A SIMPLE, GENERAL STRUCTURE FOR TEMPORAL DOMAINS

by

James Clifford
and
Ahobala Rao
Information Systems Area
Graduate School of Business Administration
New York University
90 Trinity Place
New York, N.Y. 10006

October 1986

Center for Research on Information Systems
Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #154
GBA #87-42

In Proc. Temporal Aspects in Information Systems, Sophia-Antipolos, France, May 1987.

A Simple, General Structure for Temporal Domains

James Clifford and Ahobala Rao
Information Systems Area
Graduate School of Business Administration
New York University

October 1986

In Proc. Temporal Aspects in Information Systems, Sophia-Antipolis, France, May 1987.

Abstract

Much recent research has focused on the need for, and definitions of, historical (or temporal) database and information systems to serve expanding information needs in a variety of applications. Almost all of this research has assumed that the domain of time itself was well-understood, and some representation for it simply needed to be included in the model to provide the needed temporal dimension. In this paper we present a simple, set-theoretic structure for a time domain which is independent of any particular calendric system. We concentrate on the general structure and operations necessary to support the needs arising in modelling time in information systems.

1. Introduction

Over the past several years there has been growing research interest in developing models for information systems that might provide facilities for structuring information with a temporal dimension and for accessing and managing information so structured. In a survey in 1982 [Bolour et al. 82] roughly 50 references were included; work done since that time has more than doubled that figure.

Within the classification scheme for such work proposed in [Ariav et al. 83], namely:

- Conceptual Data Modelling
- Systems Design and Implementation
- Dynamic Database
- AI-related Research

the present work falls into the first sub-category, that of data modelling. However, it address an issue that, while recognized as important by most research in the area, has almost universally been given only minor attention. We speak of the *nature* of the concept of **time** itself, i.e., what **structure** is appropriate for representing the concept of time in temporal information systems, what **operations** are needed on this domain, and ultimately what data structures and algorithms can we devise for **implementing** such operations efficiently.

The issue of the nature of the time domain arises most sharply when we attempt to define operations in a temporal information system, as for example in [Clifford 85] and [CliffordCroker 87]. For example, a particular database might represent salary history by an attribute SALARY with a temporal domain

based on **days**, and sales figures by an attribute SALES with a temporal domain based on **months**, as in Figure 1-1, using the structures described in the Historical Relational Data Model (HRDM) described in [Clifford 85] and [CliffordCroker 87].

Figure 1-1: Example Time-Series Database Attributes

Consider how the system would understand a request to **join** two such time series, or, how the system could interpret a request from a user *who does not know the way the information is structured* for the salary of an employee in a given **month**. To provide these operations, the system must clearly have a rich and well-defined structure for a set of inter-related temporal domains, and a rich set of well-defined and versatile operations for elements, sets, and intervals of time. This paper represents a proposal for such a structure, based upon naive set theory and algebra.

It is important to single out the work of Anderson [Anderson 81] as perhaps the first database researcher to study this problem in some depth and to propose a solution. Her work is both less general and more user-oriented than this work. The present work may be regarded as an attempt to present formally a completely general view of a temporal domain, in a structure that is at once simple yet powerful enough for its intended use. We believe that only after developing such a formal and "complete" view of time is it appropriate to begin designing the user interface.

2. Temporal Domains

2.1. Preliminary Definitions

Given a set $S = \{ \ldots , s_0, s_1, \ldots \}$, a **total order** on S, denoted $<$, is a relation on S satisfying the property that for any two elements s_i and s_j in S, either $s_i < s_j$, $s_j < s_i$, or $s_i = s_j$. The relation is simply the set of ordered pairs, i.e. $< = \{ <s_i , s_j > \mid s_i < s_j \}$. We will refer to S as a **totally ordered set.**

If $S = \{ s_0, s_1, \ldots , s_n \}$, i.e. S is finite, then we can define the operators **first**(S) and **last**(S) in the obvious way:

first(S) = that $s_i \in S$ such that $\forall s_j \in S \; [s_i <= s_j]$

last(S) = that $s_i \in S$ such that $\forall s_j \in S \; [s_j <= s_i]$

Given a totally ordered set S, a **finite closed interval** of S is a set $S' = \{s_1, s_2, \ldots\ldots s_n\}$ such that:

1. S' is finite
2. $S' \subseteq S$
3. $\forall y \in S \; [\text{first}(S') \leq y \leq \text{last}(S') ==> y \in S']$

Given a set $S_0 = \{ \ldots , s_{0,0}, s_{0,1}, \ldots \}$, with $<$ a total order on S_0, a **constructed intervallic partition (CIP)** of S_0 is an ordered pair $<S_1, \psi_0^1>$ where:

1. $S_1 = \{ \ldots, s_{1,0}, s_{1,1}, \ldots \}$
2. $\forall x \; [x \in S_0 \text{ -> } x \notin S_1]$
3. ψ_0^1: S_1 -> 2^{S_0} is a mapping such that:
   a. ψ_0^1 maps each element in S_1 to a finite closed interval of S_0, and
   b. $\cup \psi_0^1 (S_1) = S_0$

These properties together say that S_0 and S_1 are disjoint, that each $s_{1,j}$ in S_1 can be mapped to a subset of the elements of S_0, in particular to a *contiguous interval* (under $<$), and that all of S_0 is "covered" under the mapping. Thus, under the mapping S_1 can be viewed as partitioning S_0. We note that by definition, ψ_0^1 maps each element in (S_1) to a finite (sub)set of S_0. We can therefore consider S_1 to be totally ordered as well, inheriting the total order of the underlying set S_0. In particular, we can define the

derived relation $<'$ as: $s_{1,i} <' s_{1,j}$ iff $\text{last}(\psi_0^1(s_{1,i})) < \text{first}(\psi_0^1(s_{1,j}))$.

For simplicity we will drop the ' and simply use the symbol $<$ for this derived relation as well.

Note also that if S_1 is a CIP of S, then as a consequence of the definition, for every element s in S_0 there is one and only one element s' in S_1 such that $s \in \psi_0^1(s')$.

2.2. Temporal Domains

The structure that we propose for a set of temporal domains suitable for the needs of an information system is a set of CIPs built upon some base set which represents the smallest observable or interesting time units in the application.

Let $T_0 = \{ \ldots, C_0, C_1, \ldots \}$ be the set of *chronons*, or non-divisible time intervals. Each C_i is indivisible, and $<_{T_0}$ is a total order on T_0. It may be helpful to think of T_0 as (some subset of) the natural numbers.

A **Temporal Universe** is a finite sequence **TU**, $\mathbf{TU} = <T_0, T_1, \ldots, T_n>$ such that

1. $\forall i \forall j, [i \neq j \rightarrow T_i \cap T_j = \emptyset]$
2. each T_{i+1} is a Constructed Intervallic Partition of T_i.

These properties state that the **Temporal Universe** is finite and forms an *ordered* set, each set beyond the initial set of chronons being a CIP of the previous set. For example, $T_1 = \{ \ldots, T_{1,0}, T_{1,1}, \ldots \}$ is an intervallic partition of T_0. We refer to each T_i in the **Temporal Universe** as a **time domain**.

Again we point out that as a consequence of this definition, every element of T_i "belongs to" (under the mapping function ψ) exactly one element of T_{i+1}. The well-known, difficult problem of the domain of "weeks," [Colson 26] which can overlap months and years, is thereby ruled out as a time domain by our definition. In the "subset chain" seconds $\subseteq$ minutes $\subseteq$ hours $\subseteq$ days $\subseteq$ months $\subseteq$ years, there is nowhere to place weeks. Thus, our proposed structure is capable of modelling all of the ordinary time units except weeks.

Figure 2-1 shows an example of a **Temporal Universe**; the more common names for the elements in these sets are shown in Figure 2-2.

$\mathbf{TU}_{1985}$ = { T_0, T_1, T_2, T_3, T_4, T_5 }, where:

T_0 = { S_0, ... , $S_{31535999}$ } is the set of "seconds" in 1985

T_1 = { M_0, ... , M_{525599} } is the set of "minutes" in 1985

T_2 = { H_0, ... , H_{8759} } is the set of "hours" in 1985

T_3 = { D_0, ... , D_{364} } is the set of "days" in 1985

T_4 = { MO_0, ... , MO_{11} } is the set of "months" in 1985

T_5 = { Y_0 } is the set consisting of the single entire "year" 1985

Figure 2-1: An Example Temporal Universe

00:00:00 - 00:00:01 January 1, 1985 for S_0

23:59:59 - 24:00:00 December 31, 1985 for $S_{31535999}$

00:00:00 - 00:01:00 January 1, 1985 for M_0

23:59:00 - 24:00:00 December 31, 1985 for M_{525599}

00:00:00 - 24:00:00 January 1, 1985 for H_0

00:00 - 24:00 December 31, 1985 for H_{8759}

January 1, 1985 for D_0

December 31, 1985 for D_{364}

January 1985 for MO_0

December 1985 for MO_{11}

1985 for Y_0

Figure 2-2: Common Names for Time Elements in Example

Given a Temporal Universe $\mathbf{TU} = \{ T_0, T_1, \ldots, T_n \}$, we typically wish to assign names to the units at each "level." We can make these ideas precise:

- A time element (or interval) $t \in T_i$ is said to be at <u>level</u> i.
- The notion of units can then be characterized by a set UNITS $= \{ U_0, U_1, \ldots, U_n \}$, with the obvious correspondence to the n elements of **TU**. The term **chronon**, is sometimes used also to refer to the unit of T_0, viz. U_0.
- The often ill-defined term "*granularity*" can also be made precise in this system: The **granularity** of a given time element (or interval) t is simply the unit of its level.

Returning to our example in Figure 2-1, we see that $\text{UNITS}_{\text{TU}_{1985}}$ = { seconds, minutes, hours, days, months, year }.

Note that the concept of a **chronon** allows us to circumvent the issue of continuous time, and also to thereby avoid the messiness of half-open intervals. Time in this model is a chain of successively refined partitions, as illustrated in Figure 2-3.

Figure 2-3: A **Temporal Universe** as Successively Refined Partitions

It is only at the lowest level of this hierarchy that we have to concern ourselves with the question, is there anything smaller? That is, in the interval $[C_i, C_{i+1}]$ is there a C_j such that $C_i < C_j < C_{i+1}$? If there is, then we should more properly describe the intervals from C_i through C_{i+1}, and from C_{i+1} through C_{i+2} as $[C_i, C_{i+1})$ and $[C_{i+1}, C_{i+2})$. But note that the size of the chronon in any **Temporal Universe** is determined by the user, based on the needs of the application. It could be a day, a second, a millisecond, a nanosecond, or whatever you like. Once this choice is made, for all practical purposes there *is nothing smaller*, and then there is no longer any reason to consider the real numbers as a model -- the **TU** is isomorphic to the natural numbers. At the lowest level, the universe is a totally ordered set of discrete chronons, and the sets $\{ C_0, C_1, \ldots, C_5 \}$ and $\{ C_6, C_7, \ldots, C_{10} \}$, for example, are properly written as $[C_0, C_5]$ and $[C_6, C_{10}]$. One may *think* of a chronon, say C_i, as a continuous, dense interval, but within the algebraic structure each C_i is considered to be indivisible, and so these sub-moments do not exist.

We note in passing that the definition of the structure of a **Temporal Universe** of in bottom-up, from the chronons to the largest time units. When we look at the structure top-down, it appears as successive refinements of a partition, as in Figure 2-3.

3. Operations on the Time Domains

Given a **Temporal Universe TU** = { T_0, T_1,.....T_n} we will now look at the kinds of operations that are possible both on elements at the same level, as well as between elements at different levels. Figure 3-1 illustrates how all the levels in the Temporal Universe, and the mapping functions ψ_k^j, are related.

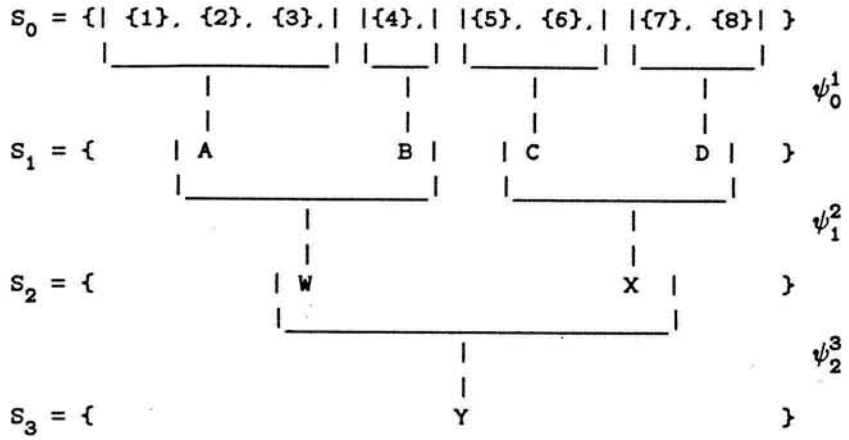

Figure 3-1: An Example of Time Levels and Mapping Functions

We will discuss three types of operators, at first a simple version which operates on objects all on the same level, and then extended versions of these which handle operands at different levels in the time hierarchy. The operators we discuss will be in two different categories:

1. Boolean Predicates on Time elements, i.e., functions which map into {0,1}
2. Unary and Binary Operators

In order to keep all of the definitions uniform it will in most cases be simpler to consider most operations as operations on *subsets* of a temporal domain, rather than on *individual elements*. (The obvious isomorphism between the set of elements at any level and the set of all singleton sets of elements provides the mapping back to individual elements, if desired.)

3.1. Operations At a Single Level

It is apparent that at any given level we are dealing with the familiar algebra of subsets. In other words, at level i, the structure $<2^{S_i}, \{\cup, \cap, -\} >$ is the Boolean algebra of subsets of S_i, and is therefore closed under all of these operations.

For individual elements at level i, the following results are obvious:

$$\{s_{i,m}\} \cup \{s_{i,n}\} = \{s_{i,m}, s_{in}\}$$

$$\{s_{i,m}\} \cap \{s_{i,n}\} = \emptyset$$

$$\{s_{i,m}\} - \{s_{i,n}\} = \{s_{i,m}\}$$

For subsets of elements at a given level, the set operators: $\subseteq$, $\supseteq$, $=$, $\in$, etc. also behave as expected.

Moreover, at any level i all of the comparators $<$, $>$, $=$, $<>$, $<=$, $>=$ are available because of the the underlying total order on S_0.

For example, we can determine whether element $s_{i,m}$ comes "before" $s_{i,n}$ by using the relation $<_i$ derived from the given order $<$. An example of some of the different types of operations follows. (Note that [Snodgrass 84] discusses some of these operations with respect to his language TQuel.)

1. Predicates: these operators are mappings in the function space: $2^{S_i} \times 2^{S_i} \rightarrow \{0,1\}$.

   a. **Precede?**

   $$X_i \textit{ precede } Y_i = \begin{cases} 1 \text{ if first}(X_i) < \text{first}(Y_i) \\ 0 \text{ otherwise} \end{cases}$$

   (Precede? is therefore just $<$.)

   b. **Overlap?**

   $$X_i \textit{ overlap } Y_i = \begin{cases} 1 \text{ if } X_i \cap Y_i \neq \emptyset \\ 0 \text{ otherwise} \end{cases}$$

2. Unary Operators on Subsets:

   a. **First** : $2^{S_i} \rightarrow S_i$.

   first(X_i) = the first element (wrt $<$) in the subset X_i

b. **Last** : $2^{S_i} \rightarrow S_i$.

$last(X_i)$ = the last element (wrt $<$) in the subset X_i

3. Unary Operators on Elements:

a. **Successor**: $S_i \rightarrow S_i$.

$$successor(s_{i,m}) = \begin{cases} \text{smallest } s_{i,n} \text{ s.t } s_{i,m} < s_{i,n} \text{ if } \exists\, s_{0,x} > \text{last}(\psi_0^i(s_{i,m})) \\ \text{undefined otherwise} \end{cases}$$

b. **Predecessor**: $S_i \rightarrow S_i$.

$$predecessor(s_{i,m}) = \begin{cases} \text{largest } s_{i,n} \text{ s.t } s_{i,n} < s_{i,m} \text{ if } \exists\, s_{0,x} < \text{first}(\psi_0^i(s_{i,m})) \\ \text{undefined otherwise} \end{cases}$$

4. Binary Operators on Subsets: these operators are mappings in the function space: $2^{S_i} \times 2^{S_i} \rightarrow 2^{S_i}$.

a. **Overlap**

X_i *overlap* $Y_i = X_i \cap Y_i$

(So overlap is just $\cap$.)

b. **Extend** (defined only over finite closed intervals)

X_i *extend* $Y_i = \{s_{i,n} \mid \text{first}(X_i) \leq s_{i,n} \leq \text{last}(Y_i)$

Additionally, we could define special operators like **precede**$_n$ for any n, to allow reference to, e.g., "the same month last year" (precede$_{12}$ on the month level), or "the same day last month" (which would have to know about the *sizes* of each month).

3.2. Operations Across Levels

In the case of operations between elements at different levels in the **Temporal Universe**, we need to map elements to a common level before performing the operation.

For this purpose, based on the mapping functions given in the definition of the Temporal Universe, we can define ψ_k^j for all $0 < j \leq n$, for all $0 \leq k < n$, where j and k represent levels as :

$$\psi_k^j(x_j) = \begin{cases} \text{undefined if } k > j \\ x_j \quad \text{if } j = k \\ \psi_k^{k+1}\{ \ldots (\psi_{j-2}^{j-1} (\psi_{j-1}^j (x_j)))) \quad \text{otherwise} \end{cases}$$

As before, we can extend this function to sets of elements at any level in the obvious way. Let $X_j = \{x_{j,1}, x_{j,2}, \ldots x_{j,n}\}$ be a set of elements at level j. Then:

$$\psi^j_k (X_j) = \bigcup_{i=1}^{n} (\psi^j_k (x_j))$$

This function ψ^j_k simplifies the definition of the various operations in the algebra.

Comparisons by order: the operators which are used for elements at different levels are simple extensions of the single level operators, and can be defined in terms of them and the mapping functions. We denote these derived operators as: $\Theta = \{ <^\circ, \leq^\circ, >^\circ, \geq^\circ, =^\circ, \neq^\circ, \text{overlap}^\circ \}$. Each operator θ° in Θ is in the function space: $\theta : 2^{S_i} X\, 2^{S_i} \rightarrow \{0,1\}$.

In general, we can define each such θ° as:

$$X_i\ \theta^\circ\ X_j = \begin{cases} \psi^i_j(X_i)\ \theta\ X_j \text{ if } i \geq j \\ X_i\ \theta\ \psi^j_i(X_j) \text{ otherwise} \end{cases}$$

The binary set constructors: $\cup^\circ, \cap^\circ, -^\circ$, and extend$^\circ$ are all operators in the function space: $\theta : 2^{S_i} X\, 2^{S_i} \rightarrow 2^{S_i}$.

In general, we can define each such operator θ° as:

$$X_i\ \theta^\circ\ X_j = \begin{cases} \psi^i_j(X_i)\ \theta\ X_j \text{ if } i \geq j \\ X_i\ \theta\ \psi^j_i(X_j) \text{ otherwise} \end{cases}$$

Two operators that map between levels can also be defined, viz. **start-of** and **end-of**, which map an element at level i+1 to an element at level i, i.e. $S_{i+1} \rightarrow S_i$.

1. **start-of**$(s_{i,n})$ = first$(\psi^i_{i-1}(s_{i,n}))$, $i > 0$.
2. **end-of**$(s_{i,n})$ = last$(\psi^i_{i-1}(s_{i,n}))$, $i > 0$.

There are many interesting properties of the **Temporal Universe** structure that we have defined here. We state a few of them here.

Theorem 1: $X_i \cup^\circ Y_i = X_i \cup Y_i$. and is an element of 2^{S_i}.

Theorem 2: $X_i \cup^\circ Y_j \in 2^{S_i}, \forall i\ [i \leq j]$.

As a Corollary to the above, we have:

Corollary 3: $X_i \cup^{\circ} Y_j \notin 2^{S_j}, \forall i\,[i \leq j]$.

Theorem 4: $\forall i\, |S_{i+1}| \leq |S_i|$

3.3. Units Revisited, or "What time did you say it was?"

The various constructor operations, under the preceding definitions, yield a unique result. Nevertheless, the question remains as to the appropriate *level* at which to display the result of an operation, say a OP b, to the user. What, for example, is the result of the **extension** of *Jan 31, 1980* and *9:00 a.m. March 3, 1980*?

We are currently working on an algorithm to provide the "best" name for the result of any constructed time element. This algorithm is related to the integer division algorithm, and also to the problem of giving correct change. The familiar relationships in currency:

tens --> fives --> ones --> quarters --> nickels --> pennies

are analogous to our structure for a **Temporal Universe.**[*] One difference is that with coins, there is only one *kind* of element at a given level, e.g. all quarters are the same size; there is no restriction of this kind in a **Temporal Universe**, and indeed, in the familiar time domain of months we have different sized intervals. Otherwise the problem is similar to the problem of giving "the smallest number of coins in the correct change?"

Also, in a Temporal Universe, the *boundaries* are crucial; this is not true with change example. The reason is that the "cents" of money are unordered, but the chronons of time, and hence by extension all of the "intervals," are also ordered. In other words, you can't just throw any old 60 seconds together to get a minute.

Nonetheless the algorithm appears to be fairly simple to compute, for a given interval of chronons, the "most concise" representation. For example (see Appendix), in TU_{1985}, the interval $[S_0,S_{3974399}]$ is just $[M_0,M_{66239}]$ since $M_i = [S_{60*i},S_{(60*i)+59}]$; but this is just $[H_0,H_{1103}]$, since $H_i = [M_{60*i},M_{(60*i)+59}]$; this in

[*] Here dimes are analogous to weeks, they are not comparable with quarters.

turn is $[D_0,D_{45}]$, since $D_i = [H_{24*i},H_{(24*i)+23}]$, which, finally, is just [January 1 1985, February 15 1985] since January 1985 is $MO_0 = [D_0,D_{30}]$ and February 1985 is $MO_1 = [D_{31},D_{58}]$

The basic idea of the algorithm is similar to the solution to the coins problem. Find the number (and names) of the largest units first, then look at the "overhangs" on the left (<) and on the right (>) and recursively find the largest units for these, etc.

4. IV. Summary

We have presented a general structure for time domains, a **Temporal Universe**, have defined operations on that structure, and have discussed some of the properties of this structure. While independent of any particular calendric system, we believe that this structure can model the usual systems and is appropriate for incorporation into information systems that model time. Further work is needed to incorporate the notion of the "week" into the structure without unduly complicating its simplicity. Moreover, additional work is required to develop appropriate data structures and algorithms for an implementation of this structure in a practical system.

Appendix

For completeness, we include the following description of our example **Temporal Universe, TU_{1985}**:

1. $T_0 = \{ S_0, \ldots, S_{31535999} \}$ is the set of "seconds" in 1985
2. $T_1 = \{ M_0, \ldots, M_{525599} \}$ is the set of "minutes" in 1985, where $M_0 = [S_0,S_{59}]$, ... , $M_{525599} = [S_{31535940},S_{31535999}]$, and in general, $M_i = [S_{60*i},S_{(60*i)+59}]$.
3. $T_2 = \{ H_0, \ldots, H_{8759} \}$ is the set of "hours" in 1985, where $H_0 = [M_0,M_{59}]$, ... , $H_{8759} = [M_{525540},M_{525599}]$, and in general, $H_i = [M_{60*i},M_{(60*i)+59}]$, or equivalently, $[S_{60*(60*i)},S_{60*((60*i)+59)+59}]$.
4. $T_3 = \{ D_0, \ldots, D_{364} \}$ is the set of "days" in 1985, where $D_0 = [H_0,H_{23}]$, ... , $D_{364} = [H_{8736},H_{8759}]$, and in general, $D_i = [H_{24*i},H_{(24*i)+23}]$, or equivalently, $[M_{60*24*i},M_{(60*((24*i)+23))+59}]$, or again equivalently, $[S_{60*60*24*i},S_{(60*((60*((24*i)+23))+59))+59}]$.
5. $T_4 = \{ MO_0, \ldots, MO_{11} \}$ is the set of "months" in 1985, where $MO_0 = [D_0,D_{30}] = [H_0,H_{743}] = [M_0,M_{44639}] = [S_0,S_{2678399}]$ /* January 1985 */

   The hours, minutes, and seconds for the following months are not given; they are easily computed.

   $MO_1 = [D_{31},D_{58}]$ /* February 1985 */

$MO_2 = [D_{59},D_{89}]$ /* March 1985 */

$MO_3 = [D_{90},D_{119}]$ /* April 1985 */

$MO_4 = [D_{120},D_{150}]$ /* May 1985 */

$MO_5 = [D_{151},D_{180}]$ /* June 1985 */

$MO_6 = [D_{181},D_{211}]$ /* July 1985 */

$MO_7 = [D_{212},D_{242}]$ /* August 1985 */

$MO_8 = [D_{243},D_{272}]$ /* September 1985 */

$MO_9 = [D_{273},D_{303}]$ /* October 1985 */

$MO_{10} = [D_{304},D_{333}]$ /* November 1985 */

And the last month, /* December 1985 */ $MO_{11} = [D_{334},D_{364}] = [H_{8016},H_{8759} = [M_{480960},M_{525599}] = [S_{28857600},S_{31535999}]$.

6. Finally, $T_5 = \{ Y_0 \}$ is the set consisting of the single entire "year" 1985

$= [S_0,S_{31535999}]$.

References

[Anderson 81] Anderson, T.L.
Database Semantics of Time.
PhD thesis, Computer Science Dept., U. of Washington, 1981.
(Unpublished).

[Ariav et al. 83] Ariav, G., Clifford, J., and Jarke, M.
Time and Databases.
In *ACM-SIGMOD International Conference on Management of Data*, pages 243-245. ACM, San Jose, CA, May, 1983.

[Bolour et al. 82] Bolour, A., Anderson, T.L., Deketser, L.J., Wong, H.K.T.
The Role of Time in Information Processing: A Survey.
ACM SIGMOD Record 12(3):28-48, April, 1982.

[Clifford 85] Clifford, J.
Towards an Algebra of Historical Relational Databases.
In *ACM-SIGMOD International Conference on Management of Data.* ACM, Austin, May, 1985.

[CliffordCroker 87]
Clifford, J., and Croker, A.
The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans,
In *Proc. Third International Conference on Data Engineering.* IEEE, Los Angeles, February, 1987.

[Colson 26] Colson, F.H.
The Week.
Cambridge University Press, Cambridge, 1926.

[Snodgrass 84] Snodgrass, R.
A Temporal Query Language.
Technical Report, Dept. of Computer Science, UNC, Chapel Hill, NC, May, 1984.
(Unpublished Technical Report #85-013).