Incomplete information in relational temporal databases

Shashi K. Gadia, Sunil S. Nair and Yiu-Cheong Poon
Computer Science Department
Iowa State University
Ames, IA 50011
(515) 294-2253; INTERNET: gadia@cs.iastate.edu

Abstract. For the conventional relational model there has been considerable research in the area of incomplete information. On the other hand, research in temporal databases has concentrated on models in which complete historical information is needed. However, the likelihood of missing information in temporal databases is greater because of the vast amount of information. Hence, a mechanism must be provided to store and query incomplete temporal information. In this paper we present a model for incomplete information in temporal databases. The model generalizes our previous model for complete temporal information. It is shown that our relational operators produce results that are reliable. We also show, with some exceptions, that if the definitions of the operators were strengthened to give more information, we may obtain results that are not reliable.

1. Introduction.

Research in temporal databases has concentrated on models in which it is essential that all the information be known [CC87, Ga88, GY88, NA89, Sa90, Sn87, Ta86]. However, in the case of temporal databases the likelihood of missing information increases because of the vast amount of information being stored. Furthermore users may want to maintain only selective portions of history, for instance only the salary history of certain employees. Therefore, there is a need to develop data models in which partial historical information can be stored and queried.

For the conventional relational model there has been considerable research in the area of incomplete information (which exists but is unknown) [Co79,Bi83,IL84,Li81,Re86]. In most models unknown values are marked by a special symbol called a null value, denoted ω. Since some attribute values are unknown a selection expression does not always evaluate to TRUE or FALSE for a given tuple. To solve this problem, a third truth-value which we call UNDEFINED is sometimes introduced. In [Co79] two forms of the selection operator are introduced: the TRUE-select and the MAYBE-select. The result of a TRUE-select operation consists of those tuples for which the selection expression yields the value TRUE. The result of a MAYBE-select operation consists of those tuples for which the selection expression evaluates to UNDEFINED. In [Bi83] each relation has an additional column called STATUS which marks tuples as definite tuples (marked with a d) or maybe tuples (marked with an m). Only a single selection operation is needed. Selection formulas are restricted to be of the form C = D or of the form C = b where C, D are attributes and b is a constant. The selection mechanism is set up in such a way that maybe tuples can lead only to maybe tuples in the result of a selection, while definite tuples can lead to definite or maybe tuples in the result.

In this paper we present a relational model for temporal databases with incomplete information. There are two points that must be considered in the storage model for incomplete information in temporal databases. First, for a given object, we may know the values for a given attribute at some points in time but the values at other points in time may be unknown. Second, at some points in time we are sure that the object must exist in the relation but at other points in time the existence of the object in the relation is not a certainty. Thus at some points in time the tuple is a definite tuple but at other points it is a maybe tuple. This second situation is more likely to occur in computed relations rather than in stored relations. However, we allow for this possibility even in stored relations.

Apart from the storage model to maintain partial histories, we also define a powerful algebra to query the incomplete historical information. The selection operation is

Proceedings of the 18th VLDB Conference,
Vancouver, British Columbia, Canada 1992

an especially interesting operator in our model. It allows us to ask questions that have no counterpart in the classical case and it represents a departure of temporal databases from the classical snapshot databases. As a consequence, our results for incomplete temporal information cannot be obtained directly from corresponding results on incomplete snapshot information. Our model has the following properties:

• The incomplete information model presented here generalizes the model for temporal databases with complete information as given in [GY88] (Theorem 5.1). Thus, if our relation had no incomplete information our operators would give the same results as the operators in [GY88]. Our generalization is seamless in the sense that queries which could be presented to a database with complete information can also be presented to our model without any change in syntax. Some remarks about further extending the querying capability of our model, without altering the structure of the syntax, are made in Section 6.

• Our algebraic expressions produce results that are reliable in the sense that they never report incorrect information (Theorem 5.2).

• Except for certain cases of selection, if the definition of the operators were strengthened to give more information, we may obtain results that are not reliable (Theorem 5.4). This theorem does not extend to (i) certain cases of selection and (ii) arbitrary algebraic expressions.

The rest of the paper is organized as follows. In Section 2 we describe a model for temporal databases with complete information on which our model is based. In Section 3 we introduce our model for incomplete temporal databases. The algebra to query the databases is given in Section 4. In Section 5 we prove some results which show that our model is theoretically sound. In Section 6 we conclude with some remarks on the inherent querying capability of our model and on how our ideas can be further investigated.

2. Model for Complete Temporal Information.

In this section we describe a complete information temporal database model [GY88], which is used as a reference point from which we define the model for partial temporal relations.

2.1. Universe of time and temporal elements. We assume a universe [0,NOW] of time instants together with a linear order $\leq$ on it. Although it is not necessary, we assume for simplicity that [0,NOW] is the discrete set {0,1,···,NOW}. Intervals are not adequate to model history of an object in a single tuple and lead to query languages that are difficult to use [GY91]. Therefore we define a *temporal element* to be a finite union of intervals. Temporal elements are closed under $\cup$, $\cap$ and $\neg$ (complementation) and form a boolean algebra (for a definition of a boolean algebra see [TM75]).

2.2. Attribute values. To capture changing value of an attribute we define a *temporal assignment* to an attribute A to be a function from a temporal element into the domain of A. An example of a temporal assignment to the attribute COLOR is ([25,32] red, [33,NOW] blue). If ξ is a temporal assignment, $[\![\xi]\!]$ denotes its domain. Thus $[\![$([25,32] red, [33,NOW] blue)$]\!]$ = [25,NOW]. $\xi\restriction\mu$ denotes the restriction of ξ to the temporal element μ. Thus ([25,32] red, [33,NOW] blue)$\restriction$[25,30] = ([25,30] red).

2.3. θ-comparisons. Our counterpart of the construct AθB of the relational model is $[\![A\theta B]\!]$, which captures the time when A is in θ-relationship to B. This is introduced through $[\![\xi_1\theta\xi_2]\!] = \{t: \xi_1$ and ξ_2 are defined at t, and $\xi_1(t)\theta\xi_2(t)$ is TRUE$\}$. For example, $[\![$([25,32] red, [33,NOW] blue) = ([0,NOW] blue)$]\!]$ = [33,NOW]. We also allow the construct $[\![A\ \theta\ b]\!]$, where b is a constant, which is evaluated by identifying the constant b with the assignment [0,NOW] b.

2.4. Tuples and relations. A *tuple* is simply a concatenation of assignments whose temporal domains are the same. The assumption that all temporal assignments in a tuple have the same domain is called the *homogeneity* assumption [Ga88]. The model collects the entire history of a real world object in a single tuple. The *restriction* of a tuple τ to a temporal element μ, denoted $\tau\restriction\mu$, is obtained by restricting each assignment in τ to the temporal element μ.

A *relation* r over a scheme R, with K$\subseteq$R as its *key*, is a finite set of non-empty tuples such that no key attribute value in a tuple changes with time, and no two tuples agree on all their key attributes. Figure 2.1 shows a database with a relation emp(NAME SALARY DEPT) with NAME as its key, and a relation management(DEPT MANAGER) with DEPT as its key. The *restriction* of a relation r to a temporal element μ, denoted $r\restriction\mu$, is the relation obtained by restricting all tuples of r to the temporal element μ. The *snapshot* of a relation r at an instant t, denoted r(t), is the relation obtained by restricting each tuple of r to t.

2.5. The nature of keys in our model. Keys play a critical role in our model. A key provides a persistent identity to

an object. In every instantaneous snapshot of the management relation in Figure 2.1 DEPT and MANAGER functionally determine each other. However, viewing MANAGER as the key would require the management relation to be restructured as shown in Figure 2.2. This is further discussed in Section 2.6.3.

NAME	SALARY	DEPT
[8,52] John	[8,39] 15K [40,52] 20K	[8,44] Toys [45,52] Shoes
[48,NOW] Doug	[48,NOW] 20K	[48,NOW] Auto

The emp relation

DEPT	MANAGER
[8,NOW] Toys	[8,39] John [40,NOW] Jack
[6,39] ∪ [48,NOW] Auto	[6,39] Jack [48,NOW] Doug

The management relation

Figure 2.1. A database

DEPT	MANAGER
[8,39] Toys	[8,39] John
[6,39] Auto [40,NOW] Toys	[6,NOW] Jack
[48,NOW] Auto	[48,NOW] Doug

Figure 2.2. management: MANAGER relation

2.6. Algebra for complete temporal information. The set of all algebraic expressions can be divided into three mutually exclusive groups: *temporal expressions*, *boolean expressions*, and *relational expressions*.

2.6.1. Temporal expressions. Temporal expressions are the syntactic counterpart of temporal elements. They are formed using temporal elements, $[\![A]\!]$, $[\![A\emptyset B]\!]$, $[\![A\emptyset b]\!]$, $\cup$, $\cap$, and $-$. If μ is a temporal expression and τ is a tuple, then $\mu(\tau)$ evaluates to a temporal element and is defined in a natural way. For example, if τ is John's tuple in Figure 2.1, $[\![\text{SALARY} = 20\text{K}]\!](\tau)$ evaluates to [40,52].

2.6.2. Boolean expressions. Boolean expressions are formed using $\mu \subseteq \nu$, where μ and ν are temporal expressions. More complex expressions are formed using $\wedge$, $\vee$, and $\neg$.

2.6.3. Relational expressions. Relational expressions are the syntactic counterpart of temporal relations.

Restructuring. The purpose of the restructuring operator is to change the key of a relation. Two relations are said to be *weakly equal* if they have the same snapshots at all instants [Ga86a]. Suppose r is a relation over R with K as its key. Then if $K' \subseteq R$ such that $K' \rightarrow R$ in all snapshots of r, then r:K′ is the unique relation weakly equal to r but with key K′. For example, management: MANAGER is shown in Figure 2.2.

Union and difference. If r and s are relations over the same scheme R and the same key K, then r∪s and r−s also have the same scheme and key. Informally, r∪s is obtained by collapsing tuples with same key values to form a single tuple. Similarly, r−s is obtained by removing portions of tuples in r which overlap with tuples in s.

Projection. To define $\Pi_X(r)$, we require that the key of r be a subset of X. Then $\Pi_X(r)$ is defined to be $\{\tau(X): \tau \in r\}$.

Selection. Selection is a powerful operator in temporal databases. If f is a boolean expression and μ is a temporal expression then the selection $\sigma(r;f;\mu)$ evaluates to $\{\tau\restriction\mu(\tau): \tau \in r \wedge f(\tau) \wedge \tau\restriction\mu(\tau) \text{ is not empty}\}$. If f evaluates to TRUE for a tuple, σ allows us to select only a relevant part of it, which is specified by μ. For example, the query *give information about employees while they were in Toys or shoes if they are currently employed* can be expressed as $\sigma(\text{emp};[\text{NOW},\text{NOW}] \subseteq [\![\text{NAME}]\!]; [\![\text{DEPT}=\text{Toys}]\!] \cup [\![\text{DEPT}=\text{Shoes}]\!])$. If the parameter f is omitted in $\sigma(r;f;\mu)$ it defaults to TRUE. If μ is omitted it defaults to [0,NOW].

The selection operation cannot be evaluated snapshot-wise [GY88]. For example, the management relation in Figure 2.1 is weakly equal to the management$_1$ relation in Figure 2.2. However, $\sigma(\text{management};[8,\text{NOW}] \subseteq [\![\text{DEPT} = \text{Toys}]\!]; [0,\text{NOW}])$ is not weakly equal to $\sigma(\text{management}_1; [8,\text{NOW}] \subseteq [\![\text{DEPT} = \text{Toys}]\!]; [0,\text{NOW}])$. Thus our selection operator represents a departure of temporal databases from classical snapshot databases.

Cross product. Suppose r and s are two relations. A tuple in r×s is obtained by concatenating a tuple in r and a tuple in s, and only preserving the instants where both the tuples are defined. This assures the homogeneity of r×s. To avoid distraction from the main theme of this paper, we confine ourselves to homogeneous relations, but our framework can be extended to *multi-homogeneous relations* [Ga86b] where a literal cross product is formed.

3. Model for Incomplete Temporal Information.

In this section, we define our model for temporal databases with incomplete information, called *partial temporal databases*, by generalizing notions of a temporal element, temporal assignment, tuples and relations to capture incomplete information.

3.1. Partial temporal element. In the case of complete information an expression like $[\![A=B]\!]$ yields a temporal element which is the set of instants during which A=B. When A and B have missing information we may not be able to compute this set exactly. Hence the knowledge of instants when we are sure A=B is TRUE, and instants when we are sure that A=B is FALSE is important. This leads to the notion of a *partial temporal element*, which is defined to be a pair $\langle \ell, u \rangle$ where $\ell \subseteq u$; ℓ and u are called the *lower* and *upper* limit of the partial temporal element, respectively (see Figure 3.1(a)). Now $[\![A=B]\!]$ yields a pair $\langle \ell, u \rangle$, where ℓ is a set of instants when A=B definitely holds, and u is a set of instants beyond which A=B could not hold. Note that a temporal element μ can be represented as a partial temporal element by $\langle \mu, \mu \rangle$. Thus partial temporal elements are a generalization of temporal elements. The operations $\cup$, $\cap$, $-$ and $\neg$ are generalized as follows:

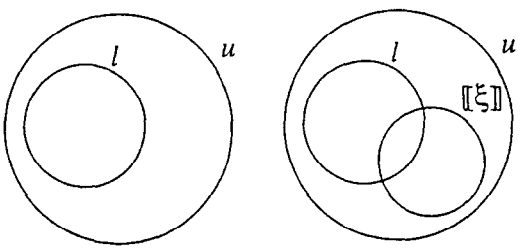
- Union: $\langle \ell_1, u_1 \rangle \cup \langle \ell_2, u_2 \rangle = \langle \ell_1 \cup \ell_2, u_1 \cup u_2 \rangle$
  e.g. $\langle [0,5],[0,20] \rangle \cup \langle [4,15],[0,15] \rangle = \langle [0,15],[0,20] \rangle$.
- Intersection: $\langle \ell_1, u_1 \rangle \cap \langle \ell_2, u_2 \rangle = \langle \ell_1 \cap \ell_2, u_1 \cap u_2 \rangle$
- Difference: $\langle \ell_1, u_1 \rangle - \langle \ell_2, u_2 \rangle = \langle \ell_1 - u_2, u_1 - \ell_2 \rangle$
  e.g. $\langle [0,5],[0,20] \rangle - \langle [4,15],[0,15] \rangle = \langle \emptyset, [0,3] \cup [16,20] \rangle$.
- Complementation: $\neg \langle \ell_1, u_1 \rangle = \langle \neg u_1, \neg \ell_1 \rangle$
  e.g. $\neg \langle [0,5] \cup [8,9],[0,20] \rangle = \langle [21,\text{NOW}],[6,7] \cup [10,\text{NOW}] \rangle$.

The operators on the left hand side are operations on partial temporal elements while the operators on the right hand side are operations on temporal elements. The set of partial temporal elements is closed under the operations defined above. The following theorem is easily proved.

THEOREM 3.1. The set of partial temporal elements together with $\cup$ and $\cap$ forms a distributive lattice (for a definition of a lattice see [TM75]).

3.2. Attributes. In our model a *partial temporal assignment* to an attribute A is a triple $\xi \ell u$, where ξ is a temporal assignment (as defined in Section 2.2), ℓ and u are temporal elements such that $[\![\xi]\!] \subseteq u$ and $\ell \subseteq u$ (see Figure 3.1(b)). The restriction of $\xi \ell u$ to a partial temporal element $\langle \ell', u' \rangle$, denoted $(\xi \ell u) \upharpoonright \langle \ell', u' \rangle$, is defined as $(\xi \upharpoonright u').(\ell' \cap \ell).(u' \cap u)$. The triple $\xi \ell u$ when it is assigned to an attribute encodes the following information:

- During ℓ we are sure that the object exists.
- Beyond u the object does not exist.
- During $u - \ell$ we are uncertain about the existence of the object.
- During $\ell \cap [\![\xi]\!]$ we know that the object exists and the values it takes.
- During $\ell - [\![\xi]\!]$ the object exists but its values are unknown
- During $u - [\![\xi]\!]$ the object may exist, but we do not know the values.
- During $[\![\xi]\!] - \ell$ if the object exists we know the values.



(a) A partial temporal element
(b) $[\![\xi]\!]$, ℓ and u in a partial temporal assignment $\xi \ell u$.
Figure 3.1.

In the model for complete information, an attribute is assigned a temporal assignment ξ. This complete information can be represented in our model for incomplete information by the partial temporal assignment $\xi [\![\xi]\!] [\![\xi]\!]$. In this sense, partial temporal assignments are a generalization of temporal assignments.

3.3. θ-comparisons. As we did in Section 2.3 for the complete temporal model, we want to introduce the constructs $[\![A\ \theta\ B]\!]$ and $[\![A\ \theta\ b]\!]$ where A and B are attributes and b is a constant. In our model an assignment to an attribute is a partial temporal assignment of the type $\xi \ell u$ where ξ is a temporal assignment and ℓ,u are temporal elements. A constant b can be identified with the assignment $\xi \ell u$ where $\xi = [0,\text{NOW}]b$, $\ell = [0,\text{NOW}]$ and $u = [0,\text{NOW}]$. Hence the constructs $[\![A\ \theta\ B]\!]$ and $[\![A\ \theta\ b]\!]$ may be introduced by first defining $[\![(\xi_1 \ell_1 u_1)\ \theta\ (\xi_2 \ell_2 u_2)]\!]$. The expression $[\![(\xi_1 \ell_1 u_1)\ \theta\ (\xi_2 \ell_2 u_2)]\!]$ evaluates to the partial temporal element $\langle [\![\xi_1 \theta \xi_2]\!] \cap \ell_1 \cap \ell_2,\ u_1 \cap u_2 - [\![\xi_1 \theta' \xi_2]\!] \rangle$ where $\theta' = \neg\theta$ (i.e. if θ is $\leq$ then θ' is $>$ etc.) The lower-limit, $[\![\xi_1 \theta \xi_2]\!] \cap \ell_1 \cap \ell_2$, is the time during which we are sure the θ-relation holds. This lower limit cannot be greater than $\ell_1 \cap \ell_2$. The upper limit, $u_1 \cap u_2 - [\![\xi_1 \theta' \xi_2]\!]$, is the time beyond which the θ-relation cannot exist. For

example, let $\xi_1 \ell_1 u_1$ be (ξ_1=([0,5]a [6,9]b); ℓ_1=[0,10]; u_1=[0,20]) and $\xi_2 \ell_2 u_2$ be (ξ_2=([0,4]a [5,8]c); ℓ_2=[0,9]; u_2=[0,15]). Then $[\![(\xi_1 \ell_1 u_1) = (\xi_2 \ell_2 u_2)]\!] = \langle [\![\xi_1 = \xi_2]\!] \cap \ell_1 \cap \ell_2,\ u_1 \cap u_2 - [\![\xi_1 \neq \xi_2]\!] \rangle = \langle [0,4],[0,4] \cup [9,15] \rangle$

The definitions of $[\![A \theta B]\!]$ and $[\![A \theta b]\!]$ given above are a generalization of the complete case. Also, by constructing a suitable example, the reader can verify that having ξ defined beyond the lower limit ℓ may help to reduce uncertainty in $[\![A \theta B]\!]$.

3.4. Tuples and relations. A tuple τ is a concatenation of partial temporal assignments whose ℓ values are the same and u values are the same. Hence, in an actual implementation the common ℓ value and u value could be stored at the tuple level rather than with each attribute. However, we will continue to have the temporal elements ℓ and u associated with the attributes in order to simplify the formalism. The ℓ values and u values have the following interpretation: During ℓ we are sure the object represented by the tuple exists in the relation and beyond u the object cannot exist in the relation. The requirement that the ℓ values of all attributes are equal and the u values of all attributes are equal makes the tuple *homogeneous*. This definition of homogeneity is analogous to the definition of homogeneity for the complete temporal case defined in Section 2.4. By $\tau\restriction\langle \ell,u \rangle$ is meant that each attribute in τ is restricted to $\langle \ell,u \rangle$ (the restriction of an attribute to a partial temporal element was defined in Section 3.2). A relation r over a scheme R with key K ($\subseteq$R) is a set of tuples such that no key attribute values of a tuple change with time, for key attributes we have $[\![\xi]\!]$ = u, and no two tuples in r agree on all their key attributes. Figure 3.2 shows a relation emp with NAME as the key. In each attribute $\xi \ell u$, the ξ part is shown first, followed by the temporal element that represents the ℓ part and then the temporal element that represents the u part.

NAME	SALARY	DEPT
[0,100]John	[10,40]30K [41,45]40K	[10,30]Toys [31,55]Shoes
[0,50]	[0,50]	[0,50]
[0,100]	[0,100]	[0,100]
[10,50]Tom	[10,45]40K [46,50]60K	[10,50]Toys
[10,50]	[10,50]	[10,50]
[10,50]	[10,50]	[10,50]

Figure 3.2. The emp relation

In the emp relation, we are sure that John was an employee at least during [0,50] and that he was not an employee beyond [0,100]. However, we have missing information for his department during [0,9]. If he was present in the organization at any time during [56,100], we have missing information on his department at that time also. If John was working for the organization during [51,55], he was in the Shoes department. We also have some missing information for the SALARY attribute.

NAME	SALARY	DEPT
[50,50]John		[50,50]Shoes
[50,50]	[50,50]	[50,50]
[50,50]	[50,50]	[50,50]
[50,50]Tom	[50,50]60K	[50,50]Toys
[50,50]	[50,50]	[50,50]
[50,50]	[50,50]	[50,50]

Snapshot as a temporal relation

NAME	SALARY	DEPT	STATUS
John	⊥	Shoes	d
Tom	60K	Toys	d

Snapshot as a static relation with nulls

Figure 3.3. Snapshot of emp relation at instant 50

NAME	SALARY	DEPT
[55,55]John		[55,55]Shoes
∅	∅	∅
[55,55]	[55,55]	[55,55]

Snapshot as a temporal relation

NAME	SALARY	DEPT	STATUS
John	⊥	Shoes	m

Snapshot as a static relation with nulls

Figure 3.4. Snapshot of emp relation at instant 55

3.5. Snapshots of partial temporal relations. An instant t can be represented as a partial temporal element $\langle [t,t],[t,t] \rangle$. A temporal snapshot of a relation at time t is obtained by restricting each tuple in r to $\langle [t,t],[t,t] \rangle$. The temporal snapshot may be represented as a static snapshot with nulls and an additional column called STATUS with domain = {d,m} which denotes whether the tuple is definitely(d) in the relation or maybe(m) in the relation. Such a relation corresponds to a classical relation with null

values as in [Bi83]. The static snapshot at t can be obtained from the temporal snapshot by placing a d in the STATUS column for a tuple whose lower limits are [t,t] or m if the lower limits are $\emptyset$, by replacing empty assignments with null values and then deleting all timestamps. We denote a temporal or static snapshot by r(t). Figure 3.3 shows the snapshot at t = 50 of the relation emp from Figure 3.2 as a temporal relation and as a static relation with nulls. Figure 3.4 shows the snapshot of emp at t = 55.

4. Algebra for Partial Temporal Databases.

In this section we generalize our algebra of Section 2 to partial temporal relations. As before there are three kinds of algebraic expressions: *partial temporal expressions*, *partial boolean expressions* and *partial relational expressions*.

4.1. Partial temporal expressions. Partial temporal expressions are the syntactic counterparts of partial temporal elements and are formed from temporal elements, $[\![A]\!]$, $[\![A\ \theta\ B]\!]$ $[\![A\ \theta\ b]\!]$, $\cup$, $\cap$ and $-$. Hence partial temporal expressions are syntactically the same as temporal expressions. However, the evaluation of these expressions on a tuple τ yield a partial temporal element as follows:

- If μ is a temporal element then $\mu(\tau) = \langle\mu,\mu\rangle$.
- $[\![A]\!](\tau) = \langle\ell,u\rangle$ where $\tau(A) = \xi\ell u$ and A is an attribute.
- $[\![A\theta B]\!] = [\![(\xi_1\ell_1u_1)\ \theta\ (\xi_2\ell_2u_2)]\!]$ where $\tau(A) = \xi_1\ell_1u_1$, $\tau(B) = \xi_2\ell_2u_2$ and A and B are attributes (for our model $\ell_1 = \ell_2$, $u_1 = u_2$ by homogeneity).
- $[\![A\theta b]\!](\tau) = [\![(\xi_1\ell_1u_1)\ \theta\ (\xi_2\ell_2u_2)]\!]$ where $\tau(A) = \xi_1\ell_1u_1$, $\xi_2 = [0,NOW]\!]b$, $\ell_2 = [0,NOW]$ and $u_2 = [0,NOW]$.
- If t_1 and t_2 are temporal expressions then

  $(t_1 \cap t_2)(\tau) = t_1(\tau) \cap t_2(\tau)$,

  $(t_1 \cup t_2)(\tau) = t_1(\tau) \cup t_2(\tau)$ and

  $(t_1 - t_2)(\tau) = t_1(\tau) - t_2(\tau)$.

4.1.1 Example. Consider the emp relation in Figure 3.2. If John's tuple is denoted by τ then $[\![\text{NAME}]\!](\tau) = \langle[0,50], [0,100]\rangle$. Similarly, $[\![\text{SALARY=30K}]\!](\tau) = \langle[10,40],[0,40] \cup [46,100]\rangle$ and $[\![\text{DEPT=Shoes}]\!](\tau) = \langle[31,50],[0,9]\cup[31,100]\rangle$. Also, $([\![\text{SALARY = 30K}]\!] \cup [\![\text{DEPT = Toys}]\!])(\tau) = \langle[10,40], [0,40]\cup[46,100]\rangle$.

4.2. Partial boolean expressions. Like the complete temporal case, essentially the atomic partial boolean expression is $\mu \subseteq \nu$, where μ and ν are partial temporal expressions. More complex boolean expressions are formed using $\vee$, $\wedge$ and $\neg$.

Since we have incomplete information, we may not always be able to determine if a particular formula applied to a tuple yields TRUE or FALSE. For instance, consider John's tuple in Figure 3.2. The atomic formula $[46,48] \subseteq [\![\text{SALARY=60K}]\!]$ may be TRUE or FALSE depending on the SALARY values during [46,48]. Hence we need to introduce the truth value UNDEFINED. Then $[46,48] \subseteq [\![\text{SALARY = 60K}]\!]$ yields the truth value UNDEFINED for John's tuple. The three-valued truth tables for $\wedge$, $\vee$ and $\neg$ are shown in Figure 4.1.

$\wedge$	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

$\vee$	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

$\neg$	
T	F
F	T
U	U

Figure 4.1. Three-valued truth tables for $\wedge$, $\vee$ and $\neg$. T≡TRUE, F≡FALSE, U≡UNDEFINED

The evaluation of $\mu \subseteq \nu$ for a given tuple τ is performed by first computing $\mu(\tau)$ and $\nu(\tau)$. Since μ and ν are partial temporal expressions, $\mu(\tau)$ and $\nu(\tau)$ yield partial temporal elements. Hence, we need to decide the truth value for one partial temporal element $\langle\ell_1,u_1\rangle$ being a subset of another partial temporal element $\langle\ell_2,u_2\rangle$. The partial temporal element $\langle\ell_1,u_1\rangle$ means "at least ℓ_1 and at most u_1." Similarly, the partial temporal element $\langle\ell_2,u_2\rangle$ means "at least ℓ_2 and at most u_2." Hence, $\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ is TRUE for sure if $u_1 \subseteq \ell_2$. Similarly, $\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ is FALSE for sure if $\ell_1 \not\subseteq u_2$. Otherwise, we say that $\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ is UNDEFINED. To further see the motivation for the definition consider the following example. Suppose from the given information in the database we can conclude that a certain condition holds at least during ℓ_1 and at most during u_1. Suppose that we can also conclude that a second condition holds at least during ℓ_2 and at most during u_2. We now want to ask if the second condition holds during the time the first condition holds. This would be TRUE for sure if $u_1 \subseteq \ell_2$ i.e. if the maximum possible time for the first condition is a subset of the minimum possible time for the second condition. Hence $\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ should be TRUE if $u_1 \subseteq \ell_2$. Similarly, $\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ should be FALSE if $\ell_1 \not\subseteq u_2$. Hence we get the following definition.

$\langle\ell_1,u_1\rangle \subseteq \langle\ell_2,u_2\rangle$ = TRUE if $u_1 \subseteq \ell_2$
= FALSE if $\ell_1 \not\subseteq u_2$
= UNDEFINED otherwise.

∩	⟨I,I⟩	⟨∅,∅⟩	⟨∅,I⟩
⟨I,I⟩	⟨I,I⟩	⟨∅,∅⟩	⟨∅,I⟩
⟨∅,∅⟩	⟨∅,∅⟩	⟨∅,∅⟩	⟨∅,∅⟩
⟨∅,I⟩	⟨∅,I⟩	⟨∅,∅⟩	⟨∅,I⟩

¬	
⟨I,I⟩	⟨∅,∅⟩
⟨∅,∅⟩	⟨I,I⟩
⟨∅,I⟩	⟨∅,I⟩

Tables for ∪ and ¬ where I = [0,NOW] (table for ∩ is similar)
Figure 4.2.

We make an observation that TRUE, FALSE, UNDEFINED, ¬, ∨, ∧ are isomorphic to ⟨I,I⟩, ⟨∅,∅⟩, ⟨∅,I⟩, ¬, ∪ and ∩ where I = [0,NOW]. This is clear from Figure 4.1 and Figure 4.2. Now we introduce a function called eval, which takes a partial boolean expression and a tuple τ, and returns one of {⟨I,I⟩, ⟨∅,∅⟩, ⟨∅,I⟩}. eval(f)(τ) is defined in such a way that eval(f)(τ) = ⟨I,I⟩ if and only f(τ) = TRUE, eval(f)(τ) = ⟨∅,∅⟩ if and only f(τ) = FALSE, and eval(f)(τ) = ⟨∅,I⟩ if and only if f(τ) = UNDEFINED. The main use of eval is to simplify the definition of the selection operator and to simplify the statements of some of our results. The function eval allows the evaluation of a partial boolean expression using the operations ∪, ∩ and ¬ for partial temporal elements. Formally, eval(f)(τ) is defined as follows:

- eval(TRUE)(τ) = ⟨I,I⟩ and eval(FALSE)(τ) = ⟨∅,∅⟩
- If μ and ν are partial temporal elements then
  eval($\mu \subseteq \nu$)(τ) = ⟨I,I⟩ if $\mu \subseteq \nu$ is TRUE
  ⟨∅,∅⟩ if $\mu \subseteq \nu$ is FALSE
  ⟨∅,I⟩ if $\mu \subseteq \nu$ is UNDEFINED.
- If μ and ν are more complex partial temporal expressions then eval($\mu \subseteq \nu$)(τ) = eval($\mu(\tau) \subseteq \nu(\tau)$)($\tau$).
- eval(f1 ∨ f2)(τ) = eval(f1)(τ) ∪ eval(f2)(τ)
- eval(f1 ∧ f2)(τ) = eval(f1)(τ) ∩ eval(f2)(τ)
- eval(¬f)(τ) = ¬(eval(f)(τ)).

4.2.1. Example. If τ is John's tuple in Figure 3.2 then let us calculate eval([46,48] ⊆ [[SALARY=60K]])(τ). For John's tuple, [[SALARY=60K]] = ⟨∅, [0,9] ∪ [[46,100]⟩. Since ⟨[46,48], [46,48]⟩ ⊆ ⟨∅, [0,9] ∪ [46,100]⟩ is UNDEFINED, eval([46,48] ⊆ [[SALARY = 60K]])(τ) = ⟨∅,I⟩.

4.3. Relational expressions. Relational expressions are the syntactic counterparts of partial temporal relations and are defined as follows.

4.3.1. Restructuring. Suppose r is a relation over R with key K. The snapshot of r at t was defined in Section 3.5. Two relations are said to be weakly equal if they have the same snapshot at each instant. If K′ ⊆ R such that K′ ⟶ R in each snapshot, then r:K′ is the relation weakly equal to r but having K′ as the key. We require that there is no missing value in the attributes in K′ in the relation r otherwise it is not possible to do the restructuring (No missing value in $\xi\ell u$ means [[ξ]] = u).

4.3.2. Union. Suppose r and s are relations with the same scheme and key. Then r ∪ s also has the same scheme and key. To arrive at r ∪ s we first compute the union of r and s treating them as sets, and then collapse each pair of tuples that agree on all key attributes into a single tuple. Hence the union is an objectwise union with the object being identified by the key values. Note that the collapsing could give an error if the two tuples being collapsed have different values at some non-key attributes at the same instant of time. Hence we need to assume that the union is being performed between compatible relations.

To formally define the union operation we first define $\xi_1\ell_1u_1 \cup \xi_2\ell_2u_2$, the union of two partial temporal assignments, to be $\xi_1 \cup \xi_2\ \ell_1 \cup \ell_2\ u_1 \cup u_2$. Suppose we are given partial temporal relations r and s with the same scheme R, and the same key K⊆R. Tuples $\tau_1 \in$r and $\tau_2 \in$s are said to be *key-equivalent* if they agree on all their key attributes. For key-equivalent tuples τ_1 and τ_2 their union $\tau_1 \cup \tau_2$ is defined attributewise. Thus, $(\tau_1 \cup \tau_2)(A) = \tau_1(A) \cup \tau_2(A)$ for each attribute A in R. We can now define r ∪ s to be the relation {τ: $\tau \in$r and τ is not key-equivalent to any tuple in s} ∪ {τ: $\tau \in$s and τ is not key-equivalent to any tuple in r} ∪ {$\tau_1 \cup \tau_2$: $\tau_1 \in$r and $\tau_2 \in$s and τ_1, τ_2 are key-equivalent}

4.3.3. Difference. Suppose r and s are relations with the same scheme R and the same key K. Then r − s has the same scheme and key. The relation r−s is computed as follows. We start with r. For each tuple τ_1 of r we check to see if there is a key-equivalent tuple in s. If there is no such tuple in s, then τ_1 does not change. If s has such a tuple τ_2, then let the lower limit of the assignments in τ_1 and τ_2 be ℓ_1 and ℓ_2 respectively, and the upper limits be u_1 and u_2 respectively. Now, at any instant in ℓ_2, if for τ_1 and τ_2 the ξ's are defined and agree on all attributes, then that instant is removed from the domain of ξ part of the assignments in τ_1. That instant is also removed from ℓ_1 and u_1, the lower and upper limits of assignments in τ_1. Also, those instants at which the ξs in τ_1 and τ_2 agree on all attributes or may have agreed on all attributes, if all the temporal assignments were completely defined, must be removed from ℓ_1.

Consider the relation emp from Figure 3.2 and the relation emp′ shown below in Figure 4.3. The result of emp − emp′ is shown in Figure 4.4.

NAME	SALARY	DEPT
[41,120]John	[41,50]40K [51,60]50K	[41,70]Shoes
[41,70]	[41,70]	[41,70]
[41,120]	[41,120]	[41,120]

Figure 4.3. The relation emp'

NAME	SALARY	DEPT
[0,40] U [46,100]John	[10,40]30K	[10,40]Toys [46,55]Shoes
[0,40]	[0,40]	[0,40]
[0,40]U[46,100]	[0,40]U[46,100]	[0,40]U[46,100]
[10,50]Tom	[10,45]40K [46,50]60K	[10,50]Toys
[10,50]	[10,50]	[10,50]
[10,50]	[10,50]	[10,50]

Figure 4.4. emp - emp'

4.3.4. Projection. The projection operation allows the user to choose certain columns of a relation. However, we require that all the attributes in the key of a relation must be projected. Thus, if r is a relation with scheme R and key K and if $K \subseteq X \subseteq R$ then $\Pi_X(r) = \{\tau(X): \tau \in r\}$.

4.3.5. Selection. The selection operator is the most powerful operator in temporal databases. The selection operator has the form $\sigma(r;f;\mu)$ where f is a partial boolean expression and μ is a partial temporal expression. As in Section 2.6.3, the computation of the result of a selection operation cannot always be done by merely considering snapshots of relations at each instant of time. The operator uses the function eval defined in Section 4.2. In the incomplete information case, the expression f when applied to a tuple τ yields TRUE, FALSE or UNDEFINED i.e. eval(f)(τ) yields $\langle I,I\rangle$, $\langle \emptyset,\emptyset\rangle$, or $\langle \emptyset,I\rangle$. If f($\tau$) is FALSE, we want to reject the tuple. If f(τ) is TRUE (i.e. eval(f)(τ) = $\langle I,I\rangle$) then we accept the tuple, but restrict it to $\mu(\tau)$ in the result because of the parameter μ. If f(τ) is UNDEFINED (i.e. eval(f)(τ) = $\langle \emptyset,I\rangle$) we are not sure if the tuple τ should be in the result or not and so the lower limits of the assignment in the tuple must be set to $\emptyset$. This lower limit says that we are sure that the object must be present in the result only during $\emptyset$. In other words, it is possible that the object should never be there in the result relation. The tuple must be further restricted to $\mu(\tau)$. All of these requirements are captured by the following definition of $\sigma(r;f;\mu)$.

Let r be a relation, f be a partial boolean expression and μ be a temporal expression. Then $\sigma(r;f;\mu)$ is defined to be $\{\tau\restriction(\text{eval}(f)(\tau)\cap\mu(\tau)): \tau\in r \wedge \tau\restriction(\text{eval}(f)(\tau)\cap\mu(\tau)) \text{ is not empty}\}$. The tuple $\tau\restriction(\text{eval}(f)(\tau)\cap\mu(\tau))$ is not empty if the upper limit of the tuple is not $\emptyset$. In $\sigma(r;f;\mu)$, f and μ, when omitted, default to TRUE and [0,NOW] respectively.

NAME	SALARY	DEPT
[0,100]John	[10,40]30K [41,45]40K	[10,30]Toys [31,55]Shoes
$\emptyset$	$\emptyset$	$\emptyset$
[0,100]	[0,100]	[0,100]
[10,50]Tom	[10,45]40K [46,50]60K	[10,50]Toys
[10,50]	[10,50]	[10,50]
[10,50]	[10,50]	[10,50]

Figure 4.3. Result of selection in Example 4.3.5.1

4.3.5.1. Example. Consider the query *give all details of employees if they had a salary of* 60K *during* [46,48] applied to the emp relation in Figure 3.2. This query can be expressed as $\sigma(\text{emp};[46,48] \subseteq [\![\text{SALARY} = 60K]\!];[0,\text{NOW}])$. For John's tuple $\text{eval}([46,48] \subseteq [\![\text{SALARY} = 60K]\!]) = \langle \emptyset,I\rangle$, which is what we expect since we cannot determine if the condition is TRUE or FALSE with the given information. The result of the algebraic expression is shown in Figure 4.3. For John's tuple the ℓ values are $\emptyset$. This means that we are sure that John's tuple belongs in the relation only during $\emptyset$ i.e. John's tuple may not belong in the relation at all. However, we do carry the ξ values forward in the result since John's tuple may belong in the relation.

NAME	SALARY	DEPT
[0,30]U[56,100] John	[10,30]30K	[10,30]Toys
$\emptyset$	$\emptyset$	$\emptyset$
[0,30]U[56,100]	[0,30]U[56,100]	[0,30]U[56,100]
[10,50]Tom	[10,45]40K [46,50]60K	[10,50]Toys
[10,50]	[10,50]	[10,50]
[10,50]	[10,50]	[10,50]

Figure 4.4. Result of selection in Example 4.3.5.2

4.3.5.2. Example. Consider the relation emp in Figure 3.2. Suppose we want to answer the following question: *give details of employees if they had a salary of* 60K *during* [46,48] *but restrict the information to the time they were in*

the Toys department. This can be expressed as σ(emp; [46,48]$\subseteq$[[SALARY=60K]]; [[DEPT = Toys]]). For John's tuple, eval([46,48] $\subseteq$ [[SALARY = 60K]]) = $\langle\emptyset,I\rangle$ as before, and [[DEPT=Toys]] = $\langle$[10,30],[0,30]$\cup$[56,100]$\rangle$. On the other hand, for Tom's tuple eval([46,48] $\subseteq$ [[SALARY=60K]]) = $\langle I,I\rangle$ and [[DEPT=Toys]] = $\langle$[10,50],[10,50]$\rangle$. The result of the algebraic expression is shown in Figure 4.4.

4.3.6. Cross product. When a tuple of r is concatenated with a tuple of s, to ensure homogeneity in the resultant tuple, we reduce its ℓ value (u value resp.) to the intersection of the ℓ values (u values resp.) of the tuples being concatenated. We also restrict the ξ's to the new u value. The key of r×s is the union of the keys of r and s.

5. Evaluation of the model.

In this section, we analyze our model for incomplete temporal information. In particular, we show the theoretical soundness of our model by proving the properties mentioned in the introduction.

Generalization of the complete information model. A relation r in the model for complete temporal information can be converted to an equivalent relation in the format of the incomplete information model by changing each temporal assignment ξ to the partial assignment ξ[[ξ]][[ξ]]. The fact that the incomplete information model is a generalization of the complete information model is captured by the following theorem. The theorem states that if all the relations had no missing information then an algebraic expression evaluated according to our model would give the same results as the complete information model described in Section 2.

THEOREM 5.1. Suppose δ is a database in the complete information model, and E is an algebraic expression. If δ' is the database in the incomplete information model obtained from δ by replacing every attribute value ξ to ξ[[ξ]][[ξ]], then E(δ') can be obtained from E(δ) by making a similar replacement.

5.2. Completions. In this section we prove results which show that our algebraic expressions give reliable results even when we have incomplete information. We first introduce the notion of completions of a relation. A relation r in our model has correct but incomplete information about the objects it describes. Informally, a relation r′ is a *completion* of a relation r if r′ has complete information but is consistent with r. Thus, if we had complete information about the objects in r it is possible that we would have r′ as our relation. Clearly, there could be many possible completions for a relation r. In fact, the more incomplete the information in r, the larger is the set of possible completions for r. On the other hand the more complete the information in r, the smaller the set of possible completions of r. If r had complete information then there would be just one completion of r, namely r itself.

We now formalize the idea of completions. If a tuple τ has lower limit ℓ and upper limit u, then the object described by τ should be present in the relation at least during ℓ and at most during u. In "reality", the object would be present during some intermediate period u′ such that $\ell \subseteq u' \subseteq u$. Also, time instants during u′ at which we do not have values for an attribute would actually have some value. This motivates the following definitions. An assignment $\xi'u'u'$ is a *completion* of an assignment $\xi\ell u$ if (i) $\ell \subseteq u' \subseteq u$, (ii) ξ' agrees with ξ everywhere that both are defined, and (iii) [[ξ']] = u′.

5.2.1. Example. Consider the assignment to NAME for John's tuple in Figure 3.2. In that assignment we had ξ = [0,100]John, ℓ = [0,50] and u = [0,100]. Thus John must be in the relation at least during [0,50] and at most during [0,100]. An assignment $\xi'u'u'$ such that ξ' = [0,60]John, u′ = [0,60] is a completion of the assignment $\xi\ell u$.

We denote the lower and upper limits of a tuple τ by $\tau.\ell$ and τ.u, respectively. $\tau.\xi$(A) denotes the temporal assignment part of the assignment to A in τ. If μ is a partial temporal expression, $\mu(\tau).\ell$ and $\mu(\tau)$.u denote lower and upper limits of the resulting partial temporal element. If τ and τ' are tuples with the same scheme R and key K such that τ and τ' agree on all attributes in K, then τ and τ' are said to be *key- equivalent.* Now we define a tuple τ' over R to be a *completion* of a tuple τ over R if for all attributes A$\in$R, $\tau'.\xi$(A) is a completion of $\tau.\xi$(A). Thus, if τ and τ' are tuples over R, then τ' is a completion of τ if (i) $\tau.\ell \subseteq \tau'.\ell = \tau'.u \subseteq \tau.u$, (ii) for all A $\in$ R, $\tau'.\xi$(A) is defined everywhere along τ'.u, and (iii) for all A $\in$ R, $\tau'.\xi$(A) and $\tau.\xi$(A) agree everywhere both of them are defined. A relation r′ is a *completion* of a relation r if (i) given a $\tau \in$ r with $\tau.\ell \neq \emptyset$ there is a $\tau' \in$ r′ such that τ' is a completion of τ, and (ii) given a $\tau' \in$ r′ there is a $\tau \in$ r such that τ' is a completion of τ. A *completion* of a database is a natural extension.

5.2.2. Example. The relation shown in Figure 5.1 is one of the possible completions of the relation emp shown in Figure 3.2.

NAME	SALARY	DEPT
[0,60]John	[0,40]30K [41,60]40K	[0,9]Auto [10,30]Toys [31,60]Shoes
[0,60]	[0,60]	[0,60]
[0,60]	[0,60]	[0,60]
[10,50]Tom	[10,45]40K [46,50]60K	[10,50]Toys
[10,50]	[10,50]	[10,50]
[10,50]	[10,50]	[10,50]

Figure 5.1. A completion of the emp relation

LEMMA 5.1. Let τ,τ' be tuples over R such that τ' is a completion of τ. Let μ be a partial temporal expression. Then $\mu(\tau).\ell \subseteq \mu(\tau').\ell$ and $\mu(\tau).u \supseteq \mu(\tau').u$.

PROOF. By induction on complexity of μ.

LEMMA 5.2. Let τ,τ' be tuples over R such that τ' is a completion of τ. Let f be a partial boolean expression. Then
1. if eval(f)(τ') = ⟨I,I⟩ then eval(f)(τ) = ⟨I,I⟩ or ⟨∅,I⟩
2. if eval(f)(τ') = ⟨∅,∅⟩ then eval(f)(τ) = ⟨∅,∅⟩ or ⟨∅,I⟩
3. if eval(f)(τ) = ⟨I,I⟩ then eval(f)(τ') = ⟨I,I⟩
4. if eval(f)(τ) = ⟨∅,∅⟩ then eval(f)(τ') = ⟨∅,∅⟩

PROOF. Parts 1 and 2 are proved together by induction on complexity of f. Parts 3 and 4 are proved by contradiction.

Part 1 of Lemma 5.2 shows that if τ' is a completion of τ (and hence τ' is consistent with τ but has complete information) and if f(τ') evaluates to TRUE then in our model f(τ) will not evaluate to FALSE in spite of τ having incomplete information. Similarly, part 2 of the above lemma states that if f(τ') evaluates to FALSE then we will not evaluate f(τ) to be TRUE in spite of incomplete information. Part 3 of the lemma states that if we evaluate f(τ) to be TRUE then f(τ') is TRUE for every tuple τ' which is a completion of τ. Similarly, according to part 4 of the lemma, if we evaluate f(τ) to be FALSE then f(τ') is FALSE for every tuple τ' which is a completion of τ.

LEMMA 5.3. Let τ,τ' be tuples over R such that τ' is a completion of τ. Let f be a partial boolean expression. Then
1. eval(f)(τ).$\ell \subseteq$ eval(f)(τ').ℓ
2. eval(f)(τ).u $\supseteq$ eval(f)(τ').u

PROOF. Immediate from statements 1 and 2 of LEMMA 5.2 and the fact that eval(f)(τ') is either ⟨∅,∅⟩ or ⟨I,I⟩.

We now state a theorem which shows that the results of our algebraic expressions are reliable. Suppose δ is the state of a database having incomplete information. Then a completion of δ (say δ') has complete information and is consistent with the information in δ. Hence, δ' is a possibility for δ in the "real" world. Since our model generalizes the complete information model we know that when there is complete information our model gives correct information. Hence, E(δ') should be a possibility for the result when E is applied to δ. In other words if E(δ') is a completion of E(δ) we may say our result E(δ) is reliable because then E(δ) does allow E(δ') as a possibility with complete information. The following theorem is proved by induction on complexity of E. The above lemmas are used in the proof for the case of selection.

THEOREM 5.2. Let δ be the state of a database and let E be an algebraic expression applied to the database. Then for any completion δ' of the database δ, E(δ') is a completion of E(δ).

The above theorem is analogous to the notion of adequacy of operators as defined in [Bi83]. However, a transition to our model is not straightforward. This is because (i) in [Bi83] AθB and Aθb are defined only when θ is the equality comparison, and (ii) our selection operation is of the form $\sigma(r;f;\mu)$ and it cannot be evaluated using snapshots of relations at instants of time (see Section 2.6.3).

5.3. Information content of a relation. The preceding theorem showed that the results of algebraic expressions are reliable in the sense that they never produce incorrect information. However, we want to produce results that have as much information as possible.

As we mentioned earlier, the more incomplete the information in r, the larger is the set of possible completions for r. On the other hand the more complete the information in r, the smaller the set of possible completions of r. This motivates the following definition for *more informative* relations. The set of completions of a relation r, denoted C_r, is given by $C_r = \{r' : r' \text{ is a completion of } r\}$. A relation r_2 is *more informative* than a relation r_1 if $C_{r_2} \subseteq C_{r_1}$. A relation r_2 is *as informative* as r_1 if $C_{r_2} = C_{r_1}$. A relation r_2 is *strictly more informative* than a relation r_1 if $C_{r_2} \subset C_{r_1}$.

To find if r_2 is more informative than r_1 based on the above definition one has to compute the set of completions for the relations r_1 and r_2. However, a relation has a potentially infinite set of completions. Hence, a more syntactic notion is needed by which one can determine if a relation r_2 is more informative than a relation r_1. Hence, we introduce the notion of *extensions*. The relationship

between extensions and information content of relations is then captured in Theorem 5.3.

Let τ and τ' be tuples over a scheme R. Then τ' is an *extension* of τ iff the following conditions hold (i) $\tau.\ell \subseteq \tau'.\ell \subseteq \tau'.u \subseteq \tau.u$, (ii) for all $A \in R$, $\tau'.\xi(A)$ is defined everywhere in $\tau'.u$ where $\tau.\xi(A)$ is defined, and (iii) for all $A \in R$, $\tau'.\xi(A)$ and $\tau.\xi(A)$ agree everywhere both of them are defined. A tuple τ' is a *proper extension* of τ if (i) τ' is an extension of τ, and (ii) τ is not an extension of τ'.

These definition can be extended to relations. A relation r′ is an *extension* of r if (i) given a tuple $\tau \in r$ with $\tau.\ell \neq \emptyset$, there is a $\tau' \in r'$ such that τ' is an extension of τ, (ii) given a tuple $\tau' \in r'$ there is a $\tau \in r$ such that τ' is an extension of τ. A relation r′ is a *proper extension* of r if (i) r′ is an extension of r, and (ii) r is not an extension of r′. The following theorem shows the relation between extensions and more informative relations. The proof is omitted due to lack of space.

THEOREM 5.3. Let r_1 and r_2 be two relations. Then (i) r_2 is more informative than r_1 iff r_2 is an extension of r_1, and (ii) r_2 is strictly more informative than r_1 iff r_2 is a proper extension of r_1.

We now examine how far our definition of the operations produce as much information as is possible. Our result is analogous to the restrictedness property of operators for classical databases with null values as developed in [Bi83]. Again, however, our result cannot be obtained as a straightforward transition from the snapshot case to the temporal case. The proof is omitted due to lack of space.

THEOREM 5.4. Let r_1, r_2 be two relations and let r_3 be a proper extension of $r_1 \odot r_2$ where $\odot$ is a relational operator. Then there are completions r_1', r_2' of r_1, r_2 respectively such that $r_1' \odot r_2'$ is not a completion of r_3. (This shows that the result of $r_1 \odot r_2$ cannot be strengthened to r_3). For unary operators we have a similarly statement. The above statement holds for the following operators

- $\cup, -, \Pi, \times$, restructuring
- σ of the form $\sigma(r; ;\mu)$, $\sigma(r; ;[\![A]\!])$, $\sigma(r; ;[\![A\,\theta b]\!])$, $\sigma(r; ;[\![A\,\theta B]\!])$, $\sigma(r; \mu\subseteq[\![A]\!];)$, $\sigma(r; \mu\subseteq[\![A\,\theta b]\!];)$, $\sigma(r; \mu\subseteq[\![A\,\theta B]\!];)$ where μ is a temporal element, and θ is one of $<, \leq, >, \geq, \neq$. (Note that it does not hold if θ is $=$, see Example 5.3.1.).

This theorem shows that if the results of the operations listed above were extended to be more informative then we would lose the property of reliability of our results.

5.3.1. Example. We show that Theorem 5.4 does not hold for selection operations of the form $\sigma(r; ;[\![A\,\theta b]\!])$ etc. where θ is $=$. This is shown by the following counterexample. Let r be the relation over the scheme AB with key A as shown in Figure 5.2(a). Then the result of $\sigma(r; ;[\![B=b]\!])$ is shown in Figure 5.2(b). The relation r_3 shown in Figure 5.2(c) is an extension of $\sigma(r; ;[\![B=b]\!])$ (the term extension was defined in Section 5). However, for every completion r′ of r, $\sigma(r'; ;[\![B=b]\!])$ will be a completion of r_3. Note that this selection could be easily defined so that r_3 is in fact the result of the selection. In that case, we would be paralleling [Bi83].

The case $\sigma(r; ;[\![B\neq b]\!])$ would work if dom(B) has at least three elements. We have implicitly assumed this to be the case. For $\sigma(r; ;[\![B<b]\!])$ we assume there are at least two elements $< b$. For $\sigma(r; ;[\![B>b]\!])$ we assume there are at least two elements $> b$.

A	B
[0,10]a	[0,5]b
[0,5]	[0,5]
[0,10]	[0,10]

(a) Relation r

A	B
[0,10]a	[0,5]b
[0,5]	[0,5]
[0,10]	[0,10]

(b) $\sigma(r; ;[\![B=b]\!])$

A	B
[0,10]a	[0,10]b
[0,5]	[0,5]
[0,10]	[0,10]

(c) The relation r_3

Figure 5.2. A counterexample for $\sigma(r; ;[\![B=b]\!])$

6. Conclusions.

We have presented a model for partial temporal databases which meet the expectations stated in the introduction. Our model clearly generalizes the complete information model. Our model allows the storage of incomplete temporal information and provides a powerful algebra to query the incomplete information. We showed that the algebraic expressions produce reliable results even when we have incomplete information in the database. We also showed that except for certain cases of selection, if the definition of our operators were strengthened to give more information, we may get results that are not reliable. We end this paper with a few remarks.

The inherent querying capability of the algebra. We term the queries in a complete information model as *standard*. The standard queries can be submitted to the incomplete information model without any changes in the syntax. This means our algebra is a seamless extension of the algebra for complete information. In addition our algebra has the inherent capability to express *nonstandard* queries i.e. queries involving uncertainty. To achieve the said inherent capability, we only have to add primitives for constructs

mentioned in Section 3.2. For example, if we define $\epsilon(A) = A.\ell - [\![A.\xi]\!]$, then this primitive captures the time when the object should be definitely present in the relation but its A-values are unknown. Then the query *give salaries of employees when their department was unknown while they surely worked for the organization* is expressed as $\Pi_{\text{NAME SALARY}}(\sigma(\text{emp; TRUE; } \epsilon(\text{DEPT})))$.

Our work as generalization of Biskup's. Except for some differences, our formalism is a generalization of Biskup's formalism when key values are required to be known. In Biskup's formalism, information is maintained only for the current instant NOW and the concept of *maybe* tuples is introduced by adding a STATUS column which has a 'd' for *definite* tuples and an 'm' for *maybe* tuples. In our model this can be achieved by restricting the upper limits of partial temporal assignments to [NOW,NOW]. Then definite tuples have lower limit [NOW,NOW] and maybe tuples have lower limit $\emptyset$.

Further work. Imielinski and Lipski [IL84] introduce *condition tables* in which they use *marked* nulls and an additional column to store conditions which must be satisfied by each tuple. This helps them obtain a theorem which states that all valid conclusions expressible by relational expressions are in fact derivable in their system. It would be useful to investigate these ideas in the context of incomplete temporal information.

Acknowledgments. We would like to thank the anonymous referees for their comments in improving the presentation of the paper.

References.

[Bi83] Biskup, Joachim. *A foundation of Codd's relational maybe-operations.* ACM Transactions on Database Systems, Vol 8, No. 4, 1983.

[CC87] Clifford, James and Albert Croker. *The historical data model (HRDM) and algebra based on lifespans.* Proc. of 3rd IEEE International Conference on Data Engineering, 1987.

[Co79] Codd, E.F. *Extending the database relational model to capture more meaning.* ACM Transactions on Database Systems, Vol. 4, No 4.

[Ga86a] Gadia, Shashi K. *Weak temporal relations.* Proc. of Fifth Annual ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1986.

[Ga86b] Gadia, Shashi K. *Toward a multi-homogeneous model for a temporal database.* Proc. of the IEEE International Conference on Data Engineering, 1986.

[Ga88] Gadia, Shashi K. *A homogeneous relational model and query languages for temporal databases.* ACM Transactions on Database Systems, Vol 13, No. 4, 1988.

[GY88] Gadia, Shashi K. and Chuen-Sing Yeung. *A generalized model for a relational temporal database.* Proc. ACM SIGMOD International Conference on Management of Data, 1988.

[GY91] Gadia, Shashi K. and Chuen-Sing Yeung. *Inadequacy of Interval Timestamps in Temporal Databases.* Information Sciences, Vol 54, pp 1-22, 1991.

[IL84] Imielinski, Tomasz and Witold Lipski, Jr. *Incomplete information in relational databases.* Journal of the ACM, Vol 13, No. 4, 1984.

[Li81] Lipski, Witold. *On databases with incomplete information.* Journal of ACM, Vol 28, 1981.

[NA89] Navathe, Shamkant and Rafi Ahmed. *A temporal relational model and query language.* Information Systems, Vol 49, 1989.

[Re86] Reiter, R. *A sound and sometimes complete query evaluation algorithms for relational databases with null values.* Journal of ACM, Vol 33, 1986.

[Sa90] Sarda, Nandlal. *Algebra and query language for a historical data model.* The Computer Journal, Vol 33, 1990.

[Sn87] Snodgrass, Richard. *The temporal query language TQuel.* ACM Transactions on Database Systems, Vol 12, pp 247-298, 1987.

[Ta86] Tansel, Abdullah U. *Adding time dimension to relational model and extending relational algebra.* Information Systems, Vol 11, No 4, 1986.

[TM75] Tremblay, J. P. and R. Manohar. *Discrete mathematical structures with applications to Computer Science.* McGraw-Hill Computer Science Series, 1975.