PARAMETRIZED ABSTRACT OBJECTS FOR LINGUISTIC INFORMATION PROCESSING

Helene Bestougeff, Gerard Ligozat
CNRS-Universite Paris VII
2,Place Jussieu 75005 PARIS FRANCE

ABSTRACT

Programming languages which have adequate primitives for linguistic information processing and a clear semantics at the formal computational level are now slowly emerging as a convergent effort from computer science, linguistics, and artificial intelligence. Our work on the processing of a special kind of linguistic information, namely temporal information ,has led us to advocate the use of a language with the following characteristic features:

- high level of abstraction;
- capacity for inference;
- modularity.

A high level of abstraction is needed to deal with complex linguistic notions which are not easily reducible to elementary data structures.

A capacity for inference is required, as most criteria or tests in linguistics make use of particular kinds of deductions, at different levels of the linguistic analysis.

As for modularity, a typical situation in linguistics has to do with a hierarchy of concepts or units, and the relations between those units at different levels.

This paper discusses the relevance of the choice of parametrized abstract objects as tools for linguistic information processing and exemplifies the use of such objects for temporal information.

INTRODUCTION

In computational linguistics, more than often there is a tendency to directly implement a model without really going through a specification step which would provide a correct abstraction of the implementation. This attitude has at least two drawbacks: Firstly, there is no formal way of comparing two models; this can lead to some pointless discussions between different approaches which, at an abstract level, can be shown to be equivalent; secondly, any extension or modification of the implemented model requires a different program instead of a mere adjustment at the abstract level which should facilitate the modular updating of the implementation and allow a formal comparison between the old and the new model.

Our work on tense and time has convinced us that, especially in linguistic domains where models are either loose or controversial, a systematic approach to linguistic information processing allowing compatible constructions at all levels is highly desirable. We therefore advocate the use of a language with the following features:

- high level of abstraction;
- capacity for inference;
- modularity.

This paper is in two parts. In the first part, we try to justify our choice of abstract parametrized objects as adequate tools for linguistic information processing; in the second part we exemplify our approach by giving a detailed account of the way we define and construct temporal objects.

PARAMETRIZED ABSTRACT OBJECTS

One of the basic difficulties in natural language processing arises from the fact that modularity is both a desirable and hardly attainable property of the systems. At first it seems quite reasonable to break the programs into manageable, modular sub-programs, especially as the linguistic data, at least at first approximation,lend themselves to a clear-cut classification in terms of morphology, syntax, semantics and pragmatics. Moreover, comparatively sophisticated techniques and methods are already available in each subfield.

Unfortunately, it has now become

commonplace knowledge that this strategy of developing separate modules for each sub-problem and integrating them smoothly is not satisfactory: the operations of parsing sentences, producing internal representations, reasoning about them, answering questions, generating text, and so on, are strongly interdependent. The degree, order and location of the interactions between different parts may vary significantly, according to individual situations.

A consequence of the realization of this fact has been the development of strongly integrated, usually procedural systems, where the individual sub-programs operate simultaneously on several deeply intricated linguistic levels . The price one has to pay for a relative success of this approach is in terms of understandability and generalization: many systems are strongly dependent on the particular type of problem they have been programmed to solve, and possible extensions or transpositions would require fundamental modifications.

What kind of software tools would allow at the same time modularity and multi-faceted, polymorphic and concurrent interactions between processes ?
Modularity and complex interactions are characteristic features of the <u>object oriented</u> paradigm .
Modularity is provided by the structuration in terms of objects. Complex interaction is a consequence of the distribution of the control between the different objects and of the (possibly multiple) inheritance facilities between hierarchically dependent objects. The possibility of using different points of views for the same objects is a consequence of this structuration.

This approach leads to focussing on the basic process of <u>abstraction</u>. In this context, abstraction is a process which, starting from a description of the data, yields an abstract specification. This involves three steps.
- define the relevant objects for the problem under study;
- define the possible functions and relations on the objects;
-give explicitly the constraints between the functions and relations.

The object-oriented approach is usually equated with the Smalltalk "vision" (Goldberg and Robson, 1983), while other views of objects are rejected as being irrelevant or even self contradictory.
Smalltalk objects can be characterized by the following properties:

- Each object is an instance of a class (a generic object).
- Each object has a local memory which can only be updated by functions (or procedures) local to the object.
- Objects are organized into a tree-like hierarchy implying tree-like inheritance.
- Communication between objects is organized through message passing.

However ,it has been argued that the object oriented approach can be fruitfully carried over into applicative language contexts (Steels ,1982) or into particular systems based on logic where the fundamental mechanism is data abstraction (Goguen et al. 1983).

As regards Smalltalk, it does not provide systematic facilities for defining abstract data types; all computations are highly dependent on side effects (assignment is systematically used in local operations) and there is no explicit typing.

We favour the approach exemplified by such languages as OBJ and their possible extensions (Goguen and Meseguer, 1984), as we think that they will allow ,in the long range, a more efficient programming style and make the systematic proof of programs possible.

The OBJ language is based upon data abstraction: an object is a type (i.e. a domain of values with functions accessing those values); objects are organized into a hierarchy (an acyclic graph) representing the dependencies among types. Computations are performed by using equational axioms as oriented rewrite rules.

Therefore , granted the availability of a theorem prover , the consistency of the specifications given in the abstract data type can be formally checked. Moreover , since the objects have a clear mathematical definition, all the techniques of abstract algebra are also available.

More generally, axioms could be given not only as equations , but also as formulas in a logical theory (such as first order predicate calculus or temporal logics) assuming those theories satisfy some given restrictions (Goguen and Burstall, 1984).

As we have no access yet to any version of OBJ, we have decided , in a first stage, to restrict the object structure to its free component, i.e. only the signatures, not the axioms are defined. Therefore, the computations have to be explicitely coded. As a consequence, no checking of consistency is possible for the time being.

The actual implementation is done in the ML language (Gordon et al., 1979). ML is a functional language which is fully higher-order. "It has a polymorphic type discipline which combines the flexibility of programming in a typeless language with the security of compile-time type checking". Moreover, one can define one's own types , which may be abstract and/or recursive.

To give a flavour of the ML programming style, consider a possible definition of the abstract recursive type of binary trees, with tip values of an arbitrary type (denoted by *) and non tip nodes of some other arbitrary type (denoted by **). (This exemple is taken from Gordon et al., 1979):

```
absrectype  (*,**) tree =
    * + ** # (*,**)tree # (*,**)tree

 with tiptree x = abs_tree (inl x)
 and  comptree  (y,t1,t2) =
          abs_tree  (inr (y,t1,t2))
 and istip t =isl(rep_tree t)
 and tipof t= outl(reptree t)
 and labelof t  = fst(outr (reptree t))
 and sonof t =snd(outr(reptree t))
```

This type is defined as recursive and abstract. The symbols "+" and "#" respectively, denote the two type constructors "disjoint sum" and "cartesian product". The functions "abs_tree" and "rep_tree" ,both of them of type (ty -> ty), are only available inside the definition of the abstract type "tree" : abs_tree maps the concrete representation of a tree unto its abstraction; rep_tree has the converse effect. Finally, isl, inl, inr, outl, outr are functions or predicates on the disjoint sum. They are defined as follows:

```
   isl: (* + **) -> bool
tests membership  of left summand;
   inl:  *  ->  (* + **)
injects  into  left summand;
   inr:   *  ->  (** + *)
injects into right  summand;
   outl:  (*  + **) -> *
projects  out  left summand;
   outr:  (*  + **) -> **
projects out right summand.
```

The signature of this type is the set of operators:

```
tiptree=-: * -> (**,*)tree
comptree=-:
* # (**,*)tree # (**,*)tree ->(*,**)tree
istip=-: (*,**)tree ->bool
tipof=-: (*,**)tree ->**
labelof=-:(*,**) tree -> *
sonof=-:
(*,**)tree ->((**,*)tree  # (**,*) tree))
```

The version of ML we use (INRIA ,1984) is written in Lisp with access to the lisp system. So our object environment is constructed as a collection of abstract data types. The hierarchy between types results from the combination and enrichment of more basic types. This hierarchy creates multiple inheritance relations between types. Some examples will be given in the context of temporal objects.
Clearly, the management of the object level must be done on top of ML . The explicit coding mixes Lisp and ML.

As we work in a functional environment there is no "local memory".However, this is , to our viewpoint, a minor drawback compared to the advantage of the abstraction facilities.
In a next stage, we intend to introduce the necessary axioms and perform the computations in a deductive style.

This approach can be used for the formal representation of natural language, or as a grammar formalism . In particular the syntactical and semantical analysis can be done in terms of objects. (De Boissieu and Forest , 1985).

PROCESSING TEMPORAL INFORMATION

Tense and time representation in natural languages is generally studied under one of the three main disciplines : logics, linguistics, and artificial intelligence. A brief overview of these different viewpoints is given in (Bestougeff an Ligozat, 1984).

The main problem is to choose the relevant objects in order to get an adequate abstraction. It must be strongly emphasized that we deny ourselves the right to assume any particular physical representation of time from the outset. The concrete properties result from the specifications.

The choice of the basic objects is somehow arbitrary, but it should nevertheless comply to the following rules :the objects must be

- close to linguistic intuition.
- general enough to be reusable as such in different contexts, or give rise to new objects by enrichment or inheritance.

The second point is required to avoid

ad-hoc and independant specifications. To achieve these goals it may be necessary to define primitive objects, which do not have any lingustic interpretation but are merely building blocks whose use enhances modularity .
In this case ,the lower level objects can be hidden to the user .

Keeping this in mind ,we can now proceed to the description of the linguistic motivations which are behind the construction of temporal objects . The idea is to give a systematic way of representing temporal information by defining abstract structures based upon the concepts and the hypotheses of a particular linguistic theory.

The linguistic theory we rely on is that of A. Culioli (Culioli, 1980), suitably adapted to computational purposes.

Temporal information can be informally characterized as information pertaining to the location and "shape" of the states and events described by natural language. In particular, this includes what is commonly referred to as aspect.

Furthermore, temporal information in natural language has both a descriptive and an operative structure: it describes and allows the users to make systematic inferences. Among these inferences are those concerned with the ordering of events, but such inferences are only part of a whole set of inferences on the factuality, the degree of completion, the type of occurrence, of the situations considered. In fact, it can be argued that the ordering relations are not necessarily of a primary nature.

Some examples will illustrate the kind of data and inferences we have in mind.

Consider the following simple sentences:

(1) John is ill.
(2) John repairs cars.
(3) John is repairing my car.
(4) John repaired my car.
(5) John has repaired my car.
(6) John was repairing my car.
(7) My car has been repaired.
(8) My car is repaired now.
(9) John was singing.
(10) John sang.
(11) John has been singing.
(12) Cats are smart.

We wish to account for some basic information imparted by the use of such sentences. For example:
- Sentence (2) does not imply (3), neither does sentence (3) imply sentence (2).
- Sentence (4) implies sentence (7), not (8).
- Sentence (5) implies sentence (8).
- Sentence (6) implies neither sentence (7) nor, a fortiori, (8), whereas sentence (9) implies sentences (10),(11).

The different uses of the simple present tense in (1) and (2) are related to a difference between the semantic types of the verbs to be ill and to repair. We will account for this difference by adapting a classification (essentially due to Vendler (1967)) into four semantic types (state, activity, accomplishement, achievement) . The usefulness of such a classification is further illustrated by comparing the behaviour of the verb to repair in sentences (6, 7, 8) with that of the verb to sing in (9, 10, 11).

The comparison between (4) and (6) in relation with (7) shows the necessity of suitably representing the difference between the simple and the progressive past, at least for verbs of the type to repair a car , which are classified as accomplishments.

To represent the difference between (4) (simple past) and (5) (present perfect), we have to express what makes (8), but not (4), derivable from (5). Reichenbach's system of temporal indexes (point of speech, point of event, point of reference) can be used to handle this phenomenon (Reichenbach, 1957). It provides a way of describing the notion of "present relevance", which is present in (5) , but not in (4).

The contrast between (1 , 2) and (12) points to another kind of distinction one has to make: (1) expresses a state, (2) a habit, which hold at the moment of speech. On the contrary, (12) states a general fact which is basically undetermined with respect to the moment of speech. Dependence on the time of speech is a fact of temporal deixis. We shall refer to it as enunciativity , (following A. Culioli) . By opposition situations such as (12) will be termed aoristic.

The preceding examples give some idea of the type of information that has to be represented. We have deliberately played down the purely sequential type of information, which is the only type of temporal information most systems are concerned with.

Moreover, the purely sequential type of information is mostly incomplete (this is stressed in particular by Smith (1978)). Consider the following examples:

(13) John saw his doctor this morning: he is ill.
(14) John saw his doctor this morning: now he is ill.

Contrasting (13) and (14) shows a potential indeterminacy in the relation between the two sentences. Smith (1978) claims that sentences like (1), where no explicit "reference time" is provided (e.g. by a time adverbial such as now) are temporally incomplete. We will be content at this point of our discussion with noting the need for a convenient notation for such a phenomenon.

TEMPORAL OBJECTS

We assume that temporal information in a text can be represented by proceeding in three steps of increasing difficulty:

- Tense in main clauses.
- Tense in subordinate clauses.
- Tense in texts.

This hierarchy follows that of Ejerhed and Janlert (1981).

To summarize the discussion of the previous paragraph, the following elements of temporal information have to be abstracted:

- temporal deixis (enunciative vs. aoristic situations) . Following Comrie (1976), we use the term "situation" as a generic term covering states, events or processes .
- inception and termination of a situation.
- information relative to the completion of the situation.
- local inferences on situations.
- mass/count properties of situations.

In our system, these elements are reconstructed from the following linguistic data:

- tenses in the finite forms of verbs.
- temporal specifiers (temporal adverbials).
-semantic types of situations (computed from the semantic type of the verbs à la Vendler, and the syntactic structure of the proposition).

At this point, most existing systems of representation make a choice, since a notion of duration has to be included in the model as well. Either one conceives of the basic elements as points, and the notion of an interval has to be introduced; or an interval is a basic element, and a second relation (of overlapping or inclusion) is introduced. In fact, in most existing models of time, the basic elements of time are conceived as elements or subsets of (a subset of) the real line (or some ramified structure built from it).

The choice of either "points" or "intervals" as basic elements leads to definite advantages and particular difficulties. The point model is basically simpler, but in some way harder to justify semantically. However, as shown by Kamp (1979) and Van Benthem (1980), both points of views are essentially equivalent.

Our claim in the matter follows the general philosophy of abstraction: Instead of the nature of the basic elements, consider their intended properties and combination rules for building derived elements. This combinatorial point of view is implicit, for example, in Allen's model (Allen 83), where a set of "intervals" is abstractly characterized by the relations holding between its elements. It can be shown (Bestougeff and Ligozat ,1984) that any set theoritic model of Allen's axioms is equivalent to (a subset) of the intervals (that is, couples of points) on a totally ordered set.

In our model,the basic elements are typed boundaries, with a (partial) order defined on them. As shown in (Bestougeff and Ligozat, 1984) an alternative way of considering the same abstraction would be in terms of "intervals", where an interval is a couple (b1, b2) of boundaries with b1 <b2 . In other words, the term "interval" has only the notions of a beginning and an end associated with it, and it is immaterial whether one or the other terminology is used. No topological properties are implied , only combinatorial properties (in terms of the types of boundaries) are retained in the abstraction. Boundary types are introduced in the model in order to represent aspectual properties of the data.

As an example, consider again sentences (1) to (12).

The state be ill in (1) holds upon an interval whose left and right boundaries are "closing" and "opening", respectively.This is a general situation for states in an enunciative setting.

The event John repair my car in (5), conversely, holds upon an interval with resp. "opening" and "closing" left and right boundaries. Consequently, the adjacent resulting state "my car is repaired" holds on an interval with a "closing" left boundary, as a state should. The combination in (5) of a verb of accomplishment with a "closing" right boundary insures that such a resulting state does indeed exist. This is to be contrasted with the situation in (6). There, the right boundary is an "opening one", which prevents the inference of a completed action from being made.

It seems that the introduction of such typed boundaries is enough to capture the intuition behind the use of topological intervals in systems representing tense and time. The approach chosen here prevents the overloading of the objects with unnecessary or undesirable properties, as is the case when a concrete model like the real line is adopted.

In terms of implemented objects, this corresponds to the definition of abstract intervals from typed boundaries and predicate information (the latter can be empty).
As an example , the explicit definition of an interval is as follows :

```
abstype intv =boundary # pred# boundary

 with make_intv (l1,l2,l3)=
                    abs_intv (l1,l2,l3)
 and left l =fst (rep_intv l)
 and right l = snd (snd (rep_intv l))
 and getp l =fst (snd (rep_intv l))
 and putl (b,i)=
    if fst (rep_intv i) =U
    then abs_intv (b,fst(snd(rep_intv)),
                  snd(snd (rep_intv i)))
    else i
 and putr(b,i)=
    if snd(snd(rep_intv i))= U
    then  abs_intv(fst(rep_intv i),
               fst(snd(rep_intv i)),b)
    else i
 and show_intv l =rep_intv l;;
```

The signature of this object is the set of typed operators:

```
make_intv=-:
    (boundary # pred#  boundary) -> intv
left=-:  intv -> boundary
right=-: intv -> boundary
getp =-: intv -> pred
putl =-: (boundary#intv) -> intv
putr =-: (boundary # intv ) -> intv
show_intv =-:
    intv  ->(intv + (intv  # nseq))
```

It seems intuitively satisfying to consider the stretch of time involved in a simple clause as totally ordered. The local inferences operate on this restricted scope.

To abstract this phenomenon we introduce interval sequences with constraints on the boundary types of adjacent intervals:

```
absrectype nseq= intv + int # nseq

force=-: intv -> nseq
ncons=-:(intv#nseq) -> nseq
make_tnseq=-:(intv # nseq) -> nseq
show_nseq=-:
   nseq  -> (intv + (intv #  nseq))
```

The central object in the model corresponds to a simple clause. It is called a polytyped string (or PTS). It is obtained from an interval sequence by adding the information about temporal indexes a la Reichenbach:

```
abstype pts =nseq #index

 make_pts=-:((nseq #index) -> pts)
 f1=-: pred -> pts
 f2=-: pred -> pts
 .
 .
 fn=-: pred -> pts
 rules=-:
 (status # tense # vendler # adverbial)
     -> pred -> pts)
 apply_rules =-:
   (pred # status) ->  pts
```

where the functional type "pred-> pts" denotes the set of functions which build PTS's from predicate information.
The predicate information is given through the " rules" where "status" is the information relative to the enunciative vs. aoristic status; "tense" denotes the morphological tense of the clause, "vendler" , the Vendler class (i.e. state, activity, accomplishement or achievement) computed from classe(s) assigned to verbs in the dictionary and the syntactical configurations ; finally "adverbial" corresponds to information attached to the time adverbials.

Up to this point , we have described the fundamentals of the system of representation. Of course, the actual construction of the representative temporal object for a text in a given language is highly language dependent. For instance, the present tense in French (which is the language we are working on) is not in a simple correspondance with the "corresponding" simple present in English. Consequently , referring to the objects described above, the functions "fi", and "rules" are quite specific to the structure of the language represented.

The temporal relations in longer units of discourse are comparatively much more loosely specified. Consider the following

example:
(15) Shakespeare is dead. John is ill. And I am not feeling well either.

Apart from considerations pertaining to real-world knowledge, no information is given about the relative order of the beginnings of the three situations considered. So the representation should allow for indeterminacy, either in listing all possible alternatives (this would be the case in Allen's model), or in leaving the order unspecified. This more economical solution is chosen here.

Compare (15) to the following:

(16) Mary got pregnant. She married John.
(17) Mary married John. She got pregnant.

Here, the order of discourse seems pertinent and should be represented.

More complex examples in this respect are:

(18) John was angry when Mary dropped the vase.
(19) Mary dropped the vase. John was angry.
(20) John was angry. Mary dropped the vase.
where (19) or alternatively (20) can be a paraphrase of (18).

The preceding discussion shows that the total ordering at the sentence level cannot in general be extended to larger units in a simple way. The eventual relations between different simple sentences are a result of a computation making use of the temporal structure of those sentences and the order of discourse.

This fact is captured as follows: The structure representing a text is constructed stepwise. At each step of the construction, the existing structure provides a context, in which the order of discourse, in particular, is represented. In technical terms, the corresponding object is called "temporal site". It is composed of a sequence of PTS's together with a set of relations on the boundaries of the constituent PTS's.So the next sentence to be examined, taken in isolation, is represented by a possibly incomplete structure (a polytyped string) with a total order on it, but with possible indeterminacies (for example, in the assignment of time indexes). This new structure is inserted into the old one, (already constructed temporal site) thereby creating new constraints resulting in the evaluation of some undetermined parameters in both structures.

Here again, the precise combination rules are language specific, as they depend on the semantic properties of the time relations in the language.

It is beyond the scope of this paper to give the rules used for French. However to give an indication of what the construction amounts to , consider the following english sentences:

(21) John was in love with Mary;
(22) John has built his house;
(23) John was building his house when I left for Rome.

The analysis of (21) yields:
tense : simple past;
status : enunciative (by default);
vendler :state;
adverbial :none.

Denoting by "p" the predication: John is in love with Mary, the structure of the representing PTS can be symbolized by the formula :

C1 p O2 ∅ O3 (S3 R2)

where the C's and O's denote closing and opening boundaries and S and R,points of speech and reference respectively . These are indexed by the order of occurrence of the corresponding boundaries. ∅ denotes a dummy predication.

Consider sentence (22) . Here
tense : present perfect;
status : enunciative (because of the present perfect tense);
vendler : accomplishment (perfect form);
adverbial : none.
The formula :

O1 p C2 reslt(p) O3 (S3 R3)

describes the associated PTS, where p is John builds his house and reslt(p) is a resulting state, obtained by local inference, which expresses : the house is built .

Finally, consider sentence (23). The associated temporal site can be symbolized by:

```
1: O1 p O2 ∅ O3   (S3,R2)
2: O1 q C2 ∅ O3   (S3,R2)

REL:  O1:1 < O1:2
      O2:1 >= O1:2
```

This temporal site contains two PTS's, with p = John builds his house and q= I leave for Rome .q is an achievement in Vendler's classification. The additional information concerns the ordering relations between the boundaries of the PTS's, numbered 1 and 2.

We have been mainly concerned with the representation of what we have termed enunciative situations (as opposed to aoristic ones). This is justified , as such situations play a central role in discourse. Concerning aoristic situations, similar representative structures are used, which are in fact more strictly constrained.

Habitual situations (e.g. sentence (2)) involve a particular treatment of the predicative component, but otherwise fit into the general scheme described above. From this point of view, they are no different from factual situations.

Dispositional sentences, on the other hand, cannot be discussed without entering the domain of modality. Although this may seem a serious limitation (especially for English, where modality is all pervasive), we leave it aside in the present consideration of tense and time.

The preceding discussion illustrates the use of linguistic inference at three distinct levels:

a) At the simple sentence level, building a representation involves a first type of inference, which makes use of morpho-syntactic and lexico-semantic information .

b) At the next higher level, as illustrated by examples (15-20), another type of inference is used to specify and build the corresponding structures (temporal sites).

c) Still another kind of linguistic inference should account for the possible derivabilities or paraphrasings mentioned à propos examples (1-12). Its formalization should make it possible to describe this inference which, starting from a given temporal site allows to deduce new sites from it.

Whereas the first two types of inference are constitutive of the derivation of temporal structures and are central to our activity, the last type has still to be defined and examined in a systematic way i.e. defined explicitely as derivation rules. In this context, the facilities for self-reference and structural inference in the software environment are of primary relevance.

REFERENCES

Allen,J.A.1983 Maintaining knowledge about temporal intervals.Comm.ACM 26 pp.832-843.

Bestougeff, H. and Ligozat, G. 1984a Processing Tense Information in French Utterances.Proceedings of the 6th European Conference on Artificial Intelligence. Pisa, Italy:209-212

Bestougeff, H. and Ligozat, G. 1984b L'inférence temporelle en situation orientée par l'action.Proceedings of the CNRS Greco Seminar.Nancy, France.

Bestougeff, H. and Ligozat, G. 1984c Temporal Intervals Revisited. CNRS Research Report, Paris 7.

Culioli, A. 1980 Valeurs aspectuelles et opérations énonciatives.In David,J.and Martin, R., Eds., La notion d'aspect. Klincksieck, Paris: 181-193.

Comrie, B. 1976 Aspect. Cambridge University Press .

De Boissieu, A. and Forest, F. 1985 Analyse linguistique en termes d'objets.CNRS Research report,Paris 7.

Ejerhed, E.I. and Janlert, L.E. 1981 Representing time in natural language processing. In Proceedings of the Workshop on Models of Dialogue . Linkoping UIT.

Goguen,J.A. Meseguer, J. Plaisted, D. 1983 Programming with Parametrized Abstract Objects in OBJ. in Theory and Practice of Software Technology. D. Ferrari, M.Bolognani and Goguen Eds. North Holland.

Goguen, J.A. and Burstall, R.M. 1984 Introducing Institutions. in Proceedings , Logics of Programming Workshop. E.Clarke and D.Kozen Eds. Lecture Notes in Computer Science, Vol 164 Springer Verlag.

Goguen J.A. and Meseguer J. 1984 Equality,Types,Modules and (why not?) Generics for Logic Programming. J. Logic Programming 1984:2.

Goldberg, A. and Robson, D. 1983 Smalltalk-80.The language and its implementation. Addison Wesley Publishing Company.

Gordon,M.J., Milner, A.J. Wadsworth,L.P. 1979 Edinburg LCF. Lecture Notes in Computer Science Vol 78 . Springer

INRIA. 1984 The ML Handbook . Version 5.1. Report of Institut National de Recherche en Informatique et Automatique. France.

Kamp, H. 1979 Events, Instants, and Temporal Reference.In Egli,U. and Von Stechow, A., Eds. Semantics from a multiple point of view. De Gruyther, Berlin: 376-417.

Reichenbach, H. 1957 The Philosophy of Space and Time.New York.

Smith, C. 1978 The syntax and interpretation of temporal expressions in English.Linguistics and Philosophy 2: 44-99.

Steels, L. 1983 ORBIT : An Applicative View of Object Programming. in Integrated Interactive Computing Systems .P.Degano, E. Sandewall Eds. North Holland.

Van Benthem, J. 1980 Points and periods. In Rohrer, C., Eds., Time, Tense, and Quantifiers, Niemeyer, Tuebingen:39-57.

Vendler, Z. 1967 Linguistics in Philosophy. Cornell University Press, Ithaca.