The Role of Time in Information Processing: A Survey

by

A. Bolour, University of California, San Francisco
T.L. Anderson, Burroughs Corporation
L.J. Dekeyser, Katholieke Universiteit Leuven
H.K.T. Wong, Lawrence Berkeley Laboratory[1]

Introduction

Numerous researchers in a handful of disciplines have been concerned, in recent years, with the special role (or roles) that time seems to play in information processing. Designers of computerized information systems have had to deal with the fact that when an information item becomes outdated, it need not be forgotten. Researchers in artificial intelligence have pointed to the need for a realistic world model to include representations not only for snapshot descriptions of the real world, but also for histories, or the evolution of such descriptions over time. Many logicians have regarded classical logic as an awkward tool for capturing the relationships and the meaning of statements involving temporal reference, and have proposed special "temporal logics" for this purpose. Finally, the analysis of tensed statements in natural language is a principal concern of researchers in linguistics.

The present survey concentrates on the first two, and to a lesser extent, on the third of these disciplines. We are concerned, in fact, with the role(s) of time in computerized information systems. But we believe that ideas from all of these fields, and perhaps from others, for example, philosophy, can find use in understanding how to deal with time in computerized information processing. Unfortunately, many workers on this topic have found it difficult to become aware of and familiar with the works of others. We offer this survey as a first step toward letting researchers on time know about each other's work, and hope that it will lead to an increased communication of ideas on information processing and time.

Of some seventy references included in the survey, about half are individually reviewed. Most of the remaining works would also have been reviewed but for a lack of time. We thought it important to have a timely presentation of these reviews, and hope that it will be possible for us or others to augment this survey at a later time. What is reviewed, however, represents a broad cross section of works in the first three disciplines mentioned above. In view of the remarks of the previous paragraph, we would not be surprised to learn that we have missed some important work in the area altogether. We would appreciate hearing about such work.

Nor is this survey intended to be exhaustive of the ideas within those works reviewed. In many cases, only a representative sample of the ideas in a work are presented. Also, in the interest of brevity and readability, we found it necessary on number of occasions to present the concepts discussed in their simplest forms, and to change some of the notation used in the works.

The items in this survey appear in chronological order, of course! An alphabetic cross reference by first author is provided at the end. Ironically, the reference processing program used to sort the references in chronological order ignored the month (and day) of publication, and some of the sorting had to be done by hand! But then there was the problem of sorting time points of different granularity: the month/year of publication for most articles, and the year of publication for books! Evidently, the problems of processing imprecise timing information addressed in many of the papers to follow are of more than just academic interest.

Bibliography and Reviews

1. W.E. Bull, "Time, tense, and the verb," University of California Publications in Linguistics Vol. 19(1960).

2. B. Langefors, "Theoretical analysis of information systems," Studentlitteratur, Auerbach, Lund, Sweden (1966).

See Sundgren [1975].

1. The work of A. Bolour was supported by NIH grant [LM03363] from the National Library of Medicine; the work of L.J. Dekeyser was supported by PHS International Research Fellowship [3 F05 TW03003-01S1]; the work of H.K.T. Wong was supported by the The Department of Energy contract [W-7405-ENG-48] from the Office of Energy Research.

Authors' present addresses: A. Bolour, Section on Medical Information Science, Room A-16, University of California, San Francisco, CA 94143; T.L. Anderson, Burroughs Corporation, 12201 Technology Blvd., Austin, Texas 78759; L.J. Dekeyser, Medische Informa., St. Rafael A.Z., Katholieke Universiteit Leuven, B-3000 Leuven, Belgium; H.K.T. Wong, Computer Science and Mathematics Dept., Room 3238, Building 50B, Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720.

3. R. Mattison, "A formal system for the logical analysis of temporal relationships between intervals of time," Memo. RM-5279-PR, Rand Corporation, Santa Monica, CA (April 1967).

4. A. Prior, Past, present, future, Oxford University Press (1967).

Arthur Prior is considered the patriarch of modern temporal logic. This book is the crowning achievement of his career, and is generally recognized as the bible of temporal logic. But we found the book difficult to read, and decided instead to review the more readable book by Rescher and Urquehardt, which approaches temporal logic from a slightly different viewpoint. A very readable book on temporal logic based on the Prior formalism is McArthur [1976].

5. R. Mattison, "An introduction to the model theory of first-order predicate logic and a related temporal logic," Memo. RM-5580-1-PR, Rand Corporation, Santa Monica, CA (June 1969).

A temporal logic is defined in which variables and terms are of two kinds: "interval" variables and terms designating time intervals, and "individual" variables and terms designating other entities. The logic is an extension of predicate logic in which there is a special binary predicate symbol, "<", designating "before", and two special unary function symbols, "max", and "min", designating functions that return the beginning and the end of an interval. Temporal terms are expressions involving interval variables and the functions "min" and "max". The language permits expressions of the form $P(t_1,\ldots,t_n)[T]$, and $f(t_1,\ldots,t_n)[T]$, where P and f are (n+1)-place predicate and function symbols, $t_1, \ldots, t_n$ are individual terms, and T is an interval term.

The properties of "<", "max", and "min" are defined in a set of "temporal axioms", which are added to the usual axioms of first order predicate logic with equality. The temporal axioms define a totally ordered and dense time, without a first or last element. They also include facts about "max" and "min", such as, "if a<b, where a and b are interval terms, then max(a)<min(b)".

The model theory of this logic is developed by using a special domain, J, for times, in addition to domains for designating individuals at particular points in time. The domain, J, should satisfy the temporal axioms. An interval variable is then interpreted as an interval (connected subset) of J. An individual variable or a one-place function symbol is interpreted as a function from some interval of J into a domain. Thus, a variable, "x", or a function, f()[T], when interpreted as, say, Abraham Lincoln, will be assigned a function that assigns to each time in Lincoln's life a domain object designating Abraham Lincoln at that time. The value of the function outside this time interval is undefined.

In general, function symbols of n+1 arguments are interpreted as functions from the Cartesian product of n domains and a time interval into a domain. Because individual terms can be interpreted as partial functions, a major complication arises in the interpretation of functions and predicates of more than one arguments. Consider, for example, a function f(t)[T]. Note that the individual term t would be assigned a function, say F_t, from some interval of J to a domain, and the interval term T would be assigned some interval of J. The problem is that the domain of definition of F_t may not cover the interval assigned to T. One solution is to say that the interpretation is undefined outside the domain of definition of F_t.

But Mattison provides another solution that can sometimes be used to extend the interpretation of the individual term t for this purpose. The idea is to assume that if the term t and some one place function, say, f'()[T], agree in their interpretation over the interval of definition of F_t, then they designate the same individual. So if the interpretation of f'()[T] is defined outside the interval of definition of F_t, then t is assumed to have the same value as the interpretation of f' outside this interval. For example, if the interpretation of a term, t, is defined only for the time period 1965, and is the constant function having the value "Lyndon Johnson" over that time period, and if we want to evaluate f(t)[T], where f is interpreted as the "wifeof" function, and T is interpreted as 1967, we try to find a one place function symbol, e.g., LBJ()[T], whose interpretation agrees with t over 1965, but which is also defined for 1967. Then we use the values of LBJ for 1967 to interpret f(t)[T]. Naturally, if this process is to define a unique interpretation, then whenever two functions, say, LBJ1()[T], and LBJ2()[T] are interpreted to have the same values over some interval, their interpretations must have the same values over all intervals. And this is a restriction that is imposed by Mattison on the interpretations of one-place function symbols. A natural question here whether a simpler semantics could be defined that captures the gist of these ideas.

6. D. Wunderlich, Tempus und Zeitreferenz im Deutschen, Max Hueber Verlag, Munich (1970).

7. N. Findler and D. Chen, "On the problems of time retrieval, temporal relations, causality, and coexistence," Proceedings of the Second International Joint Conference on Artificial Intelligence, Imperial College, (1-3 September 1971).

A working system for the storage and retrieval of temporal information is described. The language SLIP (a list processing language) is used for the user interface. A base system called The Associative Memory Parallel Processing Language, which provides processing capabilities for binary relations, was used for the storage and retrieval of temporal data. The principal application alluded to is sorting out scientific data in the process of establishing cause and effect relationships.

The specific temporal notions discussed are as follows. Each event has a start time, a finish time, and a duration. Any of these may be unknown. Two types of events are distinguished, instantaneous events, and extended events. The usual binary timing relations are defined for events. In addition, chains of events (either related by strict "before" relations, or by overlapping "before" relations) are defined. Along the same lines, sets of equitemporal events, or approximately equitemporal events are defined. (It is unclear how the approximate relations are processed.) A quantized time scale is used. Time points can be identified in one of two modes, dates, and simple integers, which are determined once and for all in each application. The start and finish times of events can be related by timing relations to the times on these time scales. Also, the interval between two events can be specified in terms of the units of time in each scale.

Some queries that can be processed by this system are: simple selection based on binary temporal relations; "time aggregate" queries involving "earliest", "longest", etc.; "chain" queries requiring chains of events leading up to another event; "generalized coexistence" queries, selecting an event based on its coexistence, and non-coexistence with other specified events.

8. J. McCawley, "Tense and time reference in English," in Studies in Linguistic Semantics, ed. T. Langendoen, Holt Reinhardt & Winston (1971).

9. N. Rescher and A. Urquhart, Temporal Logic, Springer Verlag, New York (1971).

This book is one of the classic texts of temporal logic. The first half of the book is devoted to the study of various systems of temporal logic that impose different degrees of structure on the relation "before". We shall concentrate on this half. The second half is a survey of interesting topics in temporal logic. A sampling of these topics will be sketched toward the end of this review.

SYSTEMS OF TEMPORAL LOGIC

Consider a statement that describes a situation or a state of affairs. There are two ways in which a time can be associated with such a statement. First, such a statement asserts that a particular condition exists at a particular point in time. We call this the time of reference of a statement. Second, the statement must be asserted or uttered by someone at some point in time. We call this time the time of assertion of the statement. Often the time of reference depends on the time of assertion, as in "it is now raining". Statements such as these are known as temporally indefinite statements. They use what are called "pseudo-dates", such as "now", "yesterday", and "tomorrow". By contrast, temporally definite statements have an explicit time of reference and use actual dates, for example, "it rained on January 18, 1981". In general, systems of temporal logic take as their basic building blocks (i.e., atomic propositions) temporally indefinite statements that implicitly refer to "now", and assume that such propositions can be either true or false at each point in time. In addition to the usual propositional connectives, these systems also provide operators such as "it will be the case that ...", to build up complex temporal propositions from atomic ones. When a statement is operated on by such operators, the truth value of the resulting statement depends, in general, on its time of assertion, and on the truth values of the operand at a set of times that bear some relation to the time of assertion. This relation is some derivative of the relation "before". The operators can be thought of as special quantifiers for time.

We shall discuss two basic systems of temporal logic in some detail, because they are simple, they convey the general flavor of systems of temporal logic in general, and they provide the foundation for understanding other systems.

The Basic System R of Temporal Logic

This is the most primitive system of temporal logic. There is, in fact, no notion of a "before" relation in this system. There is a single operator, called the temporal realization operator, that reads, "it is the case at time t that ...". It is denoted by R(t)(A), where t is a time, and A is a statement. Thus, we may have R(Jan. 18,1981)(it is raining), to mean "it rained on

Jan. 18, 1981". The parameter "t" here can be "now", denoted by "n". But it is an axiom that R(n)(A) is equivalent to A itself. The system R is an extension of propositional logic, and includes specific machinery for dealing with the R operator.

Formulas of R

Any propositional formula is a formula of R. In addition, we can use the R operator, and "temporal variables" which include a special symbol, "n", for "now", to construct other formulas as follows. If t and s are variables, then "t=s" is a formula. If A is a formula and t is a variable, then R(t)(A) is a formula, and (∀t)(A), and (Et)(A) are also formulas if t is not "n" (where "E" denotes the existential quantifier).

Proof Theory of R

Axioms

To the standard axioms of propositional logic, identity, and quantification, the following axioms are added to define the properties of the R operator.

1. R(t)(~A)<=>~R(t)(A)
2. R(t')[(∀t)A]<=>(∀t)R(t')(A)
3. R(t)(A&B)<=>(R(t)(A)&R(t)(B))
4. R(n)(A)<=>A
5. R(s)R(t)(A)<=>R(t)(A), except for t=n
6. R(t)(n=s)<=>t=s
7. R(t)(r=s)<=>r=s
8. (∀t)A=>A(t/n), where A(t/n) denotes the result of substituting n for the free occurrences of t in A, and t does not occur within the scope of an R operator in A.

The fourth axiom says that once a statement is made definite, it can no longer be affected by temporal realization. Note that if t is "now" in this axiom, A is not made definite by R(t), and hence the axiom does not apply.

Rules of Inference

In addition to modus ponens and the rule of specialization, we have the following rules for the introduction of the R operator.

If |-A<=>B and |-R(t)(A), then |-R(t)(B)).

If |-A, then (∀t)R(t)(A).

Model Theory of R

The main idea in the development of the model theory of R is that at each point of time each statement of R may be either true or false. The truth value of a temporally definite statement is constant over time; the truth value of a temporally indefinite statement may vary. In order to study the semantics of R, we can consider a "truth cube" (rather than a truth table), in which the third axis represents a set of times. We begin by stipulating a fixed set of times, and a truth value assignment for each of these times to each atomic proposition. The truth value of a formula that uses only logical connectives is determined at time t from the truth values of its constituents at time t in the usual way. But we also have to specify how to obtain the truth value of a temporally realized statement. In the system R, an interpretation is an assignment of a time to each temporal variable. But the interpretation of "n" must depend on the time at which a statement is being evaluated: in fact it must be identical to that time. R(t)(A) in such an interpretation is then defined to be true iff A is true at the time assigned to t.

With this machinery it is possible to define the notion of validity, and to prove the completeness of the system R.

Temporal Logic with the Relation "Before"

Fundamental to our notion of time is the relation "before" between points in time. This relation and the corresponding predicate is denoted as U. In Newtonian physics, time is modeled as the real line, and the U relation is "<". In this model, the U relation has a great deal of structure. It is possible, however, to construct theories of time in which the U relation has less structure. In studying temporal logic, the authors define a series of increasingly restrictive logics of time, and investigate their properties.

* The System K_t of Minimal Tense Logic: In this system no restrictions are imposed on the U relation.
* Branching Time: In this system the U relation is restricted to be transitive and backward linear (meaning that two predecessors of a given time must be relatable). At any point in time, the future can branch out, but the past is linear.

* Linear Time: In this system the U relation is restricted to be transitive and connected (meaning that any two times must be relatable).

* Additive Time: In additive time the P operator is interpreted as specifying the realization of a statement at a time relative to "now": by adding the specified time to "now", we obtain the time of reference of the temporally realized statement. In that case, nesting of the R operator becomes additive, that is, R(t)(R(s)(p))<=>R(t+s)(p), and "n" becomes the additive zero. Addition is assumed to define a commutative group. By further restricting addition, it is possible to think of additive time as a restriction of linear time, in which the U relation is defined in terms of addition.

* Metric Time: In this system the U relation is isomorphic to "<" for reals.

These systems can be developed by adding a specific two-place predicate symbol, U,to the basic system R, and by specifying additional axioms and rules of inference that define the properties of the U relation. Much of the development of temporal logic, however, has proceeded on the basis of so-called "tense-operators", rather than on the basis of the R operator and the relation U. Here are some important tense operators together with their definitions in the R/U basis.

1. Future: F(p), defined as, (Et)(U(n,t)&R(t)(p));

2. Past: P(p), defined as, (Et)(U(t,n)&R(t)(p));

3. Perpetual Future: G(p), defined as, (¥t)(U(n,t)=>R(t)(p));

4. Perpetual Past: H(p), defined as, (¥t)(U(t,n)=>R(t)(p)).

Note that the first two of these can be defined in terms of the last, and vice versa: F(p)<=>~G(~p), and P(p)<=>~H(~p).

We shall only discuss the system K_t in greater detail.

The System K_t of Minimal Tense Logic

The system K_t is minimal in the sense that every temporal statement that is true without assuming any special property of the U relation is a theorem of K_t.

Proof Theory of K_t

Formulas of K_t are constructed from propositional variables, propositional connectives, and the operators G and H. The following axioms and rule of inference are added to the deductive machinery of the propositional calculus.

Axioms

1. G(p=>q)=>(G(p)=>G(q));

2. H(p=>q)=>(H(p)=>H(q));

3. ~H~Gp=>p;

4. ~G~Hp=>p.

Rule of Inference:

if |-A then |-G(A) and |-H(A).

Model Theory of K_t

The semantics of tense logic are defined in terms of what are called tense structures. A tense structure consists of a set of times, a binary relation on this set (the U relation), and, for each time, a function that assigns truth values to each proposition. The truth values of complex propositions are defined in terms of the truth values of their constituents as would be expected from the definition of the tense operators. For example, G(A) is true at t iff for every time after t, A is true. A formula is said to be true in such a tense structure, iff it is true at all times. A formula is valid if it is true in every tense structure.

It is possible to prove that K_t is complete and decidable. It is also possible to prove that the translation of any provable formula of K_t into R can be proved in R.

It should come as no surprise that the "G/H language" used to express the formulas of K_t is weaker than the "R/U language" of the basic system R augmented by the predicate U. For example, there are properties of the U relation, such as asymmetry, that cannot be expressed in G/H, but can be expressed in R/U. (A property of is said to be expressible in a language if there is a formula of the language that is valid for all and only those tense structures that satisfy the property.) There are also "tenses" that cannot be expressed in G/H. A tense is defined by a statement of R/U having a single free variable (for its time of assertion). The tenses "p has been the case since q happened", and "p will be the case till q happens" cannot be expressed in G/H, for

example. But it is interesting that by assuming that time is linear, infinite, and dense, which are accepted assumptions of Newtonian time, these two tenses are sufficient as a basis for defining all tenses.

OTHER ISSUES OF TEMPORAL LOGIC

A most interesting aspect of this book is its discussion of numerous other issues that are motivated by the study of temporal logic. Here is a small sample of some of these issues.

* Temporal Categories of Statements

  The following distinctions can be made with respect to the time of reference of a statement. An atemporal statement is timeless: "5 is a prime number"; a present statement refers to now: "John is standing"; an omnitemporal statement refers to always: "copper is a conductor of electricity"; a transtemporal statement refers to the present period, but not necessarily to "always": "the earth is a planet of the sun". Evidently, different mechanisms of inference apply to each category of statements, although this issue is not formalized.

* A Three-Valued Temporal Logic

  The logics that we have so far considered assume that a temporally definite statement is either true or false at each time. But it is somewhat troublesome that the truth value of a statement is determined at all times prior to its time of reference. A simple solution is to allow temporal statements to have undefined truth values. The following restrictions on truth value assignments to definite temporal statements then seem reasonable. If a statement has a truth value T at any time, then it cannot have a truth value F at any other time (before or after). If a statement has a truth value T or F at any one time, then it retains that value for all later times. And at the time of reference of a statement, it is either true or false.

  Other approaches to a many-valued formulation of temporal logic are also explored.

* The Absolutist versus Relativist Notions of Time

  For the absolutist, times are "containers of events". They are defined logically prior to any events, and allow us to sort out events by relating them to their containers. For the relativist, events are logically prior to times. Times are constructs defined via events, a particular cluster of events defining each point in time. The significance of this distinction, according to the authors, is that the relativist must reject certain structures of time, e.g., cyclic structures. Of course, from the point of view of representing historical accounts in the physical world as we know it, such structures are of little interest. From this point of view, the question becomes whether or not we treat time as a derived notion in developing a logic of events and processes.

* Chronological Purity

  In describing a state of affairs or an event, we may or may not make reference to times other than that at which the affair took place. The latter kind of description is called "chronologically" pure, the former, "chronologically impure". Thus, a chronologically pure description would stand even if the only time in the world were the time of the occurrence of the affair described by it, whereas an impure description would lose its meaning in such a world. The distinction is important because certain deductions can lead to fallacious results when the statements involved are chronologically impure, but not when they are chronologically pure. For example, it seems plausible in some applications to say that "if a statement A is true at time t, then we know at all times s, s>=t, that A is true at time t". Now suppose A is the statement "rain, the day after tomorrow", and suppose that t is the day before yesterday, and s is yesterday. The assertion would then say that if it rains today then we knew yesterday that it rains today! By restricting A to be chronologically pure, we can remove this problem.

* Stable and Terminating Descriptions

  A predicate may begin designating a property for an object (or a relationship between objects), continue to so designate an object or objects, and then cease in this designation. When a predicate may cease to describe a property or relationship for objects, we say that the predicate is temporally unstable or terminating. The predicate "x is the president of the United States" is a terminating predicate. Other predicates, once they are applicable to a thing, are always applicable, and are termed stable. The predicate "x is the first child of someone" is stable. Clearly, we can infer the truth

of a stable predicate at times >=t from its truth at time t, but the same is not the case for a terminating predicate.

Both terminating and stable predicates may be identifying predicates, that is, they may identify a unique individual. But a terminating identifying predicate, such as "x is the president of the United States", identifies someone at each point in time: its referent may change over time. A stable identifying predicate, on the other hand, yields a unique individual irrespective of its time of assertion: "x is the first president of the United States", where the "is" is taken loosely to mean "is or was or was to become". The significance of this distinction is that we cannot substitute identical terminating descriptions for each other, unless we know that their times of assertion were the same. Also, if two terminating descriptions refer to the same individual, we cannot in general use one in place of the other.

* Quantification over Objects

Exactly what are we referring to when we use quantification in a temporal context? The axioms of a quantificational theory of temporal logic vary according to whether we are referring to all individuals existing now, all individuals who have existed up to now, or all individuals who will ever exist.

* Temporal Modality

Modality has to do with the notions of possibility and necessity. There are two classical schools of thought on temporal modality. The Diodorean School defines possible(A) as A|F(A), and necessary(A) as A&G(A). The Aristotelian/Megarian School defines possible(A) as A|F(A)|P(A), and necessary(A) as A&G(A)&H(A). It can be shown that the logical systems of temporal modality that arise from these definitions are identical to well-known modal logics. Other definitions of temporal modalities are also discussed.

* The Theory of Processes

An interesting contribution to the theory of processes is the relationship of process implication, saying that the existence of one event or process implies the existence of another at a later point in time. This is embodied in the definition, p|c|->q, for (∀ t)(R(t)(p)=>R(t+c)(q)). The notion can be generalized to a process family for which a successor function and a delay function are defined.

* Categories of Processes

The following distinctions between processes are of interest. In a homogeneous process, the process is going on at every point in time during some interval, e.g., standing. In a majoritative process, the process is going on most of the time during an interval, e.g., watching television. In an occasional process, the process is going on some of the time during an interval, e.g., drinking wine. In a wholistic process, the process as a whole is of interest.

Many of these issues are formalized to some extent (some are not). But there is room for a great deal more research in understanding and formalizing these and other issues discussed in the book.

10. J.F. Fries, "Time-oriented patient records and a computer data bank," Journal of the American Medical Association Vol. 22(12)(December 1972).

See Wiederhold [1975].

11. B. Bruce, "A model for temporal reference and its application in a question answering program," Artificial Intelligence Vol. 3(1)(1972).[2]

A formal framework is proposed for the analysis of tenses, time relations, and other references to time in natural language. A time-system is an ordered pair, (time, <), where "time" is a set whose elements are called time-points, and "<" is a relation which partially orders "time". Timing relations, such as "before", "during", "same-

2. See also:
B. Bruce and N. Singer, "CHRONOS II : The processing of incompletely specified data," Memo CBM-TR-9, Dept. of Computer Science, Rutgers University, New Brunswick (August 1972).

B. Bruce, "On the organization of event memories," Memo CBM-TM-25, Dept. of Computer Science, Rutgers University (July 1973).

B. Bruce, "CHRONOS user's manual- Version I," Memo CBM-TM-26, Dept. of Computer Science, Rutgers University (August 1973).

B. Bruce, "The processing of time phrases in CHRONOS," Memo CBM-TM-29, Dept. of Computer Science, Rutgers University, New Brunswick (September 1973).

time", and "overlaps", and functions such as "beginning", and "end", are defined in terms of "<" through the notation of first order predicate calculus with equality. A tense is defined as a special n-ary relation on time segments. The first time segment of a tense is called the "time of speech", the last is called the "time of event", and the intermediate segments are called the "times of reference". If the n time segments are represented in sequence, then the tense is defined by relating each segment to the following segment via one of the timing relations. For example, the past perfect is defined as "after(speech, reference) and after(reference, event)". The use of this framework to capture the notions of order, aspect, duration, frequency of occurrence, and modality is then exemplified.

A LISP program called CHRONOS is described that illustrates the processing of temporal information, tenses, time relations, dates, and time words. CHRONOS is a natural language system which consists of a simple English sentence parser, a theorem prover, and a database of facts and rules of inference of time relations and events. The system first accepts facts about events from the user. It can then answer questions which could include time relations between two events that are different from those explicitly entered earlier. CHRONOS makes use of the theorem prover on the facts and rules of inference on time relations to demonstrate its understanding of temporal information.

12. D.R. Dowty, Studies in the logic of verb aspect and time reference in English, Dept. of Linguistics, Univ. of Texas at Austin (1972).

13. J.T. Fraser, F.C. Haber, and G.H. Muller, "The study of time," Proceedings of the first conference of the international society for the study of time, Springer- Verlag, (1972).

14. G.G. Hendrix, "Modeling simultaneous actions and continuous processes," Artificial Intelligence Vol. 4(3)(1973).

Work on world modeling prior to this paper had relied on the notions of states, and operators. States are snapshot world descriptions; operators transform states. This paradigm promotes a world model in which the state of the world during the time in which an operator is applied is ignored. Furthermore, the choice of which operator to apply at a particular point of time was usually made by some active agent, e.g., a robot. Thus, it was difficult to model passive or involuntarily initiated state transformations. To remedy these shortcomings, Hendrix offers a view of the world as a collection of simultaneously ongoing processes. A process can be thought of as an operator that changes the state of the world gradually.

Processes are activated when certain "initiation conditions" are met. (Both passive and active process initiation can be modeled in this way, since the fact that an agent has chosen to initiate a process can be made part of the world state, and used as an initiation condition for that process.) Processes continue as long as certain "continuation conditions" are in effect. The state of the world is modeled by a set of predicates. But the positions in the predicates may be filled, not only with specific objects, but also with time-varying variables that designate things that are changing. The value of a time-varying variable at a particular time is obtained by solving a set of equations that describe its behavior over time. For example, during the filling of a bucket, the amount of water in it may be described by "content(bucket, Ycontent)", where "Ycontent" is a real variable, described by the equation "Ycontent= Cinitialcontent+Crate.time". But when no processes are active that have to do with the bucket's content, its content is described by a particular value.

Each type of process is described by a "scenario". A number of formal parameters are associated with each type of process. These are bound to particular objects once a process of that type is initiated. Process parameters can be "primary" or "secondary". Primary parameters are identifying parameters, in the sense that at any given time, only one process of a given kind can be going on for a given combination of primary parameters. For the bucket-filling example, these might be the bucket and the tap that is used to fill it. Examples of secondary parameters here are, a place designating the position of the bucket, and a real number giving the initial amount of water in the bucket. The scenario also includes initiation and continuation assertions for a process of the given kind. The initiation condition for bucket-filling includes the following assertions (among others), "at(bucket, place)", "at(tap, place)", and "content(bucket, Cinitialcontent)". As soon as such an initiation condition is satisfied by a set of objects, a monitor creates a process of the given kind, if none exists with the given primary parameters. The continuation conditions for bucket filling include such assertions as "orientation(bucket, up)". Finally, the scenario includes a description of the effect of the process in terms of changes. Gradual effects are described by systems of equations. A process may also lead to changes that are only interesting as a whole. These changes are described by add and

delete lists giving predicates to be added to or deleted from the world state when the process terminates.

The process simulator is initially given a world state. It uses this state to initiate a number of processes. But rather than stepping through time in discrete equal sized steps, it "leaps" to the next time when a process can be initiated or terminated, or when some information is needed about the world being modeled.

15. Operating Systems, Inc., "Synthesis of inference techniques: an interpreted syntax for the logical description of events," OSI: N74-003 Technical Note No. 2 (31 May 1974).

16. E.F. Falkenberg, "Time-handling in data base management systems," Report 07/1974, University of Stuttgart (July 1974).

The conceptual data model defined in the author's Ph.D. dissertation is extended to include the time dimension in this paper. The conceptual model is based on the notion of an association. An association is a set of two or more objects, in which each object plays a specific role. Associations correspond to relations that are semantically irreducible. Associations have a period of existence, hence a beginning and an end on the time axis, called events. Time is taken to be common calendar time. But it is noted that the "points" of time in a calendar system actually designate time intervals. So in defining such a time period by its end points on the calendar, it is important to specify which (if any) of the end points is to be included in the period.

Language expressions that are used to identify (sets of) objects, associations, events, time periods or points of time are called "significations". The paper enumerates a number of temporal signification concepts. Included are significations for sets of objects that exist at a specific time, for the time period or point at which a set of objects exist, and for times related by timing relations to points of time and to time intervals. Timing relations are defined to compare both time points and time periods with other time points and periods. They are essentially those of Bruce [1972].

Further development of this research is reported in Wacker [1974] and in Breutmann [1979].

17. R. Wacker, "Das Zeitproblem bei generalizierten Datenbankmanagement," Diplomarbeit, Universitat Stuttgart, Institut fur Informatik (August 1974).

This Master's thesis is based on the work of Falkenberg [1974]. Time is assumed to be isomorphic to the real numbers. The distance function on the real numbers is used to partition time into unit intervals, which become the basic time unit for a particular application, and are called "time points". Time intervals are defined as multiples of time points. Events are defined as the beginning or ending of associations. Associations always happen over intervals. But the way intervals are defined, it appears that one can't hire and fire someone in the same day in an employee database.

Many temporal relations are specified for comparing time points and time intervals. Also, a query language that permits time point and time interval signification (naming) is examined. The naming of time points and time intervals is based on various combinations of the temporal relations and scalar quantities, such as "weeks", as in "three weeks before 1/1/75". But it is unclear how "three weeks" is defined. Also, while the dependence of the time system on the application is pointed out, no general mechanism for defining an arbitrary time system is given.

18. K.M. Kahn, "Mechanization of temporal knowledge," Tecnical Report MAC-TR-155, Artificial Intelligence Laboratory, MIT (September 1975).

This work was conceived as a part of the "Present Illness Program".[3] A module, called the "time specialist" was designed and implemented to store, retrieve, and reason about temporal information. It was considered important to be able to represent inexact temporal facts.

There are three major ways to organize temporal statements in the time specialist: by relating events to dates, by relating events to special "reference events", and by relating events together into before/after chains. Reference events are events like birth, whose time is usually known quite precisely, and that are often used to time other events. Each of these methods has its own special data structures and routines to work with those structures. When information about a new event is added to the data base, the system can attempt to deduce the date of this event, or to relate it to a reference event, or to fit it into a before/after chain.

The time specialist is able to handle queries such

3. S. Pauker, et al., "Toward a simulation of clinical cognition- taking a present illness by the computer," American Journal of Medicine, Vol. 60, pp. 981-996 (June 1976).

as: "did event X happen at time-expression T?", "when did event X happen?", and "what happened at time-expression T?" In answering queries, information from all of the above categories may be combined.

The system can also check the consistency of a new fact with the present state of the data base, and to try to resolve inconsistencies through interaction with the user. In such an interaction, the user may withdraw either the new fact, or some old facts whose removal would lead to consistency. Removing old facts may involve undoing some prior deductions. In order to be able to do this, a deduced fact is marked by those facts used to deduce it. The user may also claim that all the facts involved in the inconsistency are true, and the system will be forced to accept this. Since the deduction programs of the time specialist are quite specialized, the effects of such inconsistency can usually be contained.

Fuzziness is represented by plus/minus error intervals for the dates of events, and for the lengths of time between two events. The processing of fuzziness is quite simple: in following chains of events the error intervals are added. Mention is made of more sophisticated processing of fuzziness via "probability curves".

The time specialist was tested with examples from two different domains: the recognition of a medical diagnostic pattern, and the understanding of time travel stories. For the first example a hypothesis matcher had to be devised. The second one needed a facility to interpret timing relations from different perspectives.

In the last part of the manuscript Kahn suggests that partitioning the facts into 'periods', as in history books, would be a useful extension of his system.

19. B. Sundgren, Theory of Data Bases, New York (1975).[4]

"Theory of Data Bases" is a condensed and revised version of the author's Ph.D. thesis, "An Infological Approach to Data Bases", completed in 1973. This work is an extension of "Theoretical Analysis of Information Systems" [1966], by B. Langefors. In the infological approach, the "elementary message" is a unit of information, that informs the database of some state of affairs in the real world at a given point of time. For example, "(JAY, <WEIGHT, 205 POUNDS>, 8/7/81)" is an elementary object message and "(<BURROUGHS, IBM, WIDGETS>, SELL, 4/1/81)" is an elementary relation message. The Langefors/Sundgren work is perhaps the earliest work to recognize the unique role of time in information processing. But although time is introduced as a fundamental concept in the infological approach, the semantics of time are not developed in any detail. For example, for a query like <JAY, <WEIGHT, ?>, 8/12/81>, it is suggested that the answer should be the "nearest" time version of an elementary message with the required information. But "nearest" is not defined formally. However, the effect of time-dependent information on file design is considered by Sundgren. (See also Bubenko [1976] for a discussion of this.)

20. G. Wiederhold, J.F. Fries, and S. Weyl, "Structured organization of clinical data bases," Proceedings of the AFIPS National Computer Conference, (1975).

Designers of medical information systems have long felt the need for the representation and processing of temporal information. Perhaps the earliest system to address this need is TOD (for Time Oriented Databank), which has been in operation for many years at Stanford University.

In TOD, entities (e.g., patients) are described by static and dynamic attributes. Static attributes are attributes whose value remains constant (e.g., date of birth), or whose value is only of interest in its most recent observation. "Address" might be an example of the second kind in some applications. When a static attribute changes, its old value is overwritten . Dynamic attributes may change over time, and their values at all times are of interest. The old values remains recorded while the new value is added to the data base. This dynamic information may be thought of as a three dimensional (sparse) matrix, in which the dimensions are entities, attributes, and times. The information is represented in what may be called a "discrete temporal relation", in which each tuple has a time attribute designating the time at which a set of properties are observed for an entity.

This model has been incorporated in a number of medical data base management systems. In addition to routine data management facilities, there are several statistical utilities, an interactive query language, and procedures to make data files

4. See also:
B. Sundgren, "Conceptual foundation of the infological approach to data bases," Proceedings IFIP Conference on Data Base Management, pp. 61-94 (1974).

B. Langefors and B. Sundgren, Information Systems Architecture, Petrocelli/Charter, New York (1975).

compatible with common statistical packages. This software system is available on a time-shared basis. One of its principal applications is research on rheumatic disease.

21. S.M. Weiss, K. Kern, C.A. Kulikowski, and A. Safir, "A system for interactive analysis of a time-sequenced ophthalmological data base," Computers in Biomedicine Techincal Report #67, Department of Computer Scienc, Rutgers University (March 1976).

22. J.A. Bubenko, Jr., "The temporal dimension in information modeling," RC 6187 #26479, IBM Thomas J. Watson Research Center, Yorktown Heights, New York (16 November 1976).[5]

The important distinction between a "time-unrestricted" conceptual design of a database, and its "time-restricted" implementation is discussed. At the conceptual level, a machine of potentially unlimited memory size (for storing an ever increasing body of historical information), and infinitesimally small processing speed (so that the time of a user request is the time of the satisfaction of the request) is assumed. However, the implementation must be time-restricted, that is, only a limited amount of information can be available to a user at any given time, and there is always a delay between a request and its processing. It is proposed that the mapping between the time-unrestricted view and the time restricted view be made explicit. A first order predicate calculus-based language is suggested for the specification of this mapping. The paper also examines the implementation of these ideas in the binary and the n-ary relational models. The supplier-warehouse-parts example is used for illustration.

These and other issues discussed in the paper are further developed in Bubenko [1980]. The paper also surveys previous results in representing time in data base systems.

23. I.P. Goldstein and R.B. Roberts, "Nudge, a knowledge-based scheduling program," pp. 257-263 in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, (1977).[6]

24. R.P. McArthur, Tense Logic, D. Reidel Publishing Co., Dordrecht, Holland, and Hingham, Mass. (1976).

This book is a very readable introduction to temporal logic. In less than a hundred pages, the book covers the principal ideas of the proof theory and model theory of linear tense logic, branching tense logic, and quantificational tense logic. An introductory chapter develops the model theory of tense logic in terms of "possible worlds semantics". A world state is represented as a truth value assignment to propositions. Since world states may recur, a possible world at some time is represented by pairing a world state with a real number (an index). A set of such pairs, together with a binary relation on the set (representing the relation "before") constitutes a history, and is used to develop the semantics of tense logic. A final chapter proves soundness and completeness theorems for tense logics.

25. Operating Systems, Inc., "OSI, All-source data base design study- final technical report," OSI: R76-004 (10 May 1976).

26. N.A. Palley, et al., "CLINFO User's Guide: release one," technical Report R-1543-1-NIH, Rand Corporation, Santa Monica, California (1976).

CLINFO is a medical information system based on the data model of TOD (see Wiederhold [1975]). The specification of the system resulted from a thorough requirements analysis study of NIH sponsored clinical research centers. The system was originally developed at Rand Corporation, bu is now commercially available from BB&N. Like TOD, it contains several statistical procedures, an interactive query facility, commands for file manipulation, and simple report generation. Many of these facilities use the notion of a "worksheet": a two-dimensional slice of the three-dimensional data base of patients, attributes, and times.

27. F. Jakob, "Le traitement de l'expression du temps, un developpement d'un systeme de comprehension du language naturel," these du troisieme cycle, L'Universite Pierre et Marie Curie (1977).

---

5. A later version of this work appeared in Architecture and Models in Data Base Management Systems, ed. G.M. Nijssen, North Holland, Amsterdam (1977).

6. See also:
I.P. Goldstein, "Bargaining Between Goals," Proceedings Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR pp. 175-180 (September 1975).

I.P. Goldstein and R.B. Roberts, "Using frames in scheduling," in Artificial Intelligence, An MIT Perspective, MIT Press, Cambridge Massachusetts (1979).

This Ph.D. dissertation deals with the construction of a module for handling time in the language understanding system CKBLM.[7] The representation of knowledge in this system is primarily procedural. The declarative part of the representation consists of a set of triplets (instances of binary relations). Each triplet has an associated "a priori plausibility index", a value between 0 (false) and 1 (true). An estimator calculates the plausibility of a triplet as a function of the plausibility of the triplets that are closely related to it. Understanding a statement is defined as calculating the "best" plausibility index for it based on the available knowledge.

Following are some of the major problems addressed by Jakob. First, timing information has both a numerical form (date), and a non-numerical form (e.g., tense of verb). Second, temporal statements in natural language are often vague and fuzzy. For example, in the statement "it rained last Sunday", the period of rain may be interpreted as some period during last Sunday, or as the the whole of last Sunday. Again, in "John set the table while Mary made the salad", the "while" cannot be taken too literally. About the only thing that can be said for sure here is that the two events in question are not sequential. Finally, and along the same lines, the time of an event can be thought of alternately as a point of reference, and as a duration. Thus, in "the voyage of Columbus marked the beginning of an era of great exploration", the focus is on the voyage as a whole, and the voyage is considered as a point of reference, whereas, for example, in describing the voyage itself, the focus is on the voyage as happening over a duration. An event as duration is described by the attributes begin, end, and duration. Note that since all the attributes may be imprecise, the duration attribute may not be redundant.

In natural language, there are a large number of words having temporal meaning. But many such words designate similar or related concepts. Jakob defines a set of "privileged" temporal relation and function names, including "before", "same-time", "as long as", "between", "begin", "end", "date", "duration". The temporal triplets of the underlying system are first translated into triplets of this privileged class. The procedures used for this transformation are called "expert procedures". A number of so-called "technical procedures" know about the properties of the privileged relations and functions, and can use these properties in deduction and question-answering.

The last part of the dissertation describes several experiments that highlight the issues and mechanisms mentioned above. A historical paragraph concerning the career of General Petain is used as test.

7. D. Coulon, D. Kayser, A. Bonnet, J.M. Lancel, M. Monfils, "Description generale d'un systeme de reponse aux questions", IFIA/SESORI Rocquencourt, in three parts (April 1977), (January 1978), and (June 1978).

28. K.M. Kahn and A.G. Gorry, "Mechanizing temporal knowledge," Artificial Intelligence Vol. 9(2) pp. 87- 108 (1977).
See Kahn [1975].

29. E.D. Sacerdoti, A Structure for Plans and Behaviour, Elsevier North-Holland, Inc. (1977).

This work is the first approach to the generation of plans that uses the top-down-refinement paradigm. The problem concerns having a computer tell an apprentice how to accomplish a task, e.g., assemble a piece of machinery. The program's knowledge base is a description of a set of actions in terms of procedures involving other actions. When the program determines that an action is to be performed to achieve an objective, it first prints out a natural language description of that action. The user, if he is sufficiently trained, may understand the action and perform it right away. Otherwise , by using the corresponding procedure description, the system "expands" the action into more refined actions that are performed recursively in the same way. The state of the set of objects that the apprentice is working with (the "world" of the program) is represented by a set of predicates. By using a series of actions, the program transforms a given initial state of the world to a desired final state. The actions add and delete predicates until the goal predicates are all present.

The Knowledge Base

To each action is associated an action procedure, an "add list", giving predicates that are made true by performing that action, and a "delete list", giving predicates that are no longer true after performing the action. Each action procedure has a number of formal parameters, which are instantiated to specific objects when the procedure is applied. In this way the predicates in the add and delete lists are instantiated. The action procedures are specified in a language called SOUP (Semantics of User's Problem), which is an extension of QLISP. Of particular interest in this language are a number of control structures, and an exclusion primitive.

- "PAND(statement1, statement2)" means do the two statements in an arbitrary order.

- "POR (statement1, statement2)" means do either one.

- "PGOAL(query, pattern), APPLY(proc1, proc2,...,procn)" means achieve a goal which is described to the user by the query statement, and to the system by the pattern (an assertion made true). If the user can't do what the query specifies, then try applying each of proc1, proc2,...procn, until one of them works.

- "PBUILD(class-name, query), ITERATE(statement)" means do something that builds a class of objects. The statement is to be executed for each member of the class.

- "PBREAK(class-name, query), ITERATE(statement)". The statement is to be executed for each member of an existing class.

- "PRECLUDE(variable, object)" precludes the binding of the given object to the given variable. When the variable is used later, it must be bound to some other object, so that the specified object is not affected by the later use of the variable. This instruction is used to prevent later actions from undoing an established state of an object.

Program Structure

The program begins by finding a high level action whose execution will achieve the desired transformation. This is done by taking the union of the initial state and the add list of an action, then removing the corresponding delete list and comparing the result with the goal state. The algorithm also binds the formal parameters of the corresponding procedure with actual objects. The action is then performed as described previously.

An iterative action (PBUILD or PBREAK) is expanded into two actions, a single iteration which may be expanded to any level of detail, and a replicate action that specifies how many more times the single iteration is to be repeated. As each iteration is done, the replicate count is decreased. When control reaches a replicate action, the apprentice is asked to do all the iterations that remain. If he can't, then control is passed back to another iteration.

The normal process of plan generation works by the local expansion of a node, without regard to global problems that may arise as a result of this expansion (except when such problems are anticipated in previous PRECLUDE statements). This locality allows the expansion process to be efficient. But the relative independence of the expansion of the different parts of a plan can lead to conflicts between the requirements of one subplan, and the achievements of another. For example, in painting both a ladder and a ceiling, these two tasks may be PANDed. But by painting the ladder first, we make it unusable for painting the ceiling.

Such problems are resolved in Sacerdoti's approach by means of specialists, called "critics", that look at the global situation from time to time, and arbitrate in the sequencing of actions and the binding of objects to formal parameters to avoid specific problems. Critics are also used in optimizing the use of resources. For example, if an action can be performed by means of a tool that is already at hand, a "use existing object" critic would opt for the binding of that tool as opposed to the acquisition of another.

30. R. Schank, and R. Abelson, Scripts Plans Goals and Understanding, LEA Publishers, Hillside, New Jersey (1977).

31. M.J. Steedman, "Verb, time and modality," Cognitive Science Vol. 1(2) pp. 216-234 (April 1977).

This paper is concerned with the linguistic analysis of sentences in which time is an important component. Steedman argues that the classification of verbs into so-called "aspectual categories" is better seen as a classification of propositions about the real world. Propositions, in Steedman's scheme, are initially categorized into "situations" and "events". Situations are propositions about continuous states of affairs, such as, "he is running", or "he knows her". Events are propositions about definite happenings. Events are further categorized into "accomplishments", "achievements", "activities", and "actions". An accomplishment involves an event with an intrinsic conclusion: "he ran a mile in four minutes". It designates a completed doing as a whole. An achievement, on the other hand, usually emphasizes the conclusion of a doing: "he reached the summit". Activities do not focus on the completion of a doing: "he ran". Finally, actions designate a repeated doing: "he tapped".

A model of the relationships between these categories is proposed that specifies how a proposition of one category can be transformed into that of another category by the use of adverbials and by context. The model is a system of

productions with associated semantic rules that define the condition or the context under which a "reduction" by the production is allowed. For example, "accomplishment->activity 'in' time" is a production whose associated semantic rule is "activity & achievement-of(activity) & T(activity)= time & T(achievement-of(activity)= end-of(time)". The proposition "he ran a mile in four minutes" exemplifies a proposition which may be reduced by this rule (here "running a mile" is thought of as an activity, and "four minutes" is a time). The semantic rule says that a reduction is possible by this production if we know that the activity of running a mile did go on, we know that it went on during the four minutes, that it was concluded, and that its conclusion happened at the end of the four minutes.

The second part of this paper deals with modal auxiliaries, such as "must" and "may". These verbs have an "epistemic" meaning, having to do with necessity and possibility, and a "deontic" meaning, having to do with social obligation and permission. It is shown that the epistemic meaning of the modals appears to be associated with situations: "you must be the chief plumber", whereas the deontic meaning is usually associated with events: "you must get married".

32. J.M. Martin, J.P. Pointel, J. Martin, G. Debry, "The notion of time in medical records - structure of chronology, validation and calculation of a time interval," Methods of Information in Medicine, Vol. 17(2), pp. 90-95 (April 1978).

A primary concern of this paper is the validation of chronological data. The validation procedures check for the valid range of calendar and clock times, and for constraints specified by chronological relationships between events. Two such relationships are emphasized: mutual exclusion, and inclusion of events. The latter is based on a hierarchical structure of event types. The procedures make sure that the time interval of an event is included in the time interval of its parent in a hierarchy.

33. J. Bradley, "Operations data bases," pp. 164-176 in Proceedings of the Fourth International Conference on Very Large Data Bases, (September 1978).

This paper considers the representation and querying of databases containing information about processes. A process involves one or more entities, and changes an attribute of at least one of these entities. For example, a painting process may involve a painter and a car, and changes the color of the car. It is shown that the representation of processes in conventional data models leads to an unreasonably complex representation of quite simple queries in the calculus-type query languages of these models. A number of structural properties of processes and attributes are considered. For each of these, it is first shown how the structure can be represented in the owner-coupled set model. Then, extensions to calculus-type query languages are proposed, that make it a great deal simpler to pose a query involving that structural property.

First, some attributes of entities are time-varying. The simplest way to represent a time-varying attribute is via a distinct record type that is related by an n:1 relation to the entity record type involved, and that has one instance for each time period over which the attribute is constant. But it is simpler in the query language to think of such attributes as time functions. Accordingly, a query format exemplified by "get car.color(t) where car.origin='France'" is proposed, to obtain the value of an attribute at a point in time. Since the time-varying attributes are changed by processes, there are relationships between process record types and the time-varying attribute record types. Query formats are therefore proposed to make it easy to relate process instances and the attributes affected by them.

Second, processes can be decomposed into subprocesses. The subprocesses would normally be represented by different record types. It is cumbersome, however, to explicitly relate subprocess records to encompassing process records in a query to find out how an instance of a given process was performed in terms of the subprocesses in it. A query format denoted by "get-how" is introduced for this purpose.

Finally the decomposition of processes of a given kind may not be uniform. A process can be thought of as represented by a program of a suitable programming language with conditionals. The conditionals depend on the attribute values of the entities involved in the process. The representation of such general processes in conventional data model is considerably more complex. It is suggested that decision tables be used in this representation. To retrieve a decomposition of such a process the "get-how-could" query format is introduced. In it the attributes of some of the entities involved in a process can be specified, and the decomposition for those attribute values will be retrieved, for example, the type of car to be painted may be specified as "racing car", and the details of the painting process for racing cars will be retrieved.

34. A. Flory and J. Kouloumdjiam, "A model for the description of the information system dynamics," Proceedings of the 2nd Conference of European Cooperation in Informatics, (October 1978).

This paper considers the problems involved in designing and implementing legitimate state changes in a database. The state change model evolves out of the observation that a functional dependency is sometimes associated with chronological order. For example, in "invoice-number->order-number" the invoice is created after the order. This chronological order is similar to "unconditional precedence" as described in Hemphill [1978], Codd [1979], and Anderson [1981].

Events, i.e, state changes, are categorized as internal or external. An internal event might be the calculation of the quantity on hand for an item in a supplier database. An external event might be the receipt of an order. This classification is based on whether the attribute involved in the state change is calculated or given, and on whether it is stable or unstable. (A stable attribute does not change over time.)

The concept of a tuple state is used to indicate the processing stage for a particular tuple of a relation describing an event. For example, an order may be registered, refused, waiting, or fulfilled. A finite state diagram is used to describe the transitions between tuple states. The transition specification includes a predicate that must be satisfied for the transition to take place, and a procedure for accomplishing the transition. It is unclear how the state transition information is to be stored, or indeed whether it needs to be stored in order to direct the processing sequence, or whether it is only intended to be used as a data base design aid.

35. G.E. Heidorn, "Natural language dialogue for managing an on-line calendar," Proceedings of the ACM Conference, (December 1978).

The paper describes SCHED, a natural language interface for a calendar database. The processing in SCHED is done with Augmented Phrase Structure Grammars using a LISP based natural language processor. This work is based on the author's earlier work on dialogues about simple simulation problems. The main emphasis of this paper is on the analysis and synthesis of natural language utterances in the dialogue. The initial version of the system described in the paper does not deal with resolving scheduling conflicts (c.f., Goldstein [1977]).

36. L. Hemphill and J. Rhyne, "A model for knowledge representation in natural language query systems," IBM research report #RJ2304 (31046), San Jose, California (September 28, 1978).

It is argued that scripts (Schank [1977]) can be used as a basis of many data base processing operations, especially when dealing with data bases that represent information about complex events. A script is a generalized specification of a complex event in terms of the possible sequences of subevents that may occur in it. In order to describe the temporal relationships between the subevents of an event, two relations were found necessary: "necessary predecessor", and "possible successor".

In a database of similar events, the subevents of a given kind for different events can be represented in a relation. The script for these events can be thought of as a high level description of this data base. Such scripts can be used to aid both in data input, and in query processing. On input, the script can drive an input program, and thereby help in the enforcement of certain consistency constraints. If input about the subevents of an event is received synchronously as the program steps through the script, the program can easily detect a subevent that does not belong to a particular place, or that does not belong to a particular event. In query processing, the script allows one to infer the existence of a subevent from the existence of another event to which the subevent is a necessary predecessor. This is called "scriptal implication".

Other kinds of scripts can also be used in query processing. For example, a questioner needs information to satisfy certain goals. The events needed for the attainment of the goal can be represented in a script, and used to provide useful information over and above what is absolutely necessary in answering a query.

37. A. Bolour, "The process model of data," Technical Report #38, University of California, San Francisco (March 1979).

38. B. Breutmann, "The temporal dimension of conceptual schemas," in Proceedings of the IFIP Working Group 2.6, Munich (March 1979).
See Breutmann [June 1979].

39. E.F. Codd, "Extending the data base relational model to capture more meaning," Proceedings ACM-SIGMOD International Conf. on the Management of Data, (May, 1979). Revised in ACM Transactions on Database Systems Vol.

4(4) pp. 397,434 (December 1979).

Included in the extensions proposed in this paper to the relational model are four relations between event types. These relations are defined as sets of tuples of the form (SUB: event-type-name, SUP: event-type-name).

* Unconditional Successor: An event type E1 is an unconditional successor of an event type E2, if an event of type E1 always succeeds an event of type E2, but there is no intermediate event type E such that E is an unconditional successor of E2, and E1 is an unconditional successor of E.

* Alternate Successor: This relation represents an exclusive-or of successors. An event of a given type must be followed by an event that belongs to one of its alternate successor types (if any).

* Unconditional Precedence: Similar to unconditional successor.

* Alternate Precedence: Similar to alternate successor.

These relations can be used by insertion and update programs for a data base to check the validity of a requested transaction. For example, when entering a new shipment event into a data base, the system can check for an order event that should unconditionally precede the shipment. (See also Hemphill [1978].)

40. B. Breutmann, E.F. Falkenberg, and R. Mauer, "CSL : A language for defining conceptual schemas," in Data Base Architecture, ed. G.M. Nijssen and G. Bracchi,North Holland, Inc., Amsterdam (June 1979).

A conceptual schema should specify both the static and the dynamic aspects of a database. The dynamics are defined in terms of "events" representing the beginning, ending, or change of an association between objects. Events occur at particular points in time. Hence time plays an important role in the definition of conceptual schemas. In order to keep track of the history of events, we need to model calendar systems in the conceptual schema. Furthermore, in order to specify constraints on events, for example, constraints that limit the interval between two consecutive events of specified types, or that limit the frequency of a given type of event within an interval, we may need to include other temporal notions in a language for defining conceptual schemas.

The portions of this paper that deal with time concentrate on an approach to defining calendar systems; temporal constraints on events are not further explored. The use of a language for defining calendar systems is exemplified, though the language itself is not formally defined. The salient features of this language are as follows. First, a sequence of domains, called "time segment types" can be defined, which correspond to increasingly coarse time periods, for example, days, months, and years. If there are n domains in the sequence, then a vector $(t_k, \ldots, t_n)$, $1<=k<=n$, in which t_j belongs to the j^{th} domain, $k<=j<=n$, represents a generalized point in time, e.g., (March,1979). Each such point has a beginning and an end, which are elements of the next finer domain. The language provides facilities for defining the beginning and the end of such time points by means of specific procedures for each k. (It should be mentioned that a formal definition of a calendar system appearing at the beginning of this paper is more restrictive than what the language actually allows.)

41. C. Rolland, S. Leifert, and C. Richard, "Tools for information system dynamics management," Proceedings of Fifth International Conference on Very Large Data Bases, (October 1979).

A concept called "Dynamic Schema" is proposed to be used in the specification of information systems. A dynamic schema represents the interconnections between static objects, operations, and events. The collection of objects with their property values represents a state of the information system being modelled. An operation represents an atomic action operating on certain objects. An event represents the atomic state change of the system.

An application of the dynamic schema concept allows one to represent reality as a set of interdependent and interacting processes which together work on the objects of the information system. A causal description model is used to describe the synchronization of processes. Two processes can be permanently chronologically ordered, conditionally chronologically ordered, or independent. In addition, a class of objects called triggers can be specified that act as invocation conditions of processes. Finally, integrity constraints, called "time constraints", can be specified on the dynamic schema. Examples of such constraints are "orders arrive only once a week", and "stock listings are considered every Saturday".

42. F. Jakob, Un examle de representation de connaissances sur le temps, Congres Reconnaissance des Formes et Intelligence Artificielle, Universite Paul Sabatier, Toulouse,

France (September 1979).

43. G. E. Yamami, "The Sematics of Time Series Data Modeling," Master's Thesis, Dept. of EECS, University of California, Berkeley (December 1979).

This thesis describes a data model for time series, including a data language and a set of commands to manipulate, analyze, display, and generate reports about time series. A time series is a sequence of observations for a variable taken at regular frequency over some time interval. The data model is essentially an entity-relationship model. Each time series is represented as an entity, which has as attributes its ID, description, its frequency, and its interval. The frequency of the time series is the periodicity or number of observations per year for the data values. The interval is the calendar period over which the time series data observations are available. Relationships among entities are represented by equations of the form: "Series A = expr (Series B, Series C, ...)", where "expr" is an arithmetic expression involving the arguments series. For example, suppose "agriculture", "industry", and "retail" are the names of time series that record the amount of energy used by each category for each year over a period of several years. Then, a new time series can be defined as the sum of these: "total_commercial= agriculture+industry+retail", to record the total amount of energy used each year. Expressions can also have other operators such as SQRT, ABS, LOG, MAX, MIN, SUM, and AVG. Sets of these relationships are called models. A simple query language is presented to retrieve entities of particular properties. An example will give a flavor of the language: "FRAME INT = 76:1:2 TO 76:10:3 FREQ = daily" will retrieve all the time series having the interval between the two calendar dates and daily frequency.

The time series commands can be categorized as follows:

* Data Management - includes commands to create, update, and transform series;

* Analysis - includes commands for regression analysis and modelling;

* Display - deals with plotting a series against time or a series against another series;

* Report generation

The thesis contains some interesting examples of application using the data model and its commands.

44. S. Jones, P. Mason, and R. Stamper, "LEGOL 2.0: A relational specification language for complex rules," Information Systems Vol. 4(4) pp. 293-305 (1979).

LEGOL 2.0 is a formal language for writing rules. Its development was motivated by an attempt to formalize legislation. In order to apply a law correctly, one often needs to know about the history of a case at hand. Hence, time handling plays a central role in LEGOL 2.0. The language is based on a relational database model in which each relation has three types of attributes: identifier attributes that name entities, a characteristic attribute that designates a property of an entity, and time attributes ("start-time" and "end-time") that designate a period over which an entity had a property. We call such relations "continuous temporal relations". They may be contrasted with the discrete temporal relations of of TOD (Wiederhold [1975]), in which a relation has a single time attribute designating the time at which a property or a set of properties were observed.

A primitive LEGOL rule is represented as a high level "append" operation for continuous temporal relations. The appended tuples represent the result of applying the rule. Primitive rules can be combined, e.g., in a sequence, to specify complex rules. The set of tuples to be appended is specified in a relational algebra-like language similar to ISBL.[8] The algebra includes temporal operators, the most important of which is the so-called "time intersect" operator represented by "while". This is a special join operator. It specifies a matching condition on the non-temporal attributes of two relations, but also involves an implicit intersection of the periods of time during which the condition is met by two matching tuples. For example, consider an educational application in which the relations advice(student, professor, start-time, end-time), and Sabbatical(professor, start-time, end-time) are defined. Then the expression "advice(student,professor) while Sabbatical(professor)" would produce a join of the two relations containing the professors who advised students while on Sabbatical, together with the time periods during which this happened for each student. The result may be used as part of a rule to reward (or punish!) such professors for their good deed, depending on how long it lasted. (Note that the counterpart of the time intersect operation for discrete temporal relations requires an implicit equality match of the

8. S.J.P. Todd, "The Peterlee relational test vehicle - a system overview," IBM System Journal, Vol. 15(4), pp. 285-308 (1976).

times of two tuples.)

There are also a variety of unary temporal operations (functions) in LEGOL 2.0. For example, the operation "max" will find for each entity and each maximal interval over which a property is known for that entity, the maximum value of that property over the interval. In contrast, the operation "highest" will find a maximum over all table entries. As another example, the operation "first" will select table entries with earliest start times for each entity and each value of a characteristic.

The main emphasis of this paper is on defining an algebra of temporal relations. The usual calendar system is assumed for the representation of times. Although temporal arithmetic operations, such as "start-of(person)+16", are discussed in the paper, the formal semantics of these operations are not. A working implementation of an earlier version of LEGOL was completed in 1975, by using the Peterlee Relational Test Vehicle. The present design has not been implemented, in part because an implementation appears impractical for large data bases. (Note that all changes to the data base are through append operations: nothing is ever deleted from the data base.)

45. M.W. Freeman, "KNET: an extended SI-net formalism for knowledge representation systems," TR 80-1, Burroughs Corporation, Federal and Special Systems Group, Paoli, Pennsylvania (January 1980).

46. L. Hirschman, "Retrieving time information from natural language texts," in Proceedings of the Conference in Information Retrieval Research, Cambridge, England (June 1980).[9]

An algorithm for capturing time relations from natural language accounts is presented. The algorithm makes use of both explicit and implicit timing information contained in a narrative. Explicit timing information is usually given in the form of adverbial expressions, such as, "during the first day of hospitalization." Implicit timing information is conveyed by relationships between sentences and parts of sentences. One convention is that time does not move backward in narrative unless an explicit time marker is provided. Another is that the time of a subordinate clause is equal to the time of its main clause, again assuming that no other explicit timing information is given.

The input to the algorithm is a sequence of subject-verb-object units of information about events. These are the output of a syntax analyzer for natural language. The units are connected by binary connectives which indicate phrase relations (e.g., subordinate conjunction as for the "while" in "she developed dyspnea while she was in Maryland"). The algorithm builds a graphical representation of the timing information contained in these units. In the graph, the nodes are the time points associated with the events and the edges represent the intervals between events.

The paper illustrates the time algorithm by using an excerpt from a hospital discharge summary.

47. J.E. Allchin, "Modula and a question of time," IEEE Transactions on Software Engineering Vol. SE-6(4)(July 1980).

This is a short note that examines the interaction between time and devices as handled by the real-time systems language Modula. The note outlines the process structure necessary to implement the DOIO statement and concludes that the assumptions about time made in the nucleus of a language such as Modula should be reviewed. The author states that "many languages are still without the basic construction components of time (time instances and time intervals) and do not provide the extension tools necessary to define them."

48. C. Rolland, C. Richard, and S. Leifert, "A proposal for information system design and management," SIGDA Newsletter Vol. 10(2)(August 1980).

49. J.A. Bubenko, Jr., "Information modeling in the context of system development," Proceedings of IFIP Congress 80, (October 1980).

A new level in the system development life cycle, called the "understanding level", is proposed. At this level, information requirements are first specified in terms of what knowledge is needed by users at what times. Then, these requirements are used to develop a model of the application reality, called the "conceptual information model", that is free of traditional data processing concepts. Recent developments in data modeling focus on representing the state of an application reality. Bubenko argues that transactions, "representing events in the application environment which affect the state of the model", should constitute an integral part of the conceptual

9. See also:
L. Hirschman, and G. Story, "Representing explicit and implicit time relations in narrative," pp. 289-295 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

information model. Time, then, is needed not only in requirement specification, but, perhaps more importantly, in the description of the evolution of the state of the model.

A time model similar to Breutmann's [1979] is proposed. In this model, there is an infinite set of time points, T, and a set of time interval types, TI={TI_1, ..., TI_N}. For example, T may be the real line, and TI may be {seconds, hours, days, months, years}. By a type such as month here is meant the set of all one month periods, e.g., "March, 1980", etc. There is also a partial ordering relation, "part-of", on the set {T, TI_1, ..., TI_N}, so that if the relation holds between two time types, then the members of the first type are parts of members of the second type. For each pair (TI_i, TI_j) in this relation, there is a total function from TI_i into TI_j. It determines that interval of TI_j to which a member of TI_i belongs. For each application at hand one defines the appropriate time intervals.

By using this machinery together with comparisons and the language of the predicate calculus, it is possible to make temporal specifications such as "the set of first days every month during the summer season". Other specifications, such as "the set of days in 1980 in which John's car was taken for repair", and "the week in 1980 when the total hours worked was maximum", require further machinery, but are not further discussed.

There are three basic kinds of objects in the conceptual information model: data, entities, and events. The objects in each of these are grouped into types. Time types are considered to be event types. In general, relationships between object types are represented by functions from one type (or the Cartesian product of several types) into another type. For an entity type, these are called attribute functions. Attribute functions are often time-varying, and have defined derivation rules that may depend on events that affect their values. For example, a person's address may vary from day to day, and can be defined in terms of the latest "address change event" that happened for that person. As another example, the monthly salary of an employee may be defined as the sum of all his/hers earnings on jobs according to "job report" events for that month.

An entity "exists" for an application only over specific time intervals. Each entity set, therefore, has an associated existence rule. Existence rules are defined in terms of events. For example, a hiring event signals the beginning of the existence of an employee.

To each event type is associated an occurrence condition, which specifies when it is to occur. An occurrence condition for a repair event is "a repair request has been made and there is enough capacity in the shop". An occurrence condition for a payroll event might be "the first day of each month". If an event's occurrence is to be triggered by a user (as in a change of address), and not by the system's internal knowledge, then the occurrence condition is labelled "external". Note that the appropriate time resolution for the address change example above is days. So an address change event is associated with the time type "days". In general, there is a date function for each event type that maps the events of that type into some time type.

These topics are further discussed in Bubenko [1981].

50. A. Sernadas, "Temporal aspects of logical procedure definition," Information Systems Vol. 5(3) pp. 167-187 (1980).

A model or a specification of an information system is said to be "memory independent" if it is not concerned with what the system is to memorize (or remember from its past), and what it should forget. A system designer using such a model need not worry about what facts to keep around for future use; such decisions are relegated to a lower level of design. This paper proposes a memory independent specification language for describing information systems at the message/process level. This is to be contrasted with an object level specification where the application world itself is modeled. In describing a process, the designer may need to refer to the past history of the information system. Extensive tools based modal logic are provided for such reference.

Two kinds of objects are manipulated by processes: state messages, and event messages. States are facts that are true over intervals of time. Events happen at discrete times. There are three atomic operations that processes can perform: signal the beginning of a state message, signal the end of a state message, and generate an event message. Processes are built from these operations by using the control structure, "and" (for parallel execution), and conditionals. Each process type has a number of formal parameters. For example, a process used to deal with back orders may have a product number and an order number as its formal parameters. The formal parameters are "quantified" by "parallel" (P), and "sequential" (S) quantifiers. For the back order example, the quantification "P(product#)S(order#)", would mean "for each product in parallel, process its back orders in sequence".

Considerable attention is paid to the development of a language for posing queries about the history of the information system, to be used in conditionals. For this purpose a temporal logic is used which is based on modal logic. The logic includes a variety of operators for time reference. Examples are operators for "always" and "sometimes", "before", and "after", and combinations of these, such as "always after". Also included are time generation operators. Among these is a way of specifying the set of times at which some fact is true, of specifying a time interval by its end points, and constants such as "THISWEEK", and "1980".

The back order processing example is among several examples illustrating these constructs. It is assumed that the price to be used for billing a back order is the price in effect at the time the order was placed, and that back orders for a particular stock item should only be filled if a replenishment notice occurred after the back order. Thus, the process description for this example involves facts at three different times, the times of an order, a back order, and a replenishment notice.

51. J.F. Allen, "Maintaining knowledge about temporal intervals," Tecnical Report 86, Dept. of Computer Science, The University of Rochester, Rochester, NY 14627 (January 1981).[10]

This paper describes a method for maintaining a network of relationships between a set of temporal intervals. The processing of relative temporal relationships is emphasized, and a number of relations are defined for this purpose. The representation includes the notions of the "present moment", and "persistent intervals" (intervals extending indefinitely into the past or future).

There are two extreme methods of representing the temporal relationships. On the one hand, one can explicitly represent only those relationships that are entered as input by a user. Other relationships would then be inferred procedurally. On the other hand, one can explicitly represent all the relationships between two intervals that can be deduced. In that case, whenever the user enters a new relationship between two intervals, this new "constraint" would have to be "propagated" throughout the network. Allen proposes a compromise representation based on the notion of "reference intervals" (see also Kahn [1975]). The idea is to define clusters of closely related intervals based on their being within interesting (user defined) reference intervals, and to propagate constraints only through each cluster.

It is also shown how such a scheme can be used to limit the propagation effects of updating the ever changing "now interval".

52. D. McDermott, "A temporal logic for reasoning about processes and plans," research report #196, Dept. of Computer Science, Yale University (March 1981).

This work draws upon the works of Hayes[11] and Moore.[12] Hayes advocates the development of theories that are rich in common sense knowledge for reasoning about the physical world. He calls such theories "naive physics". Moore has based his analysis of the interaction between knowledge and action on the "possible worlds" interpretation of the modal operator "know". In his approach, all statements involving someone's knowledge are converted to statements about the possible worlds that are compatible with what that person knows. It turns out that the possible worlds formulation simplifies the theory conceptually, and renders it computationally feasible. McDermott's logic is a "naive physics" of time a la Hayes, which uses the possible worlds approach.

The logic specifies the properties of a set of states: possible instantaneous worlds. States are ordered by the "no later than" order relation, "<=". There is also a date function, mapping each state into a real number. States are arranged into chronicles: possible temporal worlds. Chronicles are totally ordered convex (gapless) sets of states. Convexity means that any state that is between two states of a chronicle must itself belong to the chronicle. Chronicles can only branch into the future.

A fact is some statement of the predicate calculus. It can be thought of as the set of states for which it is true. An event can be identified with a set of time intervals, those intervals which embodied the event once. The subset relation between events can then be used to indicate

10. See also:
J.F. Allen, "An interval-based representation of temporal knowledge," pp. 221-226 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

11. P.J. Hayes, The naive physics manifesto, Department of Computer Science, University of Essex, Colchester, Essex (May 1978).

12. R. Moore, Reasoning about knowledge and action, Technical Report 191, Artificial Intelligence Center, SRI International, Menlo Park, California (October 1980).

implication.

Among the many everyday notions that this logic attempts to formalize (by appropriate axioms) are the notions of causality and change. One notion of causality is that one event always follows another. To capture this notion, a predicate "ecause" is defined that takes the two causally related event types, a delay duration, and an "exception" predicate. The meaning of this predicate is that the second type of event always follows the first within the duration unless the exception predicate becomes false.

Another notion of causality is that an event or an action causes a fact to be true. But events rarely cause instantaneous facts. Rather, they cause persistences, facts that are true over extended durations. A complication in working with persistences is that we can only rely on them in the absence of any information to the contrary that might crop up later. Thus, we can assume that a boulder, having fallen down a valley will remain there for some time unless we are told otherwise at a later time.

In order to think about changes, the notion of a thingum is introduced. A thingum corresponds to an intentional object, for example, "the President of the United States" is a thingum. A thingum has an associated value that is subject to change over time.

The paper also discusses a number of relationships between actions. Actions can be composed from other actions by sequence, parallel, and other programming language constructs. They can also be composed by things like "avoid", that is, an action can be avoided by another action. A task is an action to which a problem solver is committed. It is argued that a problem solver should not only have a plan for the action, but also have a representation for what has actually been done while trying to perform that action (i.e., for the task), since many plans are not or can not be carried out as specified.

53. G. Ariav, and H.L. Morgan, "MDM: Handling the time dimension in generalized DBMS", Working Paper, Dept. of Decision Sciences, The Wharton School, University of Pennsylvania (May 1981).

The authors designed and implemented a system called DATA (Dynamic Alerting Transaction Analysis System), in which time is handled through a concept they called "Modal Data Management". The time model is linear according to a "date line", absolute (independent of an observer), and explicit: recorded events contain an explicit, precise, and unified time. It is also "non-hierarchical", that is, time periods and their interrelationships are not dealt with.

The DATA system is a relational system which can provide users with a view of the database as it existed at a previous point in time. The system does not maintain the traditional correspondence between "records" and real world entities. Rather it contains cumulative, time ordered, lists of transactions. In this way, the status of an entity at any given point of time can be extracted from the collection of transactions relating to that entity prior to this time. Transactions are always appended to the database; they are never modified or deleted. They can be one of three kinds: addition and deletion of an entity, and modification of an attribute of an existing instance of an entity. Each transaction contains its time stamp and the new value of the data changed by the transaction.

The system has been operational since 1978. It is based on differential files. Following are some examples commands dealing with time.

* Rundown - "replays" a sequence of recorded events between two points in time.

* Settime - allows the database to be viewed from previous points in time.

* Rollback - backs up the database to permit operations on its contents at some previous point in time.

54. A. Bolour and L.J. Dekeyser, "Time relations and event relations," submitted for publication (July 1981).

This paper is concerned with the representation and processing of historical information, e.g., patient histories in a medical data base. The first part of the paper considers the dichotomy between relative timing statements and absolute timing statements. When seen from the point of view of classical first order logic, this dichotomy becomes one example of a general dichotomy between the use of a set of predicates in making statements, and the use of a model for a logical theory on these predicates to make statements. It is argued that this dichotomy is general enough that it deserves special attention in semantic data modelling. A number of generic classes of the absolute/relative dichotomy are presented. For each, a generic set of predicates, a logical theory on this set, and a model for this theory are described. The use of data base modules that know about each theory and its corresponding model, and can use this knowledge in answering queries, and in enforcing integrity constraints is

suggested.

The second part of the paper deals with the representation of events, such as surgical operations, in terms of predicates. Some of the positions of an event predicate can be used to specify the participants of an event. But often such a participant only participates in an event during some subinterval of its own and of the event's existence. To simplify the representation of events, it is proposed that some positions of an event predicate should be interpreted as "temporal parts" of entities (i.e., as the existences of entities over particular time periods).

55. K.D. Forbus, "Qualitative reasoning about physical processes," pp. 326-330 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (Aug. 1981).

56. J.L. Kolodner, "Organization and retrieval in a conceptual memory for events," pp. 227-233 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

57. T. Koyama, S. Kayhara, T. Minamikawa, and T. Kurakawa, "Time-oriented features for medical consultation systems," pp. 910-912 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

58. H. Marburger, Bernd Neumann, and H-J. Novak, "Natural language dialogue about moving objects in an automatically analyzed traffic scene," pp. 49-51 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

59. E. Mays, S. Lanka, A.K. Joshi, and B.L Weber, "Natural Language interaction with dynamic knowledge bases: monitoring as response," pp. 61-63 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

60. V. Sembugamoorthy, "Analogy-based acquisition of utterances relating to temporal aspects," pp. 106-108 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

61. J.K. Tsotsos, "Temporal event recognition: an application to left ventricular performance," pp. 900-907 in Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, British Columbia (August 1981).

62. V. De Antonellis, and B. Zonta, "Modelling events in data base applications design," pp. 23-31 in Proceedings of the Seventh International Conference on Very Large Data Bases, Cannes, France (September 1981).

63. M.R. Klopprogge, "TERM: An approach to include the time dimension in the entity-relationship model," Second International Conference on the Entity Relationship Approach, (October 1981).

TERM is a PASCAL-like language for the definition and manipulation of historical data. It extends the entity relationship model of data with modeling constructs that deal with time and time-dependent information.

The paper begins with example applications exhibiting time-dependent data modeling problems. These problems include the description of discrete versus continuous data, differences in sampling rates, measurement inanccuracy, and the derivation of missing values for attributes.

The basic data modeling constructs in TERM are called "structures". They are similar to abstract data types, except that their definition may also include the specification of a predicate that must be true for each element of the structure. Three different types of structure are described: time structures, value structures (for the values of attributes and roles), and history structures. A history is thought of as the representation of the development of some fact over time. A history is defined as a possibly partial function from a time structure to a value structure. Each corresponding pair of values in a history is called a "state". When a history has an infinite number of states, its representation consists of a set of explicitly represented "characteristic states", and a function for the derivation of other states from these. A special "existence" history is also defined, mapping object-time pairs into boolean values.

The basic operations of the model are "registration" and "correction". There are two registration operations: "initiation", for the addition of new entities and relationships, and "completion", for the addition of attribute, role, and existence values to entities and relationships at new points in time. Corrections, on the other hand, are the alteration of erroneous values.

An appendix gives examples written in the TERM

language.

64. J. Clifford, and D.S. Warren, "Formal semantics for time in databases," Technical Report 81/025, Department of Computer Science, SUNY, Stony Brook, NY 11794 (November 1981).

65. T.L. Anderson, "The Data Base Semantics of Time," Ph.D. Dissertation, University of Washington (1981).

This work is founded on Abrial's conceptual data model. There are three principal concepts in this model: categories (of objects), binary relations, and programs. Four types of operations: insertion, deletion, test of existence, and iterative access, are defined for categories and for relations. Each of these operations has a "standard semantics": a given associated program that works independently of a particular category (or relation). A principal contribution of Abrial was the provision of a facility for defining an "extended semantics" of these operations for each particular category or relation. In this approach, the applications programmer is allowed to provide programs that would be run prior to, or in place of the standard programs. For example, in a test from a relation that is known to be transitive but whose tuples are not all represented explicitly, the programmer can specify a program to test whether two objects are transitively related to each other. The work of Levesque augments part of this framework with the notion of generalization. The standard semantics are extended to use generalization knowledge whenever possible.

In attempting to use this foundation to describe the data base semantics of time, Anderson encountered a number of anomalies to which she proposes solutions. For example, when iterative access to a relation has an added extended semantics to access objects implicitly related to a given object, only one instance of each related object should be retrieved, even though it may be returned by both the standard and the extended semantics programs. Anderson also adds a way of specifying the number of objects related to a particular object by a given relation, though the actual objects may be unknown. The resulting model is used to define the semantics of time as follows. A number of categories and relations are provided to describe temporal information. A number of constraints on the relations are described, which are built into the extended semantics of the operations on these relations. Thus, when an application program uses these categories and relations to describe a particular time category, all the constraints are automatically taken care of. Many of the constraints specify one category or relation as a generalization of another.

The basic elements of time come from a discrete totally ordered set of time points, T_c (the "c" indicates the context in which this is to be used). The points in T_c may, for example, represent working days. The total ordering relation is represented as "$<_c$", and is a particular instance of the "before" relation. Thus, the relationship between "$<_c$" and "before" is a generalization. (In contrast, Bolour and Dekeyser [1981] regard the relationship between "before", and "<", as the relationship between a temporal predicate and its corresponding relation in a model of time.) The basic notion of duration is a function d_c from T_c into the reals. Normally, this is identically 1^c (each element of T_c is one unit long). But the user may specify some non-standard semantics for it. There is a function "now": context -> $\{T_c\}$, and a special object "current time" that is used whenever the "now" function is to be evaluated. Intervals are defined as contiguous elements of the superset of T_c. They are represented by pairs of times. The duration function for intervals is a metric on their representation as a pair of times. The metric is the sum of the duration of all the points between the two end points. Intervals may have unknown and also "current time" beginning and ending points. Periodic times are defined as elements of the superset of the superset of T_c. Periodic times are characterized by three things: a cover interval, a period of repetition, and a duration of each repetition. A number of binary relations are described for times: "before", "equal", "time in", "earliest time in", "latest time in", "begin", "end" (of intervals), "duration", with obvious interpretations.

The basis for describing processes in Anderson's thesis is the notion of a Graph Model of Computation. There are a set of nodes, representing the possible events in the process, and a set of "super arcs" that are pairs of sets of arcs. Each such arc has two attributes: which may be called "origin completion," and "destination activation." The attributes may take on one of the values "AND" and "OR". The basic idea is that when one (OR) or all (AND) of the events in the origin have completed, then one (OR) or all (AND) of the events in the destination can be activated. A basic notion of concern for Anderson is that of a legal sequence of events, i.e., a legal sequence of node visits in the graph. Parallelism can be represented by having an event in a process begin before another event in the process ends. A process is an event that can be decomposed into other processes and events.

There are obvious precedence constraints inherent

in the graph model. But independently of these, constraints can be specified for the sequentiality and the mutual exclusion of events in a process. Mutual exclusion is discussed in some detail, and particular versions of it called "non-reentrant", and "object non-reentrant" are introduced for processes that cannot have more than one instance in existence at a given time, and processes that cannot have more than one instance with the same object at a given time. Other constraints have to do with the duration of events. A provision is made for the user to specify the maximum and minimum durations of events. Also, the time of an event must be included in the time of an encompassing process.

66. J.A. Bubenko, Jr., "On concepts and strategies for requirements and information analysis," SYSLAB Report No. 4, Department of Computer Science, Chalmers University of Technology (1981).

67. P. Needham, "Temporal intervals and temporal order," Logic and Philosophy, (1981).

68. T.L. Anderson, "Modeling time at the conceptual level," in Proceedings of the Second International Conference on Databases (to appear), Jerusalem, Israel (June 1982).

69. P.J. Hayes, "Histories: the naive physics manifesto part two," in preparation.

Author Index