A PERSPECTIVE FOR RESEARCH ON CONCEPTUAL MODELLING

John Mylopoulos

Department of Computer Science
University of Toronto
Toronto, Ontario

1. INTRODUCTION

This position paper is intended to provide a perspective for research on conceptual modelling carried out over the past five years at the University of Toronto and to draw some conclusions from the experiences we have accumulated. "Conceptual modelling" here refers to the activity of constructing abstract models of knowledge about some world and is synonymous with the terms "knowledge representation" and "semantic data model" as they have been used in AI and Databases respectively. Much of the research on the subject has focused on the development of descriptive tools for the description of such models. Less attention has been paid, so far, on methodologies for building such models.

The basic premise that has guided much of our research is that useful descriptive tools for conceptual modelling can be developed within a framework that is characterized by the following features:

a) Is object-oriented, i.e. its atomic units are objects intended to represent entities and they can be created, have properties throughout their lifetime and can eventually be destroyed.

b) Uses procedural attachment, in the sense of [Winograd 75], to define procedurally the "meaning" of objects.

c) Pays special attention to the organization of all units constituting a model along different dimensions defined in terms of organizational principles such as aggregation, generalization, classification (as discussed in section 2) or a context dimension ([Hendrix 75]).

The premise is being tested with the design of PSN[†], a knowledge representation language, and TAXIS[††], a semantic data model. PSN treats traditional semantic network concepts within a procedural framework ([Levesque 76,77,79], [Schneider 78,79], [Kramer 80], [Lesperance 80]). TAXIS, on the other hand, is a programming language for the design of interactive information systems (IISs) such as airline reservations and credit card verification ([Mylopoulos 78,80a,80b], [Wong 80], [Barron 80]). The position paper outlines the most important similarities and differences between the two formalisms, examines the influence each one had on the other and describes some open questions that are currently under investigation.

[†] PSN: Procedural Semantic Networks.

2. AN OVERVIEW OF PSN

The PSN framework has its origins in semantic networks (see [Brachman 79] for a survey) and Abrial's work on data semantics ([Abrial 74]). As described in [Levesque 77,79], a model ("knowledge base") is a collection of *tokens* and (binary) *links* between them. Every token must be an instance of a *class* and every link an instance of some *relation*. The "meaning" of each class is defined through four procedures that specify respectively how to *insert*, *remove* and *fetch* instances of the class and how to *test* whether an object is an instance of the class. PSN departs from Abrial's proposal with the introduction of three organizational principles which are used to structure a model:

a) *Aggregation*: an important aspect of the definition of a class is its internal structure given in terms of *slots* which are bound to particular values for each instance of the class. For example, the class PERSON may have slots "name", "address" and "phone#" and this means that with each particular person token, say john-smith, we can associate a particular name, address and phone# by binding it to the corresponding slot. Each slot has a *type*, another class, and *constraints* that limit the range of objects that can be bound to it. A slot can also have a *default* that will be used when no value is provided for the slot during the creation of an instance. If class A has slots named $s_1, s_2, \ldots, s_n$ with types the classes $B_1, B_2, \ldots, B_n$, we say that $B_1, B_2, \ldots, B_n$ are the *parts* of A or that the special

[††] TAXIS: from ταξις, Greek noun meaning order as in "law and order" or class as in "social class", "university class", etc.

relation PART-OF holds between B_i and A, $1 \leq i \leq n$. PART-OF defines the first organizational principle of PSN as it allows certain objects, the instances of class A, to be viewed as *aggregations* of others, namely instances of the classes $B_1, B_2, \ldots, B_n$.

b) *Generalization*: defined through the special partial order relation IS-A which organizes classes into a hierarchy with respect to their generality/specificity. Thus the class PERSON is a generalization of the classes STUDENT, EMPLOYEE and is placed higher up along the generalization dimension. If class A is a generalization of class B then every instance of B is also an instance of A and every slot of A has a corresponding slot of B with the same name. *All* classes, including relations which are treated as classes, have a coordinate with respect to the generalization dimension.

c) *Classification*: defined through the INSTANCE-OF relation that holds between an object and each class it is an instance of. *All* objects, i.e. tokens, links and classes, must be instances of classes. This introduces the notion of a metaclass, a class whose instances are classes, and allows the definition of a third dimension along which a model can be structured.

Uniformity of representation has been a basic design principle for PSN throughout its development. Viewing relations as classes and classes as objects which are themselves instances of other classes are two particular uses of uniformity as a design criterion. A third, and perhaps more interesting, use involves the treatment of procedures in PSN. Procedures are viewed as classes whose execution instances are special tokens related to the corresponding procedures through the INSTANCE-OF relation. The body and the parameters of a procedure are defined through its class structure and procedures are organized into an IS-A hierarchy†, since they are classes. This treatment of procedures has two advantages over others that view procedures as executable atomic units. Firstly, it adds greatly to the perspicuity of a model where everything is represented and organized according to the same rules. Secondly, and in a sense more importantly, it encourages model design which proceeds from the representation of more general concepts (procedural and non-procedural alike) to the representation of more specialized ones.

Another interesting feature of PSN which is also heavily dependent on the uniformity of the formalism, is the degree of "natural" self-description that is possible within the formalism. [Levesque 77] includes a complete description of an early version of PSN in PSN which is based on a small set of primitive notions.

† The notion of IS-A hierarchies for procedures evolved out of earlier work done for the TORUS project [Mylopoulos 76] where IS-A hierarchies of events served as a basic organizational tool for the knowledge base of a natural language understanding system. The notion of a SIMULA subclass [Dahl 72] is also relevant, with the major difference between the two notions being that unlike a SIMULA hierarchy of classes, an IS-A hierarchy of procedures imposes strict semantic constraints on procedures that may be introduced at the bottom of the hierarchy [Mylopoulos 80a].

Some representational issues were not treated satisfactorily, if at all, in the original version of PSN [Levesque 77,79]. Among them we note *descriptions* which may or may not refer to any entity in the world that is being modelled (e.g. "a man who kissed Mary") and *indefinite objects*, i.e. objects whose existence has been confirmed but whose identity remains unknown. Another difficulty arises from the strict rather than default inheritance rules associated with the IS-A relation. According to these rules, if a class A has a certain slot s then every specialization of A has to have an s slot perhaps in a more restricted but never in a contradictory form. This means that classical representational problems such as representing the fact that penguins are birds but (unlike birds) they don't fly, could not be represented in an obvious way in PSN.

3. FROM PSN TO TAXIS: AN ADAPTIVE APPROACH

TAXIS was conceived as a potential solution to an existing problem: the lack of software facilities for the design of IISs. Much of its development was guided by our convinction that traditional software development methodologies which stress decomposition as a design paradigm (e.g. [Wirth 71]) are primarily applicable to situations which require the representation of a *few complex* concepts/problems and they do not address themselves to modelling situations which involve *large* amounts of *simple* detail, such as those encountered during the design of an IIS. This observation led us to the design of a programming language which offers the organizational principles of PSN but emphasizes generalization, which appears to be a major abstraction facility humans use to cope with large amounts of detail, while de-emphasizing aggregation and classification. Particular points of major difference between the two formalisms are listed below:

a) TAXIS limits the classification dimension to 3 levels which include respectively tokens, classes and metaclasses. This was meant to be a simplification of the conceptual framework used for TAXIS over that of PSN.

b) TAXIS restricts the kinds of information that can be attached to slots to type and default information. Other constraints on the values that can be bound to slots have to be represented procedurally.

c) In TAXIS two different times are recognized in a program's lifetime: program creation time and run time. Classes can only be created at program creation time. This is a clear concession to efficiency at the expense of flexibility.

d) Although the ability to associate procedures with a class is retained in TAXIS, it is modified so that procedures can define arbitrary operations on instances of a class they are associated with.

e) Since run time exceptions play an important role during the life of an IIS, much attention was paid to exceptions and exception handling. As with all other constructs in TAXIS, exceptions are treated as classes thus maintaining the uniformity of the underlying conceptual framework of the language and keeping it consistent with PSN. Moreover,

whereas in PSN a procedure had exactly four associated expressions which define its body as consisting of a precondition, an action, a postcondition and a complaint, in TAXIS a procedure can have an arbitrary number of associated expressions each of which belongs to one of four categories. This modification of the PSN framework was needed in order to allow a more flexible exception handling mechanism in TAXIS.

4. CLOSING THE FEEDBACK LOOP

The development of TAXIS has, in turn, had its influence on recent extensions to PSN. In particular, the treatment of procedures in TAXIS is adapted to PSN in [Kramer 80]. The adaptation includes the introduction of the notion of *metastructure* which allows a metaclass to limit the allowable structures of its (class) instances in the same way the structure of a class limits the allowable aggregations that can be used for its instances.

Another extension of PSN described in [Lesperance 80] involves the introduction of a mechanism for handling both *static exceptions*, i.e. having an "instance" of a class whose components don't satisfy the constraints specified in the structure of the class, and *dynamic exceptions* which arise during the execution of a procedure. A separate aspect of this work involves the development of a *mapping mechanism* between classes which makes it possible to define a class as being similar to another. For example, the class PENGUIN can be defined in terms of this mechanism as being similar to the class BIRD except that the part of the structure of BIRD which specifies that birds fly isn't inherited by PENGUIN.

A third extension of PSN, not influenced by TAXIS however, is described in [Schneider 79], [Lesperance 80] and concerns the notion of *contexts* (*partitions* in [Hendrix 75]) which is useful for modelling multiple views (e.g. reality vs a fairy tale world view), hypothetical worlds, belief spaces and time frames. Contexts can be thought as means of grouping together objects so that for any given context an object is either present (visible) or absent. Moreover, contexts are organized into a tree-structured *context hierarchy* with contexts near its top representing global views whose objects are visible in contexts lower down unless explicitly deleted from those contexts. Once this additional dimension is added to a modelling framework it is necessary to treat objects from a relativistic point of view since john-smith in the reality context can be a very different object from john-smith in the fairy tale world context. One of the issues discussed in [Schneider 79], [Lesperance 80] concerns the existence of invariants across contexts. Another the treatment of contexts as objects which requires that they be instances of classes, have properties and be visible within other contexts or even themselves.

5. FUTURE RESEARCH DIRECTIONS

We will mention two research questions which are of great interest to us at this time. The first concerns the introduction of (logical) assertions into the framework outlined in section 1. Assertions can be very useful for defining *invariants* of a class, i.e. conditions that must be satisfied at all times by their instances, and *preconditions/postconditions* of procedures. To introduce them, however, in PSN or TAXIS while maintaining the uniform framework used so far means that assertions too should be treated as objects that are instances of (assertion) classes and can be talked about in other assertions (see also [Borgida 80]).

The research carried out so far has concentrated on representing knowledge about a slice of reality. A major consideration during the design of IISs involves the user interfaces through which an IIS will communicate with its users. Some of the questions that have to be dealt with in designing a user interface are -

a) *Linguistic:* What are the allowable forms of messages to be passed back and forth between the system and its users or even between different components of the system (graphic, restricted natural language, some other special form,...)?

b) *Pragmatic:* What types of dialogues will be supported by the user interface? Should it support, for example, clarification or explanation dialogues such as those supported by RENDEZVOUS [Codd 78]?†

As with assertions, the problem is not simply to add facilities so that linguistic and/or pragmatic aspects of a system's design can be dealt with, but rather to provide these facilities within the PSN/TAXIS framework so that the designer of a PSN/TAXIS model is presented with a single conceptual framework within which he constructs his model.

6. CONCLUSIONS

The one major difference between TAXIS and PSN that we consider essential is the program creation/run time distinction of TAXIS which limits the kinds of things that can be done at run time to a TAXIS program but opens the door to gains in efficiency. This concern for efficiency at the expense of flexibility is also reflected in the implementation of the two formalisms. The implementation of PSN, described in [Kramer 80] involves an interactive LISP-based system for the creation and manipulation of a PSN knowledge base. The implementation of TAXIS (under design) involves the development of an interactive system for the creation of a TAXIS program and then its translation into a PASCAL R program [Schmidt 77] which is the final product of the design process.

In all other respects the representational issues of knowledge representation formalisms were found to have their counterparts in semantic data model design and vice versa.

7. ACKNOWLEDGEMENTS

The following colleagues contributed in a major way to the research described in this paper:

† Barron's work [Barron 80] which extends TAXIS to provide facilities for the definition of dialogues to be supported by the IIS under design is a step along that direction.

John Barron, Philip Bernstein (at Harvard University), Alex Borgida, Sol Greenspan, Bryan Kramer, Yves Lesperance, Hector Levesque, Peter Schneider and Harry Wong (at the Lawrence Berkeley Laboratories). The research was carried out in the Department of Computer Science of the University of Toronto with partial support from the Division of Applied Sciences of Harvard University.

8. REFERENCES

1. [Abrial 74]
Abrial, J.R., "Data Semantics" in Klimbie, J.W. and Koffeman, K.L. (eds.) *Data Management Systems*, North-Holland, 1-60, 1974.

2. [Barron 80]
Barron, J., *Dialogue Organization and Structure for Interactive Information Systems*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto; also CSRG TR-108, 1980.

3. [Borgida 80]
Borgida, A. and Greenspan, S., "Data and Activities: Utilizing Hierarchies of Classes" (these proceedings).

4. [Brachman 79]
Brachman, R., "On the Epistemological Status of Semantic Nets" in Findler, N.V. (ed.) *Associative Networks*, Academic Press, 1979.

5. [Codd 78]
Codd, E.F. et al., "RENDEZVOUS Version 1: An Experimental English Language Query Translation System for Casual Users of Relational Databases", IBM Research Report RJ 2144, 1978.

6. [Dahl 72]
Dahl, O.J. and Hoare, C.A.R., "Hierarchical Program Structures" in *Structured Programming*, O.J. Dahl, E. Dijkstra and C.A.R. Hoare (eds.), Academic Press, 1972.

7. [Hendrix 75]
Hendrix, G., "Extending the Utility of Semantic Networks through Partitioning", Proceedings IJCAI-75, Tbilisi USSR, 1975.

8. [Kramer 80]
Kramer, B., *The Representation of Programs in the Procedural Semantic Network Formalisms*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto; also DCS TR-139, 1980.

9. [Lesperance 80]
Lesperance, Y., Kramer, B., Schneider, P., "Topics in PSN-II: Exception Conditions in PSN; Representing Programs in PSN: Contexts in PSN", CSRG TR-112, Computer Systems Research Group, Univ. of Toronto, 1980.

10. [Levesque 76]
Levesque, H., Mylopoulos, J., McCalla, G., Melli, L., Tsotsos, J., "A Formalism for Modelling", Proceedings First CSCSI Conference, Vancouver, Canada, 1976.

11. [Levesque 77]
Levesque, H., *A Procedural Approach to Semantic Networks*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto; also DCS TR-105, 1977.

12. [Levesque 79]
Levesque, H. and Mylopoulos, J., "A Procedural Semantics for Semantic Networks" in Findler, N.V. (ed.) *Associative Networks*, Academic Press, 1979.

13. [Mylopoulos 76]
Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., and Wong, H.K.T., "TORUS: A Step Towards Bridging the Gap Between Databases and the Casual User", Information Systems, Sept. 1976.

14. [Mylopoulos 78]
Mylopoulos, J., Bernstein, P.A., and Wong, H.K.T., "A Preliminary Specification of TAXIS: A Language for Interactive Information System Design", CCA-78-02, Computer Corporation of America, 1978.

15. [Mylopoulos 80a]
Mylopoulos, J., Bernstein, P.A., and Wong, H.K.T., "A Language Facility for the Design of Interactive Database Intensive Applications", Transactions on Database Systems, vol.5, no.2, 1980; also presented at SIGMOD Conference, 1978.

16. [Mylopoulos 80b]
Mylopoulos, J., and Wong, H.K.T., 'Some Features of the TAXIS Data Model", Proceedings VLDB-6, Montreal, Canada, 1980.

17. [Schmidt 77]
Schmidt, J., "Some High Level Language Constructs for Data of Type Relation", Transactions on Database Systems, vol.2, no.3, 1977.

18. [Schneider 78]
Schneider, P., *Organization of Knowledge in a Procedural Semantic Network Formalism*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto; also DCS TR-115, 1978.

19. [Schneider 79]
Schneider, P., "A Formal Definition of Contexts in the Procedural Semantic Network Formalism", AI Memo 79-5, 1979.

20. [Winograd 75]
Winograd, T., "Frame Representations and the Declarative-Procedural Controversy" in Bobrow, D., Collins, A. (eds.) *Representation and Understanding*, Academic Press, 1975.

21. [Wirth 79]
Wirth, N., "Program Development by-Stepwise Refinement", Comm. ACM, vol.14, no.4, 1971.

22. [Wong 80]
Wong, H.K.T., *Design and Verification of Interactive Information Systems*, Ph.D. thesis, Dept. of Computer Science, Univ. of Toronto, 1980.