [arg:id2, arg:restaurant], 1, id1, id2)
fact(id4, go, [agente:id3, locat:id2], 1, tes(f5_r01), id2)
fact(infon10, isa, [arg:id4, arg:ev], 1, tes(f5_r01), id2)
fact(infon11, isa, [arg:id5, arg:tloc], 1, tes(f5_r01), id2)
fact(infon12, past, [arg:id5], 1, tes(f5_r01), id2)
overlap(tes(f5_r01), td(f5_r01))

sentence(r02.new, [there, was, a, table, in, the, corner])
ind(infon21, id6)
ind(infon22, id7)
fact(infon23, inst_of, [ind:id7, class:thing], 1, univ, univ)
fact(infon24, isa, [ind:id7, class:corner], 1, id1, id2)
fact(infon25, in, [arg:id6, locativo:id7], 1, id1, id2)
fact(infon26, isa, [ind:id6, class:table], 1, id1, id2)
fact(infon27, inst_of, [ind:id6, class:thing], 1, univ, univ)
fact(id8, there_be, [tema_nonaff:id6], 1, tes(f4_free_r02), id2)
fact(infon31, isa, [arg:id8, arg:st], 1, tes(f4_free_r02), id2)
fact(infon32, isa, [arg:id9, arg:tloc], 1, tes(f4_free_r02), id2)
fact(infon33, past, [arg:id9], 1, tes(f4_free_r02), id2)
included(tr(f4_free_r02), id1)
```

```
contains(tes(f4_free_r02), tes(f5_r01))

sentence(r03.new, [the, waiter, took, the, order, .])
ind(infon42, id10)
fact(infon43, inst_of, [ind:id10, class:[social_role]], 1, univ, univ)
fact(infon44, isa, [ind:id10, class:waiter], 1, id1, id2)
fact(infon45, role, [waiter, id2, id10], 1, id1, id2)
fact(id12, take_order, [actor:id10, goal:id3], 1, tes(f2_free_aq), id2)
fact(infon48, isa, [arg:id12, arg:pr], 1, tes(f2_free_aq), id2)
fact(infon49, isa, [arg:id13, arg:tloc], 1, tes(f2_free_aq), id2)
fact(infon50, past, [arg:id13], 1, tes(f2_free_aq), id2)
included(tr(f2_free_aq), id1)
after(tes(f2_free_aq), tes(f5_r01))

sentence(r04.new, [the, air, was, nice, and, clean])
loc(infon59, id14, [arg:main_tloc, arg:tes(f2_free_aq)])
fact(infon60, isa, [arg:id15, arg:air], 1, id14, id2)
fact(infon61, [clean, nice], [arg:id15], 1, id14, id2)
fact(id16, be, [prop:infon61], 1, tes(f5_r04), id2)
fact(infon62, isa, [arg:id16, arg:st], 1, tes(f5_r04), id2)
fact(infon63, isa, [arg:id17, arg:tloc], 1, tes(f5_r04), id2)
fact(infon64, past, [arg:id17], 1, tes(f5_r04), id2)
included(tr(f5_r04), id14)
contains(tes(f5_r04), tes(f2_free_aq))

sentence(r05.new, [he, began, to, read, his, book])
fact(infon76, isa, [arg:id18, arg:book], 1, id14, id2)
fact(infon77, poss, [arg:id18, poss:id3], 1, id14, id2)
fact(id19, read, [agente:id3, tema_aff:id18], 1, tes(finf1_free_a1), id2)
fact(infon78, isa, [arg:id19, arg:pr], 1, tes(finf1_free_a1), id2)
fact(infon79, isa, [arg:id20, arg:tloc], 1, tes(finf1_free_a1), id2)
fact(infon80, pres, [arg:id20], 1, tes(finf1_free_a1), id2)
fact(id21, begin, [agente:id3, prop:id19], 1, tes(f3_free_a1), id2)
fact(infon82, isa, [arg:id21, arg:ev], 1, tes(f3_free_a1), id2)
fact(infon83, isa, [arg:id22, arg:tloc], 1, tes(f3_free_a1), id2)
fact(infon84, past, [arg:id22], 1, tes(f3_free_a1), id2)
included(tr(f3_free_a1), id14)
after(tes(f3_free_a1), tes(f2_free_aq))
```

PART II

4. MAPPING SEMANTIC REPRESENTATIONS INTO THE KNOWLEDGE BASE FOR BACK 5.2

Conceptual Representations(CR) have been introduced by Jackendoff and others, however we refer to Dorr(1993) who introduced a number of augmentation to the original set which we also endorse. Delmonte(1990, 1996) considered CR the link from the semantics to the knowledge of the world needed to represent meaning in a general and uniform manner. The Discourse Model only contains reference to semantic roles and other semantic relations like Poss, which have a correspondence in the CR. Here below are CRs for some of the verb predicates of the text under analysis.

pred(exist[BE(<theme_unaffect>(STAYposit(AT)))
{(en,tn),(e1,t1)}])
pred(enter[CAUSE(<agent>(GOposit(FROM[x]
(INTO<locat_into>))))){(en,tn),(e1,t1)}])
pred(take_order[CAUSE(<actor>(GO(FROM<goal>)))
{(e1,t1),(en,tn)}])
pred(read[LET(<address>(GO(REP(FROM<informtn>)
))
{(en,tn),(e1,t1)}])

4.1 From Conceptual Representations to Inferential Rules

a) [CAUSE (X,E) at t1] => [E] cond = +specific(t1)
b) [STAY ([X],[AT Y]) from t1 to t2] =>
[BE ([X],[AT Y]) at t3] cond = t1<t3<t2
c) [GO([X],[FROM Y],[TO Z]) at t1] =>
[BE ([X],[AT Y]) at t2] &
[BE ([X]?[AT Z]) at t3]
cond = t2<t1<t3

The reading of these expressions is quite intuitive: in a) if an agent X causes E than E takes place, under the condition that reference time be specific; b) is the subinterval condition which is cast into J.Allen's formalism for temporal reasoning; c) shows how a motion predicate is translated into a couple of state predicates and so on.

d) [GO ([X],[(AWAY_)FROM Y],[TO(WARD) Z]
from t1, to t2] =>

NOT [STAY ([X],[AT Y])
from t1, to t2] & NOT [STAY ([X],
[AT Z]) from t1, to t2]

e) [STAY ([X],[AT Y] from t1, to t2] =>
NOT [GO ([X],[(AWAY_)FROM Y],
[TO(WARD)W]) from t3, to t4]
cond = t1<t3<t4<t2

4.2 Definition of sets, general class, social roles and time intervals.

At first, very general roles and properties are declared, like for instance the set of proper names or the set of prepositions:

set :- proper_names := aset([john,mary]),
masfem := aset([male,female]),
pol := 0..1,
special := aset([in,on,under,near]).
generic_class :- general :< anything.
int :- time :< general,
tloc :< tempo.

We then define attributes to be associated to individuals which we recover from the DM and divide up accordingly into three main general group: PERSONS, OBJECTS and PLACES.

persons :- man :< general,
has_name :< domain(man) and range(proper_names),
sex :< domain(man) and range(masfem),
get_persons.
objects :- thing :< general,
get_objects.
places :- place :< general,
get_places.
get_objects:-
fact(_,inst_of,[ind:Y,class:thing],1,_,_),
fact(_, isa, [ind:Y, class:Class], 1, _, _),
fact(_, in, [arg:Y, locativo:Loc], 1, _, _),
Class :< thing
and all(locat,place) and all(spec,special)
and all(location,place) and all(partOf,thing),!.
get_objects:-
fact(_,inst_of,[ind:Y,class:thing],1,_,_),
fact(_, isa, [ind:Y, class:Class], 1, _, _),
Class :< thing and all(location,place).
get_places:-
fact(_, isa, [arg:Y, arg:Class], 1, _, Y),
Class :< place.
get_persons:-
fact(_,inst_of,[ind:Y,class:social_role],1,_,_),
fact(_, isa, [ind:Y, class:Class], 1, _, _),
fact(_,role,[Class,X,Y],1,_,_),
Class :< man and all(work_in,X).

Aspectual information is used to individuate the appropriate internal constituency of the event (see also Delmonte, 1997 and above), and also to drive the semantics, which together with the information coming from arguments and adjuncts will be able to trigger the adequate knowledge representation. In particular, as shown in Passonneau 1988, we need to process reference to entities and events in the discourse model, in order to know what predicates are asserted to hold over what entities and when.

Finally, EVENTS, STATES and PROCESSES are special objects which possess a polarity, a reference timeand an agent that causes something: it may be a process as in the case of BEGIN, or and event of going to a place as for GO, or still a theme_nonaff (a non affected theme that stays in a given location.

events :- ev :< general
and all(time,tense) and all(polarity,pol),
specify_event_predicates.
states :- st :< general
and all(time,tense) and all(polarity,pol),
specify_state_predicates.
processes :- pr :< general
and all(time,tense) and all(polarity,pol),
specify_process_predicates.
specify_event_predicates:-
fact(_,isa,[arg:X,arg:ev],1,A,B),
fact(X,Pred,[Agent:_,locat:_],1,B,_),
Pred :< ev
and all(agent,man) and all(location,place)
and all(prim,primCause) .
specify_state_predicates:-
fact(_,isa,[arg:X,arg:st],1,A,_),
fact(X,Pred,Arguments,1,B,_),
Pred :< st
and all(theme_nonaff,thing) and all(prim,primbe).
specify_process_predicates:
fact(_,isa,[arg:X,arg:pr],1,A,_),
fact(X,Pred,[Agent:_,theme_aff:_],1,B,_),
Pred :< pr
and all(Agent,man) and all(theme_aff,thing)
and all(prim,primCause),!.

From general entities we then declare instances as they have been collected in the DM with their semantic ids, for instance for all instances of man we use,

facts :- fact(_,inst_of,[ind:Y,class:man],1,_,_),
Y :: man.

For all instances of individuals belonging to the class of social_roles,

facts :- fact(_,inst_of,[ind:Y,class:social_role],1,_,_),
fact(_, isa, [ind:Y, class:Class], 1, _, _),
Y :: Class.

Names and roles are associated to a specific relation:

facts :- fact(_,name,[X,Y],1,_,_),
Y :: has_name:X.
facts :- fact(_,role,[Class,X,Y],1,_,_),
Y :: works_in:X.

The same applies for ids associated to spatiotemporal locations, both for entities and relations like events, processes or states,

facts :- fact(_,isa,[ind:X,class:Y],1,A,Z),
X :: Y and time:A and location:Z.
facts :- fact(_,isa,[arg:X,arg:ev],1,A,B),
X :: ev and time:A and location:B.

Then we associate to a relation id all its semantic properties, from semantic roles to spatiotemporal relations, as for instance with predicates like GO,

facts :- fact(X,Pred,[agent:Z,locat:A],1,B,_),
X :: Pred and agent:Z and location:A and time:B.

4.3 Defining Semantic Fields and Inferences from Conceptual Primitives

```
fields :- semanticfield :< anything,
        spatial :< semanticfield,
        perceptive :<semanticfield,
        existential :<semanticfield,
        possessive :< semanticfield,
        circumstantial :<semanticfield .
defineGo :-  fromgo:<anything,
             togo:<anything,
             themego:<anything,
             pathgo:<oneof([to,toward,via]),
             primgo:< primitive
and all(from,fromgo) and all(to,togo)
and all(path,pathgo)
and all(semanticfieldgo,semanticfield)
and all(theme,themego).
defineStay :- themestay :< anything,
              statestay :< anything,
              pathstay :< oneof([in,at]),
              primstay :<primitive
         and all(state,statestay) and all(path,pathstay)
         and all(semanticfieldstay,semanticfield)
         and all(theme,themestay).
defineBe :- themebe :< anything,
            placebe :< anything,
            pathbe :< oneof([in]),
            primbe :<primitive
           and all(place,placebe) and all(path,pathbe)
           and all(semanticfieldbe,semanticfield)
           and all(themeNonAgent,themebe).
defineCause :- themeCause :< anything,
  primCause :< primitive and all(theme,themeCause)
         and all(primitiveIntroduced,primitive).
```

4.4 Object Building for Inferential Processes on the Basis of CRs

```
generate :- collect_ids(ev,Pred,X,Y,K),
         concat(cause,K,R),concat(Pred,K,S),
         assoc_sem_field(Pred,Field),
  R::primCause and theme:X and primitiveIntroduced:
    (S:: primgo and theme:X and from:nil and to:Y
          and semanticfieldgo:Field),
                linkverb(K,R).
 generate :- collect_ids(ev,Pred,X,Y,K),
         concat(be1,K,R),
         assoc_sem_field(Pred,Field),
 R:: primbe and themeNonAgent:X and place:Y
        and semanticfieldbe:Field,
                linkverb(K,R).
collect_ids(ev, Pred):-
fact(_,isa,[arg:X,arg:ev],1,A,B),
fact(X,Pred,[FstRole:Y,SndRole:Z],1,B,_), !.
```

4.5 BACK 5 Internal DATABASE of Classes and Instances

```
b5st_class_db(prim(waiter), (c, 0), 45, internal).
b5st_class_db(prim(works_in), (r, 0), 46, internal).
b5st_class_db(works_in, (r, 0), 47, works_in:<domain
(anything) and range(anything)).
b5st_class_db(prim(restaurant), (c, 0), 49, forward).
b5st_class_db(restaurant, (c, 0), 50,
restaurant:<anything).
b5st_class_db(waiter, (c, 0), 51, waiter:<man and
all(works_in, restaurant)).
b5st_class_db(prim(agente), (r, 0), 113, internal).
b5st_class_db(agente, (r, 0), 114, agente:<domain
(anything) and range(anything)).
b5st_class_db(prim(prim), (r, 0), 118, internal).
b5st_class_db(prim, (r, 0), 119, prim:<domain(anything)
and range(anything)).
b5st_class_db(prim(st), (c, 0), 126, internal).
b5st_instance_db(mary, (a, 0), 14, no_definition).
b5st_instance_db(john, (a, 0), 15, no_definition).
b5st_instance_db(restaurant, (c, 0), 159, ([], [])).
b5st_instance_db(id3, (c, 0), 155, ([man,
has_name:john], [])).
b5st_instance_db(id10, (c, 0), 158, ([waiter,
lavora_in:id2, locazione:id2, time:id1], [])).
b5st_instance_db(tes(f5_r01), (c, 0), 162, ([], [])).
b5st_instance_db(tes(f3_free_a1), (c, 0), 164, ([], [])).
b5st_instance_db(tes(f2_free_aq), (c, 0), 166, ([], [])).
b5st_instance_db(tes(finf1_free_a1), (c, 0), 168, ([], [])).
b5st_instance_db(tes(f4_free_r02), (c, 0), 175, ([], [])).
b5st_instance_db(id18, (c, 0), 178, ([book, location:id2,
time:id14], [])).
b5st_instance_db(id21, (c, 0), 165, ([begin, ev,
agente:id3, location:id2, prop:id19,
time:tes(f3_free_a1)], [])).
b5st_instance_db(poss, (c, 0), 179, ([have, agente:id3,
tema_aff:id18, time:id14], [])).
b5st_instance_db(id6, (c, 0), 157, ([table, thing,
locaz:id7, location:id2, spec:in, time:id1], [])).
b5st_instance_db(id7, (c, 0), 156, ([corner, thing,
location:id2, partOf:id2, time:id1], [])).
b5st_instance_db(go_id4, (c, 0), 284, ([primgo,
semanticfieldgo:spatial, from:nil, tema:id3, to:id2], [])).
b5st_instance_db(cause_id4, (c, 0), 285, ([primCause,
primitiveIntroduced:go_id4, tema:id3], [])).
```

5. SEMANTIC Ids & QUERIES TO THE SYSTEM

Here below are some of the queries that can be addressed to the system. We propose the canned sentence proposed to the user, then the internal output of the query, and finally the answer generated by the internal generator.

“Where was John?”

```
| ?- where_was(id3).
After tes(f5_r01) was in id2
```

after entering john was in the restaurant

“Where was the table?”

```
| ?- where_was(id6).
id6 was in id7 of id2
```

the table was in the corner of the restaurant

“Where was the corner?”

```
| ?- where_was(id7).
id7 part of id2
```

the corner was part of the restaurant

```
" Where was john after being entered "
| ?- where_was_after(id4,id3).
```

id2
in the restaurant
"Where was the waiter?"
| ?- where_is(waiter).
location = [[id2]]
the waiter is in the restaurant
"Who took the order?"
| ?- who_took_order.
take_order = id12
agent = [[id10]]
the waiter took the order
"Who ordered?"
| ?- who_ordered.
take_order = id12
goal = [[id3]]
john ordered

"Who was in the restaurant?"
| ?- who_was_there(restaurant).
[[id10]]
the waiter was in the restaurant

"What was in the restaurant?"
| ?- what_was_there(restaurant).
[[id7],[id6],[id17]]
in the restaurant there was a corner a table a book

6. REFERENCES

Allen J.F.(1983b), Maintaining Knowledge about Temporal Intervals, *Communications of the ACM*, 26,11, 832-843.
Allen J.F.(1984), Towards a General Theory of Action and Time, *Artificial Intelligence* 23, 123-154.
Allen J.F.(1994), **Natural Language Understanding**, The Benjaming/Cummings Pub.Co. Redwood City CA.
Barwise J.(1985), The Situation in Logic-II: Conditionals and Conditional Information, *Report No. CSLI-85-21*.
Barwise J.(1985), The Situation in Logic-III: Situations, Sets and the Axiom of Foundation, *Report No. CSLI-85-26*.
Barwise J., J.M.Gawron, G.Plotkin, S.Tutiya(eds.)(1991), Situation Theory and its Applications, Vol.2, *CSLI Lecture Notes No.26*.
Cooper R.(1991), *Three Lectures on Situation Theoretic Grammar*, in **Situation Theoretic Grammar, op.cit.**, 1-44.
Cooper R.(1991), **Situation Theoretic Grammar**, 3rd European Summer School in LLI, Universität des Saarlandes, Saarbrücken.
Delmonte R.(1990), Semantic Parsing with an LFG-based Lexicon and Conceptual Representations, *Computers & the Humanities*, 5-6, 461-488.
Fenstad J.E. et al.(1987), **Situations, Language and Logic**, Reidel, Dordrecht.
Halvorsen P.K.(1983), Semantics for Lexical Functional Grammar,*Linguistic Inquiry* 4,567-616.
Halvorsen P.K.(1988), Situation semantics and semantic interpretation in constrained-based grammars, *Proceedings of the International Conference on Fifth Generation Computer System*, Tokyo, 471-478.
Halvorsen P.K. & R.Kaplan(1988), Projections and semantic description, *Proceedings of the International Conference on Fifth Generation Computer System*, Tokyo, 1116-1122.
Hendrix G.(1978), The Representation of Semantic Knowledge, in D.Walker(ed), **op.cit.**, 121-224.
Herzog O., C.-R. Rollinger(1991)(eds), **Text Understanding in LILOG**, Springer Verlag, Berlin.
Higginbotham J.(1985), On Semantics, *Linguistic Inquiry* 16, 4, 547-593.
Higginbotham J.(1989), Elucidations of Meaning, *Linguistics and Philosophy* 12, 4, 465-518.
Hobbs J.R.(1993), Intention, Information, and Structure in Discourse: A First Draft, Artificial Intelligence Center, SRI International.
Hornstein N.(1989), **As Times Goes By: Tense and Universal** , Ms, University of Brandeis.
Jackendoff R. (1972), **Semantic Interpretation in Generative Grammar**, The MIT Press, Cambridge, MA.
Jackendoff R.(1976), Towards an Explanatory Semantic Representation, *Linguistic Inquiry* 7, 89-150
Jackendoff R.(1980), Belief Contexts Revisited, *Linguistic Inquiry* 11(2).
Jackendoff R. (1983), **Semantics and Cognition**, MIT Press, Cambridge Mass.
Jackendoff R. (1985a) Believing and Intending: Two Sides of the Same Coin, *Linguistic Inquiry* 16, 445-460.
Jackendoff R. (1985b), Multiple Subcategorization and theTheta-Criterion: the Case of *Climb*, *Natural Language and Linguistic Theory* 3, 271-296.
Jackendoff R.(1987a), **Consciousness and the Computational Mind**, The MIT Press, Cambridge Mass.
Jackendoff R.(1987b), The Status of Thematic Relations in Linguistic Theory, *Linguistic Inquiry* 18, 369-411.
Jackendoff R.(1993), The Combinatorial Structure of Thought: The Family of Causative Concepts, in Reuland & Abraham(eds), **op.cit.**, 31-50.
Jackendoff R.(1993b), On the role of Conceptual Structure in Argument Selection: A Reply to Emonds, *Natural Language and Linguistic Theory* 11, 2, 279-312.
Laresohn P.(1990), Group Action and Spatio-Temporal Proximity, *Linguistics and Philosophy* 13, 2, 179-206.
McCawley J.(1981), **Everything that Linguistis have Always Wanted to Know about Logic* - *but were ashamed to ask**, The University of Chicago Press,Chicago.
Martin W.(1981), Roles, Co-Descriptors, and the Formal Representation of Quantified English Expressions, in *American Journal of Computational Linguistics*, 7, 3, 137-147.
ter Meulen A.(1983), The representation of time in natural language, in A. ter Meulen(ed), **Studies in Modeltheoretic Semantics**, Foris, Dordrecht, 177-192.
Moens M., M.Steedman, Temporal Ontology and Temporal Reference, in *Computational Linguistics* cit., 15-28.
Partee B. (1993), Semantic Structures and Semantic Properties, in Reuland & Abraham(eds), **op.cit.** 7-30.
Pereira F.(1989), A Calculus for Semantic Composition and Scoping, in *Proceedings of the 27th Annual Meeting of ACL*, Vancouver,
Poesio M.(1991), Relational Semantics and Scope Ambiguity, in J.Barwise et al.(eds.), **op.cit.**, 469-498.
Webber B. L. (1981), Discourse model synthesis: preliminaries to reference. In A. Joshi, B. L. Webber and I. Sag (eds.), **op.cit.**.
Webber B. L.(1983), So what can we talk about now?, in Brady M. and Berwick R.(eds), **Computational Models of Discourse**, The MIT Press, 331-372.