Mariusz Giero
University of Białystok

APPLICATION OF BITEMPORAL DATABASES CONTAINING MEDICAL DATA[12]

Most databases store time-varying information. The research area of temporal databases aims to provide expressive and efficient ways to model, store, and query temporal data. The paper concentrates on the application of temporal logic in the research. It shows how the language of first-order temporal logic can be used to query temporal data.

1. Introduction

A temporal database [Etz, Ste, Tan1] is defined as the one maintaining object histories, i. e., past (before modification), present, and possibly future data (e.g. database is used for forecasts). Maintaining temporal data within traditional relational databases [Cod, Dat] is not straightforward.

Example 1

In a traditional relational database, deletion and modification operation causes physical removing of data and overwriting data, respectively. It is not possible to trace the history of records. Let us consider a database with one relation TREATMENT (id, medicine, when) where the course of patient's treatment is stored. Attribute *id* is for the patient identification number, attribute *medicine* is for the name of the medicine the patient is treated with and attribute *when* is for the date of treatment:

[12] The research reported in this paper is a part of the project entitled Temporal representation of knowledge and its implementation in the computer systems of medical conducts supported by the Polish Ministry of Science and Higher Education, grant no. 3 T11F 011 30.

ISBN 978–83–7431–231–8 ISSN 0860-150X

ID	MEDICINE	WHEN
1	A	10 X – 15 X
2	B	13 X – 16 X
3	A	13 X – 20 X

After deleting the first record and modifying the third one, we obtain:

ID	MEDICINE	WHEN
2	B	13 X – 16 X
3	C	13 X – 15 X

Data about patient 1 and treatment of patient 3 with A is lost.

There are numerous application domains that require keeping previous states: Medical Systems (e.g. patients records), Computer Applications (e.g. history of files back ups), Archive Management Systems (e.g. sporting events, publications and journals), Reservation Systems (e.g. when was a flight booked), and many others. Snodgrass [Sno] lists 16 various areas with time-varying data. In fact, it is hard to find areas that do not contain time-varying data.

One of the ways ([JenSno, Jen, Tan2]) to extend a traditional relational database to be a temporal one is by adding to each relation two attributes: *valid time* and *transaction time*. Valid time attribute (VT) is for recording time when a record is valid in the world represented by the database. Transaction time attribute (TT) is for recording time when a record is stored (from insertion to possible deletion) in the database. Applying the solution, the database from example 1 can be represented in the following way:

Table 1. Example of temporal database

ID	MEDICINE	VT	TT
1	A	10 X – 15 X	12 X – now
2	B	13 X – 16 X	10 X – now
3	A	13 X – 20 X	5 X – now

The attribute *when* has been replaced by the attribute VT and gives information about when the patient was treated. The attribute TT gives information about when the record was inserted and whether it was deleted or not. The values for the attributes VT, TT are intervals. The end date of the intervals can be value *now*. In case of *VT* it indicates that the end date of treatment has not been decided yet and means current date. In case of *TT* it indicates that the record was not deleted.

One can read the following information from table 1:

- The attribute *VT* reads: patient 1 was treated with *A* from 10 X to 15 X, patient 2 with B from 13 X to 16 X and patient 3 with A from 13 X to 20 X.
- The attribute *TT* reads: 3 records were inserted in the database, respectivly: 12 X, 10 X and 5 X and these records are still stored.

In a temporal database the basic operations: *insert*, *modification*, *deletion* are defined in a different way than in a traditional relational one.

The *deletion* operation is not a physical removing of a record but a logical one. It means that the end date of *TT* is changed from *now* to the current date. For example, after deletion of the first record (14 X) the database contains:

Table 2. Temporal database after deletion operation

ID	MEDICINE	VT	TT
1	**A**	**10 X – 15 X**	**12 X – 14 X**
2	B	13 X – 16 X	10 X – now
3	A	13 X – 20 X	5 X – now

The *modification* results in changing the end date of *TT* from *now* to the current date (this indicates that the record was deleted) and inserting the new record with new values of the attributes. For example, after modification of the third record (that took place on 10 X), the database contains:

Table 3. Temporal database after modification operation

ID	MEDICINE	VT	TT	
1	A	10 X – 15 X	12 X – 13 X	
2	B	13 X – 16 X	10 X – now	
3	**A**	**13 X – 20 X**	**5 X – 10 X**	the old version of the third record
3	**C**	**13 X – 15 X**	**10 X – now**	the new version of the third record

The *insertion* operation results in adding a new record. The start date of *TT* is the execution date of the operation. The end date of *TT* is *now*. For example:

Table 4. Temporal database after insertion operation

ID	MEDICINE	VT	TT	
1	A	10 X – 15 X	12 X – 13 X	
2	B	13 X – 16 X	10 X – now	
3	A	13 X – 20 X	5 X – 9 X	
3	C	13 X – 15 X	10 X – now	
2	**A**	**14 X – now**	**14 X – now**	the record inserted 14-th

Such a temporal database prevents from losing information. One can read both present and past states of data with respect to valid time and transactions time. For example:

State of data for the day 14 X of *TT*
in other words: records stored in the database on 14 X:

ID	MEDICINE	VT	**TT**
2	B	13 X – 16 X	**10 X – now**
3	C	13 X – 15 X	**10 X – now**
2	A	14 X – now	**14 X – now**

State of data for the day 13 X of *VT*
in other words: records (deleted and stored) about treatment taken on 13 X:

ID	MEDICINE	TT	**VT**
2	B	10 X – now	**13 X – 16 X**
3	A	5 X – 9 X	**13 X – 20 X**
3	C	10 X – now	**13 X – 15 X**

State of data for the day 14 X of *VT* and 10 X of *TT*
in other words: records stored (not deleted) in the database on 10 X
about treatment taken on 14 X:

ID	MEDICINE	**VT**	**TT**
2	B	**13 X – 16 X**	**10 X – now**
3	C	**13 X – 15 X**	**10 X – now**

Temporal databases can record only valid time (called valid–time databases), only transaction time (called transaction–time databases) or both valid time and transaction time (called bitemporal databases). The paper examines bitemporal databases. It is convenient to represent bitemporal databases graphically:

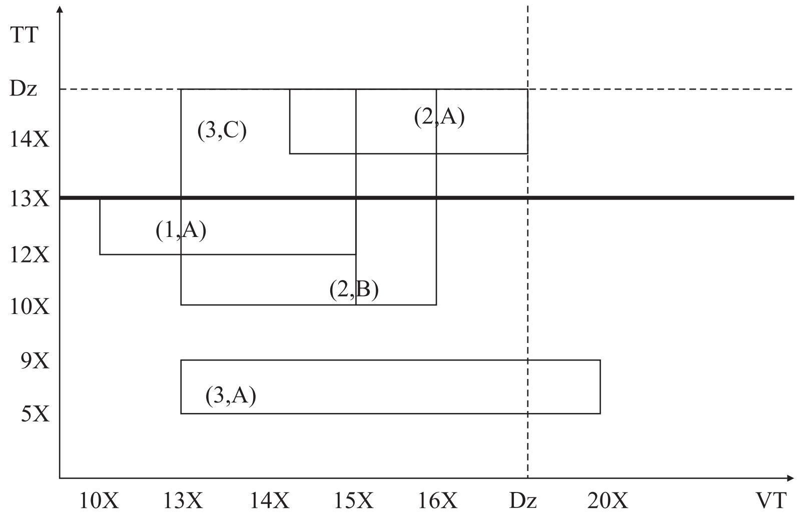

Figure 1. Graphical representation of the bitemporal database from Tab. 4

2. Time

In this paper, we assume that the flow of time $(T, <)$ is a linear, discrete and ordered structure with no end points. T is a set of time points and $<$ is a binary order relation defined on T that satisfies the following conditions:

- transitivity $\forall x, y, z(x < y \land y < z \Rightarrow x < z)$
- irreflexivity $\forall x \neg(x < x)$

3. Bitemporal database

There are various approaches to the definition of bitemporal database ([Tan1]). Bitemporal database can be defined as a structure $DB = (T, <, D, R_1, R_2, \ldots, R_n)$, where:

- $(T, <)$ is a discrete structure of time
- D is a non empty set (database domain),

- $R_1, R_2, \ldots, R_n$ are relations on $D \cup T$ such that

$$R_i \subseteq \underbrace{D \times \ldots \times D}_{k \text{ times}} \times 2^T \times 2^T$$

  where k is a natural number associated with each relation that indicates the number of non-temporal attributes, two last arguments of the relations indicate valid time and transaction time, respectively.

The state of relation R_i with k non-temporal arguments at the moment (t_v, t_t) is ${}_{tv,tt}R_i = \{(d_1, \ldots, d_k) :$ exist $\tau_v, \tau_t \subseteq T$ such that $(d_1, \ldots, d_k, \tau_v, \tau_t) \in R_i$ and $t_v \in \tau_v, t_t \in \tau_t\}$

Example 2

${}_{14\text{X},10\text{X}}$TR (where TR is the relation TREATMENT from the database presented in the Table 4):

ID	MEDICINE
2	B
3	C

4. The language of first order temporal logic

The first order temporal query language is an extension of the first order classical logic language of temporal connectives [Gab, ChoTom]. After some modifications, it is suited for formulating queries in bitemporal database. The alphabet, syntax and semantics are defined as following.

4.1. Alphabet

Alphabet has the following categories of basic symbols:

- domain variables: $x_1, x_2, \ldots$;
- domain constants: $c_1, c_2, \ldots$;
- time variables: $t_1, t_2, \ldots$;
- time constants: $e_1, e_2, \ldots$;
- predicate symbols: $P_1, P_2, \ldots, P_k$;
- equality symbol: $=$;
- classical connectives: $\neg, \wedge$;
- existential quantifier: $\exists$;
- temporal connectives: $\mathtt{U}, \mathtt{S}, \underline{\mathtt{U}}, \underline{\mathtt{S}}$;

- punctuation symbols:), (.
- additional predicate symbols: *date*, $\underline{date}$;

4.2. Syntax

A *term* is either a domain term or time term. A *domain (time) term* is either a domain (time) constant or a domain (time) variable. The atomic formulas of the language are of the form:

- $P_i(a_1, a_2, \ldots, a_n)$, where P_i is n-ary predicate symbol, $a_1, a_2, \ldots, a_n$ are domain terms
- $t = u$, where t, u are both either domain terms or time terms
- $date(t), \underline{date}(t)$, where t is time term

Formulas are finite strings of basic symbols defined in the following recursive manner:

(1) any atomic formula is a formula
(2) if φ, ψ are formulas, so also are $\neg\varphi$, $\varphi \wedge \psi$, $\exists x\varphi$, $\varphi\mathtt{U}\psi$, $\varphi\underline{\mathtt{U}}\psi$, $\varphi\mathtt{S}\psi$, $\varphi\underline{\mathtt{S}}\psi$, where x is any domain variable.
(3) there are not any other formulas than finite strings satisfying conditions (1) and (2)

Other operators are introduced as abbreviations:

- $\varphi \vee \psi \quad \equiv \quad \neg(\neg\varphi \wedge \neg\psi)$
- $\varphi \to \psi \quad \equiv \quad \neg\varphi \vee \psi$
- $\varphi \leftrightarrow \psi \quad \equiv \quad (\varphi \to \psi) \wedge (\psi \to \varphi)$
- $\forall x\varphi \quad \equiv \quad \neg\exists x\neg\varphi$
- $\mathtt{F}\varphi \quad \equiv \quad \top\mathtt{U}\varphi$
- $\mathtt{P}\varphi \quad \equiv \quad \top\mathtt{S}\varphi$
- $\mathtt{G}\varphi \quad \equiv \quad \neg\mathtt{F}\neg\varphi$
- $\mathtt{H}\varphi \quad \equiv \quad \neg\mathtt{P}\neg\varphi$
- $\mathtt{X}\varphi \quad \equiv \quad \bot\mathtt{U}\varphi$
- $\mathtt{X}^-\varphi \quad \equiv \quad \bot\mathtt{S}\varphi$,

where $\top$ denotes the propositional constant verum, $\bot$ denotes the propositional constant falsum.

Underline operators, i.e. $\underline{\mathtt{F}}, \underline{\mathtt{P}}, \underline{\mathtt{G}}, \underline{\mathtt{H}}, \underline{\mathtt{X}}, \underline{\mathtt{X}}^-$ are introduced in analogous way. For examle:

- $\underline{\mathtt{G}}\varphi \quad \equiv \quad \neg\underline{\mathtt{F}}\neg\varphi$
- $\underline{\mathtt{X}}\varphi \quad \equiv \quad \bot\underline{\mathtt{U}}\varphi$

4.3. Semantics

Let $DB = (T, <, D, R_1, R_2, \ldots, R_n)$ be a bitemporal database, Θ be an interpretation and θ be a valuation. The interpretation Θ associates every predicate symbol to a relation, every domain constant to an element of D, every time constant to an element of T. The valuation θ associates every domain term to an elelement of D, every time term to an element of T. If c is a constant term, we make $\theta(c) = c$.

We define a formula A to be true in DB at point $(t_v, t_t) \in T \times T$ under valuation θ, writing $DB, \theta, t_v, t_t \models \varphi$, by induction on the structure of formula:

(1) $DB, \theta, t_v, t_t \models P_i(x_1, \ldots, x_k)$
iff $(\theta(x_1), \ldots, \theta(x_k)) \in {}_{tv,tt}\Theta(P_i)$,

(2) $DB, \theta, t_v, t_t \models x_i = x_j$
iff $\theta(x_i) = \theta(x_j)$

(3) $DB, \theta, t_v, t_t \models \neg\varphi$
iff it is not the case that $DB, \theta, t_v, t_t \models \varphi$

(4) $DB, \theta, t_v, t_t \models \varphi \wedge \psi$
iff $DB, \theta, t_v, t_t \models \varphi$ and $DB, \theta, t_v, t_t \models \psi$

(5) $DB, \theta, t_v, t_t \models \exists x_i \varphi$
iff there exists an $a \in D$ such that $DB, \theta^*, t_v, t_t \models \varphi$,
where θ^* is a valuation which agrees with the valuation θ on the values of all variables except, possibly, on the values of x_i

(6) $DB, \theta, t_v, t_t \models \varphi \mathtt{U} \psi$
iff there exists a $t_1 \in T$ with $t_v < t_1$ and
$DB, \theta, t_1, t_t \models \psi$ and for every $t_2 \in T$,
whenever $t_v < t_2 < t_1$ then $DB, \theta, t_2, t_t \models \varphi$

(7) $DB, \theta, t_v, t_t \models \varphi \mathtt{S} \psi$
iff there exists a $t_1 \in T$ with $t_1 < t_v$ and
$DB, \theta, t_1, t_t \models \psi$ and for every $t_2 \in T$
whenever $t_1 < t_2 < t_v$ then $DB, \theta, t_2, t_t \models \varphi$

(8) $DB, \theta, t_v, t_t \models \varphi \underline{\mathtt{U}} \psi$
iff there exists a $t_1 \in T$ with $t_t < t_1$ and
$DB, \theta, t_v, t_1 \models \psi$ and for every $t_2 \in T$
whenever $t_t < t_2 < t_1$ then $DB, \theta, t_v, t_2 \models \varphi$

(9) $DB, \theta, t_v, t_t \models \varphi \underline{\mathtt{S}} \psi$
iff there exists a $t_1 \in T$ with $t_1 < t_t$ and
$DB, \theta, t_v, t_1 \models \psi$ and for every $t_2 \in T$
whenever $t_1 < t_2 < t_t$ then $DB, \theta, t_v, t_2 \models \varphi$

(10) $DB, \theta, t_v, t_t \models \mathit{date}(t)$
iff $\theta(t) = t_v$

(11) $DB, \theta, t_v, t_t \models \underline{\mathit{date}}(t)$
iff $\theta(t) = t_t$

The not underlined temporal connectives refer to valid time while the underlined ones refer to transaction time.

Semantics of the others temporal connectives $\mathtt{F}, \mathtt{P}, \mathtt{G}, \mathtt{H}, \mathtt{X}, \mathtt{X}^-$ and their underlined counterparts are the following:

(1) $DB, \theta, t_v, t_t \models \mathtt{F}\varphi$
iff there exists a $t_1 \in T$ with $t_v < t_1$ and $DB, \theta, t_1, t_t \models \varphi$

(2) $DB, \theta, t_v, t_t \models \underline{\mathtt{F}}\varphi$
iff there exists a $t_1 \in T$ with $t_t < t_1$ and $DB, \theta, t_v, t_1 \models \varphi$

(3) $DB, \theta, t_v, t_t \models \mathtt{P}\varphi$
iff there exists $t_1 \in T$ with $t_1 < t_v$ and $DB, \theta, t_1, t_t \models \varphi$

(4) $DB, \theta, t_v, t_t \models \underline{\mathtt{P}}\varphi$
iff there exists $t_1 \in T$ with $t_1 < t_t$ and $DB, \theta, t_v, t_1 \models \varphi$

(5) $DB, \theta, t_v, t_t \models \mathtt{G}\varphi$
iff for every $t_1 \in T$ whenever $t_v < t_1$ then $DB, \theta, t_1, t_t \models \varphi$

(6) $DB, \theta, t_v, t_t \models \underline{\mathtt{G}}\varphi$
iff for every $t_1 \in T$ whenever $t_t < t_1$ then $DB, \theta, t_v, t_1 \models \varphi$

(7) $DB, \theta, t_v, t_t \models \mathtt{H}\varphi$
iff for every $t_1 \in T$ whenever $t_1 < t_v$ then $DB, \theta, t_1, t_t \models \varphi$

(8) $DB, \theta, t_v, t_t \models \underline{\mathtt{H}}\varphi$
iff for every $t_1 \in T$ whenever $t_1 < t_t$ then $DB, \theta, t_v, t_1 \models \varphi$

(9) $DB, \theta, t_v, t_t \models \mathtt{X}\varphi$
iff $DB, \theta, t_v + 1, t_t \models \varphi$

(10) $DB, \theta, t_v, t_t \models \underline{\mathtt{X}}\varphi$
iff $DB, \theta, t_v, t_t + 1 \models \varphi$

(11) $DB, \theta, t_v, t_t \models \mathtt{X}^-\varphi$
iff $DB, \theta, t_v - 1, t_t \models \varphi$

(12) $DB, \theta, t_v, t_t \models \underline{\mathtt{X}}^-\varphi$
iff $DB, \theta, t_v, t_t - 1 \models \varphi$

Graphical representations of defined connectives presents Fig. 2.

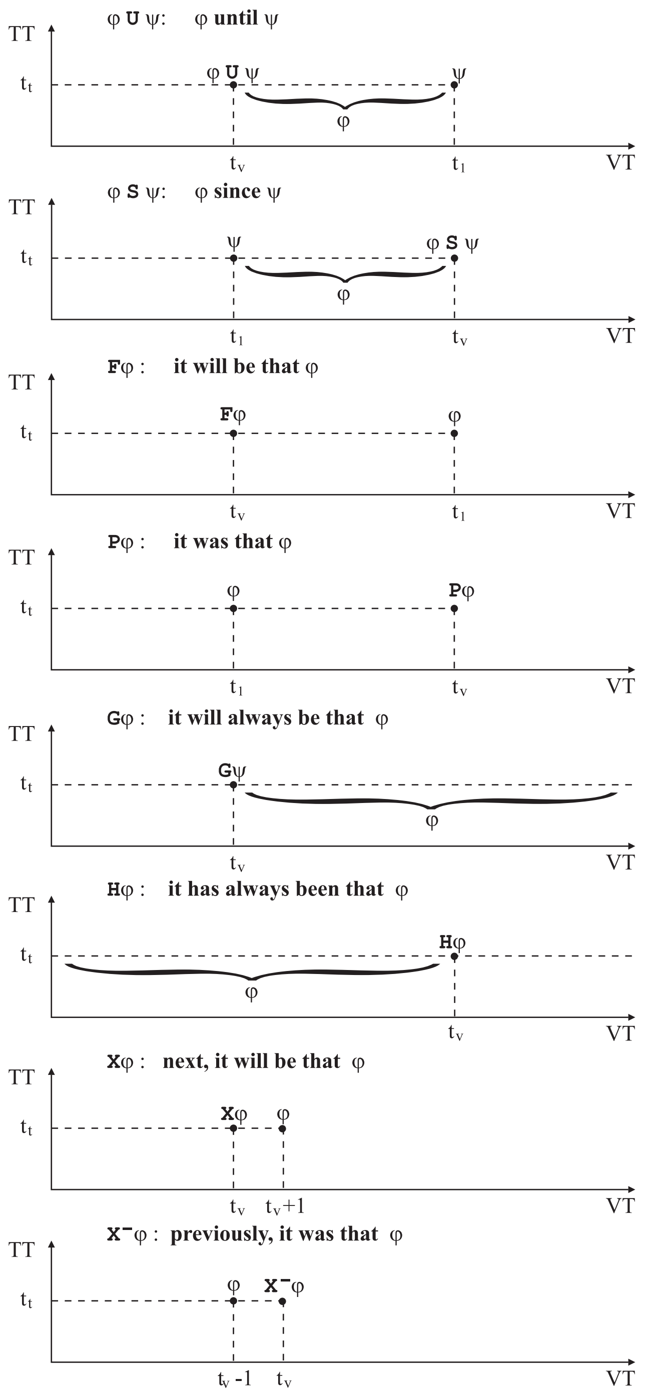

Figure 2. Graphical representation of temporal connectives referred to valid time

Graphical representation of temporal connectives with underlining is analogical (Fig. 3):

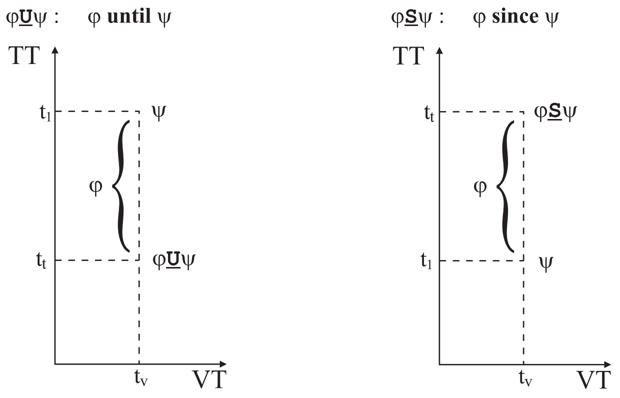

Figure 3. Graphical representation of connectives $\underline{\mathtt{U}}$ and $\underline{\mathtt{S}}$

Example 3

Let us interpret T_1 as the relation TREATMENT (Tab. 4).

- DB,θ,14X,10X $\models T_1(2,B)$,
  because $(2,B) \in {}_{14X,10X}$TREATMENT
- DB,θ,16X,12X $\models T_1(2,B)$ $\mathtt{S}$ $T_1(1,A)$,
  because there exists a t_1<16X with DB,θ,t_1,12X $\models T_1(1,A)$
  and for every $t_2 \in T$ whenever $t_1 < t_2 <$ 16X then
  DB,θ,t_2,12X $\models T_1(2,B)$ (e.g. t_1 can be associated to 13X:
  $(1,A) \in_{13X,12X}$TREATMENT and $(2,B) \in_{t2,10X}$TREATMENT,
  where 13X<t_2<16X)
- DB,θ,14X,11X $\models T_1(1,A)$ $\underline{\mathtt{U}}$ $T_1(2,B)$,
  because there exists a t_1>11X with DB,θ,14X,$t_1 \models T_1(2,B)$
  and for every $t_2 \in T$ whenever 11X<t_2<t_1 then
  DB,θ,14X,t_2 $\models T_1(1,A)$ (e.g. t1 can be associated to 13 X:
  $(2,B) \in_{14X,13X}$TREATMENT and $(1,A) \in_{14X,t2}$TREATMENT,
  where 11X<t_2<13X)

5. Querying a bitemporal database

The language defined in the previous section enables to query a bitemporal database in a convenient way. One can query a bitemporal database with respect to valid time (to retrieve data about what happened in the reality modelled by a database during given time), transaction time (to retrieve data stored in a database during given time) and both valid and transaction time (to retrieve data about what happened in the reality modelled by a database during given time according to records stored in a database during given time). In the first case, one uses temporal operators without underlining, in the second one temporal operators with underlining, and in the third case both operators.

A formula φ is a bitemporal database DB query if it contains at least one free variable and all the variables are domain ones. The answer of the query is defined as follows:

$$\varphi(DB) = \{(\theta(x_1), \ldots, \theta(x_n)) : DB, \theta, t_v, t_t \models \varphi\},$$

where $x_1, \ldots, x_n$ are the only free variables of the formula xx.

5.1. The examples of queries

The queries refer to the bitemporal database containing two relations: PATIENTS(id, name, VT, TT) and TREATMENT(id, medicine, VT, TT). The patients of a hospital and their course of treatment are recorded in this database. Two first attributes in both relations are non-temporal ones, the third attribute is for recording valid time, and the last one is for recording transaction time. The predicate symbol P_1 is interpreted as the relation PATIENTS and the predicate symbol L_1 is interpreted as the relation TREATMENT. We assume that the following constraint is defined in the database: the period of treatment of a patient is contained in (or is equal to) the period of their staying in the hospital.

Query 1

What medicines have been given every day till today (including today) to Kowalski (id. 1) during his last stay in hospital. The treatment could start on any day during his stay in hospital. The query refers to the current state of the database.

$$
\begin{aligned}
\varphi = (((&\mathrm{L}_1(1,x)\ \mathrm{S}\ \mathrm{P}_1(1,\mathrm{Kowalski}))\\
&\ \wedge \mathrm{X}^-\ \mathrm{P}_1(1,\mathrm{Kowalski}) \wedge \mathrm{X}^-\mathrm{X}^-\mathrm{P}_1(1,\mathrm{Kowalski})\\
&)\\
&\vee\\
&(\mathrm{X}^-\mathrm{P}_1(1,\mathrm{Kowalski}) \wedge \mathrm{X}^-\mathrm{X}^-\neg\mathrm{P}_1(1,\mathrm{Kowalski}) \wedge \mathrm{X}^-\mathrm{L}_1(1,x)\\
&)\\
) \wedge\ &\mathrm{date}(\mathrm{now}+1) \wedge \underline{\mathrm{date}}(\mathrm{now})
\end{aligned}
$$

The predicate symbols `date` i <u>`date`</u> determine with respect to what kind of time one queries a database. According to the definition, one searches those valuations of a variable x in order to meet requirements::

```
DB,Θ,now+1,now ⊨
     ((L₁(1,x) S P₁(1,Kowalski))
      ∧ X⁻ P₁(1,Kowalski) ∧ X⁻X⁻P₁(1,Kowalski)
     )
     ∨
     (X⁻P₁(1,Kowalski) ∧ X⁻X⁻¬P₁(1,Kowalski) ∧ X⁻L₁(1,x))
```

The valid time is `now+1` to include the present day to the period of treatment. In case the valid time is `now`, the treatment finished „yesterday" would satisfy formula φ (it is a consequence of the assumption that time is irreflexive). The following part of the query:

```
X⁻P₁(1,Kowalski) ∧ X⁻X⁻P₁(1,Kowalski)
```

is a consequence of time being irreflexive. If this part was deleted, every treatment of patient admitted to hospital today would satisfy formula φ. The last part of the query is to ensure that among answers of the query is the following case: a patient is admitted to hospital today and his/her treatment starts that very day.

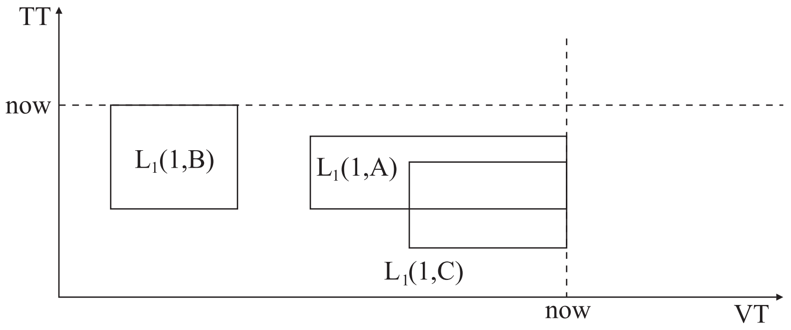

Figure 4. Contents of database at the moment of execution of query 1

The result of the query is medicine A (Fig. 4). The treatment with medicine B is finished and the record about treatment with medicine C does not belong to the current state of database.

Query 2

What medicines have been given to Kowalski? Search only records that have been inserted to the database since the record about admitting Kowalski to hospital was inserted.

$$\begin{aligned}
\varphi = (((&\mathtt{L_1(1,x)}\ \underline{\mathtt{S}}\ \mathtt{P_1(1,Kowalski)} \wedge \underline{\mathtt{X}}^{-}\mathtt{P_1(1,Kowalski)} \\
&\wedge \underline{\mathtt{X}}^{-}\underline{\mathtt{X}}^{-}\mathtt{P_1(1,Kowalski)} \\
&) \\
&\wedge \\
&(\underline{\mathtt{X}}^{-}\mathtt{P_1(1,Kowalski)} \wedge \underline{\mathtt{X}}^{-}\underline{\mathtt{X}}^{-}\neg\mathtt{P_1(1,Kowalski)} \wedge \underline{\mathtt{X}}^{-}\mathtt{L_1(1,x)} \\
&) \\
&) \wedge \underline{\mathtt{date}}\mathtt{(now+1)}
\end{aligned}$$

We refer to the transaction time, hence, the operators with underlining, $\underline{\mathtt{S}}$ and $\underline{\mathtt{X}}^{-}$, are used. We search for records about treatment but valid time of the treatment is not of our interest, hence, the predicate symbol `date` (without underlining) is not used. The usage of the operator $\underline{\mathtt{X}}^{-}$ is to take into account the case that a patient is admitted to hospital today (analogical to query 1).

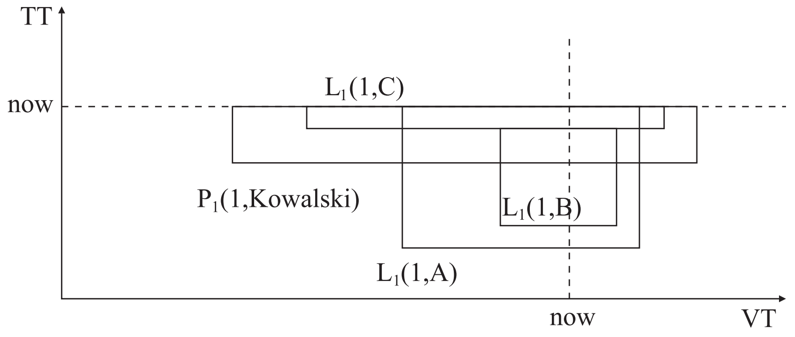

Figure 5. Contents of database at the moment of execution of query 2

The results of the query are medicines A and C (Fig. 5). The record about treatment with B does not belong to the current state of database.

Query 3

What are the identifiers of patients who are no longer treated with A according to the current state of the database.

$$\begin{aligned}
\varphi = &\neg\mathtt{L_1(x,A)} \wedge \mathtt{P\ L_1(x,A)} \wedge \neg\ \mathtt{F\ L_1(x,A)} \\
&\wedge \mathtt{date(now)} \wedge \underline{\mathtt{date}}\mathtt{(now)}
\end{aligned}$$

There might be records about future treatments in the database. The subformula $\neg$`F L₁(x,A)` is to exclude those records.

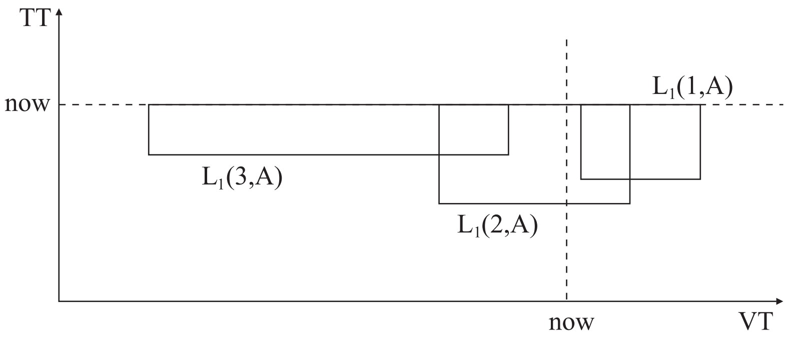

Figure 6. Contents of database at the moment of execution of query 3

The result of the query is identifier 3 (Fig. 6).

Query 4

What are the medicines that were given or are given or will be given to Kowalski according to the current state of database.

```
φ = ∃x (P(L₁(x,y) ∧ P₁(x,Kowalski))
        ∨
        L₁(x,y) ∧ P₁(x,Kowalski) ∨
        F (L₁(x,y) ∧ P₁(x,Kowalski))
       )
       ∧ date(now) ∧ date(now)
```

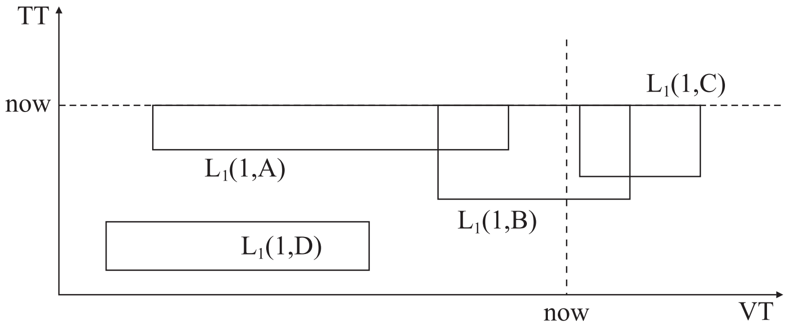

Figure 7. Contents of database at the moment of execution of query 4

The results of the query are medicines A, B, C (Fig. 7). The record with medicine D does not belong to the current state of the database, hence does not satisfy the query.

Query 5

Who will be treated with A during his/her whole stay in hospital in the future according to the current state of database?

$$
\begin{aligned}
\varphi = \exists x\ \mathtt{F}(&\mathtt{X}^{-}\neg \mathtt{P}_1(x,y) \wedge \mathtt{L}_1(x,A) \\
&\wedge \\
&(\mathtt{L}_1(x,A)\ \mathtt{U}\ (\neg \mathtt{P}_1(x,y) \wedge \mathtt{X}^{-}\mathtt{L}_1(x,A)) \\
&) \\
)& \\
\wedge\ &\mathtt{date}(now) \wedge \underline{\mathtt{date}}(now)
\end{aligned}
$$

The subformula $\mathtt{X}^{-}\neg \mathtt{P}_1(x,y) \wedge \mathtt{L}_1(x,A)$ defines the day when a patient is admitted to hospital. The subformula $\neg \mathtt{P}_1(x,y) \wedge \mathtt{X}^{-}\mathtt{L}_1(x,A)$ defines the day when a patient is discharged from hospital. Patients that satisfy the query are treated with A these both days but the connective U does not ensure it (under assumption that time is irreflexive). Hence, additional subformulas $\mathtt{X}^{-}\neg \mathtt{P}_1(x,y) \wedge \mathtt{L}_1(x,A)$ and $\neg \mathtt{P}_1(x,y) \wedge \mathtt{X}^{-}\mathtt{L}_1(x,A)$ are put in the query.

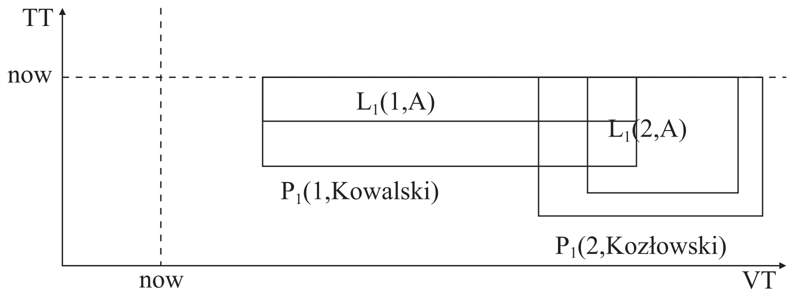

Figure 8. Contents of database at the moment of execution of query 5

The result of the query is the name *Kowalski* (Fig. 8).

Query 6

What are the identifiers of patients whose records about treatment with A have been deleted (logically) from database. We assume that the time of treatment is irrelevant.

$$\varphi = \underline{\mathtt{P}}(\mathtt{L}_1(x,A) \wedge \mathtt{X}\neg \mathtt{L}_1(x,A)) \wedge \underline{\mathtt{date}}(now)$$

The results of the query are identifier 1 (first rectangle from the left) and 2 (Fig. 9).

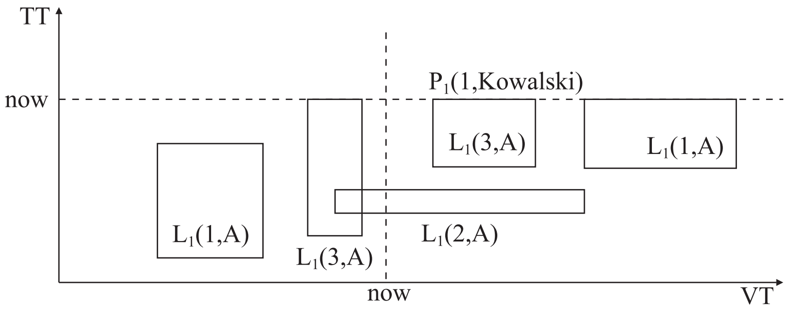

Figure 9. Contents of database at the moment of execution of query 6

Query 7

What records from table PATIENTS were inserted on the day of d_1 (< *now*) and eventually deleted (logically). The time of staying in hospital is irrelevant.

```
φ = P₁(x,y) ∧ X̲⁻¬P₁(x,y)
    ∧
    F̲(¬(P₁(x,y)
       ∧
       (F̲ date̲(now) ∧ date̲(now)
       )
     )
    ∧ date̲(d₁)
```

The subformulas $\mathtt{P_1(x,y)} \wedge \underline{\mathtt{X}}^{-}\neg\mathtt{P_1(x,y)}$ and $\underline{\mathtt{date}}\mathtt{(d_1)}$ define the day (d_1) of insertion of a record about staying of a patient in hospital. The subformula $\underline{\mathtt{F}}\mathtt{(\neg P_1(x,y)} \wedge \mathtt{(}\underline{\mathtt{F}}\ \underline{\mathtt{date}}\mathtt{(now)} \vee \underline{\mathtt{date}}\mathtt{(now)))}$ defines the day of deletion of the record. The subformula $\underline{\mathtt{F}}\ \underline{\mathtt{date}}\mathtt{(now)} \vee \underline{\mathtt{date}}\mathtt{(now)}$ ensures that the day is between d_1 and *now* (including *now*). If the query did not contain this subformula every record would satisfy the query. This is because transaction time can not be later than *now* (transaction time is always the real time of the execution of an operation).

The result of the query is record (3, Nowak) (Fig. 10). Record (2, Kozłowski) was inserted before d1 and record (1, Kowalski) has not been deleted till *now*.

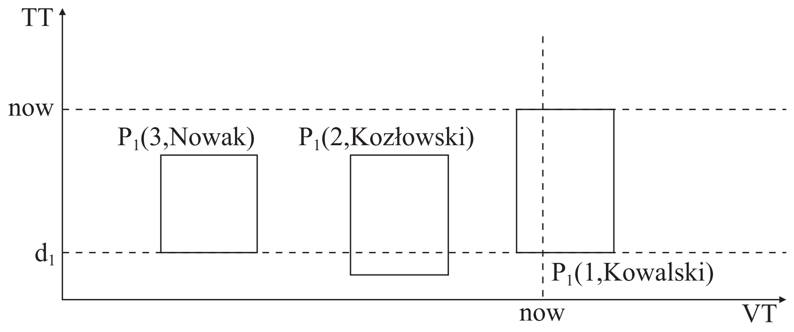

Figure 10. Contents of database at the moment of execution of query 7 and 8

Query 8

What records from table PATIENTS about staying in hospital inserted in the day of d_1 are in the current state of database (have not been deleted). The time of staying in hospital is irrelevant.

$$
\begin{array}{l}
\varphi = (\mathtt{P}_1(\mathtt{x,y}) \\
\quad\quad \underline{\mathtt{S}} \\
\quad\quad (\mathtt{P}_1(\mathtt{x,y}) \wedge \underline{\mathtt{X}}^{-}\neg\mathtt{P}_1(\mathtt{x,y}) \wedge \underline{\mathtt{date}}(\mathtt{d}_1) \\
\quad\quad) \\
\quad) \wedge \mathtt{P}_1(\mathtt{x,y}) \wedge \underline{\mathtt{date}}(\mathtt{now})
\end{array}
$$

The subformula $\mathtt{P}_1(\mathtt{x,y}) \wedge \underline{\mathtt{X}}^{-}\neg\mathtt{P}_1(\mathtt{x,y}) \wedge \underline{\mathtt{date}}(\mathtt{d}_1)$ defines the day (d_1) of insertion of a record about staying in hospital. The subformula $\mathtt{P}_1(\mathtt{x,y})$ in the last part of the query ensures the record has not been deleted. If the query did not contain the subformula, records deleted the present day (henceforth not in the current state of database) would satisfy the query. The use of connective $\underline{\mathtt{S}}$ is for the case that a record was deleted and inserted again between $\mathtt{d}_1$ and `now`. Such a record should not satisfy the query.

The result of the query is record (1, Kowalski) (Fig. 10). Record (3, Nowak) was deleted. Record (2, Kozłowski) was inserted before d_1 and deleted as well.

Summary

The paper presented a formal definition of a bitemporal database, i.e., a database recording valid time and transaction time of data. It concentrated on the definition of a query language for this kind of database. The language is a slight modification of the language of the first order temporal logic. It enables to form queries in a convenient and intuitive way. One does not need to use variables to refer to time, hence the transcript of a query is shorter and less complicated. The paper presented a few examples of queries referring to the database containing medical data.

References

[ChoTom] Chomicki J., Toman D.: *Temporal Logic in Information Systems.* In Logics for Databases and Information Systems, Kluwer Academic Publishers, 1998, s. 31–70.

[Cod] Codd E. F.: *A Relational Model of Data for Large Shared Data banks.* In Communications of the ACM, 1970, 13 (6), s. 377–387.

[Dat] Date C. J.: *Introduction to Database system.* Addison-Wesley, 2003.

[Etz] Etzion, Jajodia, Sripada: *Temporal Databases: Research and practice.* Springer, 1998.

[Gab] Gabbay D. M., Reynolds M. A., Finger M.: *Temporal Logic, Mathematical Foundations and Computational Aspects.* Vol2, Clarendon Press, Oxford, 2000.

[Jen] Jensen, C. S.: *Temporal database Management.* PhD Thesis, Department of Computer Science, Aalborg University, 2000, [http://www.cs.aau.dk/∼csj/Thesis/].

[JenSno] Jensen C. S., Snodgrass R. T.: *Semantics of Time-Varying Information.* In Information Systems, 21(4), 1996, s. 311–352.

[Sno] Snodgrass R., editor.: *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[Ste] Steiner A.: *A Generalisation Approach to Temporal Data Models and Their Implementations.* PhD Thesis, Departement Informatik, ETH Zurich, Switzerland, 1997.

[Tan1] Tansel A. U. i in.: *Temporal Databases: Theory, Design, and Implementation.* Benjamin-Cummings Publishing Co., Inc., 1993.

[Tan2] Tansel A., Tin E.: *Expressive Power of Temporal Relational Query Languages and Temporal Completeness.* In Temporal Databases: Research and Practice, LNCS 1399, Berlin Heidelberg, Springer-Verlag, 1998, s. 129–149.

Mariusz Giero
Chair of Logic, Informatics and Philisophy of Science
University of Białystok
ul. Sosonowa 64, 15–887 Białystok, Poland
giero@uwb.edu.pl