

Almeida, Michael J. (2006) Incomplete Timestamps in Temporal Databases, *Proceedings of the 2nd International Advanced Database Conference*, US Education Service, pp. 93-98.

INCOMPLETE TIMESTAMPS IN TEMPORAL DATABASES

Michael J. Almeida
Fayetteville State University
Mathematics and Computer Science
1200 Murchison Road
Fayetteville, NC 28301
910-672-1363
malmeida@uncfsu.edu

ABSTRACT: *This paper addresses problems in representing and reasoning about a type of temporal indefiniteness that derives from having incomplete timestamps for some states, where a timestamp is called incomplete if it corresponds to only the beginning, ending, or some other portion of the full extent of the state. The approach to incomplete timestamps described in this paper combines two major components: (1) a method for representing timestamps based on the four possible ways that a state can be related to its interval, and (2) the use of a form of default reasoning called persistence to determine when a state may be assumed to hold beyond the time it is known to hold. The use of state-time connectors provides great ease and flexibility in representing the different possible types of timestamps, while state persistence provides a method for inferring when a state may hold that changes dynamically as information is added to, or deleted from, the database. A particularly important application of incomplete timestamps occurs in the interaction of currently holding states with the “moving now-point”.*

Keywords: *temporal databases, incomplete timestamps, state-time connectors, persistence.*

1. INTRODUCTION

Ideally, temporal databases should be able to deal with all types of temporal information, both definite and indefinite. One important form of indefinite temporal information is where we know that an event occurs or a state holds but not exactly when, a type of indefinite temporal knowledge sometimes called *valid-time indeterminacy* [5]. One way of representing and reasoning about such information is to use temporal constraint networks [1]. Other methods, more specific to temporal databases, for handling valid-time indeterminacy are discussed in [3], [5] and [6].

This paper is concerned with another form of temporal indefiniteness, one based on the “incompleteness” of a state’s timestamp. When a timestamp begins with the initiation of the state and ends with the termination of the state, then we say that the timestamp is *complete*. On the other hand, a timestamp is called *incomplete* if it corresponds to only the beginning, ending, or some other portion of the full extent of the state. A particularly important application of incomplete timestamps occurs in the interaction of currently holding states with the “moving now-point”, in which we represent states that started in the past but continue into the present and possibly into the future.

The approach to incomplete timestamps described in this paper is an expansion and refinement of ideas we first presented in [2]. Our approach combines two major components: (1) the use of explicit *state-time connectors* for representing timestamps based on the four possible ways that a state can be related to its interval, and (2) the use of a form of default reasoning called *persistence* to determine for what intervals a state may be assumed to hold in the absence of contrary information. The use of state-time connectors provides great ease and flexibility in representing the different possible types of timestamps, while state persistence provides a method for inferring when a state may hold that changes dynamically as information is added to, or deleted from, the database.

2. TEMPORAL REPRESENTATIONS

Allen [1] first made popular the purely interval-based approach to temporal representation. In Allen’s system there are 13 basic, i.e., mutually exclusive and collectively exhaustive, relations that can hold between two intervals of time. (Note that intervals of time are sometimes called *periods*, e.g., [8].) These

relations are *before* (b), *after* (bi), *meets* (m), *met_by* (mi), *overlaps* (o), *overlapped_by* (oi), *starts* (s), *started_by* (si), *during* (d), *contains* (di), *finishes* (f), *finished_by* (fi) and *equals* (e). In order to compactly represent temporal ambiguity, the relation between two intervals is a set of basic relations representing a disjunction. For example, the formula “ $t1 \{m,o,s\} t2$ ” represents the disjunction $(t1 \text{ meets } t2) \vee (t1 \text{ overlaps } t2) \vee (t1 \text{ starts } t2)$.

With granularity-based intervals, each interval T has a begin time, a single granule designated by $begin(T)$, and an end time, a single granule designated by $end(T)$. We will therefore sometimes write intervals in the form $[begin(T), end(T)]$. In addition, in the closed-closed version of interval representation, the begin and end points are both actual parts of the interval. Assuming the closed-closed interval representation, the thirteen basic relations between two intervals can be defined in terms of the relations between their begin times and end times as follows [8]:

1. $T1 \{b\} T2$ iff $end(T1) + 1 < begin(T2)$
2. $T1 \{bi\} T2$ iff $T2 \{b\} T1$
3. $T1 \{m\} T2$ iff $end(T1) + 1 = begin(T2)$
4. $T1 \{mi\} T2$ iff $T2 \{m\} T1$
5. $T1 \{o\} T2$ iff $begin(T1) < begin(T2)$
 $\wedge end(t1) < end(T2) \wedge begin(T2) \leq end(T1)$
6. $T1 \{oi\} T2$ iff $T2 \{o\} T1$
7. $T1 \{s\} T2$ iff $begin(T1) = begin(T2)$
 $\wedge end(T1) < end(T2)$
8. $T1 \{si\} T2$ iff $T2 \{s\} T1$
9. $T1 \{d\} T2$ iff $begin(T1) > begin(T2)$
 $\wedge end(T1) < end(T2)$
10. $T1 \{d\} T2$ iff $T2 \{d\} T1$
11. $T1 \{f\} T2$ iff $begin(T1) > begin(T2)$
 $\wedge end(T1) = end(T2)$
12. $T1 \{fi\} T2$ iff $T2 \{f\} T1$
13. $T1 \{e\} T2$ iff $begin(T1) = begin(T2)$
 $\wedge end(T1) = end(T2)$

In these formulas, P+1 designates the immediate successor point of P. Similarly P-1 designates the immediate predecessor of P.

Throughout this paper, we will assume that (1) intervals of time are composed of one or more granules in some fixed time granularity, (2) the begin times and end times of intervals are known exactly with respect to the granularity and therefore there is no valid-time indeterminacy and so no ambiguity in the temporal relationships between intervals, and (3) intervals are closed-closed.

3. STATE-TIME CONNECTORS

In order to represent incomplete timestamps, we propose the use of four different *state-time connectors* (STCs) that relate a state and a time interval for which that state exists. These four STCs constitute an expansion of those introduced in [2]. The STCs are:

1. HA(F,T) – An instance of state type F *holds at* (HA) time interval T if that instance of F is initiated at the start of T and is terminated at the end of T. In other words, T is the complete interval for this instance of F. HA is the implicit state-time connector in most temporal databases.

2. IA(F,T) – An instance of state type F *initiates at* (IA) time interval T if that instance of F is initiated at the start of T and holds for all of T. In other words, we know when F came into existence and we have partial knowledge of its extent, but we do not know if or when it went out of existence and therefore we do not have complete knowledge of its extent.

3. TA(F,T) – An instance of state type F *terminates at* (TA) time interval T if that instance of F holds for all of T and terminates at the end of T. In other words, we know when F went out of existence and we have partial knowledge of its extent, but we do not know when it came into existence and therefore we do not have complete knowledge of its extent.

4. HF(F,T) – An instance of state type F *holds for* (HF) time interval T if that instance of F holds for every part of T but we do not know when F was initiated or terminated. Therefore we have only partial knowledge of the extent of F.

State-time relations 2 through 4 connect a state with a (potentially partial) interval for that state. Of course, it is always possible that the interval is in fact complete, but we don't know it. It is therefore very important to distinguish between the start/begin and finish/end times of time intervals and the initiation and termination times of state instances. The truth values of the four state-time connectors are related to one another as follows:

HA(F,T) is true iff $\exists T1. [HA(F,T1) \wedge T \{e\} T1]$

IA(F,T) is true iff
 $\exists T1. [(IA(F,T1) \vee HA(F,T1)) \wedge T \{s,e\} T1]$

TA(F,T) is true iff
 $\exists T1. [(HA(F,T1) \vee TA(F,T1)) \wedge T \{f,e\} T1]$

HF(F,T) is true iff
 $\exists T1. [(HF(F,T1) \vee HA(F,T1) \vee IA(F,T1) \vee TA(F,T1)) \wedge T \{s,d,f,e\} T1]$

It will be noticed that HA(F,T) implies that all of the other STCs are true for T, while IA(F,T) and TA(H,T) each imply only HF(F,T). The state-time connectors can be easily added to a temporal database schema as a new attribute with the possible values ‘HA’, ‘IA’, ‘TA’ and ‘HF’.

3.1 Inferring Initiation and Termination Times

Knowledge of the initiation and termination times of state instances (collectively, their *boundaries*) can come from three different sources: (1) being told about them, (2) inferring them from the proximity of boundaries of other instances of the same state type, and (3) inferring them from the proximity of instances of incompatible state types. The boundaries of other instances of a given state type are important given the assumption that a state cannot come into, or go out of, existence during the time occupied by another instance of that same state type. So, for example, someone cannot wake up if they are already awake. As for incompatible state types, given a state type F, we will use $\sim F$ to designate any state type incompatible with F in the sense that they cannot both hold at the same time. The results of these rules typically feed into the merger rules of section 3.2.

1. $HF(F,T1) \wedge (IA(F,T2) \vee HA(F,T2)) \wedge T1 \{m\} T2 \rightarrow TA(F,T1)$
2. $IA(F,T1) \wedge (IA(F,T2) \vee HA(F,T2)) \wedge T1 \{m\} T2 \rightarrow HA(F,T1)$
3. $TA(F,T1) \wedge (TA(F,T2) \vee HA(F,T2)) \wedge T1 \{mi\} T2 \rightarrow HA(F,T1)$
4. $HF(F,T1) \wedge (TA(F,T2) \vee HA(F,T2)) \wedge T1 \{mi\} T2 \rightarrow IA(F,T1)$
5. $HF(F,T1) \wedge (HA(\sim F,T3) \vee IA(\sim F,T3) \vee TA(\sim F,T3) \vee HF(\sim F,T3)) \wedge T1 \{m\} T2 \rightarrow TA(F,T1)$
6. $IA(F,T1) \wedge (HA(\sim F,T3) \vee IA(\sim F,T3) \vee TA(\sim F,T3) \vee HF(\sim F,T3)) \wedge T1 \{m\} T2 \rightarrow HA(F,T1)$
7. $TA(F,T1) \wedge (HA(\sim F,T3) \vee IA(\sim F,T3) \vee TA(\sim F,T3) \vee HF(\sim F,T3)) \wedge T1 \{mi\} T2 \rightarrow HA(F,T1)$
8. $HF(F,T1) \wedge (HA(\sim F,T3) \vee IA(\sim F,T3) \vee TA(\sim F,T3) \vee HF(\sim F,T3)) \wedge T1 \{mi\} T2 \rightarrow IA(F,T1)$

3.2 Merging Intervals

Since our knowledge of the time occupied by a state can grow in a piecemeal fashion, the process of merging or coalescing intervals is of fundamental importance. Such mergers are possible when the times for which a state exists meet or overlap in an allowable way. The following rules describe the conditions under which such mergers are possible and the results of these mergers. The special symbol \Rightarrow can be read as “is replaced by”, in other words these rules describe modifications to be made in the database. It is assumed that the appropriate mergers are performed every time a new piece of temporal data is added to the database since many of our other rules depend on these results.

1. $HF(F,T1) \wedge HF(F,T2) \wedge T1 \{s,d,f,e\} T2 \Rightarrow HF(F,T2)$
2. $HF(F,T1) \wedge IA(F,T2) \wedge T1 \{s,d,f,e\} T2 \Rightarrow IA(F,T2)$
3. $HF(F,T1) \wedge TA(F,T2) \wedge T1 \{s,d,f,e\} T2 \Rightarrow TA(F,T2)$
4. $HF(F,T1) \wedge HA(F,T2) \wedge T1 \{s,d,f,e\} T2 \Rightarrow HA(F,T2)$
5. $HF(F,T1) \wedge HF(F,T2) \wedge T1 \{m,o\} T2 \Rightarrow HF(F,[begin(T1),end(T2)])$
6. $HF(F,T1) \wedge TA(F,T2) \wedge T1 \{m,o\} T2 \Rightarrow TA(F,[begin(T1),end(T2)])$
7. $HF(F,T1) \wedge IA(F,T2) \wedge T1 \{mi,oi\} T2 \Rightarrow IA(F,[begin(T2),end(T1)])$
8. $IA(F,T1) \wedge TA(F,T2) \wedge T1 \{m,o,s,fi,e\} T2 \Rightarrow HA(F,[begin(T1),end(T2)])$
9. $IA(F,T1) \wedge HA(F,T2) \wedge T1 \{s,e\} T2 \Rightarrow HA(F,T2)$
10. $IA(F,T1) \wedge IA(F,T2) \wedge T1 \{s,e\} T2 \Rightarrow IA(F,T2)$
11. $TA(F,T1) \wedge HA(F,T2) \wedge T1 \{f,e\} T2 \Rightarrow HA(F,T2)$
12. $TA(F,T1) \wedge TA(F,T2) \wedge T1 \{f,e\} T2 \Rightarrow TA(F,T2)$
13. $HA(F,T1) \wedge HA(F,T2) \wedge T1 \{e\} T2 \Rightarrow HA(F,T2)$

3.3 Detecting Inconsistency

The following formulas specify the conditions under which our temporal information is not consistent and therefore needs repairing. In conditions 1 – 9, the temporal inconsistency comes from situations where at least one boundary of a state instance is asserted to be within the time of another instance of that same state type. Condition 10 prohibits any kind of temporal overlap between states of incompatible types.

1. $IA(F,T1) \wedge IA(F,T2) \wedge T1 \{o,oi,d,di,f,fi\} T2$
2. $IA(F,T1) \wedge TA(F,T2) \wedge T1 \{oi,si,d,di,f\} T2$

3. $IA(F,T1) \wedge HA(F,T2) \wedge T1 \{o,oi,si,d,di,f,fi\} T2$
4. $IA(F,T1) \wedge HF(F,T2) \wedge T1 \{oi,d,f\} T2$
5. $TA(F,T1) \wedge TA(F,T2) \wedge T1 \{o,oi,s,si,d,di\} T2$
6. $TA(F,T1) \wedge HA(F,T2) \wedge T1 \{o,oi,s,si,d,di,fi\} T2$
7. $TA(F,T1) \wedge HF(F,T2) \wedge T1 \{o,s,d\} T2$
8. $HA(F,T1) \wedge HA(F,T2) \wedge T1 \{o,oi,s,si,d,di,f,fi\} T2$
9. $HA(F,T1) \wedge HF(F,T2) \wedge T1 \{o,oi,s,d,f\} T2$
10. $(HA(F,T1) \vee IA(F,T1) \vee TA(F,T1) \vee HF(F,T1))$
 $\wedge (HA(\sim F,T2) \vee IA(\sim F,T2) \vee TA(\sim F,T2)$
 $\vee HF(\sim F,T2)) \wedge T1 \{o,oi,s,si,d,di,f,fi,e\} T2$

4. STATE PERSISTENCE

The *closed-world assumption* is frequently used in databases in that a database is often considered to contain a complete description of the relevant aspects of its domain. It follows from the closed-world assumption that if we cannot prove that some proposition is true in our database, then we can assume that it is false, a type of default reasoning known as *negation-as-failure*. Negation-as-failure is an example of defeasible, or nonmonotonic, reasoning, reasoning which produces results that are taken to be true by default but which could become false given additional information.

Default reasoning based on the closed-world assumption can be used in databases with incomplete timestamps to make inferences about the temporal extent of states beyond the intervals where they are known to hold, an effect called *persistence* [7]. In general, persistence can occur in either of the two possible temporal directions, either into the future (*forwards persistence*) or into the past (*backwards persistence*). There are potentially many different logics of persistence based on different criteria.

4.1 A Basic Persistence Logic

A straightforward logic of persistence can be based on the idea that a state is assumed to persist unless it is blocked or “clipped” by a boundary of the state type in question or by an incompatible state [7]. Under this criterion, states tend to temporally expand to the maximum extent possible, i.e., for as long as there is no inconsistency. This logic also allows for both forwards and backwards persistence and treats them symmetrically. In the clipping rules below, T1 is the known time of the state F and T is the target interval, i.e., the interval to which we want F to persist.

forwards_clipped(T1,F,T) iff
 $\exists T2. [(TA(F,T2) \vee HA(F,T2))$
 $\wedge T1 \{b,m,o,s,d,f,fi,e\} T2 \wedge T2 \{b,m,o,s,d\} T]$
 $\vee [(IA(F,T2) \vee HA(F,T2)) \wedge T1 \{b,m\} T2$

$\wedge T2 \{b,m,o,oi,s,si,d,di,f,fi,e\} T]$
 $\vee [(HA(\sim F,T2) \vee IA(\sim F,T2)$
 $\vee TA(\sim F,T2) \vee HF(\sim F,T2))$
 $\wedge T1 \{b,m\} T2 \wedge T2 \{b,m,o,oi,s,si,d,di,f,fi,e\} T]$

backwards_clipped(T1,F,T) iff
 $\exists T2. [(TA(F,T2) \vee HA(F,T2)) \wedge T1 \{bi,mi\} T2$
 $\wedge T2 \{bi,mi,o,oi,s,si,d,di,f,fi,e\} T]$
 $\vee [(IA(F,T2) \vee HA(F,T2))$
 $\wedge T1 \{bi,mi,oi,s,si,d,f,e\} T2$
 $\wedge T2 \{bi,mi,oi,d,f\} T]$
 $\vee [(HA(\sim F,T2) \vee IA(\sim F,T2)$
 $\vee TA(\sim F,T2) \vee HF(\sim F,T2))$
 $\wedge T1 \{bi,mi\} T2$
 $\wedge T2 \{bi,mi,o,oi,s,si,d,di,f,fi,e\} T]$

The rules of persistence for the four state-time connectors follow. The rule for HF is the most complex because it contains three separate cases, one using only backwards persistence, one using only forwards persistence, and one using both backwards and forwards persistence. In all of these rules, “not” is understood as negation-as-failure.

HA(F,T) is true by persistence if
 $\exists T1,T2. IA(F,T1) \wedge T1 \{s\} T \wedge TA(F,T2) \wedge T2 \{f\} T$
 $\wedge (\text{not forwards_clipped}(T1,F,T)$
 $\vee \text{not backwards_clipped}(T2,F,T))$

IA(F,T) is true by persistence if
 $\exists T1. IA(F,T1) \wedge T1 \{s\} T$
 $\wedge \text{not forwards_clipped}(T1,F,T)$

TA(F,T) is true by persistence if
 $\exists T1. TA(F,T1) \wedge T1 \{f\} T$
 $\wedge \text{not backwards_clipped}(T1,F,T)$

HF(F,T) is true by persistence if
 $[\exists T1. (TA(F,T1) \vee HF(F,T1)) \wedge T1 \{bi,mi,oi,f\} T$
 $\wedge \text{not backwards_clipped}(T1,F,T)]$
 $\vee [\exists T1. (IA(F,T1) \vee HF(F,T1)) \wedge T1 \{b,m,o,s\} T$
 $\wedge \text{not forwards_clipped}(T1,F,T)]$
 $\vee [\exists T1. HF(F,T1) \wedge T1 \{d\} T$
 $\wedge \text{not backwards_clipped}(T1,F,T)$
 $\wedge \text{not forwards_clipped}(T1,F,T)]$

4.2 A More Complex Persistence Logic

Naturally, the existence of an incompatible state can serve to block persistence, but there is also the possibility that while some state F persists to an interval T , an incompatible state $\sim F$ also persists to (at least a part of) T . In the case of such “incompatible persistence”, a decision has to be made about how to interpret this result. The first persistence logic ignored it, but in this second persistence logic we assume that persistence fails under such circumstances.

In order to implement this new persistence logic it is necessary to distinguish between two types of clipping, *standard clipping*, in which F may partially overlap the target interval before it is blocked, and *external clipping*, in which F is blocked before it can overlap any part of the target interval [2].

$\text{forwards_ext_clipped}(T1,F,T)$ is true iff

$$\begin{aligned} & \exists T2. [(TA(F,T2) \vee HA(F,T2)) \\ & \quad \wedge T1 \{b,m,o,s,d,f,fi,e\} T2 \wedge T2 \{b,m\} T] \\ \vee & [(IA(F,T2) \vee HA(F,T2)) \\ & \quad \wedge T1 \{b,m\} T2 \wedge T2 \{b,m,o,s,si,di,fi,e\} T] \\ \vee & [(HA(\sim F,T2) \vee IA(\sim F,T2) \vee TA(\sim F,T2) \\ & \quad \vee HF(\sim F,T2)) \\ & \quad \wedge T1 \{b,m\} T2 \wedge T2 \{b,m,o,s,si,di,fi,e\} T] \end{aligned}$$

$\text{backwards_ext_clipped}(T1,F,T)$ is true iff

$$\begin{aligned} & \exists T2. [(TA(F,T2) \vee HA(F,T2)) \\ & \quad \wedge T1 \{bi,mi\} T2 \wedge T2 \{bi,mi,oi,si,di,f,fi,e\} T] \\ \vee & [(IA(F,T2) \vee HA(F,T2)) \\ & \quad \wedge T1 \{bi,mi,oi,s,si,d,f,e\} T2 \wedge T2 \{bi,mi\} T] \\ \vee & [(HA(\sim F,T2) \vee IA(\sim F,T2) \vee TA(\sim F,T2) \\ & \quad \vee HF(\sim F,T2)) \\ & \quad \wedge T1 \{bi,mi\} T2 \wedge T2 \{bi,mi,oi,si,di,f,fi,e\} T] \end{aligned}$$

The new rules of persistence for the four state-time connectors follow.

$HA(F,T)$ is true by persistence if

$$\begin{aligned} & \exists T1,T2. IA(F,T1) \wedge T1 \{s\} T \wedge TA(F,T2) \wedge T2 \{f\} T \\ & \quad \wedge (\text{not forwards_clipped}(T1,F,T) \\ & \quad \vee \text{not backwards_clipped}(T2,F,T)) \end{aligned}$$

$IA(F,T)$ is true by persistence if

$$\begin{aligned} & \exists T1. IA(F,T1) \wedge T1 \{s\} T \\ & \quad \wedge \text{not forwards_clipped}(T1,F,T) \\ & \quad \wedge \text{not } \exists T1. [(HF(\sim F,T1) \vee TA(\sim F,T1)) \\ & \quad \quad \wedge T1 \{bi,mi\} T \\ & \quad \quad \wedge \text{not backwards_ext_clipped}(T1,\sim F,T)] \end{aligned}$$

$TA(F,T)$ is true by persistence if

$$\begin{aligned} & \exists T1. TA(F,T1) \wedge T1 \{f\} T \\ & \quad \wedge \text{not backwards_clipped}(T1,F,T) \\ & \quad \wedge \text{not } \exists T1. [(HF(\sim F,T1) \vee IA(\sim F,T1)) \\ & \quad \quad \wedge T1 \{b,m\} T \\ & \quad \quad \wedge \text{not forwards_ext_clipped}(T1,\sim F,T)] \end{aligned}$$

$HF(F,T)$ is true by persistence if

$$\begin{aligned} & \{[\exists T1. (TA(F,T1) \vee HF(F,T1)) \wedge T1 \{bi,mi,oi,f\} T \\ & \quad \wedge \text{not backwards_clipped}(T1,F,T)] \\ & \quad \wedge \text{not } \exists T1. [(IA(\sim F,T1) \vee HF(\sim F,T1)) \\ & \quad \quad \wedge T1 \{b,m\} T \\ & \quad \quad \wedge \text{not forwards_ext_clipped}(T1,\sim F,T)]\} \\ \vee & \{[\exists T1. (IA(F,T1) \vee HF(F,T1)) \wedge T1 \{b,m,o,s\} T \\ & \quad \wedge \text{not forwards_clipped}(T1,F,T)] \\ & \quad \wedge \text{not } \exists T1. [(TA(\sim F,T1) \vee HF(\sim F,T1)) \\ & \quad \quad \wedge T1 \{bi,mi\} T \\ & \quad \quad \wedge \text{not backwards_ext_clipped}(T1,\sim F,T)]\} \\ \vee & \{[\exists T1. HF(F,T1) \wedge T1 \{d\} T \\ & \quad \wedge \text{not backwards_clipped}(T1,F,T) \\ & \quad \wedge \text{not forwards_clipped}(T1,F,T)] \\ & \quad \wedge \text{not } \exists T1. [(IA(\sim F,T1) \vee HF(\sim F,T1)) \\ & \quad \quad \wedge T1 \{b,m\} T \\ & \quad \quad \wedge \text{not forwards_ext_clipped}(T1,\sim F,T)] \\ & \quad \wedge \text{not } \exists T1. [(TA(\sim F,T1) \vee HF(\sim F,T1)) \\ & \quad \quad \wedge T1 \{bi,mi\} T \\ & \quad \quad \wedge \text{not backwards_ext_clipped}(T1,\sim F,T)]\} \end{aligned}$$

5. THE MOVING NOW POINT

The present moment in time, the *now-point*, presents unique representational problems for temporal databases. One of the more difficult issues concerns the representation of states that hold in the present and whose end times are unknown. The obvious solution of using the present moment as the value of the end point fails for two reasons. In the first place, because the now-point keeps moving forward in time, there is the “update” problem in which we must constantly change the value of the end time of all current states in order to prevent them from becoming terminated [4]. In addition, using the current value of the now-point as the end time of current states implies that we believe that all of these states will terminate at the present moment, something that we generally do not want to imply. Three additional methods for providing an end time value for such states are discussed in [8]: (1) use a special time value that otherwise makes no sense in the database, e.g., a time which is outside the bounds of the coverage of the database; (2) use NULL; and (3) use the latest future date that is representable in the database. As is pointed out in [4], all of these attempted solutions either have logical problems or

they are dishonest in that what they are actually asserting is false.

Fundamentally, the problem with all of these attempted solutions is that their underlying representations assume that all timestamps are complete, and it simply makes no sense to represent a state whose end time is unknown with a complete timestamp. Recognizing this, [4] proposes a representation for current states in which only the initiation time, i.e., the time when that state instance came into existence, is represented. Such tables are called *semitemporal* because they can only represent one boundary of a state's complete timestamp [4]. Our representation for currently holding states is similar to this proposal, except that we can use either the IA or HF state-time connector to represent states whose termination times are unknown. By using explicit state-time connectors, we get a unified framework for representing all types of incomplete temporal data along with complete data, so we do not require separate tables for each of the different possibilities, as happens with a semitemporal table.

The interpretation of a semitemporal table is that any state represented in it is understood to definitely hold from its initiation time up to the present. We do not make such an assumption in our approach, instead we distinguish between two components of the extent of a state, the part given by its timestamp where the state is known to hold, and the part where the state is assumed to hold by persistence. If we can determine that a previously existing state is still in existence, then we can update the database to reflect this new information. On the other hand, if we don't check but merely assume from our lack of knowledge of a change that the state is still in existence, then we are using persistence. In either case, we have a representation that naturally accommodates the moving now-point.

6. CONCLUSION

In this paper we have described an approach to representing and reasoning about temporal indefiniteness based on incomplete timestamps, where a timestamp is called *incomplete* if it corresponds to only the beginning, ending, or some other portion of the full extent of the state. Our approach consists of two components, (1) a method for representing incomplete timestamps based on the use of four state-time connectors, and (2) the use of persistence to determine where a state may possibly hold beyond where it is known to hold. The use of state-time connectors provides great ease and flexibility in representing the different possible types

of incomplete timestamps, while state persistence provides a method for inferring when a state may hold that changes dynamically as information is added to, or deleted from, the database.

In the future we hope to combine our solution to timestamp incompleteness with methods designed to handle other types of temporal indefiniteness. We also plan to incorporate more sophisticated persistence logics, such as those that make use of probabilities based on knowledge of the inherent or typical longevities of different types of states.

References

- [1] Allen, J. Maintaining knowledge about temporal intervals. *Comm. of the ACM*, 26, 11 (Nov. 1983), 832-843.
- [2] Almeida, M. J. Ambiguity and persistence in the interval algebra. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI '03)* (Las Vegas, NV, June 23-26, 2003). CSREA Press, 2003, 394-399.
- [3] Cowley, W. and Plexousakis, D. An interval algebra for indeterminate time. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)* (Austin, TX, July 30-August 3, 2000). AAAI Press, 2000, 470-475.
- [4] Date, C. J., Darwen, H., and Lorentzos, N. A. *Temporal Data and the Relational Model*. Morgan Kaufmann, San Francisco, CA, 2003.
- [5] Dyreson, C. E. and Snodgrass, R. T. Supporting valid-time indeterminacy. *ACM Transactions on Database Systems*, 23, 1 (March 1998), 1-57.
- [6] Koubarakis, M. Database models for infinite and indefinite temporal information. *Information Systems*, 19, 2 (1994), 141-173.
- [7] Shanahan, M. *Solving the Frame Problem*. MIT Press, Cambridge, MA, 1997.
- [8] Snodgrass, R. T. *Developing Time-Oriented Database Applications in SQL*. Morgan Kaufmann, San Francisco, CA, 2000.