Analysis of Temporal Abstraction in Medical Databases

Mira Balaban[1], David Boaz[2] and Yuval Shahar[2]

Department of Information Systems Engineering,
Ben Gurion University, Beer Sheva 84105, Israel

[1]mira@cs.bgu.ac.il , [2]{dboaz,yshahar}@bgumail.bgu.ac.il

Abstract

Physicians and medical decision-support applications, such as for diagnosis, therapy, monitoring, quality assessment, and clinical research, reason about patients in terms of *abstract,* clinically meaningful concepts, typically over significant *time periods*. Clinical databases, however, store only *raw, time-stamped* data. Thus, there is a need to bridge this gap. We introduce the Temporal Abstraction Language (TAR) which enables specification of abstract relations involving raw data and abstract concepts, and use it for defining typical medical abstraction patterns. For each pattern we further analyze finiteness properties of the answer set.

1 Introduction: Temporal Abstraction

In a historical database, the information that is stored includes temporal attributes stating when the information is valid. Temporal abstraction [17, 18, 10, 5, 6] is an approach for reasoning about historical databases that allow multiple abstraction levels. The use of temporal abstraction is important especially for decision support applications which consume abstract concepts, while databases usually contain primitive concepts. Abstractions can be performed on the atemporal attributes (e.g., "150 kgs" might be abstracted to heavy), or on the temporal attributes (e.g., two distinct "heavy" facts that hold on Monday and Friday might be abstracted into one "heavy" fact that holds during the interval from Monday to Friday).

One of the first in-depth ontologies for handling many aspects involved in the temporal-abstraction task is the knowledge-based temporal-abstraction (KBTA) ontology of Shahar [17], which was implemented within the RÉSUMÉ system and further extended in CAPSUL [5, 6]. The KBTA distinguishes between the following relation types: events, parameters, and contexts. Events represent external interventions (e.g., insulin injections, a chemotherapy protocol), parameters represent the subject state (e.g., blood-glucose values), and contexts represent situations (e.g., diabetes).

Facts can be abstracted (derived) from other facts. The KBTA contains several built in abstraction mechanisms. In this research we handle the following:

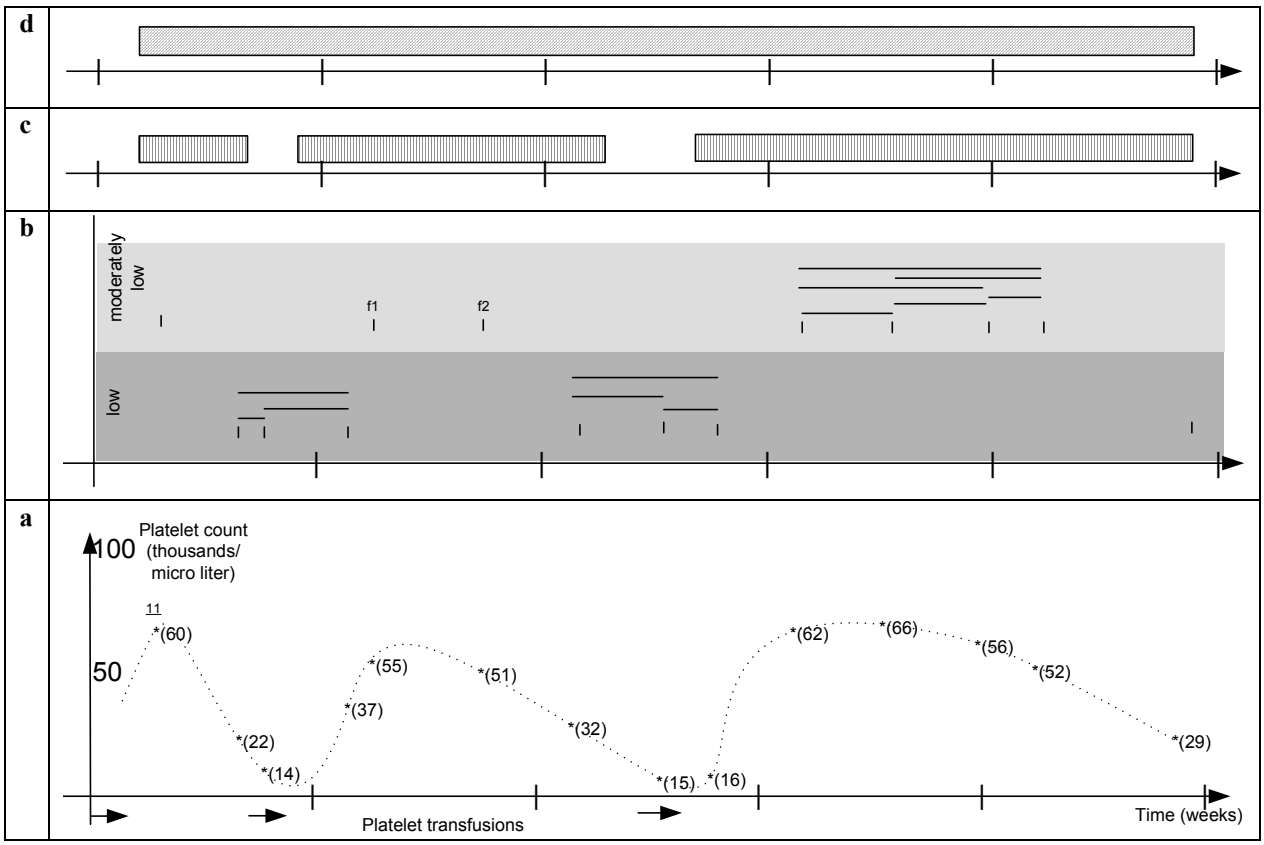


Fig. 1. A schematic visualization of temporal abstraction in the bone-marrow transplantation (BMT) domain. A patient who has undergone a BMT receives frequent platelet transfusions until the patient's body recovers and starts to produce platelets on its own again. (a) displays extensional facts: *platelet_count* (*) and *platelet_transfusion* (➙). (b) Displays *platelet_state* intensional facts. These facts are generated using two rules: The *vertical abstraction* rule maps *platelet_count* values less than 50 into *low*, and values between 50 and 100 into *moderately_low*. The *interpolation* rule concatenates two adjacent close enough *platelet_state* facts (points or intervals) with the same value into one long fact. Note that the facts *f1* and *f2* were not close enough. (c) Displays for each *platelet_transfusion* fact its corresponding *platelet_half_life* value. *The platelet_half_life* is computed as the time (days) it takes for the *platelet_count* to decrease to one half of the level that was measured immediately post-transfusion. (d) Displays a *recovery* fact. *recovery* is defined as successive increasing *platelet_half_life* fact durations, which indicates that the patient produces platelet again, and recovering.

- The *Vertical Abstraction* Mechanism – responsible for abstracting from one, or more, simultaneous base facts into a new derived fact. The interval of the derived fact is the *intersection* of base fact time-intervals. The value of the derived fact is the result a specific map function application on the base facts. This function maps between different value domains: $D \rightarrow R$. The restriction in the KBTA is that the range is finite (e.g. *{low, moderately low, normal, high}*). Actually, the restriction may be relaxed, such that the range is "smaller" than the function domain. For finite domains, smaller means smaller in size. For infinite domains, for example the function can round real numbers to the closest integer. Figure 1 presents a vertical abstraction relation, where *platelet_count* (panel a) are the base facts and *platelet_state* (panel b) are the derived facts.

- The *Temporal Interpolation* Mechanism (details in [18]) – responsible for bridging two different temporal intervals or points that do not meet, provided that the temporal gap is smaller than a specific time measure that is determined by the durations of the facts and their values. Figure 1 in panel b presents an interpolation relation for *platelet_state* facts. Note how short facts create longer facts and that the facts *f1* and *f2* are not close enough to be bridged.
- The *Linear Patterns Detection* Mechanism – responsible for detection of phenomena that occur only once. This mechanism generalizes the two previous ones, since it enables the user to provide an arbitrary set of constraints and functions. Figure 1 in panel c presents a linear pattern, called *platelet_half_life*, in which the phenomenon is a *platelet_transfusion* followed by a *platelet_count* with value *V1*, followed by another *platelet_count* with value *V2*, where $V2 < \frac{V1}{2}$. The inferred fact interval is the interval that starts in the *platelet_transfusion* fact start and finishes at the end of the second *platelet_count* fact.
- The *Periodic Patterns Detection* Mechanism – infers new intensional facts based on detection of maximal sequences of repeating phenomena that satisfy given criteria. A maximal repeating sequence is a sequence of facts of the same kind that follow in time and can not be extended without violating the criteria. A periodic pattern specifies the repeated fact type, a set of constraints on the sequence and the inferred fact. Figure 1 in panel d presents a *recovering* fact. *Recovery* is specified using a periodic pattern, where the repeating elements are successive *platelet_half_life* facts (derived by a linear pattern rule, as explained above) and the constraints are that the duration of fact intervals is increasing and that there should be at least three repetitions. More examples of periodic patterns appear in [5, 6, 10]. There are several kinds of constraints on sequences: *"Each_constraints"* are constraint on separate elements (facts), *"Successor_constraints"* are constraints between successive elements in sequences and "*Sequence_constraints"* are constraints on whole sequences. In the example presented in figure 1 panel d, the constraint that the durations are increasing is a successor constraint, since it is applied on adjacent facts in the sequence; the cardinality constraint "at least three repetitions" is a sequence constraint, since it applies to the sequence as a whole.

In this paper we formalize and generalize the above temporal abstraction mechanisms. We present a temporal abstraction mechanism that subsumes CAPSUL's linear pattern, as well as other mechanisms types in the KBTA ontology and the RÉSUMÉ system, and analyze the finiteness property of its answer set. We note that the mechanisms can not be expressed neither in relational algebra due to the usage of recursion, nor in Datalog due to the usage of functions and the support of the time dimension. This mechanism is implemented as the core reasoning in the Idan system [3].

In section 2 we present the TAR language and we define condition for the finiteness of answer sets. In section 3 we present its account for the KBTA abstraction mechanisms and analyze their finiteness properties. Section 4 discusses future research.

2 The Temporal-Abstraction Rules (TAR) Language

A TAR knowledge-base consists of *Rules* and *Facts,* and can be viewed as a subset of deductive databases.

2.1 TAR Language Syntax

The TAR language contains symbols of three types (for each there are constant and variable symbols; variables start with upper case letters): 1) *Individual* symbols (e.g., *john*), 2) *Time-interval* symbols (e.g., the day of 1/1/2000) and 3) *Value* symbols (e.g., *2.3 cm, low, abnormal,* and *green*). The value symbols (constants or variables) are further partitioned into sub-types, like: *Integer, Size= {small, medium, big},* etc. Each value symbol/ variable is associated with a single type. These types are termed *value-types*.

A fact is a TAR atomic formula specified using a predicate symbol with the signature: *Individual* $\times$ *Time-interval* $\times$ *vt*, where *vt* is a value-type. For example *height(john, 1/1/1990-1/1/1990, 152* cm*)* is a fact. That is, *atomic formulae* have the structure *p(d, i ,v)*, where *p* is a predicate symbol, *d* is a term of type individual*, i* is a term of type time-interval and *v* is a term of an appropriate value-type. A *fact* is a ground (without variables) atomic formula.

The language contains also a set of external evaluable *functions* and *constraints*: A *time-function* is a function symbol with the signature $(Time\text{-}interval \cup Value)^n \rightarrow Time\text{-}interval$ (i.e. the function accepts time-intervals and values, and returns a time-interval). A *value-function* is a function symbol with the signature $(Time\text{-}interval \cup Value)^n \rightarrow Value$. A *constraint* is an external n-ary predicate symbol that accepts time-intervals and values symbols as arguments, and evaluate to TRUE/ FALSE by calling an external constraints package (e.g., *greater_than*, *during*, *monday* etc).

A TAR *rule* (to be defined from now on, simply as *rule*) is a statement of the form:

$h(D, I, V) \leftarrow b_1(D, I_1, V_1),\ldots,b_n(D, I_n, V_n) \mid \{c_1,\ldots,c_m\}, ifn, vfn$

where: $h(D, I, V)$, $b_1(D, I_1, V_1),\ldots,b_n(D, I_n, V_n)$ are atomic formulae (note that the same individual appears in all atomic formulae, and $n \geq 1$), $c_1,\ldots,c_m$ are constraints, *ifn* is a time-function symbol, and *vfn* is a value-function symbol. The signature of *vfn* is determined by the types of the value variables (e.g. if the head predicate is associated with a value-type *Size*, then the function returned value-type must be *Size* as well). $h(D, I, V)$ is the *head* atom of the rule, *h* is the head predicate, each $b_i(D, I_i, V_i)$ is a *body atom* of the rule, b_i is a *body predicate* in the rule, the conjunction of the body atoms is the *body* of the rule. Variables that appear in the head are *head variables*, and variables that appear in the body are *body variables,* denoted $\vec{B}$. The Interval and Value head variables are distinct from all body variables. A *rule instance* is a pair of substitutions (σ_h, σ_b), where σ_h is a substitution for the head variables, and σ_b a substitution for the body variables. A *ground rule instance* is a rule instance in which all body variables are ground.

Later in this paper (in section 3) we extend the language so that the body of a rule functions as a general means for retrieving information from the knowledge base (extensional and intensional). In the simple case of TAR rules, as above, information is retrieved by explicit call to facts. In the general case, information can be retrieved by other meta-level tools (e.g., aggregation).

2.2 Semantics of TAR

The Temporal Model: We assume a time line that can be identified with the integers. The model distinguishes among three temporal data-types. A *Time-point* is identified with the integers, a *Time-*

measure that denotes sizes on the time-line and a *Time-interval* that denotes a segment on the time line. For example, 3 o'clock is time-point, t*wo hours* is a time measure, and it can be from *3* to *5 o'clock* or from *8* to *10 o'clock*. A measure can be a positive, negative or zero.

Language of Constraints: In the TAR language facts can be constrained by temporal or value constraints. Temporal constraints are built-in, and value constraints depend on the application domain. Constraints can be combined using logical connectives and possibly by cardinality based connectives like *at-least-N.*

The language temporal built-in constraints are: *Calendar constraints* over a single time-point. e.g., *wednesday(P1)* holds if P1 happened on Wednesday. *Point-* and *Measure-constraints* enables comparison of time-points and time-measures. Following Allen time-interval algebra [1], we define *interval-constraints* as predicates equivalent to the 13 basic binary relations between intervals.

TAR language is domain independent. It can be applied on various value types, e.g., numbers, strings, images, structures etc. For each value type, the user adds external value constraints that are invoked (evaluated) on query processing. It is the user responsibility to apply the correct constraints on the appropriate value types.

A TAR semantic structure is a pair $J= (D, \cdot^J)$ of a domain *D*. The Domain is partitioned into three parts: Individual, Time-Interval and Value.

- *Individual* is a finite non-empty domain of individuals (in the medical domain patients are individuals), e.g., the person whose name is John.
- *Time-Interval* is an infinite domain of time intervals. An interval is a segment on a discrete time line, e.g., the segment of time from 1/1/2000 00:00 to 04:00.
- *Value* is a finite or infinite domain of values. Different value domains are possible in different application domains, e.g., *182 centimeters* and *high-fever*.

The time-interval and value symbols in the language are identified with the entities in the Time-Interval and the various Value domains. That is, the collection of symbols is the semantic domain for these types.

A time-function symbol denotes a function that accepts time-intervals and values, and returns an interval. Similarly, a value function symbol denotes a function with the same argument types that return a value. A relation symbol denotes I*ndividual* $\times$ *Time-Interval* $\times$ *Value.* For example: the meaning of the relation *height* is a finite set of triplets: {*(john, 1/1/1990-1/1/1990, 152* cm*), (marry, 1/1/1990-1/1/1990, 160 cm), (john, 1/1/2000-1/1/2000, 185 cm)…}*. Constraint symbols denote external constraint predicates. The constraints are partitioned into *temporal constraints* which are built-in, and *value constraints* which are evaluated using external, packages. A set of constraints must be evaluable for every ground instance of a rule.

Temporal-Abstraction Property: Given the rule $r=h(D, I, V) \leftarrow b_1(D, I_1, V_1),\ldots,b_n(D, I_n, V_n) \mid \{c_1,\ldots,c_m\}$, *ifn, vfn*, and an interpretation *J*, we say that the rule is true in *J*, written $J \models r$, if for every ground instance (σ_h, σ_b) of *r*, where σ_h is empty, the following holds:

If:

1. *All body atoms are satisfied in J, i.e.,* $J \models (b_1(D, I_1, V_1),\ldots,b_n(D, I_n, V_n))\sigma b$
2. *All constraints evaluate to TRUE, i.e.,* $(c_i)\sigma b$*=TRUE for every* $1 \le i \le m$

Then:

1. *The time variable, I, and the value variable, V, in the head are substituted to the values obtained by the time function, ifn, and value function, vfn, i.e., define σ_h' to be {I=i, V=v} such that $i^J=ifn((\vec{B}\ \sigma_b)^J)$ and $v^J=vfn((\vec{B}\ \sigma_b)^J)$*
2. *The rule head holds in J, i.e., $J \models (h(D, I, V)\sigma_h$'*

This condition is called the *Temporal-Abstraction Property* of *r* with respect to interpretation *J*.

An interpretation *J* is a *model* of a knowledge-base if every rule in it fulfills the temporal-abstraction property with respect to *J*. An inference mechanism for TAR can be either query driven i.e., find whether the query holds, in every model of the knowledge base, or can compute the intensional relations. We say that an inference mechanism is *complete* if it can compute the intensional relations.

2.3 Well Defined TAR Knowledge-Bases

Datalog programs restrict the rules to guarantee finite intensional relations. In contrast, the rules in the TAR language contain function symbols and can define infinite relations. This is because the time and value functions can create unboundedly new values. Clearly, our interest is to characterize TAR rules that guarantee finiteness. Such knowledge bases are termed *well defined*. The existence of a complete terminating inference mechanism implies finiteness. The common approach for computing intensional relations is bottom-up evaluation of the rules [2, 11, 12], which repeatedly applies rules until no new fact computed. Termination is guarantied [21] if each round is finite, and the number of rounds is finite.

Claim 2.1: TAR rules guaranty finite rounds, i.e., in each round, a finite number of facts are added (note that in the general case, if a rule has an infinite number of instances, this is not necessarily true).

Proof: (sketched) The claim results from the fact that all body relations are finite, and the head variables are limited by the functions in the rule body.□

An extensive amount of research was devoted to the study of finiteness in deductive databases and in extended Datalog in particular. There are approaches that study finiteness constraints on infinite arithmetic relations [11, 12, 21]; others study restrictions on concrete domains [14, 15].

Based on claim 2.1, infinity of intensional relations in TAR can arise only when a complete bottom-up inference procedure has an infinite number of rounds. This can happen only if the time and value functions keep generating new values. For that purpose we introduce the notions of converging/ diverging functions and the related diverging dependency graph that is derived from the rules. The idea is that if the latter is restricted to be stratified diverging, the intensional relations are finite. A similar restriction was used in [14, 15] for guarantying finiteness of answers to queries over sequence databases.

Intuitively, a function is converging if it can be associated with some norm over some associated well-founded domain such that the values of functions applications properly decrease. A function that is not converging is *diverging*. The d*iverging dependency graph* of a TAR knowledge-base is a directed-graph, where *nodes* are labeled by the predicates, and arcs are either converging, la-

beled "c", or diverging, labeled "d". There is a *converging arc* from *b* to *h* if there is a rule with *h* as the head predicate, *b* as a body predicate and the time and value functions are converging; and there is a *diverging arc* from *b* to *h* if at least one of the rule functions is diverging. A knowledge-base has a *stratified-diverging* structure if its diverging dependency graph does not contain cycles with a diverging arc. Figure 2 shows an example of a knowledge base that is not stratified diverging.

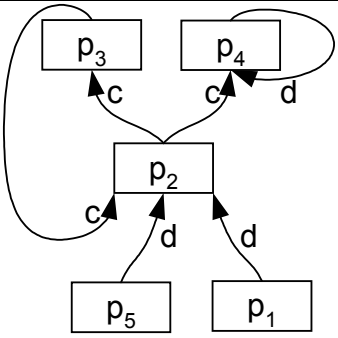
TAR rules	The diverging dependency graph
r1: p2(V) ←p1(V1)| V= V1*2 r2: p2(V) ←p3(V3), p5(V5)| max(2*V5, V3) r3: p3(V) ←p2(V2)| V= V2 r4: p4(V) ←p2(V2)| V= V2 r5: p4(V) ←p4(V4)| V= s(V4)	

Fig. 2. An example of a knowledge base and its diverging dependency-graph (for the sake of simplicity, we omit the time variables, the constraints and the time function of the rules). *p1* and *p5* are finite, since they are extensional predicates. The value functions used in r1, r2, and r5 are diverging. *p2* and *p3* are finite because they are not participating in a cycle with a diverging arc. *p4* is not finite because it participates in a cycle with a diverging arc (p4→p4). The knowledge base does not have a stratified-diversion structure, and hence it is not well defined.

Claim 2.2: A TAR knowledge base with a stratified-diversion structure is well defined.

Proof: (sketched) Recursive applications produce decreasing values in a well-founded domain, while non-recursive rules can be applied only a finite number of times. All in all, the amount of new terms is finite.□

In conclusion, the finiteness of intensional relations can derive from the dependency links between predicates and the convergence of the rule functions. Note that finiteness is not affected by the constraints.

3 Temporal Abstraction in TAR

The KBTA includes several abstraction mechanisms. In this paper we deal only with the vertical abstraction, interpolation, linear pattern detection and periodic pattern detection mechanisms. In this section we show how these mechanisms are handled in TAR. For that purpose we first describe the essential knowledge base structures required by these mechanisms.

The herbrand base of a TAR knowledge base is totally ordered by a *follow* relation. The follow relation is a binary relation defined on facts. Fact f_2 follows fact f_1, written $follow(f_2, f_1)$, if $f_2.start > f_1.start$. In order to characterize periodic patterns we need to identify temporal sequences. A *temporal sequence* (shortened *sequence*) is a finite ordered set of facts that follow in time. In this paper we use the following notations: *f* denotes facts, and *s* and *t* denote sequences. $[f_1, \ldots, f_n]$ denotes the sequence of facts f_1 to f_n; the empty sequence ε is denoted []; $s[i]$ denotes the *i*-th element. In every sequence, $follow(s[i+1], s[i])$ for every element of the sequence.. For a sequence $s = [f_1, \ldots, f_n]$, the start of *s*, denoted *s.start*, is $f_1.start$; the end of *s*, denoted *s.end*, is $f_n.end$; and the interval of *s*, de-

noted *s.interval*, is [*s.start, s.end*]. For the sequence s = [$f_1,\ldots, f_n$], new facts can be concatenated to its start or end: $s{\cdot}f$ = [$f_1,\ldots, f_n, f$] is defined if *follows*(f, f_n), and $f{\cdot}s$ = [$f, f_1,\ldots, f_n$] is defined if *follows*(f_1, f).

For a concrete TAR knowledge base (i.e. the collection of facts) we can define *adjacency* and *maximal sequence* relations. The followness relationship induces an adjacency relationship between the knowledge base facts. Fact f_2 is adjacent to fact f_1, written *adjacent*(f_2, f_1), if $f_2.start > f_1.start$, and there is no different fact f_3 such that $f_1.start < f_3.start < f_2.start$. The adjacency relation is functional, i.e., for every fact f_1, there is a single adjacent fact f_2, not symmetric and not transitive. Note that adjacency is meaningless for the herbrand base. Sequence *s* is *tight* if for every $1\leq i<n$: *adjacent*($s[i], s[i+1]$).

A sequence *s* is *maximal* with respect to a set of constraints *C*, if for no *f* in the knowledge base: $s{\cdot}f$ or $f{\cdot}s$ satisfies *C*. Maximal sequences *s*, *t* are *disjoint* if for no *i*, *j*: $s[i]=t[j]$; otherwise they are *intersecting*. *s* and *t* are *overlapping in time* if *s.interval* intersects *t.interval*. Practically, it appears that medical information systems are interested in sequences that are disjoint [4]. Figure 3. describes several maximal sequences.

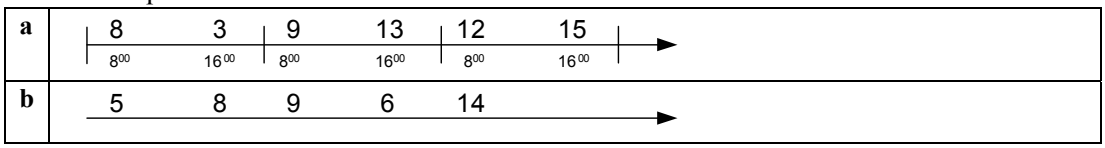


Fig. 3. Examples of maximal sequences. The panels present sets of arbitrary measures. (a) The measures are performed every day at 8 and 16 o'clock. The set of constraints, *C1*, constrains successive elements to be in gap of 24 hours, and with increasing values. There are two maximal sequences with respect to *C1*: *s1* = [8, 9, 12] and *s2* = [3, 13, 15]. The sequences are disjoint, but overlapping in time. Note that the sequence [8, 9] is not maximal because the fact 12 can be added to the sequence without violating *C1*. (b) The set of constraints, *C2*, constrains successive elements to have increasing values. In this case there are two maximal sequences with respect to *C2*: *s3* = [5, 8, 9, 14] and *s4* = [5, 6, 14]. In this case, the sequences are intersecting since the elements 5 and 14 appear in both. If the maximal sequence is specified as tight, the only sequences are *s5* = [5, 8, 9] and *s6* = [6, 14]. These sequences are disjoint and not overlapping in time.

Property 3.1: For a finite knowledge base all maximal sequences are bounded by the size of the knowledge base. This property results from the absence of repetitions in temporal sequences. It also implies that the number of maximal sequences is finite.

3.1 TAR Account for KBTA Mechanisms

There are four KBTA mechanisms. We now consider each separately:

- *Vertical Abstraction* Mechanism*:* For each abstract parameter *a* which is abstracted from $\{p_1,\ldots,p_n\}$, we create a rule:

  $a(D, I, V) \leftarrow p_1(D, I_1, V_1), \ldots, p_n(D, I_n, V_n)|$
  $simultaneous(\{I_1,\ldots,I_n\}),$
  $I=intersection(\{I_1,\ldots,I_n\}),$
  $V=map_a(V_1,\ldots,V_n).$

  where, *simultaneous*($\{I_1,\ldots,I_n\}$) is a compound constraint, that checks that $I_1,\ldots,I_n$ have not empty intersection; *intersection*($\{I_1,\ldots,I_n\}$) returns the interval intersection of $I_1,\ldots,I_n$;

$map_a(V_1,...,V_n)$ applies the value map of a on $V_1,...,V_n$, and return a value. For example, two simultaneous *platelet_state* and *wbc_state* (state of white blood cells) facts create a *myelotoxicity* fact. If the base facts values are *low*, then the *myelotoxicity* fact value is *grade_3*. This rule can be formulated as:

$$myelotoxicity(D,I,V) \leftarrow platelet\_state\ (D, I_1, V_1),\ wbc\_state\ (D, I_2, V_2) \mid simultaneous(\{I_1,I_2\}),\ I=intersection(\{I_1,I_2\}),\ V=map_a(V_1,V_2).$$

$$map_a(\{low,low\})=grade\_3$$

Hence, the database instance: {*platelet_state (john, 1/2001-12/2002, low), wbc_state (john, 1/2000-12/2001, low)}* implies the fact: *myelotoxicity (john, 1/2001-12/2001, grade_3)*.

- *Temporal Interpolation* Mechanism: the interpolation mechanism interpolates *adjacent* facts that are *close enough*. That is, if facts $p(D,I_1,V)$ and $p(D,I_2,V)$ are found to be adjacent and close enough the temporal interpolation mechanism implies a new fact $p(D,I_3,V)$ where I_3 is the interval that bridges the interval I_1 and I_2 by adding the gap between them.

Adjacent facts are *close enough* if the gap between their intervals is sufficiently small. The notion of sufficiently small depends on the predicate, the facts value and their duration. For example, if a patient has *over-weight* facts on the start and end of the month, we imply that the patient is *over_weighted* during all the month. On the other hand, if the patient had *fever* (temperature above 39^o Celsius) at the start and end of the month, we can not imply that the *fever* last during all the month.

For each predicate, p, that has a close enough function, $close\_enough_p$, we create a rule:

$$p(D, I, V) \leftarrow p(D, I_1, V), p(D, I_2, V) \mid p(D, I_2, V)=adjacent(p(D, I_1, V)),\ close\_enough_p(I1,I2,V),\ I=(I_1.start, I_2.end)$$

- *Linear Pattern* Mechanism: A linear pattern specification does not specialize the time or value functions and the form of the rules itself. Therfore, a general linear pattern can be captured by an arbitarary TAR rule. Clearly, vertical abstraction and temporal intepolation are special cases of linear patterns.
- *Periodic Pattern* Mechanism detects maximal sequences of phenomena. In the special case where the repetition has an exact upper bound (on the length) the periodic pattern can be specified as a set of regular TAR rules. For example, if the predicate p is implied from two to four repetitions of q, it can be defined by three regular TAR rules:

$$p \leftarrow q,q \mid C',\ ifn,\ vfn$$
$$p \leftarrow q,q,q \mid C'',\ ifn,\ vfn$$
$$p \leftarrow q,q,q,q \mid C''',\ ifn,\ vfn$$

where the constraints C', C'' and C''' restrict the repetition sequences to be maximal.

However, in the general case, where there is no upper bound (or it is too large), detecting maximal sequences involves *recursion*. The recursion is handled by employing an interme-

diate recursive mechanism that accumulates maximal sequences. Let $maxSeq_q(D, C, Seq)$ stand for the application of maximal sequence detection on predicate q for individual D, and the set of constraints C. Seq is the detected sequence. The periodic pattern is captured by the following extended TAR rule:

$$p(D, I, V) \leftarrow maxSeq_q(D, C, Seq) \mid \{\}, ifn, vfn.$$

Note that the knowledge base level operation *maxSeq* serves the role of a usual TAR rule body, since it provides arguments for the constraints (empty in this case), *ifn* and *vfn*.

3.2 Analysis of the KBTA rules

A practical question for a KBTA knowledge base is whether it ensures finiteness of intensional relations? If no, which restriction can be imposed on it in order to ensure finiteness? As described in the previous section, the main limitation is that the knowledge-base diverging-dependency-graph does not contain diverging arcs. We will scan the KBTA mechanisms and check this feature.

- *Vertical Abstraction:* The time function of this mechanism is the *intersection* function. *Intersection* is converging over the temporal domain, since the result is smaller or equal to the input arguments, and that the temporal domain is discreet. The value function is restricted to reduce its domain (the function range is smaller than its domain), as explained in the introduction. Therefore, vertical abstraction can not be applied recursively. Hence, when applying only vertical abstraction, the intensional relations are finite.
- *Temporal Interpolation*: This mechanism is recursive by definition (two *low platelet_state* facts are bridged into one long *low platelet_state* fact). Finiteness of interpolated predicates results from the properties of the temporal interpolation function (the value function is simply identity). The temporal interpolation function has the property that if $I1 \supseteq I2$ then the *temporal_interpolation(I1,I2)=I1*. Since the number of intervals in the database instance is finite, and temporal interpolation produces intervals that include the original intervals, then when applying only temporal interpolation the intensional relations are finite.
- *Linear pattern:* This mechainsm does not impose any restriction on the rule format or on the time and value functions. Therefore, linear pattern rules should be restricted by general finiteness methods, like divergence-dependency graphs.
- *Periodic Pattern*: periodic patterns are similar to simple TAR rules (linear patterns), except that the body of the rules consists of the $maxSeq_q$ application instead of calls to explicit knowledge base predicates. The general finiteness analysis for simple TAR rules is based on the assumption that all body relations are finite. For periodic patterns, the $maxSeq_q$ application functions as a body relation. Therefore if it is finite, the finiteness of the head predicate derives on the regular analysis for TAR rules. Property 3.1 implies that for every finite predicate q, invidual D and constraints C, $maxSeq_q(D, C, Seq)$ is finite.

4 Related Work

Summarization of patient's state is an important task of current medical information systems. The goal is to create an automatic summarization of patient's current state based on the patient's data. Systems like TrenDx of [9], VIA-VENT of [16], M-HTP of [13] and Resume of [17] implement various abstraction means in order to summarize the data. Temporal abstraction is one such means. The language suggested in this paper is a first attempt to formulize temporal abstraction.

The TAR language is related to deductive databases and in particular to temporal databases, since it is an extended Datalog (Datalog with function symbols) language with temporal arguments [7]. Significant efforts were devoted to the study of finiteness in the presence of function symbols. One direction involves decision procedures for finiteness of queries [11, 12]. Another direction involves restriction on the knowledge base structure when applied to concrete domains, e.g. sequences [14, 15]. Constraint databases [19] are also closely related to our subject.

Temporal deductive databases rely on the structure of time, and especially assume the availability of a successor function. One direction involves finding a finite representation for an infinite temporal knowledge base [4, 8]. Another direction imposes restriction on the query language in order to guarantee finite answer sets.

5 Discussion and Future Research

In this paper we have described common medical temporal abstraction mechanisms, based on the KBTA ontology of Shahar[17, 18]. A language, termed TAR, for temporal abstraction was introduced, finiteness restrictions for TAR knowledge bases are shortly discussed. It is shown that TAR language abstracts the common temporal abstraction mechanisms of the KBTA, and finiteness of these mechanisms is analyzed.

In the future, we intend to the further investigate the properties of TAR and its application to other mechanisms of the KBTA (e.g., contexts, gradients, and rate mechanisms). In particular, we intend to study maximal sequences and their constraints, and further study interaction between mechanisms. Finiteness analysis should be deepened, especially for the study of convergence of functions in the temporal domain. Adding negation is another issue. All improvements and extension will be evaluated through the current implementation in the IDAN system [3].

Bibliography:

1. Allen, J. F.: Maintaining knowledge about temporal intervals. CACM 26 (11) (1983) 832-843.
2. Bancilhon, F. and R. Ramakrishnan: An amateur's introduction to recursive query-processing strategies. ACM SIGMOD Intl. Conf. on Management of data, (1986) 16-52.

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

4. Baudinet, M., Chomicki, J., and Wolper, W.: Temporal Deductive Databases. In Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, pages 294- 320, 1993.

5. Chakravarty, S. and Y. Shahar: CAPSUL: A Constraint-Based Specification of Repeating Patterns in Time-Oriented Data. Annals of mathematics and artificial intelligence (AMAI); Vol. 30 (2000) 3-22.

6. Chakravarty, S., and Y. Shahar. Specification and detection of periodicity in clinical data. *Methods of Information in Medicine* 40(5), 410-420, 2001. Reprinted in: Haux, R., and Kulikowski, C. (eds), *Yearbook of Medical Informatics* 2003, Stuttgart: F.K. Schattauer and The International Medical Informatics Association, forthcoming.

7. Chomicki, J.: Temporal Query Languages: A Survey. In Dov M. Gabbay and Hans J. Ourgen Ohlbach, editors, Temporal Logic: International Conference on Temporal Logics (ICTL'94), volume 827, pages 506{534. Springer-Verlag, 1994.

8. Chomicki, J., Imielinski, T.: Temporal deductive databases and infinite objects. In Seventh ACM Symposium on Principles of Databases Systems, pages 61-73, Austin, Texas, Mar. 1998.

9. Haimowitz, I.J. and Kohane, I.S., "Managing temporal worlds for medical trend diagnosis." Artificial Intelligence in Medicine, 8(3): 299–321 (1996).

10. Keravnou, E.T., "*Temporal abstraction of medical data: Deriving periodicity*." In: Intelligent Data Analysis in Medicine and Pharmacology (Lavrac, N., Keravnou, E.T. and Zupan, B., eds.), Kluwer, 1997, pp.61-79.

11. Kiffer, M.: On the Decidability and Axiomatization of Query Finiteness in Deductive Databases. JACM vol 45, no 4, Jully 1998, pp. 588-633.

12. Krishnamurthy, R., R. Ramakrishnan, and O. Shmueli: A Framework for Testing Safety and Effective Computability of Extended Datalog. Proc. Sixth ACM Symposium on Principles of Database systems, (1988) 154-163.

13. Larizza, C., Moglia, A., and Stefanelli, M., "M-HTP: A system for monitoring heart transplant patients,"Artificial Intelligence in Medicine, 4: 111–126 (1992).

14. Mecca, G. and A. J. Bonner: Sequences, Datalog and Transducers. In Fourteenth ACM SIGMOD Intern. Symposium on Principles of Database Systems (PODS'95), San Jose, California, (1995) 23-35.

15. Mecca G. and A.J. Bonner. Query Languages for Sequence Databases: Termination and Complexity. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(3): 519-525, May/June 2001.

16. Miksch, S., Horn, W., Popow, C. and Paky, F., "Utilizing temporal data abstraction for data validation and therapy planning for artificially ventilated newborn infants," Artificial Intelligence in Medicine, 8: 543–576 (1996).

17. Shahar, Y.: A Framework for knowledge-based temporal abstraction in clinical domains. Artificial intelligence 90(1-2) (1997) 79-133.

18. Shahar, Y.: Dynamic Temporal Interpretation Contexts for Temporal Abstraction. Annals of Mathematics and Artificial Intelligence. 22(1-2) (1998) 159-92.

19. Revesz, P. Z.: Constraint databases: A survey. In L. Libkin and B. Thalheim, editors, Semantics in Databases, number 1358 in LNCS, pages 209–246. Springer, 1998.

20. Tansel, A., J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors. Temporal Databases: Theory, Design and Implementation. Benjamin/ Cummings, 1993.

21. Ullman, J. D.: Principles of database and knowledge-base systems. (1988) volume1 chapter 3.