Effective Data Integration in the Presence of Temporal Semantic Conflicts

Hongwei Zhu, Stuart E. Madnick, Michael D. Siegel
MIT Sloan School of Management
Cambridge, MA 02142, USA
{mrzhu, smadnick, msiegel}@mit.edu

Abstract

The change in meaning of data over time poses significant challenges for the use of that data. These challenges exist in the use of an individual data source and are further compounded with the integration of multiple sources. In this paper, we identify three types of temporal semantic heterogeneities. We propose a solution based on extensions to the Context Interchange framework, which has mechanisms for capturing semantics using ontology and temporal context. It also provides a mediation service that automatically resolves semantic conflicts. We show the feasibility of this approach with a prototype that implements a subset of the proposed extensions.

1. Introduction

Effective management of temporal data has become increasingly important in application domains from day-to-day record keeping to counterterrorism efforts. It is often even required by law for organizations to store historical data and make sure it is accurate and easy to retrieve[1]. While temporal databases can be used to manage the data, ensuring that the retrieved data is meaningful to the users is still an unsolved problem when data semantics changes over time.

As an example, suppose an arbitrage analyst in New York needs to compare Daimler-Chrysler's stock prices at New York and Frankfurt exchanges. He retrieved the data from Yahoo's historical database, see Figure 1. Two anomalies caught his eyes at a quick glance at the data. First, prices at two exchanges differ substantially; and second, the price at Frankfurt dropped by almost 50% at the turn from 1998 to 1999!

Date	Open	High	Low	Close	Volume	Adj Close*
6-Jan-99	105.25	105.74	103.92	105.13	2,061,200	90.49
5-Jan-99	99.05	103.43	98.93	103.31	2,634,600	88.92
4-Jan-99	99.66	100.69	98.08	98.99	3,441,400	85.20
31-Dec-98	94.49	94.55	93.21	93.51	506,900	80.49
30-Dec-98	94.97	95.34	94.18	94.18	391,300	81.06
29-Dec-98	96.13	96.25	95.64	95.95	1,195,700	82.58
28-Dec-98	95.64	96.43	95.16	95.64	1,707,800	82.32

6-Jan-99	90.10	92.40	89.30	92.30	13,950,500	86.67
5-Jan-99	86.80	88.60	86.10	86.80	12,329,300	81.51
4-Jan-99	83.50	88.50	82.50	87.50	13,660,200	82.16
30-Dec-98	166.30	167.90	164.50	164.50	4,934,820	154.47
29-Dec-98	166.00	166.50	164.50	165.00	5,039,660	154.94
28-Dec-98	159.50	167.80	159.30	166.50	9,748,480	156.34

* Close price adjusted for dividends and splits.

Figure 1. Historical stock prices for Daimler-Chrysler. *Top*: New York; *Bottom*: Frankfurt

These anomalies result from unresolved semantic conflicts between the data sources and the data receiver. In this case, not only are the currencies for stock prices different at the two exchanges, but the currency at Frankfurt also changed from German Marks to Euros at the beginning of 1999. Once the data is normalized using this knowledge, it can be seen there is neither significant arbitraging opportunity nor abrupt price plunge for this stock. We call metadata knowledge such as the currency for price *context knowledge*, and the history of time varying metadata *temporal context*.

To allow data receivers like the analyst to effectively use data from temporally heterogeneous sources, we need to represent temporal context knowledge and incorporate it into data integration and query answering systems. Temporal database research has primarily focused on management of temporal data in a homogeneous constant context environment. Semantic data integration techniques developed so far are based on snapshot data models that ignore the time dimension.

The objective of this research is to fill this gap by developing techniques to effectively resolve temporal semantic conflicts between data sources and receivers. Specifically, we extend the Context Interchange

[1] See Robert Sheier on "Regulated storage" in ComputerWorld, 37(46), November 17, 2003. Health Insurance Portability Act requires healthcare providers keep records till two years after death of patients; Sarbanes-Oxley Act requires auditing firms retain records of financial statements.

Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME'04)
1530-1311/04 $20.00 © 2004 **IEEE**

IEEE COMPUTER SOCIETY

(COIN) framework [5, 9] with temporal contextual knowledge representation and reasoning capability.

2. Challenges of temporal data integration

2.1. A simple integration example

A temporal database is one that supports some aspect of time, not counting user-defined time such as birthday and hiring date [12]. This rather informal definition is due to the fact that the temporal dimensions are often application specific, therefore it is either difficult or unnecessary to support all aspects of time. Nevertheless, most *temporal data* can be viewed as time-stamped propositions and represented as relational tuples with timestamps. Table 1 gives an example of some time series data for a company.

Table 1. Company time series data

Year	Num_Employee	Profit	Tax
…			
1999	5100	4.2	1.1
2000	12000	13000	2500
2001	25.3	20000	4800
2002	30.6	35.3	7.97
…			

Intuitively, the example describes how the values of several attributes change over time. Each tuple represents a fact that can be viewed as a predicate with a timestamp argument and other non-temporal arguments. However, there are other unspecified metadata attributes, such as currency type and scale factor, that critically determine the truth value of each predicate. We call the specification of metadata attributes *context* knowledge. For metadata attributes whose value changes over time, a specification of their history is termed *temporal context*.

Table 2. Examples of temporal context

	Source	Receiver
Currency	Francs(FRF), year ≤ 2000 Euros, year≥2001	USD, always
Scale factor for **profit** and **tax**	1M , year =< 1999 1K, 2000≤year≤2001 1M, year≥2002	1K, always
Scale factor for **Num_Employee**	1, year≤2001 1K, year≥2002	1K, always
Profit	Exclude tax, year≤2000 Include tax, year≥2001	Include tax, always

Table 2 gives examples of the context knowledge in a simple integration scenario involving the source in Table 1 and a receiver. The receiver context can be time varying as well. Semantic conflicts arise because the source and the receiver have different contexts, which need to be reconciled for the receiver to meaningfully use the data. Imagine the complexity of scenarios that involve dozens of sources and receivers, each with time varying heterogeneous contexts. We need effective technologies to manage this complexity.

2.2. Temporal semantic heterogeneities

We see at least three categories of issues in the integration of temporal data.

Representational heterogeneity – the same relational attribute may be represented differently in the time span of a data source. In addition to *currency* changes for monetary concepts like *profit* and *tax*, there are also *scale factor* changes, as described in Table 2.

Ontological heterogeneity – the ontological concept represented by an attribute may change over time. In Table 2, profit on and before 2000 excludes taxes, afterwards it includes taxes. There are also cases where the entity referred to by an identifier changes over time. For example, stock symbol "C" used to refer to Chrysler but changed to refer to Citigroup on December 4, 1998 after Chrysler merged with Daimler-Benz. The derivation method of concepts, such as unemployment rate, often change over time.

Heterogeneity in temporal entity – the abstraction and representation of time domain differs across systems and time. Although a temporal entity is just another data type, it has special properties and operations that warrant a category of its own. The example in Table 1 uses point representation for the timestamp attribute *year*. Another system may choose to use intervals, e.g., [1/1/1999, 12/31/1999] for the year 1999. Differences in calendar systems, time zones, and granularities present many challenges for integration.

The semantics of the association between propositions described by the non-temporal attributes in a tuple and the temporal entity may differ across attributes. How the truth of a proposition over an interval is related to its truth over subintervals is described by the proposition's *heredity* properties [21]. Recognizing this property is useful for temporal query language design. For example, if in a bank account database the balance over an interval is known and the user queries the balance at a time within the interval, the query language should use the *liquidity* property of balance attribute to infer the result [3]. We observe that heredity is often attribute dependent and does not change over time or across data sources. Thus we need not consider heterogeneity of this property in the data integration setting.

In an effective integration framework, data receivers should not be burdened by these context heterogeneities; rather, it should be a system service to record con-

IEEE COMPUTER SOCIETY

text and use it to reconcile context differences before delivering data to the receiver. Our temporal extension to COIN framework will provide such a solution.

3. Review of Related Research

Related research can be found in the areas of temporal database, temporal reasoning, and data integration. Although each provides useful concepts and techniques, none address all the temporal semantic heterogeneity problems identified in this paper. The following brief review is not intended to summarize or criticize the findings in each area; rather, it is to identify the most relevant results and show what is missing from a temporal semantic data integration point of view.

Temporal databases. The time domain is often represented as time points with certain granularities. An interval is a set of contiguous points and can be represented as a pair of begin and end points. A time point may have a duration and thus is not an *instant* in time ontology [10].

Over 40 temporal data models have been proposed [20]. Many of the models let the system manage timestamps, which effectively hide the timestamp attribute from the user. This approach is inconsistent with relational theory [4]. As commonly practiced, databases that store temporal data often have a schema with explicit timestamp attribute(s); standard SQL is used to retrieve data and temporal operations are selectively implemented in the application layer. Our framework will target the common situation where data sources have limited temporal support.

As in the case of conventional databases, temporal databases also fail to facilitate context knowledge management. As a result, context is often hard-coded into data transformations in data warehouses. This ad-hoc approach lacks flexibility and scalability.

Temporal reasoning. While a restricted set of temporal logics can be executed using logic programming, there seems to be a trend where temporal logics are transformed into temporal constraints to take advantage of the efficiency of constraint solvers. The framework provided in [16] combines qualitative and quantitative (metric) temporal relations over both time points and time intervals. These relationships can be considered as temporal constraints in constraint logic programming. Therefore, temporal reasoning can be treated as a constraint solving problem, to which a number of constraint solving techniques [11] can be applied. We will use a solver implemented using constraint handling rules (CHR) [8] as demonstrated in [6].

Temporal granularity research, both logic [17] and algebraic [3] based, has developed techniques for representing and reasoning about granularities and user-defined calendars. Conversions between granularities [2] will be useful in dealing with heterogeneities in temporal entity.

Data integration. Approaches to achieving data integration largely fall into tight-coupling and loose-coupling categories depending on whether a global schema is used [5, 9]. In these approaches, intended data semantics in sources and receivers are explicitly incorporated in either the view definitions or the user queries. The computation complexity [14] in rewriting user queries for the former approach and the user's burden of writing complex queries for the latter limit the flexibility and scalability of these approaches.

COIN [5, 9] is a middle ground approach that, avoids these shortcomings by encapsulating data semantics into a context theory and maintaining accessibility of source schema by users. In COIN, a user issues queries as if all sources are in the user's context and a mediator is used to automatically rewrite the queries to resolve semantic differences.

Unfortunately, existing approaches, including COIN, use a static data model and ignore temporal context. Consequently, temporal concepts are missing in the ontologies used in these systems. This research will focus on the representation and reasoning of temporal context. Our framework incorporates context into the query evaluation process to automatically detect and reconcile temporal semantic conflicts. By combining the concepts and techniques from the three relevant research areas, we develop a scalable solution to temporal heterogeneity problems.

4. Temporal COIN approach

A recent extension to COIN demonstrates its new capability in solving ontological heterogeneity problems [5]. With various temporal representation and reasoning techniques, we can further extend COIN to handle temporal semantic heterogeneities.

4.1. The COIN framework

The COIN framework [5] uses an *ontology* as a formal mechanism for representing context knowledge and a *mediation* service to dynamically detect and reconcile semantic conflicts in user queries. Implemented in abductive constraint logic programming (ACLP) [14], the mediator generates *mediated queries* (*MQs*) that serve as intensional answers to the user. A distributed query executioner generates a query plan and brings extensional answers to the user. *Conversion functions* defined internally or externally are called

IEEE COMPUTER SOCIETY

during query execution stage to convert extensional dataset from its source context into receiver's context.

The existing COIN uses a snapshot data model that does not allow temporal context representation; the mediator also lacks temporal reasoning capability.

4.2. Temporal extensions to representation

The extended framework admits temporal data sources, which are assumed to be relational with an explicit timestamp attribute in their schema. They accept SQL queries with usual comparison operators (=, >, <, *etc.*) on timestamp domain.

Definition An *ontology* defines a common semantic domain that consists of semantic data types and their relationships, including 1) *is-a* relation, indicating a type is a subtype of another; and 2) named attribute relation, indicating the types that correspond to the domain and the range of the attribute.

The ontology is augmented with temporal concepts as defined in the time ontology [10]. The most general one is *Temporal Entity*, which can be further specialized as *Instant* or *Interval*. Each element in the source schema is mapped to a corresponding semantic data type in the ontology by an *elevation axiom*. A timestamp can be elevated to Temporal Entity or a subtype. For types whose values are time dependent, we relate them to a temporal entity type via *temporal attribute*.

There are two kinds of attributes in the ontology. A regular attribute represents a relation in source schema and obtains its value from the extensional database (EDB). A *contextual attribute*, also called a *modifier*, is an implicit attribute whose value, defined by context, functionally determines the interpretation and the value of the regular attribute [20]. For example, **currency** in Table 1 is a contextual attribute for **profit** and **tax**.

Definition The *temporal context* of a data source or a receiver is a specification of the history of all contextual attributes in the ontology. Mathematically, it is set of <*contextual_attribute, history*> pairs, where *history* is a set of <*value, valid_interval*> pairs.

In existing COIN, a context is simply a set of <*contextual_attribute, value*> pairs. The temporal extension allows us to represent the entire history of each contextual attribute. If the value does not change over time, the *history* set is simply a singleton with the *valid_interval* covering entire time span of interest. We achieve backward compatibility by treating <*contextual_attribute, value*> as the shorthand for <*contextual_attribute, {<value, entire_time_span>}*>.

Temporal entity type may also have contextual attributes, *e.g.*, granularity, time zone, etc., to account for various contexts.

4.3. Temporal extensions to mediation

Given a user query expressed in user's context, the mediator detects and reconciles semantic differences by examining involved contextual attributes and applying appropriate conversion functions if the values differ between any source and the receiver. With temporal extensions, contextual attributes are no longer singly valued. However, at any point in time, there is only one valid value for each attribute. The mediator needs to find the maximum time interval over which all involved contextual attributes are singly valued. Over this interval, a $\mathcal{MQ}$ can be generated as in the case of existing COIN; the interval appears in the $\mathcal{MQ}$ as additional constraints over the attribute of temporal entity type.

The mediator translates *history* pairs for contextual attributes into temporal constraints, which are posted into a constraint store concurrently solved by solvers in CHR. Through back tracking, all intervals over which contextual attributes are singly valued are found.

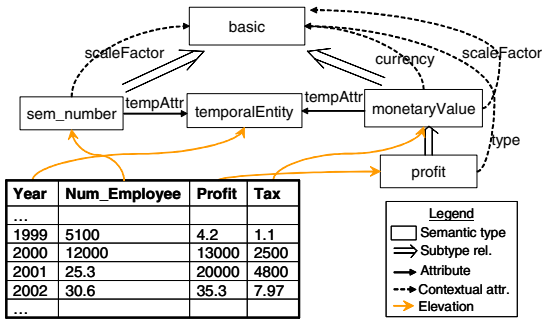
In our framework, contexts are declaratively defined using First Order Logic rules. The mediator is a general abductive reasoner. When new sources are added, we only need to add context rules for them. External functions can be called to convert between contexts. These features lend COIN great flexibility and scalability.

5. Prototype and preliminary results

We are able to solve a range of temporal heterogeneity problems exemplified in Section 2 by implementing a fraction of the suggested extensions.

5.1. Representation and mediation

Representation. Figure 2 shows the ontology for the example. Here, we use the most general concept *temporal entity*. Using elevation axioms, we create the mappings between attributes in data source and the types in the ontology.



Year	Num_Employee	Profit	Tax
…			
1999	5100	4.2	1.1
2000	12000	13000	2500
2001	25.3	20000	4800
2002	30.6	35.3	7.97
…			

Figure 2. Example ontology and elevation

IEEE COMPUTER SOCIETY

We model the time line as discrete and unbounded with both points and intervals as primitives. The past and future infinities are represented by constants *bottom* and *top*. We implement the ≤ relation between points as a *tle* constraint. The *contains* relation between an interval and a point is translated into *tle* constraints; the *overlaps* relation between intervals are also translated into *tle* constraints.

This simple model has sufficient expressive power to represent the temporal knowledge needed in Table 2. For example, internally we use the following Prolog statements to represent the source context for currency:

```
modifier(monetaryValue,O,currency,c_src,M):-
     containObj([bottom, 2000], O),
     cste(basic, M, c_src, "FRF").
modifier(monetaryValue,O,currency,c_src,M):-
     containObj([2001, top],O),
     cste(basic, M, c_src, "EUR").
```

The head of the statement reads: O of type *monetaryValue* has a contextual attribute (*i.e.*, modifier) *currency*, whose value in source context *c_src* is M. Its body has two predicates. *containObject(I, O)* will use the *tempAttr* of O to obtain its temporal attribute T (which corresponds to *Year* attribute in the EDB) of type *temporalEntity* and add constraint *contains(I, T)*. The helper predicate *cste* specifies the primitive value of M in *c_src* context. Thus, the history of each contextual attribute is now a set of pairs $\langle V_i, I_i \rangle$, where $\bigcup_i I_i = [bottom, top]$.

For context that does not change over time, we could have used *[bottom, top]* interval in *containObj* predicate. Since the translated constraints are always true, we will not include this predicate for this case.

Mediation. As described earlier, the mediation service needs to find the maximum interval over which all contextual attributes are singly valued. Figure 3 helps visualize this task by graphically representing the context knowledge in Table 2. For example, *[bottom, 1999]* is such an interval where the source context can be described with a set of singly valued contextual attributes:

```
{<monetaryValue.currency, "FRF">,
   <monetaryValue.scale, "1000000">,
   <profit.type, "taxExluded">,
   <sem_num.scale, "1">}.
```

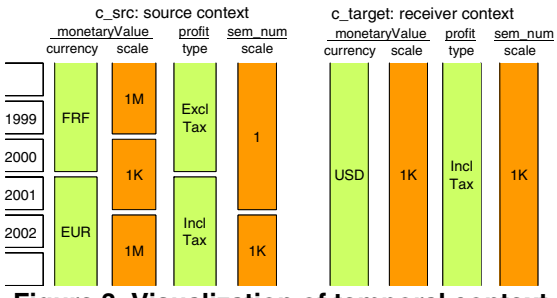


Figure 3. Visualization of temporal context

Recall that we translate all temporal relations into *tle* constraint over points. Each contextual attribute will generate two *tle* constraints for the temporal variable. The above problem is thus turned into to a problem of solving the constraints generated by all the contextual attributes, which is solved concurrently using a solver implemented in CHR.

Constraints over *bottom* and *top* can be removed using simplification rules so that these two literals do not appear in the list of abducted predicates. Constraints over other time points can be pair-wise simplified. We also implement *overlaps* to simplify *tle* constraints over four points at a time. These rules tighten the bounds of the temporal variable or signify a failure if inconsistencies are found.

Along with rules that handle equality constraint, this point based temporal constraint solver covers the 13 relations for temporal intervals in Allen [1]. Relations *before*, *after*, *meets*, and *met_by* generate a failure, all the rest relations are subsumed into *overlaps*.

Through backtracking, the recursive algorithm finds all intervals over which contextual attributes are singly valued. Conversions are applied as in the case of existing COIN implementation. This simple temporal constraint based extension transforms a temporal context problem into a set of non-temporal problems, thereby allows us to reuse the non-temporal implementation of the COIN mediator.

5.2. Preliminary results

These temporal extensions to COIN framework enable semantic interoperability for the integration example. The prototype can generate *MQs* that reconcile temporal context differences.

As an example, suppose a user in the receiver context wants to retrieve data from the company time series relation named *Financials* in Table 1 using the following SQL query:

```
Select Year, Num_Employee, Profit
From Financials;
```

and expects the returned data to be in his context. The query is translated into a well formed Datalog query in our prototype. The extended COIN mediator takes this query and the representation of the integration as input, and produces the following mediated query in Datalog (which COIN eventually converts to SQL):

```
%1. =<1999: currency,scale,type;scale
answer('V32', 'V31', 'V30') :-
  'V29' is 'V28' * 1000.0,   'V31' is 'V27' * 0.001,
  olsen("FRF", "USD", 'V26', 'V32'),
  'V28' is 'V25' * 'V26',
  financials('V32', 'V27', 'V25', 'V24'),
  'V32' =< 1999, 'V23' is 'V24' * 'V26',
  'V22' is 'V23' * 1000.0, 'V30' is 'V29' + 'V22'.
%2. 2000: currency and type;scale
```


IEEE COMPUTER SOCIETY

```
answer(2000, 'V21', 'V20') :-
  'V21' is 'V19' * 0.001, 'V20' is 'V15' + 'V14',
  financials(2000, 'V19', 'V18', 'V17'),
  olsen("FRF", "USD", 'V16', 2000),
  'V15' is 'V18' * 'V16', 'V14' is 'V17' * 'V16'.
%3. 2001: currency;scale
answer(2001, 'V13', 'V12') :-
  'V13' is 'V11' * 0.001, 'V12' is 'V10' * 'V8',
  financials(2001, 'V11', 'V10', 'V9'),
  olsen("EUR", "USD", 'V8', 2001).
%4. >=2002: currency,scale;none
answer('V7', 'V6', 'V5') :-
  olsen("EUR", "USD", 'V4', 'V7'),
  financials('V7', 'V6', 'V3', 'V2'),
  2002 =< 'V7',  'V1' is 'V3' * 'V4',
  'V5' is 'V1' * 1000.0.
```

The mediated query has four subqueries, each resolves a set of semantic conflicts that exist in the time specified by the timestamp attribute. Note that *olsen* predicate corresponds to a currency conversion data source introduced by the conversion function for *currency* contextual attribute. These subqueries resolve all the semantic conflicts in Table 2 or in Figure 3.

6. Discussion and future plan

We identified three types of semantic heterogeneity in the integration of temporal data. There is an ever increasing need to efficiently handle temporal heterogeneities as more historical data is used for auditing, forecasting, investigation, and many other purposes. We proposed temporal extensions to the COIN framework. A prototype of the extensions shows that our approach is capable of solving temporal context problems. With its declarative knowledge rule base and its capability of calling external functions, this approach is flexible and extensible.

Our future research aims to develop this approach in several aspects. Current representation of temporal context explicitly compares an interval with the temporal attribute of an object. The representation may be made cleaner by using an annotated temporal constraint logic [7]. We need to investigate how this logic can be integrated with the ACLP based COIN mediator.

An important part of future research will be focused on the heterogeneities of temporal entities. We plan to add various contextual attributes to the temporal entity type in the ontology and use external functions to convert between contexts. If this is not expressive enough to represent the diversity of time, a richer time ontology may be necessary. We also need to incorporate metric temporal reasoning, which often involves computations of one or more calendars. We will investigate the feasibility of leveraging web services like those in [2]. This is a challenging and important research area because misunderstanding date and time can have serious consequences, as history has shown in an 1805 event [15] where the Austrian troops were forced to surrender largely because of the misunderstanding of a date in two different calendar systems.

Acknowledgements: The study has been supported, in part, by MITRE Corp., MIT-MUST project, the Singapore-MIT Alliance, and Suruga Bank.

References

[1] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, 26, pp. 832-843, 1983.
[2] C. Bettini, "Web services for time granularity reasoning," TIME-ICTL'03, 2003.
[3] C. Bettini, S. Jajodia, and X. S. Wang, *Time Granularities in Databases, Data Mining, and Temporal Reasoning*: Springer, 2000.
[4] C. J. Date, H. Darwen, and N. A. Lorentzos, *Temporal Data and the Relational Model*: Morgan Kaufmann Publishers, 2003.
[5] Firat, "Information Integration using Contextual Knowledge and Ontology Merging," PhD Thesis, MIT, 2003.
[6] T. Frühwirth, "Temporal Reasoning with Constraint Handling Rules," ECRC-94-5, 1994.
[7] T. Frühwirth, "Temporal Annotated Constraint Logic Programming," *Journal of Symbolic Computation*, 22, pp. 555-583, 1996.
[8] T. Frühwirth, "Theory and Practice of Constraint Handling Rules," *Journal of Logic Programming*, 37, pp. 95-138, 1998.
[9] H. Goh, S. Bressan, S. Madnick, and M. Siegel, "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information," *ACM TOIS*, vol. 17, pp. 270-293, 1999.
[10] J. R. Hobbs, "A DAML Ontology of Time," LREC, 2002.
[11] J. Jaffar and M. J. Maher, "Constraint Logic Programming: A Survey," *Journal of Logic Programming*, vol. 19/20, pp. 503-581, 1999.
[12] S. Jensen, et al., "The Consensus Glossary of Temporal Database Concepts-February 1998 Version," 1998.
[13] C. Kakas, A. Michael, and C. Mourlas, "ACLP: Integrating Abduction and Constraint Solving," *Journal of Logic Programming*, 44, pp. 129-177, 2000.
[14] M. Lenzerini, "Data integration: a theoretical perspective," 21st ACM SIGMOD-SIGACT symposium on Principles of database systems, 2002.
[15] S. E. Madnick, "Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity," 1999 IEEE Meta-Data Conference, 1999.
[16] Meiri, "Combining Qualitative and Quantitative Constraints in Temporal Reasoning," presented at AAAI 91, 1996.
[17] Montanari, "Metric and Layered Temporal Logic for Time Granularity," PhD Thesis, University of Amsterdam, 1996.
[18] H. Nguyen, Y. Shahar, S. W. Tu, A. K. Das, and M. A. Musen, "Integration of Temporal Reasoning and Temporal-Data Maintenance Into A Reusable Database Mediator to Answer Abstract, Time-Oriented Queries: The Tzolkin System," *Journal of Intelligent Information Systems*, 13, pp. 121-145, 1999.
[19] G. Özsoyoglu and R. T. Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 7, pp. 513-532, 1995.
[20] Sciore, M. Siegel, and A. Rosenthal, "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems," *ACM TODS*, 19, pp. 254-290, 1994.
[21] Y. Shoham, "Temporal Logics in AI: Semantical and Ontological Considerations," *Artificial Intelligence*, 33, pp. 89-104, 1987.

IEEE COMPUTER SOCIETY