NATURAL LANGUAGE SEMANTICS AND LOGIC PROGRAMMING

Patrick Saint-Dizier, Evelyne Viegas

IRIT Université PaulSabatier
118, route de Narbonne
31062 TOULOUSE Cedex France
e-mail: stdizier/viegas @irit.irit.fr

Introduction

Very different goals, motivations and methods are at the origin of the development of the three main trends in natural language semantics:

- the logico-philosophical trend, mainly oriented towards defining formal models,
- the psycological trend, which carries out models based on human observations,
- the computational linguistic trend whose goal is to develop softwares which, in a certain sense simulate the understanding of natural language.

These trends obviously have meeting points and overlap to some extend, we think that a real, complete account of our language is understood has to take into account these three levels.

Natural language semantics is a very complex domain, it is in fact the major problem of language processing. We can distinguish three levels where semantics factors occur:

- the lexical level, dealing with the meaning of words. Meanings of words are not defined in isolation, but a lexicon is a kind of lattice in which words are related by lexical semantics relations.
- the phrase levels, whose goal is to build an adequate representation of a proposition, taking into account argument structures as well as modifiers.
- the discursive level, whose goal is to relate propositions. This level has in particular to solve references (pronominal, temporal locative, etc...) and to relate objects between propositions.

These levels clearly have relations and the distinction between them is not always trivial. In this document, we will mainly develop the first level, on which there is now a lot of emerging work. This work will be done within the framework of logic programming, more specifically within the framework of logical types and active constraints. Logic Programming turns out, as shall be seen to be a very convenient formalism to describe lexical semantics. Besides lexical semantics, logic programming has not yet been much used practically and for its formalism within natural language semantics. Of interest are the development of lambda-Prolog to deal with lambda-abstractions of the kind encountered in Montague semantics and CIL (Mukai 86) designed to deal with complex indeterminates of the type found in Situation Semantics (Barwise and Perry 85).

In this document we will mainly focus on *how world and domain specific knowledge can be associated with an appropriate lexical organization of word-senses*. Our experience indeed has shown us that the lexicon is one of the best media between, on the one hand linguistic knowledge, and on the other hand world knowledge. Reasoning procedures will be specified in a separate module. They will be used in the different stages of the translation system and will operate on these lexical descriptions, considering them as (nonprimitive) objects.

The basic motivation behind this approach is that most current lexical knowledge systems are too weak to account for one of the central problems: ambiguity because of the under-specification of the meanings of words (since words can be polysemic, we will use hereafter the notion of **word-sense,** assuming that a particular sense of a word can be isolated and characterized). There are several reasons for this situation (as noted also in Pustejovsky, 1991): most lexical knowledge systems are centered around the semantics of verbs and do not account for the semantics of other objects like nouns, adverbs and other major lexical categories. Secondly, they represent lexical ambiguity by means of exhaustive lists of possible senses (sometimes using tricky notational devices). This approach is too limited since it is not flexible enough *to account for the-creative uses of words in a given domain.* It is clear to us that a lexical entry should be much more than a mere enumeration of attribute-value pairs: deeper aspects of meaning and knowledge have to be taken into account. The dynamic nature of word meaning (and also of phrase meaning, to some extent) is reflected in the fact that the meaning of a word often depends on the meanings of the other words in the sentence and on the way they are syntactically organized. Finally, this research will be a substantial contribution to a theory of lexical meaning, a central problem for most natural language processing systems.

Finally, in order to have a system which is efficient, we will give to lexical semantics relations a **constraint logic programming interpretation.** Besides increasing efficiency, this approach also increases the overall expressivity of the finalism since active constraints are executed only whenever necessary and have to be true and consistent with the other constraints throughout the whole proof procedure. Translating a sentence of a given language into a sentence of another language is viewed as **a constraint satisfaction problem.**

1. A formalism for world knowledge representation associated to lexical descriptions

Lexical organization, to be fully adequately treated and to be operational, requires the inclusion of areas previously thought beyond the scope of grammar and linguistics, such as common sense knowledge, inheritance, default reasoning, collocational relations and domain knowledge. One of the underlying, long-term, goal of our research is to build a theory of lexical meaning, with its own peculiarities, e.g., by considering the relations between lexical semantics knowledge and world knowledge.

In this section, we present our view on the way semantic and pragmatic information about linguistic objects can be represented and dealt with to more accurately parse and generate natural language utterances. At the level of semantic and pragmatic knowledge representation, we consider three main representational media for 'word-senses':

- **feature systems:** pairs of attribute-values, used at the level of the word representation,
- **lexical semantics relations** to organize a lexicon as a whole (relations like taxonomies, meronomies, synonyms and opposites holding between word senses),
- **logico-pragmatic representations** of words by means of abstract representations like scenarios or scripts, which reflect world knowledge about that word-sense.
- **general commonsense reasoning rules,** which are specified in a separate module and which are refered to by translations rules. These reasoning rules use the lexical entries as objects. We now present these aspects and explain their use for our problematics.

1.1 Feature systems

Features mainly permit an efficient and simple representation of attribute-values pairs having discrete values. They usually denote properties which take their value in finite sets. Among them, we have for instance:
- morphological characteristics,
- syntactic category,
- semantic features such as: semantic class (e.g; human, inanimate, etc. . .),
- aspectual value,
- thematic role (agent, patient, etc. . .)
- and argument structure (in simple cases).

Feature structures may be represented by trees, their procedural semantics is well-established within most NLP systems. Feature attributes may be subject to default value assignments or to constraints. The determination of feature values may be complex and may involve different levels of reasoning, refering to world knowledge and to the information present in the other elements of the utterance to parse or to generate. However, to preserve efficiency and simplicity, these types of reasoning should be kept minimal.

Features mainly permit to represent linguistic knowledge and some very simple aspects of world knowledge. They are often prefered to any other representation modes because of their ease and efficiency of treatments. World knowledge representation and also several aspects of linguistic and lexical semantics knowledge representation indeed require representation modes which cannot be accounted for by means of feature-value pairs but they rather involve relational aspects (lexical semantics relations) or reasoning (by means of logical formulae).

The formalism we have used so far in our work is very close to the feature system used in Eurotra. It is based on Login (Ait-Kaci and Nasr, 1986), which is a type-based structure with an extended unification mechanism which integrates a built-in treatment of taxonomies. In Login, the syntactic representation of a structured term is called a y-term. It consists of:

(l) a *root symbol*, which is a type constructor and denotes a class of entities,

(2) attribute labels, which are record field symbols. Each attribute denotes a function in extenso, from the root to the attribute value. The attribute value can itself be a reference to a type or to an instance of a type.

(3) coreference constraints among paths of labels, indicate that the corresponding attributes denote the same function. They are indicated by variables.

Here is an example, the description of the concept of person:

```
person( id => name(first => string,
                        last => X: string),
           born => date(day => integer,
           month => monthname, year=>integer),
           father => person( id => name(last => X ))).
```

The root symbol is *person; id, born* and *father* are three sub-y- terms which have either constants or types as values. X indicates a coreference. All different type structures are tagged by different symbols. Notice also that in the latter field only relevant information about person is mentioned. Infinite structures can also be specified by coreference links. Variables are in capital letters, constants in small letters.

To this framework, we have added features necessary to language processing:
- *conjunction, disjunction and negation of feature values,*
- *use of (simple) formulae as feature values,*
- *inheritance of attributes,*
- *association with active constraints of the constraint logic programming framework.*

Within our approach, feature values are viewed as types, instead of just atoms. This permits us to have recursive type descriptions. The reference we make to types permits the introduction of active constraints.

Three main classes of operations at the basis of that typed-based language allow:
- the expression of type construction to describe phrase structures and lexical entries,
- the expression of relations (either local or long-distance) between types,
- the expression of well-formedness constraints on types.

The two last operations are more global than the first one and do not appear in the usual NLP systems. To these operations are also associated common-sense reasoning rules which will help making decisions when there several ways of processing a structure.

This formalism permits us to define **type inheritance** and the possibility to define in a clean way classes and subclasses corresponding to **structured linguistic objects.** The use of active constraints on these structures permits us to have a very elegant way of specifying well-formedness constraints on linguistic objects: using a certain linguistic object in a certain context contributes to a **constraint satisfaction problem.**

1.2. Lexical semantics:

Lexical semantics relations, by means of lexical inheritance structures, determine the ways in which a word-sense is related to other word-senses in the lexicon. First, it answers questions concerning the **organization of a lexical knowledge base,** second, it makes it possible to link lexical knowledge with general world knowledge. Moreover, since we aim at multilinguism, organizing lexical knowledge by means of lexical semantics relations is in keeping with **psycholinguistics models** for mental lexicon organization. These general models have shown that people tend to access words by means of routines such as the is-a and part-of relations.

Lexical semantics offers a large variety of **semantic relations:**

(1) **essential relations** include:
- taxonomies (the well-known *isa* relation and more refined taxonomic relations),
- meronomies (the *part of* relation), this relation is complex, it is decomposed into several subrelations such as: component of an integral part (pedal-bicycle), stuff of material (steel-car), member of a collection (student-class), etc...
- synonymies, usually defined with reference to a given context,
- complementaries, opposites, also given with refence to a context (e.g. by reference to arrive, stay and leave are opposites, (Saint-Dizier 91b)).

(2) **functional relations** express more pragmatic factors such as, for example, the instrumental role of an object. These relations are usually described by means of abstract structures like scenarios.

The role of lexical semantics relations is essential in semantics. They permit a very precise characterization of words in relation with other words. They permit to factor out regularities, similarities, differences, counterparts, etc... between words. Finally, they also permit a more modular, uniform, principle-based, reliable and declarative description of the meanings of words. The complete characteristics of a word are determined on the basis of its relations with other words by means of operations like: inheritance in taxonomies, partial equality (or overlapp) for synonyms, complements for complementaries, etc... Descriptions may not be completely compatible, introducing thus non-monotonicity in lexical descriptions.

We think that lexical semantics relations mainly permit to represent world knowledge. Linguistic tests (such as the A is part of B for relating A and B in a meronomy) permit to characterise in a quite reliable way the relations holding between word-senses; however, these relations essentially reflect world knowledge. Our aim is to explore how lexical semantics relations interact with the world knowledge expressed within the functional descriptions (the scenarios in our case) and how these lexical semantics relations can be used to describe in a more precise, accurate and reliable way these scenarios. Ultimately, these relations could originate a principled way of defining scenarios.

From an operational point of view, lexical semantics relations introduce properties like transitivity or inheritance of features which are very convenient and elegant, but which are often very heavy to manipulate, introducing a lot of extra-processing, and thus, a lot of inefficiency. Techniques like pre-compilation or partial evalution have been explored, but they have not given very satisfactory results. The advantages of using active constraints is to avoid a lot of useless computation while maintaining an overall coherence of the system. For example, inheritance of features is only performed when required by the parser/generator, otherwise, the inheritance is just kept as a potentiality by the active constraints. We have evalutated that up to about 80% of useless features values are computed in standard systems. Besides taxonomies, meronomies permit to make very refined inferences which are very useful for world knowledge representation

The formalism we have to describe lexical semantics relations is very simple. Binary relations like taxonomies and meronomies are represented by a binary operator, e.g.:
A isa B.
C part_of B.
while ternary relations are represented with an annotation prefix:
A : B opp C.
within the context of A, B is an opposite of C. These relations are active constraints (SaintDizier 91b).

1.3. Ontologies, predicates and scenarios

In order to more accurately and more fully describe semantic data, it is necessary to develop methods to define the **ontology of the domain** of application considered. We have also explored and want to go on exploring and defining some **'primitive' relations** operating on these ontologies such as: identification, possession, localization, etc... This domain is extremely complex, but it is, to our sense, essential to give an appropriate semantics to words and sentences. In real applications, the degree of precision, i.e. the degree of atomicity, of the primitives can be more or less refined depending on the complexity of the natural language utterances and on the complexity of the treatment which is forseen. Primitive relations are hypothetized to be **multi-lingual** (at least for our European languages), **multi-purpose** (translation as well as man-machine interfaces), **not committed to any linguistic theory** and, finally, they can be made more refined in an incremental, monotonic way (i.e. changes in the degree of precision of the representations will not affect the organization of the results already elaborated but will just make them more precise). Finally, these primitive relations constitute the basis of contextual reasoning.

A scenario represents the meaning of a word-sense in a more dynamic way than feature-value pairs. It uses the ontology and the primitive relations to describe the 'operational' aspects of a word-sense in the world which is considered. It can roughly be represented by a triple of the following form:
< preconditions,
description of the action,
evolution of the world >.

To a given word can be attached several scenarios describing its different facets, each of them being a word-sense. In the above representation, the preconditions part is a set of formulae (built from the basic predicates and the ontology of the domain) which have to be true for the representation of the concept to be the second element of the triple and for the evolution of the world to be the third element of the triple.

A scenario describes world knowledge. This world knowledge is clearly monolingual, as the remainder of the lexical entry (the attribute-value pairs, the lexical semantics relations it has with other word-senses). Application-specific knowledge as well as general, common-sense knowledge cannot fully be represented in lexical entries, it is also represented by a set of inference rules which may operate on these lexical entries or may be used by them. In the case of machine translation applications, these inference rules may, for example, also encode the transfer (or contrastive) knowledge.

Let us now consider some simple, informal examples. Scenarios were first used for the representation of verbs, e.g. the verb to screw can be informally (for the sake of readability, a more precise logical formulation is however quite simple) represented by the following scenario:

SCREW:
preconditions:
the following objects are at disposal in the world: screw, screw-driver, wall there is an agent capable of screwing
action:
put a screw in a wall w at location 1 and time t
evolution of the world:
there is a screw in the wall w at location 1.

Nouns, adjectives and adverbs can also be represented by means of scenarios. For example, a possible scenario for a *car* will express *its functional role,* namely that a car permits people to be carried from one place to another. This mode of expression (here by reference to the function of the noun) is much more dynamic than a mere feature value:

CAR:
preconditions:
there are human beings capable of driving in the current world
action:
move, carry persons P from location 11 at time tl to location 12 at time t2
evolution of the world:
persons P are at location 12.
Several recent approaches indeed claim that noun meanings should make reference to related word-senses via lexical semantics relations.

An adjective like *fast* can also be represented by an informal scenario like the following, in which we introduce prototypical information about speed: FAST: *preconditions:* object to which adjective is applied can move *action:* functionality: object modifier such that: object = car --> speed of object > 100 km/h object = snail ==> speed of object > 0. lcm/sec, etc. . .

The formalism of scenarios can be easily integrated into the lexical semantics relations system. Indeed, the scenario of a given word-sense is defined with respect to the scenarios of the other word-sense it is in relation with. For example, a word-sense will inherit of the scenario descriptions of its superordinates. From another point of view, this entails that the semantic representation in terms of scenario of a given word-sense must be coherent with the semantic representations of its superordinates. The way predicates in scenarios combine is still an open problem, however. The lexical semantics approach will be used in our system to control the coherence of some aspects of scenarios when they are defined by the expert.

Finally, the scenario system can be further refined by having scenarios with several sub-fields for the action part, each of these sub-fields describing a diferent aspect of the use of the scenario. For example, we may decompose action into:

- its telic aspect,
- its aspectual function, whenever appropriate,
- its argument structure.

1.4 Common-sense and application-dependent reasoning

Several aspects of common sense and application-dependent knowledge and reasoning procedures cannot directly be encoded within lexical entries for two major reasons:

- they cannot be associated, by their very nature, to lexical entries (this is the case for example for preferences, which express a kind of **pragmatic relation** between lexical entries),
- they are more general and require to be represented at a more abstract level.

The reasoning procedures and the general and domain-dependent knowledge use the lexical knowledge (primitive values like feature values as well as complex values like the formulae associated to the fields of a scenario) to block or to allow a certain transition rule to be activated. This perspective is often refered to as a constraint satisfaction problem.

The information represented in these general and domain-dependent knowledge and reasoning procedures can be very different. In this project, we will only consider very simple, but useful, rules like the following:

- preference rules among objects or actions,
- rules associated to lexical semantics relations such as the incompatibility between the sisters in a taxonomy,
- default reasoning rules,
- domain-dependent rules expressing the semantics of the domain.

These rules will be expressed within a Horn Clause schema, even if their theoretical foundations go beyond this simple, but operational, framework. One of our strategy is to limit the number of these rules and to have as many semantic information as possible in the lexicon, where it can be defined in a more homogeneous way.

2. Development of active constraints on typed constructions

2.1 Introduction to constraints

Our goal is to develop active constraints of the Constraint Logic Programming framework applied to typed constructions (representing feature structures in grammatical constructions and in lexical entries) in order to:

- **improve efficiency** of the actual system,
- **improve expressivity** since active constraints have a different semantics,
- **improve genericity and reusability** of the tools developed for language processing so that they can be used for different subareas and purposes.

Active constraints of the constraint logic programming framework (Jaffar and Lassez, 1987, Colmerauer, 1990) are:

- **active throughout the whole proof construction process,** that means that they have to be true throughout the whole proof construction process,
- **they are computed** (or evaluated) **when there is available sufficient knowledge** about their variables,
- **their coherence is checked at each step** of the proof construction,
- they are **more modular,**
- the result of a query is a set **of constraints defining domains of values for variables,** however, the constraints also express deep relations between variables which should be kept,
- **fully independent of the way they are used** (e.g. for parsing or generating sentences, with a bottom-up or a top-down strategy), - **fully declarative** and easy to use since they are directly encoded.

Using a certain lexical entry, translating a certain construction into another one is then viewed as **a constraint satisfaction problem.**

To summarize, we can say that CLP offers a global rule-based framework to handle constraints. CIJP associates unification and constraint solving. CLP programs are highly declarative, soundly based within a unified framework of formal semantics, they exhibit a greater expressive power, and a greater ease of programming since constraints are directly stated (there is no need to encode them by means of terms).

2.2. Our constraint and type-based framework

Within the framework of natural language processing (parsing and generation), we have developed basic constraints to deal with linear precedence restrictions, subcategorization satisfaction, feature inheritance management and expression of long-distance dependencies.

Lexical semantics relations like taxonomies (A isa B), meronomies (A is part of B), synonymy and opposition can also be represented by active constraints. We then have a more dynamic treatment of these relations. For example, in the case of the isa relation, attribute inheritance will be processed only when required. If we say that *John isa human.*, John has his own characteristics and dynamically inherits of the attributes of human. In another context, if we state, within the context of to give, that offer and donate are synonyms, then, this relation will be kept preserved when lexical knowledge will be updated (in a way similar, but more principle-based, as integrity constraints). At a more global level, the overall coherence of a is-a system is managed at a high level of abstraction by the coherence checker of the constraint resolution mechanism associated to that active constraint.

So far, the general constraint management system is simply meta-interpreted, while constraint resolution mechanisms themselves have received more attention and have efficient resolution mechanisms. These constraints, of general purpose to NLP, have been used in real parsing and generation systems we have developed in our laboratory. The meta-interpretation level introduces a greater flexibility but some inefficiency. We want to improve the efficiency of this level without loosing too much of its flexibility by having constraints more directly managed within a Prolog system. We also want to specialize these constraints and to refine their resolution mechanisms.

The experiments we have conducted so far on active constraints show that there is a gain of a factor ranging from 3 to 5, depending on the complexity of the treatment. The more complex the treatment is, the higher is the gain.

3. An illustration of the use of world knowledge to solve ambiguities

We now give a few simple examples that illustrate and motivate the formalisms presented in section 1.

3.1 An example:

Let us consider the following sentence:

Walking back home, Mary spotted a heron.

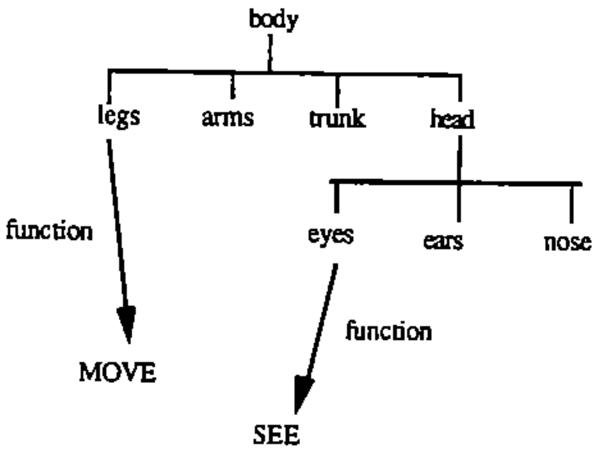
To deal with this sentence, we will have to consider features and lexical semantics relations. General common-sense knowledge knowledge will be eventually useful too, as shall be seen. The actions of walking and of spotting are related to two elements of the body of a human being. They are in fact two functions of two different parts, namely the legs and the eyes. Let us suppose that we have the following, simple, knowledge organization:

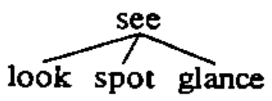
for the part_*of relation* (meronomy):
{ legs, trunk, arms, head } are parts_of body
{ ears, eyes, mouth } are parts_of head.

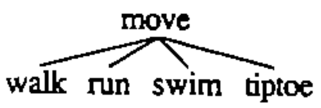
The attribute functionality of legs refers to the action of moving: LEG: functionality—> move.
the attribute functionality of the noun eyes refers to the action of seeing:
EYE: functionality --> see.

Now, if we consider the lexical semantics structures associated to move and see, we have, for example, the following taxonomies (relation is_a):
{ walk, run, swim, tiptoe) is_a move.
{ look, spot, glance } is_a see

These lexical semantics information can be represented as follows on a simple diagram:







Finally, consider the general rule:
- if the feature durative is associated to a verb in the gerundive form of an adjunct clause modifying a main clause with a verb in the preterit with feature punctual,
(so: if the feature durative is associated to *walk* (and to *move* more generally) and the feature punctual is associated to the verb *spot),*
- and if we assume that only the actions which are sisters, or which have sister parents, are incompatible (i.e. they cannot co-occur), then we can deduce that the actions under *see* are not incompatible with the actions under *move,* thus that walk and spot are compatible and can occur at the same time.

- then we can infer that *to spot is* a punctual action which takes place within a *walking* action,thus **walking temporally includes spotting.**

We could also have a more constraining inference rule describing world knowledge of the form:

walk(X, t0, tl) ==> see(X, t0, tl).

which can roughly be paraphrased by:

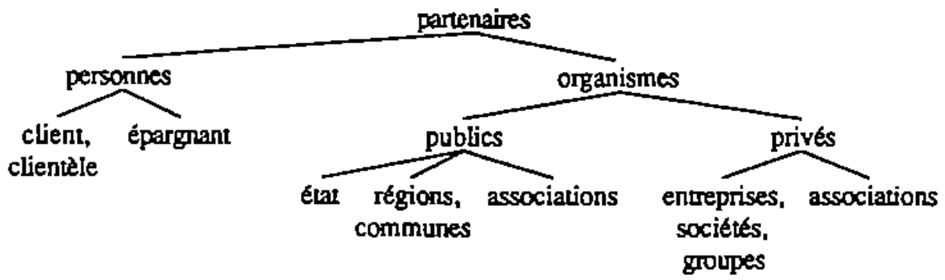
if X can walk on [t0, tl], then X can see (then spot) on [t0, tl].

This rule would save some processing of the lexical semantic. structures, but the reasoning used above without that rule is more general because it entirely based on a general reasoning procedure which uses the properties of the lexical semantics relations.

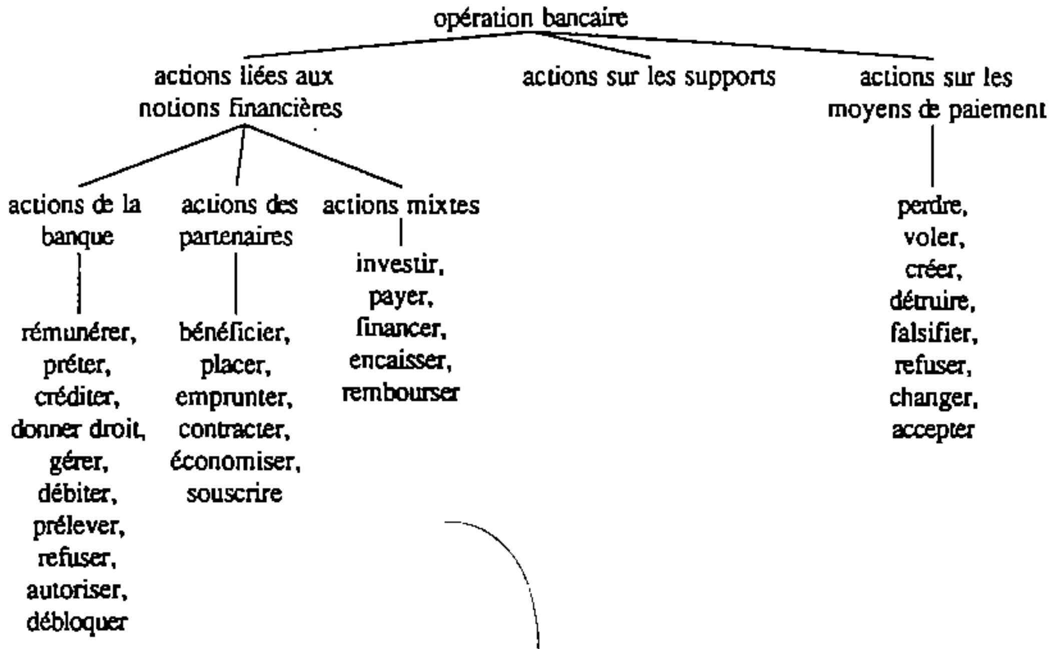
3.2. Modelling an application domain: the banking domain

We have chosen the domain of bank objects and loan operations since it involves a lot of various temporal relations, within a domain where operations are non-ambiguous and operate on precise objects like accounts.

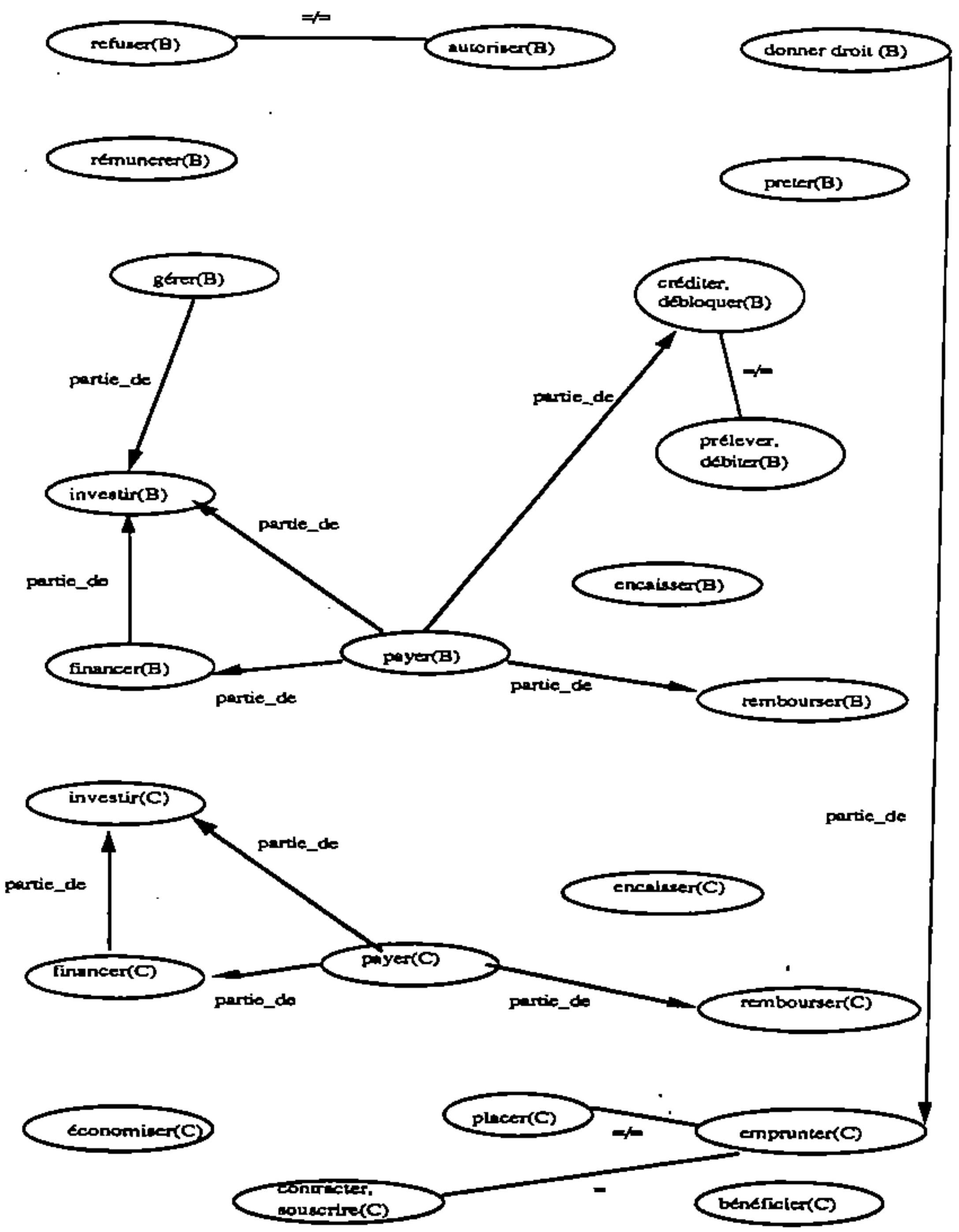
We now give a small diagram of some lexical semantics relations. This diagram has been established for French terms, it may, of course, be somewhat different for another language since the meanings of words do not necessary fully overlap from one language to another. Here is the partners diagram (relation is-a):



For banking operations, we have, for example:


opération bancaire
actions liées aux notions financières
actions sur les supports
actions sur les moyens de paiement
actions de la banque
actions des partenaires
actions mixtes
rémunérer, prêter, créditer, donner droit, gérer, débiter, prélever, refuser, autoriser, débloquer
bénéficier, placer, emprunter, contracter, économiser, souscrire
investir, payer, financer, encaisser, rembourser
perdre, voler, créer, détruire, falsifier, refuser, changer, accepter

We can now visualize the relations part ot, synonym and opposite on these actions:



There are several opposition relations, as for example (C= Customer, B= Bank):

complementaries:

demander(C): autoriser(B) / refuser(B)

or directional opposition: the direction being the movement of money

créditer(B) / débiter(B)
débloquerl(B) / prélever(B)

créditer(B) / prélever(B)

With a domain ontology composed of individuals, money, materials, abstract supports (like accounts) and time points, it is possible to define basic predicates and scenarios. For example, we have the following scenarios for the underlined word-senses:

somme(A: money), capital(A): A > O .

mouvement(Somme: A, Origine: I ou S, Destination: I ou S, moyen: S, Instant: T): appartenir à(I1: I, Somme, T1: T) H appartenir_à(I2: I, Somme, T2: T) H I1 =/= I2 HT2 > T1.

chèque(Ch: S): matériau(Ch, papier) H support_pour(Ch, Somme: A), H écrire(I1: I, Somme, Ch) .

Acknowledgements

Parts of this text have emerged from discussions and exchanges of documents with Annie Gal, Palmira Marrafa, James Pustejovsky and Jesús Vidal. We thank them for their contribution. This work has been supported by the French Ministry of research via the GDR-GRECA Communication Homme-Machine and by the Industries francophones de la langue.

References

Aït-Kaçi, H., Nasr, R., LOGIN: A Logic Programming Language with Built-in Inheritance, *Journal of Logic Programming*, vol. 3, pp 185-215,1986.

Anick, P., Pustejovsky, J., An Application of Lexical Semantics to Knowledge Acquisition from Corpora, *Coling 90*, Helsinky, 1990.

Barwise, J., Perry, J., *Situations and Atntudes, MIT* Press, 1985.

Boguraev, B., Briscoe, E., *Computational Lexicography for Natural Language Processing*, Wiley, London, 1989.

Briscoe, T., Copestake, A., Boguraev, B., Enjoy the Paper: Lexical Semantics via Lexicology, *Coling90*, Helsinky, 1990.

Colmerauer, A., An Introduction to Prolog III, *CACM* 33-7, 1990.

Jaffar, J., Lassez, J.L., Constraint Logic Programming, *Proc. 14th ACM Symposium on Principles of Programming Languages*, 1987.

Mukai, K., CIL: Complex Indeterminates Language, *Fifth Generation Computing journal*, 1985.

Pustejovsky, J., Current Issues in Computational Lexical Semantics, *ACL89*, European Chapter, Manchester, 1989.

Saint-Dizier, P., Processing Language with Types and Active Constraints, proc ACL91,Berlin.

Saint-Dizier, P., A Denotational Semantics for Synonyms and Opposites, in proc ACHIALLC conference, Tucson, AZ, 1991.