In order to obtain a suitable model of the universe of discourse of some information systems, it is important to allow the description of imprecise objects with temporal indeterminacy. These objects are called indeterminate temporal objects. This paper presents a formalism, called LITO (Logic of Indeterminate Temporal Objects), for the representation, querying and modification of such objects.

1 Introduction
In certain applications, it is fundamental to consider historical objects. Moreover, the information about the objects of the application frequently cannot be completely determined. However, the current computer information systems (including the database management systems) do not feature resources to represent, retrieve and modify such objects, here called indeterminate temporal objects.

Researches about incomplete information in information systems exist since the 1970’s [11]. However, their integration with database systems is restricted to the null value concept for an attribute. Similarly, research about maintenance of that object was not so much undertaken [5]. The TSQL2 language, which was created to Temporal Databases, has a limited concept of indeterminate information [14]. So, both formalisms and temporal query languages present partial solutions to representing and manipulating indeterminate temporal objects.

The main objective of this paper is to show a formalism that gives definitions about representation and manipulation of indeterminate temporal objects in indeterminate temporal database (ITDB). More precisely, we herein introduce LITO (Logic of Indeterminate Temporal Objects), a formalism that allows for retrieval query, logical query, insertion, logical deletion, refinement and update of indeterminate temporal objects. Furthermore, a computer system was implemented in LPA-Prolog [9] to show how LITO can be applied in practice.

Section 2 shows the theoretical foundation for temporal database and indeterminacy. Related works are treated in section 3. Section 4 describes LITO. The last section encompasses final comments.

2 Temporal Database and Indeterminacy
We consider time as a discrete, infinite, and totally ordered set of points called *instants*. Given two instants \( t_1 \) and \( t_2 \), the set of all instants between them, including \( t_1 \) and \( t_2 \), is called an *interval* and is denoted as \([t_1, t_2]\). A set of one or more disjoint intervals is called a *time element*.

A temporal database is a database where time elements are associated to some *data elements* (attributes and objects). These time elements determine the *valid time*, that is, they specify the time when the corresponding information holds in the real world modeled by the database. Another time dimension, known as transaction time [7], represents the times when a fact was stored in the database. Data elements as well as time elements may be well determined or more or less indeterminate. For data elements, this corresponds to situations where “we know exactly what” or “we do not know exactly what” while, for time elements, the meaning is that “we know exactly when” or “we do not know exactly when”.

A database models a part of the real world called universe of discourse. If the available information is complete and precise, then there is a clear correspondence between the database and the universe of discourse. When the knowledge about the real world is incomplete, several scenarios are possible, but it is not known which of them represents the real status of the world. A database containing incomplete information implicitly represents a set of possible worlds.

Possible-world scenarios are supported by the so-called open-world assumption (OWA) [8]. Under the usual closed-world assumption, facts not explicitly stored as *true* nor deducible from true facts are considered *false*. Under the OWA, a statement is considered false only if it is explicitly stored as *false* or if its negation can be logically derived from the database. The information neither explicitly present in the database nor derivable from explicit information is considered *unknown*; the information contained in one but not all possible worlds is considered *possible*. The information contained in all possible worlds is considered *true* while information not present in any possible worlds is considered *false*.

Temporal indeterminacy characterizes situations where an instant is not perfectly defined but is described instead as an interval of possible values. For instance, *during* 2001 means, with a granularity of day, “some day between 1/1/2001 and 31/12/2001”.

An *indeterminate interval* is an interval delimited by at least one indeterminate instant [7]. A temporal database where both *data elements* and *time elements* can be indeterminate is called an Indeterminate Temporal Database (ITDB).

### 3 Related Work

The formalization of a theory for modification operations in temporal database does not exist. The same happens for manipulation operations in ITDB. TSQL2 language [14] only considers retrieval operation to obtain registers in valid timetables, i.e., it does not consider modification operations. This limitation compromises the language’s usability for real situations.

TSQL2 provides a form of representation for indeterminate instants and other way to represent indeterminate periods [4]. For instance, the notation 10/5 ~ 29/5 is used to denotes an instant between 10 and 29 of May, and the notation [1/3/99 ~ 31/3/99 – 1/6/99 ~ 30/6/99] is used to denotes a set of possible periods.

TSQL2’s indeterminate periods are considered in LITO by the temporal operator *during*. Besides this one, LITO supports other temporal operators such as *before*, *after* and * providing more options for time representation and more realistically modeling the universe of discourse [1,2].

The Logic of Incomplete Knowledge Base [10] is an adequate method to represent objects in OWA. Wherever, it treats neither temporal indeterminacy nor modification of ob-
jects. For this reason, LITO has developed new axiomatic and deduction rules for indeterminate temporal objects.

The majority of proposed query languages to temporal database extend only the SQL’s select command. Elsewhere, SQL/Temporal [12] and an extension proposal of SQL [13] are the first approaches for temporal modification. Unfortunately, they evolve only periods with determinate time.

4 A Formalism for Indeterminate Temporal Objects

This formalism aims at supporting manipulation operations in both temporal database and ITDB. It is out of the paper’s scope to present the complete formalism that can be found in [1]. So, this section shows just a part of the formalism.

4.1 The Language

LITO is formed by a triple \(<L, A, R>\) where \(L\) is the logic’s language, \(A\) is the set of axioms that characterizes the temporal and modal operators, and \(R\) is the set of rules to be used for deduction of new facts (modus ponens) and to process querying of objects.

4.1.1 Alphabet

The alphabet of LITO is the minimal set of basic symbols

**Logic symbols:**
- **Punctuation:** “(“, “)”, “[“,”]”, “,”
- **Connectives:** \(\neg, \lor\)
- **Quantifier:** \(\forall\)
- **Modal operators:** \(M, K\)
- **Temporal operators:** during, begin, end
- **Non-temporal operators:** not, or, lt, in
- **Variable:**
  - non-temporal: \(x, x_1, x_2,\ldots\)
  - temporal instant: \(\infty, now, t, t_1, t_2,\ldots\)
  - temporal interval: \(I, I_1, I_2,\ldots\)

The basic modal operator \(M\) (Maybe) describes indeterminate information and the basic modal operator \(K\) (Known) characterizes information represented in ITDB. These operators have the same signification used in both modal logic [6] and incomplete knowledge base logic [10], respectively. Non-temporal operators describe indeterminate non-temporal information, i.e., in this paper, they describe imprecise, unknown, and negative information, according to [1]. Temporal operators will be explained in section 4.2.

**Non-logic symbols:**
- **constants:**
  - temporal: \(*\)
  - non-temporal: \(*, none\) e outras cadeias de caracteres.
- **Predicative symbols:** \(Time, <, =, P, P_1, P_2, \ldots\)

The temporal constant \(\ast\) is necessary for representing both temporal indeterminate values and non-temporal indeterminate values, while the non-temporal constant \(none\) denotes a non-existent unknown information.

The set of predicative symbols is open. Each such symbol (except the three predicate symbols \(Time, <, \) and \(=\)) must begin with the letter \(P\). The symbol \(Time\) denotes the valid
time of an object. Its first argument is a non-temporal variable that identifies the object. The second argument is a temporal term (section 4.1.2).

4.1.2 Terms

Terms are expressions of the language $L$, they are used to denote information capable of being stored in ITDB. They are classified in non-temporal terms, temporal instant terms, and temporal interval terms. The $L$’s set of terms is the minimal set satisfying the following conditions:

Non-temporal terms:
- Each non-temporal constant is a non-temporal term;
- Each non-temporal variable is a non-temporal term;
- if $x$ is a non-temporal variable, and $T_x$, $T_{x_1}$ and $T_{x_2}$ are non-temporal terms, then $\text{or}(T_{x_1}, T_{x_2})$, $\text{not}(T_x)$, and $\text{lt} x$ are non-temporal terms.

Definite temporal instant and indefinite temporal instant terms:
- the temporal constants * and $\infty$ are indefinite temporal instant terms;
- each temporal instant variable is a definite temporal instant term;
- if $T_I$ is a temporal interval term, then $\text{during} T_I$ is an indefinite temporal interval term;
- if $T_I$ is a temporal interval term, then $\text{begin} T_I$ and $\text{end} T_I$ are indefinite temporal instant terms;

The distinction between indefinite and definite terms occurs because the order relation ($<$) is not applicable to terms * and $\text{during} T_I$. These terms do not precisely define the temporal instant represented. The order relation is applicable only to definite temporal instant terms. The general denomination “temporal instant term” is used to describe both types of temporal instant terms.

Temporal interval terms:
- each temporal interval variable is a temporal interval term;
- if $T_{t_1}$ and $T_{t_2}$ are temporal instant terms, then $[T_{t_1}, T_{t_2}]$ is a temporal interval term.

4.1.3 Formulas

Formulas are expressions of the language $L$, they are used to denote facts in the domain of the represented application. A well-formed formula (wff) is recursively defined as follows:

- if $x$ is a non-temporal variable, $T_x$, $T_{x_1}$ and $T_{x_2}$ are non-temporal terms, and $T_I$ is a temporal interval term, then $T_{x_1} = T_{x_2}$, $P(x)$, $P(x, T_x)$, $\text{Time}(x, T_I)$ are wffs;
- if $F$ and $G$ are wffs, then $\neg F$, $F \lor G$, $M F$ and $K F$ are wffs;
- if $F$ is a wff and $v$ is a non-temporal variable, a temporal instant or some temporal interval, then $\forall v (F)$ is a wff;
- if $T_{t_1}$ and $T_{t_2}$ are definite temporal instant terms, then $T_{t_1} = T_{t_2}$ and $T_{t_1} < T_{t_2}$ are wffs;
- if $T_{I_1}$ and $T_{I_2}$ are temporal interval terms, then $T_{I_1} = T_{I_2}$ is a wff.

Note that the predicative symbol $P$ can express both a class of objects and the attributes of this class. In the first case, $P$ is the name of class and $x$ contains the identifier of the class’ objects. An attribute of the object $x$ will be represented by a binary predicative symbol $P(x, T_x)$ such that $T_x$ represents the attribute value and $P$ is its name.

4.1.4 Definitions

Definitions are useful to construct new symbols of the $L$ from other symbols already specified. So, if $F$ and $G$ are wffs then we can define the following symbols:

1. $\exists x (F) \iff_{\text{def}} \neg \forall x (\neg F)$
2. \( F \land G \iff \neg (\neg F \lor \neg G) \)

3. \( F \rightarrow G \iff \neg F \lor G \)

4. \( F \leftrightarrow G \iff (F \rightarrow G) \land (G \rightarrow F) \)

These four definitions are classical. They define existential quantification, conjunction, implication and logic equivalence, respectively. The next two definitions are applicable only to definite temporal instant terms.

5. \( T_{t1} \in [T_{t2}, T_{t3}] \iff (T_{t2} \leq T_{t1}) \land (T_{t1} \leq T_{t3}) \)

This relation of pertinence characterizes temporal interval as a linear and connected set.

6. \( I = [t_1, t_2] \iff \text{begin } I \land t_2 = \text{end } I \)

An interval \( I \) represented by \([t_1, t_2]\) is delimited by definite temporal instants \( t_1 \) and \( t_2 \), where \( t_1 \) is the initial instant of \( I \) and \( t_2 \) is its final instant.

7. \( I_1 \subseteq I_2 \iff (\text{begin } I_2 \leq \text{begin } I_1) \land (\text{end } I_1 \leq \text{end } I_2) \)

The relation “is contained in or is equal to” applied to intervals is defined according to their limits.

Definitions 8 and 9 use the linear and discrete structures of the time and they respectively express the successor and predecessor operators for temporal instants.

8. \( t_2 = \text{succ } t_1 \iff (t_1 < t_2 \land \forall t_3 (t_1 < t_3 \rightarrow (t_2 \leq t_3))) \)

9. \( t_2 = \text{pred } t_1 \iff \text{succ } t_2 = t_1 \)

A temporal instant completely undetermined is indicated by \(*\).

10. \(* \iff \text{during } [\infty, \infty] \)

11. \( \text{before } t \iff \text{during } [\infty, \text{pred } t] \)

12. \( \text{after } t \iff \text{during } [\text{succ } t, \infty] \)

### 4.2 Axiomatics

The axioms of LITO are grouped in basic, temporal and non-temporal axioms, according to their predicative symbols.

**Basic axioms:**

Basic axioms encompass the axioms of first-order logic plus the total order predicate \((\leq)\).

**Modal axioms:**

Modal axioms characterize the modalities Maybe \((M)\) and Known \((K)\), besides negation.

Facts quantified by the \(K\) operator represents the database and those with \(M\) are derived possibilities.

1. \( \forall x (F \lor G) \rightarrow MF \)

   Each component of a conjunction can be true.

2. \( \forall x \neg K \neg F \rightarrow MF \)

   Facts that are not known as false possibly are true facts (OWA).

**Non-temporal axioms:**

Non-temporal axioms encompass only binary predicate \(P(x, Tx)\) where \(P\) denotes a non-temporal attribute of an object.

3. \( \forall x \forall x_1 \forall x_2 (KP(x, or(x_1, x_2)) \rightarrow MP(x, x_1)) \)

   When the value of a predicate \(P\) stored in ITDB is a disjunction of values, then each of
them is possibly true. Note that $MP(x, x_2)$ also holds because disjunctions are commutative.

4. $\forall x \forall x_1 (KP(x, *) \rightarrow \exists x_1 P(x, x_1) \land \neg KP(x, x_1))$

This is a generalization of axiom 8 for cases when nothing is known about some value, besides its existence.

5. $\forall x (KP(x, none)) \rightarrow \forall x_1 \neg P(x, x_1)$

The null value is not applicable to the considered object.

Temporal axioms:
Temporal axioms include the predicates $Time(x, TI)$, i.e., predicates determining the time when an object is valid.

6. $\forall x \forall I (MTime(x, I) \leftrightarrow \exists t (t \in I \land Time(x, [t, t]) \land \neg KTime(x, [t, t])))$

The time of an object is possibly valid in the interval $I$, iff it is valid for at least an instant in $I$ unknown by ITDB.

7. $\forall x \forall I \forall t_1 (Time(x, [during I, t_1]) \rightarrow \exists t (t \in I \land Time(x, [t, t_1]) \land \neg KTime(x, [t, t_1])))$

The interval represented by $[during I, t_1]$ is valid starting from an $I$’s initial instant $t$ which is unknown by ITDB.

8. $\forall x \forall t_1 \forall I (Time(x, [t_1, during I]) \rightarrow \exists t (t \in I \land Time(x, [t_1, t]) \land \neg KTime(x, [t_1, t])))$

The interval represented by $[t_1, during I]$ is valid starting from an $I$’s final instant $t$ which is unknown by ITDB.

4.3 Rules
In order to simplify our explanation, rules are categorized in basic rule and retrieval rules.

Basic rule:
1. if $\neg F$ and $\neg F \rightarrow G$ then $\neg G$ (modus ponens)

Retrieval rules:
The rules for processing queries are given as Horn clauses or formulas reducible to Horn clauses. When a query is formulated the $x$ variable of the head predicate should be instantiated [3]. The other variable is free and will contain the result of the rule’s application, i.e., it will contain the answer.

Non-temporal retrieval rules:
These rules do not entail the retrieval of temporal values. The value of the argument $x_1$, denoted by predicate $P_1(x, x_1)$ of a query, can match completely or partially with a value of the same predicate in ITDB. The set of possible values of an undetermined attribute (for instance, salary $(x, \lt 1000)$ is called the predicate’s domain.

The matching is complete if both query and ITDB predicate’s domains are precisely similar. Elsewhere, i.e., if there is an intersection between these domains, the matching is partial. If there is no intersection between these domains, the resulting value is the unknown value.

The second argument of the predicate $P_2(x, x_2)$ specifies retrieval values from ITDB. These values are qualified by the true value $M$ (maybe), $\neg$ (no) ou yes (when a symbol is absent). For instance, the query “What is Mary’s salary?” will be satisfied by the following rule:

$$
employee(x, mary) \land Kemployee(x, mary) \land Ksalary(x, x_2) \rightarrow
$$
salary(x, x2)

There are rules where the values of the queried argument x1 match completely, and there are rules where they match partially.

2. \( P_1(x, x_1) \land KP_1(x, x_1) \land KP_2(x, or(x_2, x_3)) \rightarrow MP_2(x, x_2) \)

3. \( P_1(x, x_1) \land KP_1(x, x_1) \land KP_2(x, *) \rightarrow MP_2(x, x_2) \)

4. \( P_1(x, x_1) \land KP_1(x, or(x_1, x_3)) \land KP_2(x, x_2) \rightarrow MP_2(x, x_2) \)

5. \( P_1(x, x_1) \land KP_1(x, *) \land KP_2(x, x_2) \rightarrow MP_2(x, x_2) \)

Temporal retrieval rules:
These rules entail only the retrieval of temporal values. The answers of a query depend of the relation between the times of the query predicates and the ITDB predicates.

6. \( Time(x, I_1) \land KP(x, x_1) \land KTime(x, I_2) \land I_1 \subseteq I_2 \rightarrow P(x, x_1) \)

7. \( Time(x, I_1) \land KP(x, x_1) \land KMTTime(x, I_2) \land I_1 \subseteq I_2 \rightarrow MP(x, x_1) \)

When the object’s time is queried, the true value returned will be determined by the precision in the Time predicate and/or in the non-temporal predicate.

8. \( P(x, x_1) \land KP(x, x_1) \land KTime(x, I) \rightarrow Time(x, I) \)

9. \( P(x, x_1) \land KP(x, x_1) \land KMTTime(x, I) \rightarrow MT(x, I) \)

4.4 Application

We illustrate the application of the logic with a simple example. Assume a class of objects of type employment(employee, company, time) containing objects O1 and O2 represented as follows:

\( employment(O1) \land employee(O1, or(Peter, Mary)) \land company(O1, or(UFPE, UFPB)) \land time(O1, [1/1/1994, *]) \)

\( employment(O2) \land employee(O2, nor(Mary)) \land company(O2, (UFC) \land time(O1, [before 01/01/1983, now]]) \)

O1 represents an employment of a person, which is Pedro or Maria, working for UFPE or UFPB since 1/1/1994 up to an unknown date. The object O2 is an employment of a person different from Maria, working at UFC since before 1983.

The query "does Mary work for UFPB at 1/1/2001?" is formalized as

\( employee(x, Mary) \land company(x, UFPB) \land time(x, 1/1/2001) \)

So, since company(x, UFPB) matches partially with rule 4, the predicate employee(x, Mary) of the query is instantiated in this rule together with the facts of the database and results

\( employee(O1, Mary) \land Kemployment(O1, or(Peter, Maria)) \land Kcompany(O1, UFPB) \rightarrow Mcompany(O1, UFPB) \)

The same reasoning can be applied to rule 9 resulting

\( employee(O1, Mary) \land Kcompany(O1, UFPB) \land KMTtime(O1, 1/1/2001) \rightarrow MTtime(O1, 1/1/2001) \)

Combining these results and interpreting M as a possibility, we get the final answer possible.

5 Final Remarks
The representation and manipulation of indeterminate temporal objects is fundamental to some applications using database management systems. In spite of this, the formalisms so far proposed and widely accepted have not satisfactorily tackled the problem.

This paper suggests a formalism called LITO that contains complete definitions about representation and manipulation of objects in both ITDB and temporal database. New forms were proposed to represent the non-temporal and temporal indeterminacy present in these objects. Also, conception and construction of mechanisms for logic queries, retrieval queries, insertions, logic deletions, refinements and updates in these databases were elaborated. All these new characteristics make this formalism a model to treat indeterminate temporal objects. A system was implemented in LPA-Prolog [9] to show the formalism’s applicability.

References