

# Temporal XML

M.G.Manukyan<sup>1</sup>, L.A.Kalinichenko<sup>2</sup>

<sup>1</sup> Department of Computer Science, Yerevan State University,  
email: mgm@ysu.am

<sup>2</sup> Russian Academy of Sciences, Institute for Problems of Informatics,  
email: leonidk@synth.ipi.ac.ru

**Abstract.** An approach for temporal extension of semi-structured data models has been investigated in frame of the INTAS n°11109 and RFBR n°00-07-90086 projects. Temporal extension of XML (*TXML*) appeared as an outcome of this research. *TXML* intention is to define such platform independent temporal data representation on WWW that various program packages could interchange this data without loss of semantic content. It is essential that a possibility for representing temporal knowledge on WWW be provided. We analyze introducing a temporal constituents for XML. XML documents as well as DTD are represented evolving in time. To each XML construct such as element, attribute, entity, etc. temporal constituent of existence and temporal constituent of belief are attached. *TXML* is treated as an XML application. Problems of support of point, interval, relative and periodical events are analyzed. The semantics of temporal data is rather complicated and generally is defined by some formulae. As formalisms we use arithmetic of Presburger, temporal logic of Allen and special temporal facilities to support such events. The encoding of temporal data and knowledge is based on  $\lambda$ -calculus.

## 1 Introduction

Problems of temporal information representation and manipulation is a research issue for several directions in computer science, such as artificial intelligence, concurrent program verification and databases. However, the paradigms used for temporal reasoning in these three areas vary widely (depending on models used). We will limit ourselves with the issues relevant to logical modeling of time in databases that remain to be important and timely. At the same time the approach used can be easily adapted for temporal modeling in other areas.

The temporal data model should support three orthogonal kinds of time[14]: *Valid time*, *Transaction time* (sometimes called 'belief time') and *User-defined time*. *Valid time* and *Transaction time* have system semantics. The temporal data model should provide representation of the following event types[2,15]: *Point event*, *Interval event*, *Relative event*, *Periodical event*.

Nowadays, several models of temporal database description with respective query languages are known [11,15,16,17,19,22]. The detailed analysis of language facilities for logical modeling of time in databases can be found in [20].

In this paper we analyze an approach for temporal facilities provisions for the semistructured data models. Possibly such extensions have never been touched before. For application we emphasize Web databases where data semantics often assume temporal orientation for information systems. This analysis is done in frame of the INTAS n°11109 and RFBR n°00-07-90086 projects.

To be specific, for semistructured data model we consider Extensible Markup Language (XML) [5], a new important and widespread standard for representing hierarchical tagged data. Popularity using XML quickly gains as a textual language for representation and exchange data over the Web. Intensive development of XML-oriented databases is performed considering XML documents to be a database and DTD (Document Type Definition) to be a database schema. The respective DBMSs come from research (e.g., [12] migrating the Lore data model and query language to work with XML) or from industry (e.g., [23]). The data according to XML is of a semi-structured nature. It is important that in such databases DTD and DTDless XML document types could be supported. XML query data model and XML query algebra are being developed [9,10].

For temporal extension of XML in *TXML* applications, three layers of representation of a temporal object are assumed. An internal layer provides internal representations used by an application. An abstract layer uses representation as a *TXML* object. A communication layer translates the *TXML* object representation to a stream of bytes. An application dependent program manipulates the temporal objects using its internal representation, it can convert them to *TXML* objects and communicate them by using the byte stream representation of *TXML* objects. *TXML* objects are representations of temporal entities that can be communicated among various applications in a meaningful way, that is, preserving their semantics. Applications that meet the *TXML* requirements may be considered as *TXML* compliant. *TXML* compliancy is intended to maximize the potential for interoperability amongst *TXML* applications. An abstract layer of *TXML* objects representation is considered here.

The paper is structured as follows. Problems of temporal constituents definition and an approach for formal representation of *TXML* objects are discussed in the next section. Encodings of *TXML* objects and content dictionaries are given in section 3 and 4. An example of temporal objects representation in *TXML* follow.

## 2 Temporal XML Objects

XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. By DTD the document structure can be defined. Since it is possible to create tags having specific meaning and helping to define meaningful values of the documents. A characteristic property of a document is its conformance to XML schema (such as DTD). For an XML application defining and preserving temporal object (in particular, temporal constituent) semantics, a serious deficiencies remain: it is

impossible to present information on types and on constraints over element type values.

## 2.1 Temporal Constituents

For temporal modeling, each XML construct, such as element, attribute, entity, etc. should be considered as evolving in time. To achieve temporal assignments of such XML constructs, the temporal constituent of existence and temporal constituent of belief are introduced. Temporal constituents provide representation of point, interval, relative and periodic events. Facilities introduced are suitable for DTD and DTDless XML document types. The temporal constituent of *existence* (denoted further as *valid*) defines *time period* during which the XML construct exists according to user beliefs. The temporal constituent of *belief* defines *time period* during which the system “believes” into existence of XML construct as values. The value of temporal constituent of *existence* is provided by user’s temporal assignment during construct specification or modification. The value of temporal constituent of *belief* is provided by the system. Temporal constituent’s value assignment expressed by XML-attribute inevitably leads to XML syntax change. The semantics of temporal data is rather complicated and generally is defined by some mathematical formulae. In our approach for support of point, interval, relative and periodic events the Allen’s temporal logic[1,2,3,4], Presburger’s arithmetic[21] and specially developed temporal facilities are used. Thus the problem generally consists in representing mathematical formulae on WWW. *TXML* should allow the encoding of mathematical notation as well as formal semantics. Usually, for such purpose special XML applications are being developed. OPENMath [6] is among them. On the first glance it seems that DTD of OPENMath could have been extended by special temporal facilities to represent temporal information on WWW. But such approach would be inefficient, because DTD of OPENMath covers all computational mathematics and the parser every time would have to check the syntactic validity of more general constructs than those that are required for defining just temporal constituents. Thus, special DTD for defining temporal data is needed. It should be noted, that DTD for defining temporal data is the extension of a DTD core for defining mathematical objects [6]. The encoding of temporal data and knowledge (similarly to OPENMath) is based on  $\lambda$ -calculus[13].

## 2.2 TXML as an XML Application

*TXML* is an XML application. Its syntax is defined by syntactical rules of XML, its grammar is partially defined by its own DTD. In other words, details of element, attribute names, element nesting, etc. are specified by DTD. But because of DTD deficiencies pointed out above and of specificity of the given application, only syntactical validity of *TXML* object representation can be provided on the DTD level. For semantic representation, in addition to general rules inherited by XML applications, *TXML* defines new rules. This is reached by means of introduction of Content Dictionaries (CDs) analogously to OPENMath. Note

that a Content Dictionary is an XML construct containing *TXML* objects semantics description. Definition of temporal object can be checked for semantic validity if it contains information on types of its constituent parts. The type system can be used for formalization of signature of mathematical symbols and for check of semantic validity of temporal objects representation. In *TXML* as well as in OPENMath the formal type system is introduced as Extended Calculus of Constructions (ECC) [18]. The *tecc* CD represents such system in *TXML*. It consists of primitive constructors symbols by means of which a typed object can be created. Several additional files are related to CDs signature files including symbols defined in some *TXML* CD and their format. Note that in defining CD for *TXML* there is no need for development of special DTD, as in this case OPENMath facilities are sufficient and natural.

### 2.3 Formal Definition of TXML Objects

In this section we provide a self-contained description of *TXML* objects similarly to OPENMath. *TXML* represents temporal objects as terms or as labelled trees that are called *TXML* objects or *TXML* expressions. Thus, the definition of an abstract *TXML* object is the following.

**Basic TXML Objects** Basic *TXML* objects form leaves of the *TXML* object tree. Basic *TXML* object is one of the following:

(b1) Integer. Represents integers in the mathematical sense, with no pre-defined range.

(b2) Datetime. Is represented as a Unicode Character string. Corresponds to "characters" in XML.

(b3) Interval. Is represented as a Unicode Character string. Corresponds to "characters" in XML.

(b4) Timeperiod. Is represented as a Unicode Character string. Corresponds to "characters" in XML.

(b5) Reference. Is represented as a sequence of characters matching a regular expression.

(b6) Symbol. Encodes two fields of information, a *name* and a *Content Dictionary*.

Each is a sequence of characters matching a regular expression.

(b7) Variable. Consists of a *name* which is a sequence of characters matching a regular expression.

**Compound TXML Objects** A compound *TXML* object is defined in terms of bindings and applications in  $\lambda$ -calculus. *TXML* objects are built recursively as follows.

- (c1) Basic *TXML* objects are *TXML* objects.
- (c2) If  $F, A_1, \dots, A_n (n > 0)$  are *TXML* objects, then  $application(F, A_1, \dots, A_n)$  is a *TXML* object. This corresponds to  $\lambda$ - term  $(FA_1 \dots A_n)$ .
- (c3) If  $S_1, \dots, S_n$  are *TXML* symbols, and  $A, A_1, \dots, A_n (n > 0)$  are *TXML* objects, then  $attribution(A, S_1, A_1, \dots, S_n, A_n)$  is a *TXML* object and  $A$  is the object *stripped of attributions*. The operation of recursively applying stripping to the stripped object is called *flattening of the attribution*. When the stripped object after flattening becomes a variable, the attributed object is called *attributed variable*.
- (c4) If  $B$  and  $C$  are *TXML* objects, and  $v_1, \dots, v_n (n \geq 0)$  are *TXML* variables or attributed variables, then  $binding(B, v_1, \dots, v_n, C)$  is a *TXML* object. For example,  $\lambda$ - term  $\lambda x.x + 1$  is defined as  $binding(\lambda lambda, x, application(plus, x, 1))$ .
- (c5) If  $S$  is a *TXML* symbol and  $A_1, \dots, A_n (n \geq 0)$  are *TXML* objects, then  $error(S, A_1, \dots, A_n)$  is a *TXML* object.

### 3 TXML Encodings

The XML encoding of a *TXML* object is defined by the DTD given in Figure 1 below. An encoded *TXML* object is placed inside *TXMLOBJ* elements. By means of a temporal object *TXMLOBJ* we model temporal constituent of existence and temporal constituent of belief.

```

<!-- DTD for TXML objects- sb 22.06.2000 -->
<!ENTITY txmlel “((TXMLS | TXMLBIND+),
                  ((TXMLV | TXMLATTR)+)?, (TXMLDT |
                  TXMLTP | TXMLTI | TXMLI | TXMLA)?) |
                  TXMLE”
<!-- reference -->
<!ELEMENT TXMLR EMPTY >
<!ATTLIST TXMLR ref CDATA #REQUIRED >
<!-- symbol -->
<!ELEMENT TXMLS EMPTY >
<!ATTLIST TXMLS name CDATA #REQUIRED
           cd CDATA #REQUIRED >
<!-- variable -->
<!ELEMENT TXMLV EMPTY >
<!ATTLIST TXMLV name CDATA #REQUIRED >

```

```

<!-- integer -- >
<!ELEMENT TXMLI (#PCDATA) >

<!-- datetime -- >
<!ELEMENT TXMLDT (#PCDATA) >
<!ATTLIST TXMLDT qualifier CDATA #REQUIRED >

<!-- time interval -- >
<!ELEMENT TXMLTI (#PCDATA) >
<!ATTLIST TXMLTI qualifier CDATA #REQUIRED >

<!-- time period -- >
<!ELEMENT TXMLTP (#PCDATA) >
<!ATTLIST TXMLTP qualifier CDATA #REQUIRED >

<!-- apply constructor -- >
<!ELEMENT TXMLA (%txmel;) >

<!-- binding constructor -- >
<!ELEMENT TXMLBIND (TXMLS, TXMLBVAR, TXMLA) >
<!ELEMENT TXMLBVAR (TXMLV | TXMLATTR)+ >

<!-- error -- >
<!ELEMENT TXMLE (TXMLS, TXMLA*) >

<!-- attribution constructor & attribute pair constructor -- >
<!ELEMENT TXMLATTR (TXMLAP, TXMLV) >
<!ELEMENT TXMLAP (TXMLS, (TXMLS | TXMLR))+ >

<!-- TXML object constructor -- >
<!ELEMENT TXMLOBJ (TXMLA) >

```

Figure 1 : DTD for the *TXML* xml encoding of objects.

## 4 Content Dictionaries

As we mentioned above, to check semantics, in addition to general rules inherited by an XML applications, *TXML* defines new rules. This is done by Content Dictionaries (CDs). CDs are central units in *TXML* for preserving temporal information. They are used to provide formal and non-formal semantics description for all symbols used in *TXML*. A Content Dictionary holds the meanings of (various) temporal words. These words are *TXML* basic objects referred to as symbols. CDs (similarly to ontologies) define symbols used to represent concepts arising in temporal models. The Content Dictionaries are public, they represent actual common knowledge among *TXML* applications. Content Dictionaries fix the meaning of objects independently of the application. Symbols are uniquely defined by the Content Dictionary in which they occur and by a name. Note that all symbols appearing in a *TXML* object are defined in a Content Dictionary.

*TXML* application object is viewed as a tree by applications that do not understand Content Dictionaries, whereas the semantics of the symbols, as defined in the Content Dictionaries, should interpret the object as functional application, constructor, or binding accordingly. A compliant application must declare the names and version numbers of the Content Dictionaries that it supports. The following CDs are considered in *TXML*:

- *tecc* and *tfns1* are defined analogously to *ecc* and *fns1* of *OPENMath*.
- *tarith1*, *tarith2*, *tarith3* define operations and constructors related to temporal types *DATETIME*, *INTERVAL* and *TIMEPERIOD*.
- *tlogic1*, *tlogic2* define operations and constructors related to the Allen's temporal logic.
- *tsetname* - are common sets of *TXML*.
- *time* - are basic time concepts.
- *tset1* - is the set of functions over set *SOFTP*.
- *tset2* - is set of functions over set *TP*.
- *dtd* - are set nodes of *DTD*.
- *db* - are set nodes of *DB*.

In *TXML* the structure *CDDefinitions* coincides with the structure of *CDDefinitions* in *OPENMath*.

**Example.** The *tlogic1* Content Dictionary File.

```

< CD >
< CDName > tlogic1 < /CDName >
.....
< Description >
This CD defines symbols for common temporal predicates.
< /Description >
< CDDefinition >
< Name > equals < /Name >
< Description >
< CMP > equals(i, j) ≡ equals(j, i) ≡
∃(k, l)meets(k, i)&meets(i, l)&meets(k, j)&meets(j, l)
< /CMP >
< /Description >
< FMP >
< OMOBJ >
< OMBIND >
< OMS cd = "quant1" name = "forall" / >
< OMBVAR >
< OMV name = "i" / >
< OMV name = "j" / >
< /OMBVAR >
< OMA >
< OMS cd = "logic2" name = "equivalent" / >

```

```

< OMA >
  < OMS cd = "logic2" name = "equivalent" / >
  < OMA >
    < OMS cd = "tlogic1" name = "equals" / >
    < OMV name = "i" / >
    < OMV name = "j" / >
  < /OMA >
  < OMA >
    < OMS cd = "tlogic1" name = "equals" / >
    < OMV name = "j" / >
    < OMV name = "i" / >
  < /OMA >
  < OMBIND >
    < OMS cd = "quant1" name = "exists" / >
    < OMBVAR >
      < OMV name = "k" / >
      < OMV name = "l" / >
    < /OMBVAR >
  < OMA >
    < OMS cd = "logic1" name = "and" / >
  < OMA >
    < OMS cd = "tlogic1" name = "meets" / >
    < OMV name = "k" / >
    < OMV name = "i" / >
  < /OMA >
  < OMA >
    < OMS cd = "tlogic1" name = "meets" / >
    < OMV name = "i" / >
    < OMV name = "l" / >
  < /OMA >
  < OMA >
    < OMS cd = "tlogic1" name = "meets" / >
    < OMV name = "k" / >
    < OMV name = "j" / >
  < /OMA >
  < OMA >
    < OMS cd = "tlogic1" name = "meets" / >
    < OMV name = "j" / >
    < OMV name = "l" / >
  < /OMA >
< /OMA >
< /OMBIND >
< /OMA >
< /OMBIND >
< /OMOBJ >

```

```

< /FMP >
< /CDDefinition >
.....
< /CD >

```

#### 4.1 Signature Files

*TXML* may be used with any type system. One just needs to produce a respective Content Dictionary. After that one may build other objects representing types in the given type system. These are typically associated with *TXML* objects via the OPENMath *attribution* constructor. A Small Type System, called STS, has been designed to define semi-formal signatures for OPENMath symbols documented in [8]. The signature files are based on this formalism. Using the same mechanism, it is shown [7] how pure type systems can also be applied to assign types to OPENMath symbols.

Thus, *TXML* Signature files contain the signatures of symbols defined in some *TXML* Content Dictionary. Signature files have a header which specifies the Content Dictionary and determines the type system used, and the Content Dictionary which contains the symbols for which the signatures are being given. Each signature takes the form of an XML encoded *TXML* object.

**Example.** *tlogic1* STS signature file.

```

< CDSignatures type = "sts" cd = "tlogic1" >
< CDComment >
Date: 2000-08-12
Author: Manuk Manukyan
< /CDComment >
< Signature name = "meets" >
< OMOBJ >
< OMA >
< OMS name = "mapsto" cd = "sts" / >
< OMS name = "TP" cd = "tsetname" / >
< OMS name = "TP" cd = "tsetname" / >
< OMV name = "boolean" / >
< /OMA >
< /OMOBJ >
< /Signature >
.....
< /CDSignatures >

```

#### 4.2 Example of TXML Object

**Temporal attribute.** The following fact is encoded in this example: an employee whose IDREF is equal to "e200" got married after graduating school.

```

< TXMLOBJ >

```

```

< TXMLA >
  < TXMLS cd = "tlogic2" name = "after" / >
  < TXMLATTR >
    < TXMLATP >
      < TXMLS cd = "tecc" name = "type" / >
      < TXMLS cd = "tecc" name = "relativetime" / >
      < TXMLS cd = "db" name = "time" / >
      < TXMLS cd = "db" name = "valid" / >
      < TXMLS cd = "db" name = "element" / >
      < TXMLR ref = "employee" / >
      < TXMLS cd = "db" name = "occurrence" / >
      < TXMLR ref = "e200" / >
      < TXMLS cd = "db" name = "attribute" / >
      < TXMLR ref = "married" / >
    < /TXMLATP >
    < TXMLV name = "x" / >
  < /TXMLATTR >
< TXMLATTR >
  < TXMLATP >
    < TXMLS cd = "tecc" name = "type" / >
    < TXMLS cd = "tecc" name = "relativetime" / >
    < TXMLS cd = "db" name = "time" / >
    < TXMLS cd = "db" name = "valid" / >
    < TXMLS cd = "db" name = "element" / >
    < TXMLR ref = "employee" / >
    < TXMLS cd = "db" name = "occurrence" / >
    < TXMLR ref = "e200" / >
    < TXMLS cd = "db" name = "attribute" / >
    < TXMLR ref = "grad_school" / >
  < /TXMLATP >
  < TXMLV name = "y" / >
< /TXMLATTR >
< /TXMLA >
< /TXMLOBJ >

```

## 5 Conclusion

The paper proposes an approach for temporal extension of semi-structured data models. Temporal extension of XML is oriented on DTD and DTDless XML document types. *TXML* intention is to define a platform independent representation of temporal data on WWW in such way that program packages could interchange this data without loss of semantic content. It is essential that a possibility for representing temporal knowledge on WWW would be provided. XML document as well as DTD are considered evolving in time. The extension consists in providing temporal assignments for different XML constructs in DTD

and in documents by means of assignment of temporal constituents of existence and belief to them. The semantics of temporal data is rather complicated and in general is defined by some formulae. In our approach for support of point, interval, relative and periodic events the Allen's temporal logic, Presburger's arithmetic and specially developed temporal facilities are used. Thus *TXML* should allow the encoding of mathematical notation as well as mathematical semantics. For that purpose special XML applications are being developed. OPENMath is among them. For efficiency reasons instead of extending OpenMath by temporal facilities it is required to develop specific DTD for temporal data as an extension of DTD core for defining mathematical objects. *TXML* is an XML application. Its syntax is defined by means of syntactic rules of XML, its grammar is partially defined by own DTD. Thus only syntactical validity of *TXML* object representation can be provided on DTD level. For semantic check Content Dictionaries (CDs) are introduced analogously to OPENMath. Note that Content Dictionary is a construct represented in accordance with the rules of XML and containing *TXML* objects semantic description. The mathematical object representation can be checked for semantic validity if it includes information on types of its constituent parts. The type system can be used for formalization of the signature of mathematical symbols and for check of semantic validity of mathematical objects representation. In *TXML* as well as in OPENMath the formal type system is introduced by Extended Calculus of Constructions. The *tecc* CD represents this system in *TXML*. It consists of symbols of primitive constructors by means of which a typed object can be created. Several additional files are related to CDs signature files to contain symbols defined in some *TXML* CD and their format. It is obvious that in defining CDs for *TXML*, there is no need for development of special DTD, as in this case OPENMath facilities are sufficient and natural. Content markup in *TXML* is based on the concept of a tree. The encoding of temporal data and knowledge is based on  $\lambda$ -calculus. The extension proposed is a heterogeneous temporal model (to each XML construct the temporal constituents of existence and belief are attached).

Without loss of generality we can assume that periodical events are interval events truth set of which can be a finite or infinite time periods. In this case we base definition of *time period* on Presburger's arithmetic. By means of arithmetical formulae of Presburger one can constructively describe truth sets of interval as well as periodical events. *Time period* is being built from discrete time moments by means of *time period* constructor. In our approach arithmetical formulae of Presburger are considered as *time period* constructors. Definitions of *time period* in this case provides for application of temporal logic of Allen to interval as well as to periodical events. This formalism will provide also for the possibility of representation of relative events. Relative events assume representation of the event sequence in databases. It is essential that this formalism preserves the transitivity property of temporal relations.

## References

1. J. F. Allen, P. J. Hayes. A Common Sense Theory of Time. In Proc. of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, pp. 528-531, 1985.
2. J.F.Allen. Towards a General Model and Action and Time. Artificial Intelligence, vol. 23, no. 2, pp. 123-154., 1984.
3. J. F. Allen. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, vol. 26, no. 11, pp. 832-843, 1983.
4. J. F. Allen An Interval Based Representation of Temporal Knowledge. In Proc. of the International Joint Conference on Artificial Intelligence, pp. 221-226, Vancouver, B. C., 1981.
5. T. Bray, J. Paoli, and C. Sperberg-McQueen (Editors). Extensible markup Language ( XML ) 1.0. February 1998. W3C Recommendation available at <http://www.w3.org/TR/1998/REC-xml-19980210>.
6. O. Caprotti, D. P. Carlise, and A. M. Cohen (Editors). The OPENMath Standard. August 1999. Available at <http://www.nag.co.uk/projects/OPENMath/omstd>.
7. O. Caprotti, A. M. Cohen (Editors). A Type System for OPENMath. OPENMath Deliverable 1.3.2b, OPENMath Esprit Consortium, February 1999. Available at <http://www.nag.co.uk/projects/OPENMath.html>.
8. J. Davenport (Editor). A Small OPENMath Type System. OPENMath Deliverable 1.3.2c, April 1999. Available at <http://www.nag.co.uk/projects/OPENMath/omstd>.
9. P. Fankhauser, et al. The XML Query Algebra. W3C Working Draft 15 February 2001. Available at <http://www.w3.org/TR/2001/WD-query-algebra-20010215>.
10. M. Fernandez, J. Robie. XML Query Data Model. W3C Working Draft 15 February 2001. Available at <http://www.w3.org/TR/2001/WD-query-datamodel-20010215>.
11. S. K. Gadia. A Homogeneous Model and Query Language for Temporal Databases. ACM TODS, vol. 13, no. 4, pp. 418 - 448, 1988.
12. R. Goldman, J. McHugh, J. Widom, From Semistructured Data to XML : Migrating the Lore Data Model and Query Language. 1998, <http://www-db.stanford.edu/lore>.
13. J. R. Hindley, J. P. Seldin. Introduction to Combinators and  $\lambda$ - Calculus. Cambridge University Press, 1986.
14. C. S. Jensen et al. A Consensus Glossary of Temporal Database Concepts. ACM SIGMOD Record, Vol. 23, No. 1, pp. 52-64, 1994.
15. F. Kabanza, J-M. Stevenne, P. Wolper. Handling Infinite Temporal Data. Journal of Computer and System Sciences, no. 51, pp. 3 - 17, 1995.
16. L. A. Kalinichenko. *SYNTHESIS* : the language for description, design and programming of the heterogeneous interoperable information resource environment. Institute for Problems of Informatics, Russian Academy of Science, Moscow, 1993.
17. M. Koubarakis, J. Mylopoulos, M. Stanley. A. Borgida, Telos: Features and Formalization. Technical Report KRR - TR - 89 - 4, University of Toronto, Department of Computer Science, pp. 1 - 84, 1989.
18. Z. Luo. An Extended Calculus of Constructions. Ph.D.'s thesis. University of Edinburgh, Holland. 1990.
19. M. G. Manukyan, Temporal Data Model. In Proc. of First East-European Symposium on Advances in Databases and Information Systems - ADBIS'97, pp. 371-378, St.-Petersburg, 1997.

20. G. Özsoyoğlu, R. T. Snodgrass. Temporal and Real - Time Databases: A Survey. IEEE Transactions on Knowledge and Data Engineering, vol. 7, no. 4, pp. 513 - 532, 1995.
21. M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchen die Addition als einzige Operation hervortritt. In "Comptes Rendus, I. Congres des Mathematiques des Pays Slaves, Warsaw, 1929", pp. 192-201, 395.
22. R. T. Snodgrass et al. TSQL2 Language Specification. ACM SIGMOD Record, vol. 23, no. 1, pp. 65 - 86, 1994.
23. TAMINO: the Information Server for Electronic Business. <http://www.softwareag/tamino/default.htm>.