Momentum – an Active Time-Oriented Database for Intelligent Abstraction, Exploration and Analysis of Clinical Data

Alex Spokoiny and Yuval Shahar

Department of Information Systems Engineering, Ben Gurion University, Beer Sheva 84105, Israel
{spokoiny, yshahar}@bgumail.bgu.ac.il

Abstract. The temporal-reasoning task focuses on intelligent analysis of time-oriented data, while the temporal-maintenance task focuses on effective storage, query, and retrieval of these data. Both are highly relevant for biomedical applications such as monitoring, therapy, quality assessment, and visualization and exploration of time-oriented data, which cannot be expected to resolve each time both tasks, or to understand the internal details of specialized modules that perform them. Thus, it is imperative to supply a system, known as a temporal mediator, which integrates these tasks. One potential problem in existing temporal-mediation approaches is lack of sufficient responsiveness when querying the database for complex abstract concepts that are derived from the raw data, especially regarding a large patient group. We propose a new integration approach: The active time-oriented database. This approach is a temporal extension of the active-database concept, a merger of temporal reasoning and temporal maintenance within a persistent database framework. The approach preserves the efficiency of databases in handling data storage and retrieval while enabling specification and performance of complex temporal reasoning using an incremental-computation approach. The new approach provides persistence and truth-maintenance of the resultant abstractions. We implemented the active time-oriented database approach within the Momentum system. Initial experiments with the Momentum system are encouraging, and an evaluation is underway to assess its validity.

1 Introduction: Temporal Reasoning and Temporal Maintenance in Medicine

Representation, querying, and analysis of time-oriented clinical data are crucial tasks for both care providers and automated decision-support systems; both types of users need to extract certain information from one or more patient records to support diagnosis, therapy, monitoring, quality assessment, or clinical research. For example, during treatment by an experimental oncology protocol, a complex temporal query might be, "locate all patients who have had, within the past 9 months, more than two episodes of *grade II or higher bone marrow toxicity* (as defined by the protocol), each lasting at least 2 weeks." **(Figure 1.)**

Handling the time dimension typically requires performance of two distinct tasks: **temporal reasoning** (**TR**) and **temporal data maintenance** (**TM**). TR supports inference regarding time-oriented data, such as when monitoring patients, diagnosing disorders, planning and applying therapy. A major aspect of TR is **temporal abstraction**: creating high-level temporally extended concepts from raw time-stamped data. TM deals with storage and retrieval of data that have heterogeneous temporal dimensions. Decision-support applications that involve time-oriented medical data require performance of both tasks. Thus, it is highly desirable to have a single module that performs both tasks, with a well-defined interface.

Effective querying of abstract, temporally extended concepts, in particular within large sets of patients, is a major potential problem of current TR and TM integration efforts, often referred to as **temporal mediation**. To overcome these deficiencies, we propose a new approach, an **active time-oriented database**. The active time-oriented database approach is a temporal extension of the *active database* concept (see below), and might be considered as merging the TR and TM tasks within a persistent database framework. The approach preserves the efficiency of databases in handling data storage and retrieval, while supporting specification and performance of complex temporal reasoning (in particular, temporal abstraction) using an incremental computation approach. The approach we suggest provides persistence and truth-maintenance of the resultant temporal abstractions. We demonstrate the new approach in the **Momentum** system[1], which we have implemented as part of an architecture for *active temporal-abstraction mediation*.

2. Background: Temporal Abstraction, Temporal Mediation, and Active Databases

To clarify the ideas leading to the Momentum system, it is best to start by a brief presentation of the *temporal-abstraction*, *temporal-mediation*, and *active database* concepts.

2.1 Temporal Abstraction

A crucial part of temporal reasoning is creating high-level temporally extended concepts from raw time-stamped data. This task is often called **temporal abstraction** (**TA**) (see Figure 1).

[1] Momentum: A force that increases the rate of development of a process

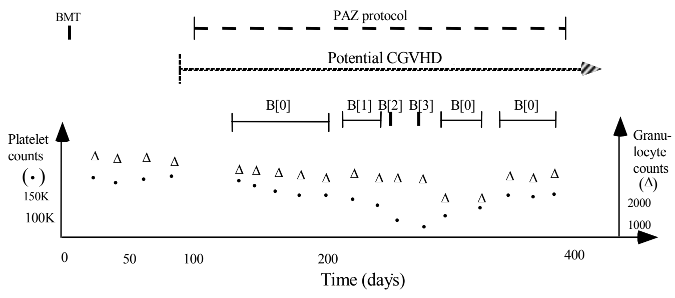

Figure 1. Abstraction of time-oriented data in the bone-marrow transplantation domain. Raw data are plotted over time at the bottom. External events and the abstract concepts computed from the data are plotted as intervals above the data. ⊢ —⊣ = an external event (medical intervention); • = platelet counts; Δ = granulocyte counts; ⊢—► = a context interval; ⊢—⊣ = a derived abstract concept interval; BMT = bone-marrow transplantation (an external event); PAZ = a protocol for treating chronic graft-versus-host disease (CGVHD), a complication of BMT; B[*n*] = bone-marrow–toxicity grade *n*, an abstract concept of platelet and granulocyte counts.

The **knowledge-based temporal-abstraction** (**KBTA**) problem-solving method [Shahar, 1997] was originally proposed specifically to solve the TA task. The KBTA method was designed with clear semantics for both the problem-solving method and its domain-specific knowledge requirements. The KBTA input includes a set of time-stamped *parameters* (e.g., blood-glucose values), external *events* (e.g., insulin injections), and, optionally, one or more user *abstraction goals* (e.g., therapy of patients who have insulin-dependent diabetes). All three input types can induce temporally extended *contexts* that can change the interpretation of one or more parameters [Shahar, 1998]. The KBTA output includes a set of interval-based, context-specific abstract concepts of several types: *state* (e.g., High), *gradient* (e.g., Decreasing), *rate* (e.g., Slow), and *pattern* (e.g., Multi-organ-toxicity). The KBTA method uses five TA *mechanisms* (e.g., *context-formation*) for five different inference tasks, which require four domain-specific TA *knowledge types* for any particular domain: (1) *structural knowledge* (e.g., ABSTRACTED-INTO relations); (2) *classification knowledge* (e.g., definition of a parameter range as High); (3) *temporal-semantic* knowledge (e.g., episodes of anemia can be aggregated into a longer single episode, since they are *concatenable*, but consecutive pregnancies cannot); (4) *temporal-dynamic* knowledge (e.g., persistence of a proposition over time when data is unavailable [Shahar, 1999]). Figure. 1 shows an example of input for the TA task, and the resulting output, in the case of a patient who is being treated by a clinical protocol for treatment of chronic graft-versus-host disease (GVHD), a complication of bone-marrow transplantation. The KBTA method was implemented within the **RÉSUMÉ** system [Shahar and Musen, 1996], which was tested in several different clinical domains, such as protocol-based care of patients who have AIDS or who have undergone bone-marrow transplantation; monitoring of children who have potential growth problems or of patients who have insulin-dependent diabetes, as well as in engineering domains such as traffic management [Shahar and Molina, 1998].

A part of the KBTA method was implemented in the **RASTA** system [O'Connor et al., 2001]. In addition, the RASTA system provided a distributed algorithm to speed up the execution of the TA task. Neither the RÉSUMÉ system nor the RASTA system allowed external querying or dynamic exploration of the high-level time-oriented abstract concepts generated by the TA process, as an inherent part of the system. Thus, *temporal-database mediators* were introduced to provide a more complete solution. (Indeed, the Momentum system is a part of a more general TA mediation architecture).

2.2 Temporal Mediation

A general approach to the TR and TM integration, called a *temporal-database mediator*, was suggested in an early architecture, the **Tzolkin** system [Nguyen et al., 1999] and in the more recent work describing the **Chronus-II** temporal database mediator [O'Connor et al., 2002]. This approach encapsulates the TR and the TM capabilities in a reusable software component that can answer raw or abstract, time-oriented queries. Such a system is called a *mediator* because it serves as an intermediate layer of processing between client applications and databases [Wiederhold, 1992]. As a result, the mediator is tied to neither a particular application, nor to a particular database [Wiederhold and Genesereth, 1997]. Furthermore, the temporal reasoning method encapsulates the task-specific TA reasoning algorithm that uses the domain-specific knowledge (thus, Tzolkin is really a *temporal-abstraction mediator*). The Tzolkin system consisted of the RÉSUMÉ temporal-reasoning module and the **Chronus** temporal-maintenance module. RÉSUMÉ generated all abstractions mentioned in the query and wrote them into a temporary database; then, the Chronus module applied the temporal constraints of the query to the temporary database (which now included also the desired abstractions), to generate the complete answer. A similar relationship exists, respectively, between the RASTA and Chronus-II systems in the Chronus-II temporal database mediator.

The mediator approach to temporal-data management is relatively novel, and thus raises new problems. One such problem is the difficulty in evaluating the general time complexity of a system that uses two very different computational methods, one of which is executed within working memory, and the other embedded within a database; the basic computational operation is difficult to define. Execution of complex temporal queries raises the important issues of the *defeasibility* (*nonmonotonicity*) of the computed temporal abstract concepts, and maintenance of the validity of the conclusions over time. The RÉSUMÉ system used a **truth-maintenance system** (**TMS**) that allows specific temporal abstractions to be withdrawn from working memory when new, contradictory data arrive (e.g., the result of a laboratory test that was taken 2 weeks ago). However, the TMS did not extend to the external database. To avoid the non-monotonicity problem, Tzolkin computes (de-novo) all abstractions based on the content of the database at the moment that the query is evaluated, a strategy that can obviously lead to significant response-time problems in the case of queries referring to complex temporal patterns, large longitudinal patient records, or a large set of patients. Furthermore the persistence of the computed abstract concepts cannot be managed between queries. Consequently, without a TMS at the database level, optimization techniques such as performance of batch computations are useful only when used with a data set that is guaranteed to be consistent over time (an unlikely situation in clinical databases). The Chronus-II mediator faces the same kind of problems. The query languages of the mediators also raise the issue of temporal knowledge maintenance and reuse. Temporal constraints in the Tzolkin and Chronus-II mediators can be defined only by using their querying languages, TSQL and TSQL2, respectively. This means that there is no way to share and reuse temporal-pattern knowledge among different temporal queries. This restriction makes concise query expression difficult. For example, many typical clinical temporal queries have to be written as two or more subqueries, which makes the query less concise and certainly less readable. In the Momentum system, we provide a solution to all the presented problems, while preserving the strengths inherent in the TA mediation approach.

2.3 Active Databases

Active databases (**ADBs**) extend "passive" databases by enabling the specification of reactive behavior as part of the database [ACT-NET, 1996; Widom and Ceri, 1996]. **Event-condition-action rules** (**ECA-rules**) in ADBs consist of events, conditions and resulting actions. The meaning of an ECA rule is: "when an event occurs, check the condition and if it holds, execute the action". Once a set of rules has been defined, the active database system monitors the relevant events. Whenever it detects the occurrence of a relevant event, it notifies the component responsible for rule execution. Subsequently, all relevant rules are triggered and executed. Rule execution incorporates condition evaluation and action execution. First, the condition is evaluated; if it is satisfied, the ADB executes the action. An ADB provides a **rule definition language** (**RDL**) as a means of specifying ECA-rules. Thus, an ADB supports rule specification, event detection, and rule execution.

We based the Momentum system on an *active time-oriented database* approach, which is a temporal extension of the *active database* concept. The presented approach preserves the efficiency of databases in handling data storage and retrieval, while adding the power of expressing and performing complex temporal reasoning using, as will be shown, an incremental-computation approach.

3. The Temporal-Abstraction Mediation Process

Before describing the Momentum system in detail, we present the context in which the system operates.

We define a *temporal-abstraction mediation (TAM) process* as a sequence of high-level tasks required to eventually generate temporal abstractions and to answer temporal queries. The process involves all phases, from data and knowledge input, through processing input raw data and delivering it in a form of knowledge-based abstract concepts to the TA mediation client applications (e.g., patient monitoring, application of clinical guidelines, assessing of the quality of guideline application, visualization and exploration of clinical data, etc.).

The TAM process (**Figure 2**) includes three phases: temporal knowledge and data acquisition, abstract-concept generation (including truth and persistence maintenance), and the abstract-concept access. The proposed process deliberately separates the data and knowledge acquisition and abstract concept generation from the abstract concept access. First, all the available abstract concepts are generated; only then they can be accessed. The queries do not need to generate abstract concepts on the fly, and thus they can be quickly answered even for complex queries and large sets of patients. Although all abstract concepts are generated before the data access (querying) starts, the process allows ad-hoc abstract concepts definition during the data access phase. In this case, the ad-hoc defined abstract concepts are generated during the query processing. This allows some level of flexibility for those client applications which cannot or do not desire to define in advance all abstract concepts in the knowledge base.

To support the TAM process, we have developed the *temporal-abstraction mediation architecture*.

3.1 The Temporal Abstraction Mediation Architecture

The *temporal-abstraction mediation architecture* (**Figure 3**) has two major aspects that are sufficiently different to warrant independent consideration. We labeled these two parts as the *back room* and the *front room*[2]. The *back room* supports the first two phases of the temporal abstraction mediation lifecycle–from the data and knowledge acquisition to the high-level abstract-concepts generation, truth and persistence maintenance, while the *front room* supports the third phase–the abstract-concepts access.

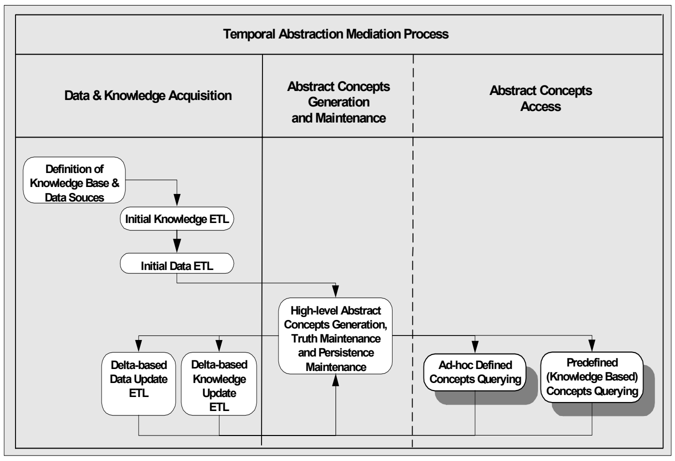
Temporal Abstraction Mediation Process
Data & Knowledge Acquisition
Abstract Concepts Generation and Maintenance
Abstract Concepts Access
Definition of Knowledge Base & Data Souces
Initial Knowledge ETL
Initial Data ETL
High-level Abstract Concepts Generation, Truth Maintenance and Persistence Maintenance
Delta-based Data Update ETL
Delta-based Knowledge Update ETL
Ad-hoc Defined Concepts Querying
Predefined (Knowledge Based) Concepts Querying

Figure 2. The temporal abstraction mediation process. The process is a sequence of high-level tasks required to generate temporal abstractions and answer temporal-abstraction queries. The process includes three phases: temporal knowledge and data acquisition, abstract-concept generation, truth and persistence maintenance, and access of the resultant abstract concepts. During the first phase, an extract, transform, and load (ETL) process is performed on the knowledge base and operational data sources: the sources are defined, the knowledge and data are extracted, transformed (for example, in accordance with a controlled medical dictionary) and loaded for the subsequent temporal abstract concepts generation.
We distinguish initial knowledge/data ETL from incremental knowledge/data updates (delta updates). During the second phase, the new knowledge-defined high-level abstract concepts are generated and the truth is maintained for the previously generated abstract concepts for each ETL (initial or delta-based). During this phase the persistence is also maintained for the generated abstract concepts. During the third phase, the generated abstract concepts can be accessed. Abstract concepts can be predefined in the underling knowledge base, or defined ad-hoc during the abstract-concepts access phase.

The *presentation server* is the "glue" which brings the two rooms together, hiding the complexities of the high-level abstract-concepts generation and enabling the decision support data to flow virtually seamlessly to the users community in form of the high-level abstract concepts generated from the data sources, according to the knowledge defined in the knowledge bases. The presentation server in fact integrates the TR tasks (abstract concept generation, truth and persistence maintenance) with the TM tasks (data and knowledge load, and the query services).

The Momentum system implements the presentation server of the architecture presented in **Figure 3**.

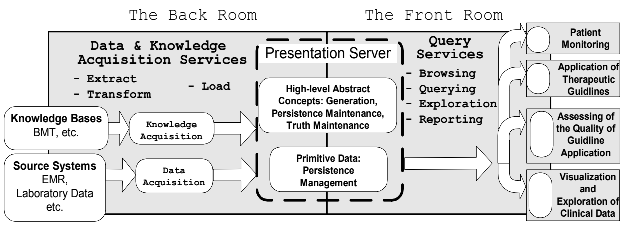
The Back Room
The Front Room
Data & Knowledge Acquisition Services
- Extract
- Transform
- Load
Presentation Server
Query Services
- Browsing
- Querying
- Exploration
- Reporting
Knowledge Bases BMT, etc.
Source Systems EMR, Laboratory Data etc.
Knowledge Acquisition
Data Acquisition
High-level Abstract Concepts: Generation, Persistence Maintenance, Truth Maintenance
Primitive Data: Persistence Management
Patient Monitoring
Application of Therapeutic Guidlines
Assessing of the Quality of Guidline Application
Visualization and Exploration of Clinical Data

Figure 3. The temporal abstraction mediator architecture. The architecture includes two parts: the *back room* and the *front room*. The back room is the part of the architecture that is responsible for data and knowledge acquisition (using the knowledge and data acquisition services), high-level abstract concepts generation, truth and persistence maintenance. Data and knowledge moves from the source systems and the knowledge bases to the acquisition areas using the applications provided as part of the data and knowledge acquisition services layer. This flow is driven by the definitions of the sources and targets and the transformation details (for example medical dictionaries mapping). The front room is the part responsible for delivering the generated abstract concepts to the user community: clinicians and computer based decision support systems (using the query-services layer). Both rooms share the *presentation server*, which loads the primitive data and knowledge in the case of the back room, generates abstract concepts, provides truth and persistence maintenance to the abstract concepts, and, in the case of the front room, presents the generated abstract concepts to the query services in a fashion that facilitates answering the user-issued temporal queries.

4. The Momentum System

Systems integrating both the TR and the TM tasks have requirements that span from efficient data-manipulation, typical of databases, to inference capabilities, typical of TA systems. Active databases may be considered as a bridge between these two kinds of systems: they have both the efficiency of databases in handling data storage and retrieval and the power of expressing complex inference mechanisms.

The Momentum system extends the approach of active databases to active time-oriented databases, by treating the temporal dimension in a unique fashion. In analogy to ECA-rules and rule execution of active databases, we provide Momentum with the domain knowledge required by the KBTA method and with the TA mechanisms. We also provide a **knowledge definition language** (**KDL**), which, in analogy to RDL, allows us to specify domain knowledge according to the KBTA ontology, and a **temporal abstraction query and instruction language**

[2] The back and front room notations originated in the data warehouse engineering community [Kimball and Ross, 2002].

(**TAQILA**). TAQILA supports the tasks of both raw-data specification and querying, abstract concepts querying, explanation of derived concepts, and dynamic sensitivity analysis of the derived concepts.

4.1. The Momentum Architecture

The Momentum architecture (**Figure 4.**)is modular, and is independent of any domain or application. The Momentum system includes, for each knowledge-base/data-source configuration, an instance of a *dynamic-processing environment*–the run-time environment, and an instance of a *dynamic repository*–the storage. The dynamic-processing environment is where the abstract-concept generation and querying take place. The dynamic repository provides persistence to the loaded knowledge, the loaded raw data, and the generated abstract concepts, which are stored along with their generation details (i.e., logical justifications such as *abstracted-from* and *abstracted-into* links). The dynamic-processing environment uses the repository to manage its persistence. Each knowledge base and data source, or a group of conformed data sources (sources containing different data aspects of the same patients), define a unique dynamic repository, which allows Momentum clients to work with different data and knowledge sources according to the temporal-abstraction mediation architecture.

The Momentum system implements the presentation server's back room functionality by providing KDL rules for knowledge definition and TAQILA raw-data specification instructions for the data loaded from data sources. The front room functionality is implemented by providing TAQILA querying capabilities which allow retrieval and exploration of both raw data and abstract concepts.

4.2 The Temporal Abstraction Query and Instruction Language (TAQILA)

TAQILA is simple yet expressive general-purpose query language. It enables data addition, asking runtime queries regarding the generated (knowledge based) or ad-hoc defined abstract concepts; its default arguments for such concepts include: start time, end time, and value.

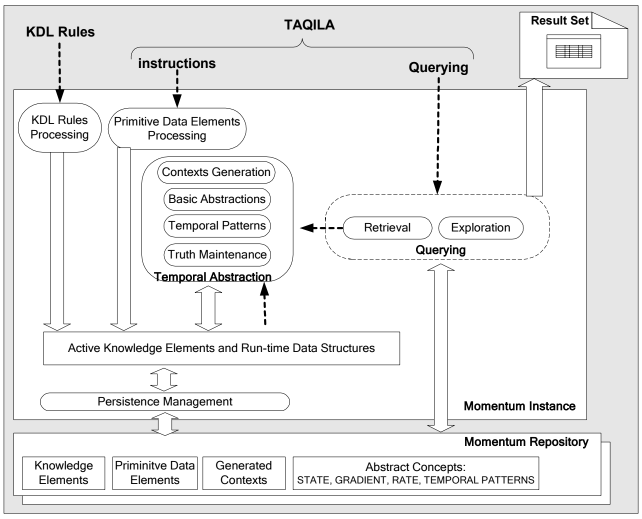

Figure 4. The Momentum Architecture: The Momentum system includes a dynamic processing environment–the run-time environment (where the abstraction and querying take place), and a dynamic repository (which provides persistent storage to primitive data, knowledge, and generated abstract concepts). The dynamic processing environment uses the repository to manage its persistence. Each dynamic processing environment consists of five modules: (1) KDL-rule- processing, (2) primitive-data processing, (3) abstraction, (4) querying, and (5) persistence management. Module (1) loads knowledge into the system and generates active knowledge elements, which detect the occurrence of relevant primitive or abstract concepts and activate the abstraction process; module (2) loads the primitive data into the run-time data structures; module (3) generates abstract concepts and updates the run-time data structures; module (4) enables retrieval and exploration (sensitivity analysis, etc.) of the generated abstract concepts set; module (5) manages persistence of the primitive and generated abstract concepts, as well as of the active knowledge elements.

TAQILA also supports exploration querying such as *explain* queries*,* which provide the data instances from which an abstract concept instance was derived, and *what-if* dynamic sensitivity-analysis queries, which support propagation of the implications of hypothetical modifications to either data or knowledge. TAQILA provides: (1) raw data instructions–for inserting (ADD), updating (UPDATE), and removing (REMOVE) primitive data elements; (2) retrieval (FIND) querying–for querying raw and derived concepts; and (3) exploration querying–for justification of the abstraction process (EXPLAIN) and for dynamic sensitivity analysis (WHAT-IF), which simulates the effect of modifying the data or the knowledge, using the TMS. Retrieval and exploration querying return sets of temporal intervals satisfying the query constraints. The TAQILA queries and instructions form an application interface (API) that is typically manipulated by various client applications.

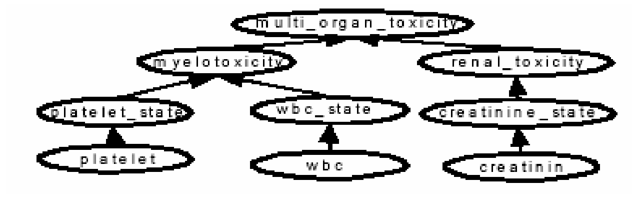

Figure 5. The abstract concept dependency tree of post-BMT multi-organ toxicity, part of the oncology-domain knowledge base. Directed arcs represent dependency relations. All concepts require a **Post-BMT Context** generated by the BMT event. Multi-organ toxicity is a temporal pattern which is defined as one week overlapping of meyelotoxicity and renal toxicity (GRADE_3 lasting for at least one week). Platelet, wbc and creatinin are raw data elements which can be abstracted into platelet_state, wbc_state and creatinie_state respectively. Myelotoxicity is abstracted from either platelet_state or wbc_state and is computed as a maximum of their values. Renal toxicity is abstracted from creatinine state.

In the reminder of this section we present in detail the data and knowledge flow that demonstrate how, given the knowledge presented in **Figure 5**, Momentum handles several different types of TAQILA data-addition instructions and data- and concept-retrieval and exploration queries. The examples also present some of the TAQILA syntax.

Table 1. Processing of a set of TAQILA instructions and queryies, using the knowledge represented within the concept dependency tree depicted in Figure 5 as an example.

TAQILA	Momentum Processing
ADD (ID_1, BMT, , 'Jan 3,2002', 'Jan 3,2002') ADD (ID_1, Creatinin, 2.5, 'Jan 12,2002', 'Jan 12,2002');	BMT event creates a Post-BMT context. All abstract concepts with Post-BMT as a necessary context can now be generated. Momentum adds a creatinin instance for the patient ID_1 with value 2.5 on Jan 12, 2002, to the repository and generates creatinine_state abstract concept on 'Jan 12,2002' with value HIGH (using a pre-existing creatinin_state definition).
ADD (ID_1, Platelet , 70,000, 'Jan 12,2002', 'Jan 12,2002');	Momentum adds a platelet instance to the repository, generates Platelet_state instance on 'Jan 12,2002' with value NORMAL, and Myelotoxicity on the same day with value GRADE_0.
ADD (ID_1, WBC , 2000, 'Jan 12,2002', 'Jan 12,2002');	Momentum adds a WBC instance to the repository, generates WBC_state instance on 'Jan 12,2002' with value LOW; the TMS then updates myelotoxicity on that day to GRADE_2

UPDATE WBC
WHERE CASE.id = ID_1
WHEN time = 'Jan 12,2002'
SET value = 600;
	Momentum updates the WBC instance; the TMS updates the value of the WBC_state instance to VERY_LOW and the value of the Myelotoxicity instance to GRADE_3

ADD (…), UPDATE (…), REMOVE (…)	Momentum adds primitive instances to the repository, generates new abstract concepts, and propagates updates via the TMS.

FIND SECOND Myelotoxicity WHEN
start-time < 'Jan 12,2002' AND
stop-time BEFORE NOW
RETURN CASE.id, value;
 	
`FIND Result:`
`PID    CID    VALUE`
`017    ID_1   GRADE_3`
`021    ID_2   GRADE_2`
 	
EXPLAIN 017
 	
<Myelotoxicity>
abstracted from:
Wbc_state and Platelet_state
using StateMaxOr mapping function.
Details:
005
Wbc_state(ID_1,VERY_LOW, 'Jan 12,2002' , 'Jan 12,2002')
007
Wbc_platelet(ID_1, NORMAL, 'Jan 12,2002' , 'Jan 12,2002')
	For each case (e.g. each patient) the query searches patients whose myelotoxicity starts after 'Jan12,1998' and finishes before NOW. For each found instance the case id (e.g. patient id) and the instance value are returned.
	The FIND query allows querying of primitive and derived concepts as well as ad-hoc detection of temporal patters (ad-hoc temporal pattern specification). The basic FIND query consists of 4 clauses: FIND, WHERE, WHEN and RETURN. The FIND clause constrains the resulting set on the element type (e.g. Myelotoxicity) and on ordinal ordering (FIRST, SECOND, THIRD, or LAST). The WHERE clause specifies constraints on the case attributes (e.g., CASE_ID) and non-temporal instance attributes (e.g., value). The WHEN clause constrains the temporal attributes of the time intervals of the data element-instances (start, end time points, and duration). Finally, the RETURN clause specifies the desired projection on the data-element attributes.
	In this example, the result set consists of three fields: PID – derived instance id, CID – case id and the value;
	The EXPLAIN instruction provides the concept instances from which an abstract concept instance was derived, within the KDL rule that generated them (i.e., both the data and the knowledge).

FIND SECOND Multi_organ_toxicity AS { OVERLAPS((GRAIN = WEEK, Renal_toxicity(value=GRADE_3 AND duration >= 2), Myelotoxicity (value=GRADE_3 AND duration >= 2), (overlap >= 1))	This query demonstrates the ability of TAQILA to define the temporal pattern on the fly. The query searches for instances of multi-organ_toxicity (a temporal pattern which is defined as a one week overlapping of myelotoxicity and creatinine_state, both of which have a value of GRADE_3 for at least two weeks), which starts before 'Jan 12,2002' and finishes before now. Momentum first retrieves all

}
WHEN
start-time < 'Jan 12,2002' AND stop-time BEFORE NOW
RETURN CASE.id, EXISTS?;
	renal_toxicity and myelotoxicity instances, then generates all available Multi_organ_toxicity, and finally constrains the generated instances according to the WHEN clause. The query returns the boolean indication whether the temporal pattern was found for a patient.

SET WHAT-IF ON
 	
ADD (…), UPDATE (…), REMOVE (…), FIND (..) …
 	
SET WHATIF OFF
	The WHAT-IF instruction allows dynamic sensitivity analysis of hypothetical modifications of either data or knowledge.
	SET WHAT-IF ON starts the what-if session. TAQILA instructions can then be performed. After the SET WHAT-IF OFF instruction, the repository returns to the initial pre-session state.

4.3. The Momentum Dynamic-Processing Data Structures and Algorithms

Due to lack of space, we will not present here the full internal workings of Momentum. However, several points are worth mentioning:

1. Momentum uses (a) several specialized time-oriented structures in the database to represent the data, both for the runtime evaluation environment and the repository, and (b) specialized knowledge structures to represent KDL rules.
2. The data structures are designed to optimize concept generation, using computational mechanisms similar to those of the KBTA method, but implemented quite differently. For example, interpolation between similar concepts, using a context-sensitive interpolation table [Shahar, 1999] is highly efficient; its worst case complexity is O(N^2), but typically (apart from a one-time sorting operation) it is a linear process.
3. The data structures facilitate incremental modification of the repository through a built-in TMS. An *inference network* maintains links among data, knowledge, and derived concept instances.

4.4. Implementation Details

We implemented the Momentum system in Java Programming Language. KDL and TAQILA as well as results sets are described as XMLs. For the data repository we use a native XML database which is compliant to the XML:DB specification and allows us to store unstructured data.

5. Discussion and Future Work

Momentum is an active time-oriented database for intelligent abstraction, exploration and analysis of time-oriented data, and, in particular clinical data. In this prototyping phase of the system, although we are using real clinical data and knowledge (mostly from chronic-disease domains, such as oncology), no complete clinical validation has been carried out yet. Nevertheless, the results of the described work are significant mainly because they prove that an active time-oriented database approach may be effectively used for integration of temporal reasoning and temporal maintenance.

Continuing the heritage of the Tzolkin system, Momentum is tied to neither a particular application nor a particular database or knowledge base. Momentum is thus independent of any particular domain and application. The reuse of Momentum in a new application involves modification of only the domain-specific knowledge, does not require the application to understand the details of the TR and TM tasks performed by Momentum, other than the KDL and TAQILA APIs, and requires no reprogramming.

Momentum tackles the responsiveness issue. A complex temporal query execution process may be intractable because the task of temporal reasoning is inherently computationally expensive, especially when population querying is involved (one of the main problems of the Tzolkin and Chronus-II mediator architecture). To address the problem, Momentum provides a new focus on the persistence and reuse of previously derived abstract concepts. Momentum dynamically generates and manages the persistence of all relevant abstractions for a given set of data previous to their use, and then enables applications to perform querying and exploration, which are very effective because all the possible abstractions have already been generated and stored (somewhat similar to Tzolkin's *batch computation* mode, but performed dynamically and incrementally). However, not all types of abstractions have to be predefined in the knowledge and be computed before querying can be performed. Temporal patterns can be defined in the querying phase itself and computed on the fly. This allows flexibility that resembles the Tzolkin TSQL and Chronus-II TSQL2. Unlike the Tzolkin and Chronus-II mediators, however, Momentum generates new abstract concepts only when relevant primitive (raw) data is added to the system, without re-computing previously generated abstractions (i.e., an incremental computational approach) while maintaining truth in the already generated abstract concepts, using dynamically formed logical links.

Unlike the Tzolkin and Chronus-II systems, the Momentum abstraction and querying modules share the same run-time data structure and truth maintenance systems. This unification solves the nonmonotonicity and basic computational operation problems encountered by the early temporal mediators.

Unlike Tzolkin's TSQL and Chronus' TSQL2, Momentum provides an expressive and simple general-purpose TAQILA, which is not SQL-based. TAQILA querying is being done on the generated abstract concepts' time intervals, where the default arguments are: start time, end time, and value. This simplifies the query syntax, and complex temporal queries can be very concise and readable, yet equally expressive. Unlike the early mediators, TAQILA also allows exploration querying such as *explain* which provides the data instances from which an abstract concept instance was derived, and *what-if* dynamic sensitivity analysis queries, which allows tracking of the hypothetical modifications of either data or knowledge.

Finally, Momentum provides specification of TAQILA instructions and KDL knowledge instances through well-defined APIs. The clean APIs allow Momentum to be used as part of a larger architecture. An example is this supporting a temporal-abstraction mediation process, in which the data and the knowledge is asserted and abstract concepts are generated in advance; another example is a Tzolkin–like architecture, in which the data and the knowledge are asserted (possibly retrieved from an external process) on the fly. One of the potentially highly interesting experiments would be substituting Momentum and its persistent storage system instead of the pure transient-memory temporal-abstraction component of the IDAN temporal-mediation architecture [Boaz and Shahar, 2003].

In the short term, we plan to perform clinical validation of the Momentum system on one or more large clinical databases. We also plan to further improve the performance by making the abstraction algorithms parallel and distributed. Multiple applications exist, such as supporting guideline-based care within the DEGEL architecture [Shahar et al., 2003a] and dynamic visual exploration of time-oriented clinical data within the KNAVE-II architecture [Shahar et al., 2003b].

7. Acknowledgments

This research was supported in part by NIH award No. LM-06806. We thank Martin O'Connor for useful discussions regarding the Chronus-II and RASTA systems, and David Boaz for illuminating discussions of temporal mediation.

References

ACT-NET (1996). The Active Database Management System Manifesto: A Rulebase of ADBMS features. *ACM SIGMOD Record*, 25(3):20--49, September.

Boaz, D., and Shahar, Y. (2003). Idan: A Distributed Temporal-Abstraction Mediator for Medical Databases. *Proceedings of the 9th Conference on Artificial Intelligence in Medicine—Europe (AIME) '03,* Protaras, Cyprus

Kimball R, and Ross M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling* (2nd Edition). John Wiley & Sons.

Nguyen JH, Shahar Y, Tu SW, Das A, and Musen MA. (1999). 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** (1/2), 121-145

O'Connor M.J., Grosso W.E., Tu S.W., and Musen M.A.. (2001). RASTA: A Distributed Temporal Abstraction System to Facilitate Knowledge-Driven Monitoring of Clinical Databases. *Proceedings of MEDINFO-2001, the Tenth World Congress on Medical Informatics*, pp. 508-512, London, UK

O'Connor M. J., Tu S. W., Musen M. A.. (2002). The Chronus II Temporal Database Mediator. *Proceedings of the American Medical Informatics Association (AMIA) 2002 Annual Fall Symposium*, San Antonio, TX.

Shahar Y, Musen MA. (1996) Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine* **8**, 267-298

Shahar, Y. (1997). A framework for knowledge-based temporal abstraction. *Artificial Intelligence* **90** (1–2), 79–133

Shahar, Y. (1998). Dynamic temporal interpretation contexts for temporal abstraction. *Annals of Mathematics and Artificial Intelligence* **22** (1-2) 159-192.

Shahar, Y., and Molina, M. (1998). Knowledge-based spatio-temporal linear abstraction. *Pattern Analysis and Applications* **1** (2) 91-104

Shahar, Y. (1999). Knowledge-based temporal interpolation. *Journal of Experimental and Theoretical Artificial Intelligence* **11**, 123-144

Shahar Y, Young O., Shalom E., Mayaffit A., Moskovitch R., Hessing A., and Galperin M. (2003a). DEGEL: A hybrid, multiple-ontology framework for specification and retrieval of clinical guidelines. *Proceedings of the 9th Conference on Artificial Intelligence in Medicine—Europe (AIME) '03,* Protaras, Cyprus

Shahar Y, Boaz D., Tahan G., Galperin M., Goren-Bar D., Kaizer H., Basso L.V., Martins S.B., and Goldstein, M.K. (2003b). Interactive Visualization and Exploration of Time-oriented Clinical Data Using a Distributed Temporal-Abstraction Architecture. *Proceedings of the 2003 AMIA Annual Fall Symposium*, Washington, DC

Widom J, Ceri S. (1996). *Active Database Systems: Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann, San Mateo, CA.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–50.

Wiederhold, G. and Genesereth, M. (1997). The Conceptual Basis of Mediation Services. *IEEE Expert*, 12(5), 38–47