ELSEVIER

Artificial Intelligence 90 (1997) 79–133

Artificial Intelligence

A framework for knowledge-based temporal abstraction

Yuval Shahar [1]

Section on Medical Informatics, Knowledge Systems Laboratory, Medical School Office Building x215, Stanford University, Stanford, CA 94305-5479, USA

Received March 1995; revised April 1996

Abstract

A new domain-independent knowledge-based inference structure is presented, specific to the task of abstracting higher-level concepts from time-stamped data. The framework includes a model of time, parameters, events, and contexts. A formal specification of a domain's temporal abstraction knowledge supports acquisition, maintenance, reuse, and sharing of that knowledge.

The *knowledge-based temporal abstraction method* decomposes the temporal abstraction *task* into five *subtasks*. These subtasks are solved by five domain-independent *temporal abstraction mechanisms*. The temporal abstraction mechanisms depend on four domain-specific *knowledge types*: structural, classification (functional), temporal semantic (logical), and temporal dynamic (probabilistic) knowledge. Domain values for all knowledge types are specified when a temporal abstraction system is developed.

The knowledge-based temporal abstraction method has been implemented in the *RÉSUMÉ* system, and has been evaluated in several clinical domains (protocol-based care, monitoring of children's growth, and therapy of diabetes) and in an engineering domain (monitoring of traffic control), with encouraging results. © 1997 Elsevier Science B.V.

Keywords: Temporal reasoning; Temporal abstraction; Knowledge acquisition; Knowledge representation

1. Introduction: temporal abstraction and the knowledge level

Many domains of human endeavor require the collection of substantial numbers of data over time, and the abstraction of those data into higher-level concepts meaningful for that domain. Much work had been done in philosophy and in computer science regarding the structure of time and the nature of general temporal reasoning. Here, we focus

[1] E-mail: shahar@smi.stanford.edu.

PII S0004-3702(96)00025-2

on the specific task of context-sensitive abstraction and interpretation of time-stamped data. We investigate the nature and semantics of this task; the knowledge required for performing it in uniform fashion in different domains; and the way that knowledge should be represented, acquired, maintained, used, shared, and reused efficiently. We mention briefly a computer system that implements our theoretical framework, and that has been evaluated in several medical and engineering domains. Finally, we provide preliminary evidence that our approach might be generalized to linear abstraction over other dimensions, such as space.

The examples presented in this paper focus on several subdomains of clinical medicine, in which the task of abstraction of data over time occurs frequently for various reasons. The ideas discussed, however, are general, and are applicable to other domains in which data need to be abstracted over time.

Most clinical tasks require measurement and capture of numerous time-stamped patient data. Physicians who have to make diagnostic or therapeutic decisions based on these data may be overwhelmed by the number of data if their ability to *reason* with the data does not scale up to the data storage capabilities. Thus, it is highly desirable for an automated knowledge-based decision support tool that assists physicians who monitor patients over significant periods to provide short, informative, context-sensitive summaries, at various levels of abstraction, of time-oriented clinical data stored on electronic media. Meaningful summaries include abstractions that hold over both time points, such as dates when data were collected, and time intervals, such as "5 months of decreasing liver-enzyme levels in the context of recovering from hepatitis". Interval-based abstractions are useful for both the physician and an automated system for

(1) planning interventions for diagnostic or therapeutic reasons,
(2) monitoring therapy plans during execution,
(3) creating high-level summaries of medical records that reside on a clinical database,
(4) providing explanations, and
(5) comparing a plan prescribed by one agent (e.g., a clinical guideline) with the actions performed by another agent (e.g., a physician). Often, overall and intermediate goals of the plan can be described as temporal patterns to be achieved, maintained, or avoided.

Currently, most decision support systems do not have sufficient general temporal reasoning knowledge. The few systems that include temporal reasoning capability usually encode both the general temporal reasoning knowledge and the temporal reasoning knowledge specific to the particular domain in application-specific rules and functions. These frameworks do not make explicit the domain-independent temporal reasoning tasks that need to be solved, the domain-independent methods used to solve these tasks, and the domain-specific knowledge required for applying these methods. Application-specific approaches enable little *reuse* of domain-independent temporal reasoning knowledge for other domains; they also do not enable *sharing* of domain-specific temporal reasoning knowledge accumulated for the particular encoded task with other tasks and problem-solving methods that involve reasoning about time in the same domain. In addition, due to the idiosyncratic nature of the knowledge representation scheme of temporally oriented

knowledge used in most systems, it is difficult to *acquire* the required knowledge from domain experts in a uniform, well-defined (possibly automated) way, or to maintain that knowledge, once acquired.

In this paper, we present a unifying, *knowledge level* [40] view of the task of abstraction of data over time. We rely on several insights from the past decade. Chandrasekaran [4, 5] defined the concept of *generic tasks*. For example, *diagnosis* is a generic task, that implies no particular control structure; it can be performed by various *problem-solving methods*, such as the *heuristic classification* method (or, rather, *inference structure*, since no control structure is specified) [8]. Other methods, more specific with respect to the *knowledge roles* that they assume, have been defined for diagnosis tasks (e.g., *cover and differentiate* [16]) and for *design* tasks (e.g., *propose and revise* [32]). Preliminary taxonomies of problem-solving methods have been suggested [6, 36]. Common to all task-specific problem-solving methods is the limitation of the potential *role* (e.g., creation of a context) that a domain knowledge item (e.g., a rule) can play, thus facilitating the use and maintenance of this knowledge [36]. As the emphasis on task-specific architectures became more pronounced, it became increasingly clear that *designing a new knowledge-based system, and acquiring the appropriate domain-specific knowledge needed to solve the tasks defined in that domain, are primarily modeling tasks*. The modeling defines the domain's *ontology*—a theory of entities, their properties, and their relations in the domain—and maps that ontology into knowledge roles that exist in the problem-solving methods chosen for performing the particular required tasks [20]. Problem-solving methods are highly reusable, and can be used to model and perform real-world tasks efficiently, by committing to a method-specific ontology [15]. The PROTÉGÉ-II project [15, 39, 55] is an example of development of a library of such reusable, domain-independent, problem-solving methods. These problem-solving *methods* solve *tasks*—a set of inputs and outputs and semantic constraints over their relationships. Methods decompose tasks into *subtasks*, which can be solved by other methods or by *mechanisms*, which are defined as nondecomposable methods [15]. Methods and mechanisms have a *control structure* and a set of *knowledge roles* that are instantiated by domain-specific knowledge.

1.1. A guide to this paper

In Sections 1.2 and 1.3, we present a brief, high-level overview of the temporal abstraction task and of our knowledge-based approach to solving that task. In Section 2.1, we define in technical detail the task-specific *ontology* of the knowledge-based temporal abstraction method. In Section 2.2, using that ontology, a knowledge level overview is presented of the five tasks defined by that method, and of five knowledge-based mechanisms that perform these tasks. The description of these mechanisms and the knowledge they require is presented in detail in Sections 2.3–2.7. In Section 3, we discuss briefly the *RÉSUMÉ* system, an implementation of our knowledge-based framework, and its application to several different clinical and engineering domains: guideline-based care, monitoring of children's growth, therapy of patients who have insulin-dependent diabetes, and monitoring of traffic control. Section 4 presents a higher-level, unified summary and discussion of the framework, its computational advantages,

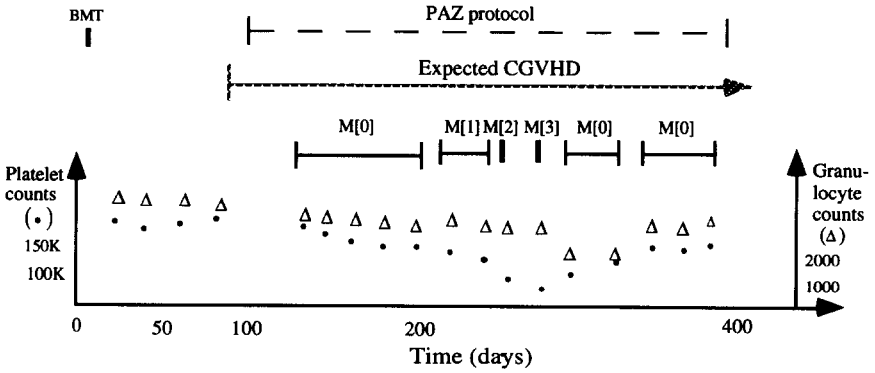

Fig. 1. Temporal abstraction of Platelet and granulocyte values during administration of a prednisone/azathioprine (PAZ) clinical protocol for treating patients who have chronic graft-versus-host disease (CGVHD). The time line starts with a bone-marrow transplantation (BMT) external event. The Platelet and granulocyte count parameters and the PAZ and BMT external event are typical inputs. The abstraction and context intervals are typically part of the output. • = Platelet counts; Δ = granulocyte counts; dashed interval = event; shaded arrow = open context interval; ⊢⊣ = closed abstraction interval; $M[n]$ = myelotoxicity (Bone-marrow toxicity) grade n.

its relationship to other temporal abstraction frameworks, and its implications. Section 5 contains our final remarks.

1.2. The temporal abstraction task: an informal overview

The *temporal abstraction* (*TA*) task can be viewed informally as a type of a generic *interpretation task*: given a set of time-stamped data, external events, and abstraction goals, produce abstractions of the data that interpret past and present states and trends and that are relevant for the given set of goals.

For instance, in clinical domains, a final diagnosis is not always the main goal. What is often needed is a coherent intermediate level interpretation of the relationships between data and events, and among data, especially when the overall context (e.g., a major diagnosis) is known. The goal is then to abstract the data into higher-level concepts useful for one or more tasks (e.g., planning of therapy or summarization of a patient's record). Thus, the goal might be to create, from time-stamped input data, interval-based temporal abstractions, such as "Grade II anemia for 3 weeks in the context of administration of the drug AZT as part of clinical therapy protocol CCTG-522" and more complex patterns, involving several intervals (Fig. 1).

A method solving the TA task encounters several conceptual and computational problems:

(1) the input parameter values might be of several data types and at various abstraction levels (e.g., Blood-glucose level = 192 mg/100cc; Glucose state = HIGH), and so might be the user's queries;
(2) input data might arrive out of temporal order, and existing interpretations must be revised nonmonotonically;

(3) several alternate interpretations might need to be maintained and followed over time;
(4) parameters have domain-specific and context-specific temporal properties, such as expected persistence of measured values; and
(5) acquisition of knowledge from domain experts and maintenance of that knowledge should be facilitated.

The method should enable *reusing* its *domain-independent* knowledge for solving the TA task in other domains, and enable *sharing* of the *domain-specific* knowledge with other tasks in the same domain.

1.3. Knowledge-based temporal abstraction theory: an informal overview

The approach that we present for performing the TA task employs an inference structure and related required knowledge that are specific to the task of abstracting higher-level concepts from time-stamped data in knowledge-based systems, but are independent of any particular domain. The theory underlying the method is specified in a general, domain-independent way by a model of time, events (e.g., administration of a drug), parameters (e.g., Platelet count), and the data interpretation contexts that these entities create (e.g., a period of chemotherapy effects), by a *knowledge-based temporal abstraction* (*KBTA*) *theory*.

Using the KBTA *theory*, we define the *KBTA method*. The KBTA method decomposes the *task* of abstracting higher-level, interval-based abstractions from input data into five *subtasks* (Fig. 2). The five subtasks are performed by five domain-independent *TA mechanisms*. The TA mechanisms depend on four domain-specific *knowledge types*.

The five TA mechanisms are defined in a formal, uniform, explicit way, such that the *knowledge* needed to instantiate them in any particular domain and task can be parameterized and acquired from domain experts manually, with automated tools, or by other methods (e.g., machine learning). The domain-specific knowledge required for instantiating the TA mechanisms is organized as four separate *TA knowledge types* (or categories) to emphasize the nature of the knowledge contained in each category and the *role* that each knowledge type plays in the reasoning performed by each mechanism.

2. Knowledge-based temporal abstraction theory

To describe the nature of the KBTA method and mechanisms, we start by defining a formal knowledge-based model of the TA task. The formal model will enable us to define clearly the inputs and outputs of the five subtasks defined by the KBTA method and of the five mechanisms performing these subtasks. We then describe each of the five TA mechanisms, the task that each mechanism performs, and the formal specifications of the mechanism's knowledge requirements.

The KBTA method and the five TA mechanisms performing the five subtasks posed by that method make several assumptions with respect to the underlying model of the

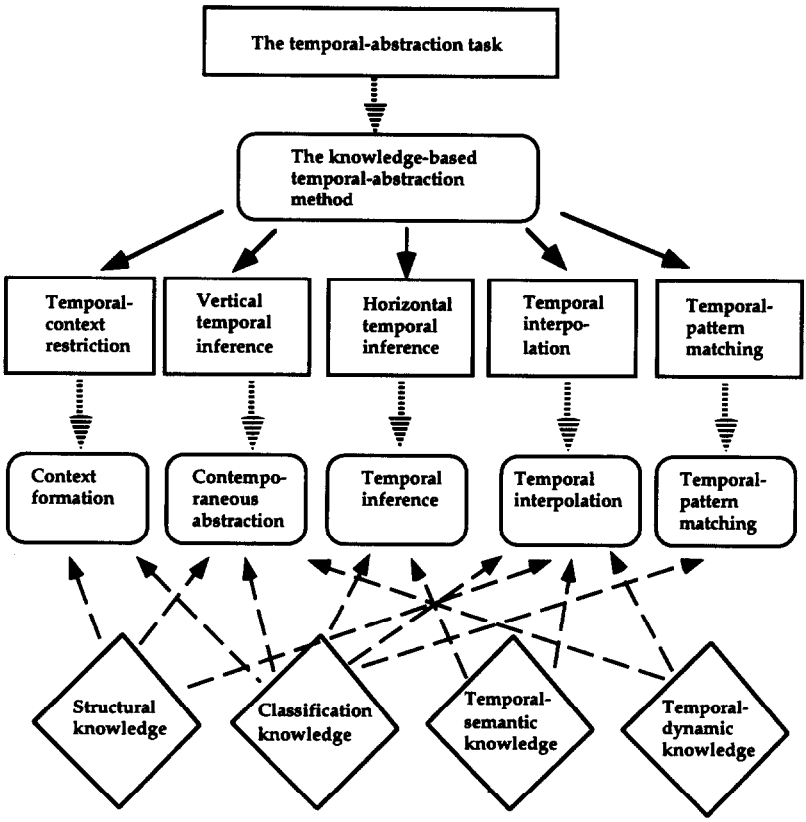

Fig. 2. The knowledge-based temporal abstraction (KBTA) method. The TA task is decomposed by the KBTA method into five subtasks. Each subtask can be performed by one of five TA mechanisms. The TA mechanisms depend on four domain- and task-specific knowledge types. Rectangle = task; ellipse = method or mechanism; diamond = knowledge type; arrow = DECOMPOSED-INTO relation; shaded arrow = PERFORMED-BY relation; dashed arrow = USED-BY relation.

domain. In particular, they assume a certain structure of time and of the propositions that can be interpreted over time. Thus, the TA mechanisms assume a task-specific *TA ontology*—a theory of what entities, relations, and properties exist in any particular domain from the point of view of the TA task and, in particular, of the KBTA method. Our goals in this section are to define ontologies of *events* (e.g., drug administrations), of *parameters* (e.g., Blood-glucose values), and of *interpretation contexts* within which parameters will be interpreted and abstracted into higher-level concepts, and to define the relationships among these ontologies. Together, these ontologies compose the domain's TA ontology. The TA task, the KBTA method, and especially the TA mechanisms can then be defined formally in terms of the TA ontology. It is therefore necessary to examine carefully the individual entities of the TA ontology.

2.1. The temporal abstraction ontology

In this section, we define formally the terms used by the KBTA method regarding time and propositions that are interpreted over time, and provide some intuition regarding the siginificance of these terms and the relations between them. Informally, the temporal model used by the KBTA theory includes both time *intervals* and time *points*. Time points are the basic temporal primitives, but propositions, such as occurrence of events and existence of parameter values, can be interpreted (in the logical sense) only over time intervals. All propositions are *fluents* [34], and, in our model, must be interpreted over a particular time period (e.g., the value of the Temperature parameter at time $[t,t]$).

Notation. The various types of KBTA theory entities and propositions can be seen as logical *sorts*. The set of symbols denoting each sort is usually clarified immediately; thus, the notation "a parameter $\pi \in \Pi$" means that each individual parameter, denoted by π, will be a member of a new set of symbols Π (denoting only parameters). Other conventions are used; for instance, relation names (e.g., PART-OF) are denoted by small capital letters. Parameter names (e.g., Hemoglobin) are denoted by initial capital letters; a particular value of a parameter (e.g., LOW) is denoted by small capital letters. Semantic properties (e.g., *concatenable*) are denoted by italics.

The KBTA theory defines the following set of entities:

(1) The basic time primitives are *time stamps*, $T_i \in T$. Time stamps are structures (e.g., dates) that can be mapped, by a time standardization function $f_s(T_i)$, into an integer amount of any element of a set of predefined *temporal granularity units* $G_i \in \Gamma$ (e.g., DAY). A *zero point* time stamp must exist, with relationship to which time stamps are measured in the G_i units. (Intuitively, the 0 point might be grounded in each domain to different absolute, "real-world", time points: the patient's age, the start of the therapy, the first day of the twentieth century.) The domain must have a time unit G_0 of the lowest granularity (e.g., SECOND); there must exist a mapping from any integer amount of granularity units G_i into an integer amount of G_0. (The time unit G_0 can be a task-specific choice.) A finite negative or positive integer amount of G_i units is a *time measure*.

The special symbols $+\infty$ and $-\infty$ are both time stamps and time measures, denoting the furthest future and the most remote past, respectively. Any two time stamps must belong to either a precedence relation or an equivalence relation defined on the set of pairs of time stamps. The precedence relation corresponds to a temporal order; the equivalence relation denotes temporal equivalence for the domain. The $-\infty$ time stamp precedes any other time stamp; the $+\infty$ time stamp follows (is preceded by) all other time stamps. Subtraction of any time stamp from another must be defined and should return a time measure. Addition or subtraction of a time measure to or from a time stamp must return a time stamp.

Time stamps resemble the result of applying a *date function* to McDermott's [35] *states*. However, McDermott maps states (snapshots of the world) to the real numbers line. The discreteness of time stamps is due to the natures of the TA *task* and of

that task's *input* (i.e., time-stamped data). The TA mechanisms do not rely on that discreteness. The TA mechanisms *do* depend on the temporal computational properties. Temporal uncertainty is modeled by the *processing* of time-stamped data, rather than by the time stamps.

(2) A *time interval* I is an ordered pair of time stamps representing the interval's end points: $[I.\text{start}, I.\text{end}]$. *Time points* T_i are therefore represented as zero length intervals where $I.\text{start} = I.\text{end}$. Propositions can be interpreted only over time intervals.

As will be clear from the rest of the definitions of the TA ontology, the set of points included in a time interval depends on the proposition interpreted over that interval. A time interval can be closed on both sides (in the case of state-type and pattern-type parameter propositions and interpretation contexts), open on both sides (in the case of gradient- and rate-type abstractions), or closed on the left and open on the right (in the case of event propositions).

(3) An *interpretation context* $\xi \in \Xi$ is a proposition. Intuitively, it represents a state of affairs that, when interpreted (logically) over a time interval, can change the interpretation (abstraction) of one or more parameters within the scope of that time interval. Thus, "the drug insulin has an effect on Blood glucose during this interval" changes the interpretation of the state of the Blood-glucose level, by suggesting a different definition of the value LOW. IS-A and SUBCONTEXT relations are defined over the set of interpretation contexts. *Basic* interpretation contexts are atomic propositions. An interpretation context in conjunction with one of its subcontexts can create a *composite interpretation context*. Composite interpretation contexts are interpretation contexts. Formally, the structure $\langle \xi_i, \xi_j \rangle$ is an interpretation context, if the ordered pair (ξ_j, ξ_i) belongs to the SUBCONTEXT relation (i.e., ξ_j is a subcontext of ξ_i). In general, if the structure $\langle \xi_1, \xi_2, \ldots, \xi_i \rangle$ is a (composite) interpretation context, and the ordered pair (ξ_j, ξ_i) belongs to the SUBCONTEXT relation, then the structure $\langle \xi_1, \xi_2, \ldots, \xi_i, \xi_j \rangle$ is a (composite) interpretation context. *Generalized* and *nonconvex interpretation contexts* are defined in Section 2.3.2.

Intuitively, composite interpretation contexts allow us to define increasingly specific contexts (e.g., a specific drug regimen that is a part of a more general clinical protocol), or combinations of different contexts, for interpretation of various parameters. Composite interpretation contexts represent a combination of *contemporaneous* interpretation contexts whose *conjunction* denotes a new context that has significance for the interpretation of one or more parameters.

(4) A *context interval* is a structure $\langle \xi, I \rangle$, consisting of an interpretation context ξ and a temporal interval I. Intuitively, a context interval represents an interpretation context interpreted over a time interval; the interpretation of one or more parameters is different within the temporal scope of that interval. Thus, the effects of chemotherapy form an interpretation context that can hold over several weeks, within which the values of hematological parameters might be abstracted differently.

(5) An *event proposition* $e \in E$ (or an *event*, for short, when no ambiguity exists) represents the occurrence of an external volitional action or process, such as the administeration of a drug (as opposed to a measurable datum, such as temperature). Events have a series a_i of *event attributes* (e.g., dose) and a corresponding series v_i of *attribute values*. (Typically, events are controlled by a human or an automated agent,

and thus neither are they measured data, nor can they be abstracted from the other input data.)

An IS-A hierarchy (in the usual sense) of *event schemata* (or event types) exists. Event schemata have a list of attributes a_i, where each attribute has a domain of possible values V_i, but do not necessarily contain any corresponding attribute values. Thus, an *event proposition* is an event schema in which each attribute a_i is mapped to some value $v_i \in V_i$. A PART-OF relation is defined over the set of event schemata. If the pair of event schemata (e_i, e_j) belongs to the PART-OF relation, then event schema e_i can be a *subevent* of an event schema e_j (e.g., a clinical protocol event can have several parts, all of them medication events).

(6) An *event interval* is a structure $\langle e, I \rangle$, consisting of an event proposition e and a time interval I. The time interval I represents the duration of the event.

(7) A *parameter schema* (or a *parameter*, for short) $\pi \in \Pi$ is, intuitively, a measurable aspect or a describable state of the world, such as a patient's temperature. Parameter schemata have various *properties*, such as a domain V_π of possible symbolic or numeric values, measurement units, and a measurement scale (which can be one of NOMINAL, ORDINAL, INTERVAL, or RATIO, corresponding to the standard distinction in statistics among types of measurement).[2] Not all properties need have values in a parameter schema. An IS-A hierarchy (in the usual sense) of parameter schemata exists. The combination $\langle \pi, \xi \rangle$ of a parameter π and an interpretation context ξ is an *extended parameter schema* (or an *extended parameter*, for short). Extended parameters are parameters (e.g., Blood glucose in the context of insulin action, or Platelet count in the context of chemotherapy effects). Note that an extended parameter can have properties, such as possible domains of value, that are different from that of the original (nonextended) parameter (e.g., in a specific context, a parameter might have a more refined set of possible values). Extended parameters also have a special property, a *value* $v \in V_\pi$. Values often are known only at runtime.

Intuitively, parameters denote either input (usually raw) data, or any level of abstraction of the raw data (up to a whole pattern). For instance, the Hemoglobin level is a parameter, the White blood cell count is a parameter, the Temperature level is a parameter, and so is the Bone-marrow toxicity level (which is abstracted from Hemoglobin and other parameters).

The combination of a parameter, a parameter value, and an interpretation context—that is, the tuple $\langle \pi, v, \xi \rangle$ (i.e., an extended parameter and a value)—is called a *parameter proposition* (e.g., "the state of Hemoglobin has the value LOW in the context of therapy by AZT"). A mapping exists from all parameter propositions and the parameter properties of their corresponding parameter (or extended parameter) schema into specific property values.

[2] Nominal-scale parameters have values that can be listed, but that cannot be ordered (e.g., color). Ordinal-scale parameters have values that can be ordered, but the intervals among these values are not meaningful by themselves and are not necessarily equal (e.g., military ranks). Interval-scale parameters have scale with meaningful, comparable intervals, although a ratio comparison is not necessarily meaningful (e.g., temperature measured on a Celsius scale). Ratio-scale parameters have, in addition to all these properties, a fixed zero point (e.g., height); thus, a ratio comparison, such as "twice as tall", is meaningful regardless of the height measurement unit.

Much of the knowledge about abstraction of higher-level concepts over time depends on knowledge of specific parameter proposition properties, such as persistence over time of a certain parameter with a certain value within a particular context. Different TA mechanisms typically require knowledge about different parameter properties of the same parameter propositions.

- *Primitive parameters* are parameters that play the role of raw data in the particular domain in which the TA task is being solved. They cannot be inferred by the TA process from any other parameters (e.g., laboratory measurements). They can appear in only the input of the TA task.
- *Abstract parameters* are parameters that play the role of intermediate concepts at various levels of abstraction; these parameters can be part of the output of the TA task, having been *abstracted* from other parameters and events, or they may be given as part of the input (e.g., the value of the state of Hemoglobin is MODERATE_ANEMIA). There is an ABSTRACTED-INTO relationship between one or more parameters and an abstract parameter. Each pair of parameters that belongs to an ABSTRACTED-INTO relation represents only one abstraction step; that is, the ABSTRACTED-INTO relation is not transitive. It is also areflexive and antisymmetric.
- *Constant parameters* are parameters that are considered atemporal in the context of the particular interpretation task that is being solved, so that their values are not expected to be time dependent (e.g., the patient's gender, the patient's address, the patient's father's height). There are few, if any, truly constant parameters. (In the discussion of temporal semantic inference in Section 2.5.1, we suggest a representation of constants as fluents with a particular set of temporal semantic inferential properties, thus removing, in effect, the traditional distinction between temporal and atemporal variables.) It is often useful to distinguish between (a) *internal* (or *runtime*) *constants*, which are specific to the particular case being interpreted and appear in the runtime input (e.g., the patient's date of birth), and (b) *external* (or *static*) *constants*, which are inherent to the overall task, and are typically prespecified or appear in the domain ontology (e.g., the population's distribution of heights).

(8) *Abstraction functions* $\theta \in \Theta$ are unary or multiple argument functions from one or more parameters to an abstract parameter. The "output" abstract parameters can have one of several *abstraction types* (which are equivalent to the abstraction function used). We distinguish among at least three basic abstraction types: *state*, *gradient*, and *rate*. (Other abstraction functions and therefore types, such as *acceleration* and *frequency*, can be added.) These abstraction types correspond, respectively, to a classification (or computational transformation) of the parameter's value, the sign of the derivative of the parameter's value, and the magnitude of the derivative of the parameter's value during the interval (e.g., LOW, DECREASING, and FAST abstractions for the Platelet count parameter). The state abstraction is always possible, even with qualitative parameters having only a nominal scale (e.g., different values of the Skin-color parameter can be mapped into the state abstraction value RED); the gradient and rate abstractions are meaningful for only those parameters that have at least an ordinal scale (e.g., degrees of physical fitness) or an interval scale (e.g., Temperature), respectively. In addition, a special type of abstraction function (and type) is *pattern*: the abstract parameter is

defined as a temporal pattern of several other parameters (e.g., a QUIESCENT-ONSET pattern of chronic graft-versus-host disease).

The θ abstraction of a parameter schema π is a new parameter schema $\theta(\pi)$—a parameter different from any of the arguments of the θ function (for example, STATE(Hemoglobin), which we usually write as Hemoglobin_state). This new parameter has its own domain of values and other properties (e.g., scale), typically different from those of the parameters from which it was abstracted. It can also be abstracted further (e.g., GRADIENT(STATE(Hemoglobin))).

Statistics such as *minimum*, *maximum*, and *average value* are not abstraction types in this ontology. Rather, these statistics are *functions* on parameter *values* that return simply a *value* of a parameter, possibly during a time interval, often from the domain of the original parameter (e.g., the minimum Hemoglobin value within a time interval I can be 8.9gr/100cc, a value from the domain of Hemoglobin values), rather than a parameter *schema*, which can have a new domain of values (e.g., the Hemoglobin_state can have the value INCREASING).

(9) A *parameter interval* is a tuple $\langle \pi, \nu, \xi, I \rangle$, where $\langle \pi, \nu, \xi \rangle$ is a parameter proposition and I is a time interval. If I is in fact a time point (i.e., I.start = I.end), then we can also refer to the tuple as a *parameter point*. Intuitively, a parameter interval denotes the value ν of parameter π in the context ξ during time interval I. The value of parameter π at the beginning of interval I is denoted as I.start.π, and the value of parameter π at the end of interval I as I.end.π.

(10) An *abstraction* is a parameter interval $\langle \pi, \nu, \xi, I \rangle$, where π is an abstract parameter. If I is in fact a time point (i.e., I.start = I.end), we can also refer to the abstraction as an *abstraction point*; otherwise, we can refer to it as an *abstraction interval*.

(11) An *abstraction goal* $\psi \in \Psi$ is a proposition that denotes a particular goal or intention that is relevant to the TA task during some interval (e.g., diagnosis).

(12) An *abstraction goal interval* is a structure $\langle \psi, I \rangle$, where ψ is an abstraction goal and I is a temporal interval. Intuitively, an abstraction goal interval represents the fact that an *intention* holds or that a TA *goal* (e.g., the goal of monitoring AIDS patients) should be achieved during the time interval over which it is interpreted. An abstraction goal interval is used for creating correct interpretation contexts for interpretation of data.

(13) *Induction of context intervals*: intuitively, context intervals are inferred dynamically (at runtime) by certain event, parameter, or abstraction goal propositions being true over specific time intervals. The contexts interpreted over these intervals are said to be *induced* by these propositions (e.g., by the event "administration of 4 units of regular insulin"). Certain predefined temporal constraints must hold between the inferred context interval and the time interval over which the inducing proposition is interpreted. For instance, the effect of insulin with respect to changing the interpretation of Blood-glucose values might start at least 30 minutes after the start of the insulin administration and might end up to 8 hours after the end of that administration. Two or more context-forming propositions induce a *composite interpretation context*, when the temporal spans of their corresponding induced context intervals intersect, if the interpretation contexts that hold during these intervals belong to the SUBCONTEXT relation.

Formally, a *dynamic induction relation of a context interval* (*DIRC*) is a relation on propositions and time measures, in which each member is a structure of the form

$\langle \xi, \phi, ss, se, es, ee \rangle$ (see explanation later in this subsection). The symbol ξ is the interpretation context that is induced. The symbol $\phi \in P$ is the *inducing proposition*: an event, an abstraction goal, or a parameter proposition. (An event schema is also allowed, as shorthand for the statement that the relation holds for any event proposition representing an assignment of values to the event schema's arguments.) Each of the other four symbols denotes a time measure or the wildcard symbol $*$. A proposition ϕ that is an inducing proposition in at least one DIRC is a *context-forming proposition.*

We can use the knowledge represented by DIRCs to infer new context intervals at runtime. Intuitively, the inducing proposition is assumed, at runtime, to be interpreted over some time interval I with known end points. The four time measures denote, respectively, the temporal distance *ss* between the *start* point of I and the *start* point of the induced context interval, the distance *se* between the *start* point of I and the *end* point of the induced context interval, the distance *es* between the *end* of I and the *start* point of the context interval, and the distance *ee* between the *end* point of I and the *end* point of the induced context interval (see Fig. 4). Note that, typically, only two values are necessary to define the scope of the inferred context interval (more values might create an inconsistency), so that the rest can be undefined (i.e., they can be wildcards, which match any time measure), and that sometimes only one of the values is a finite time measure (e.g., the *ee* distance might be $+\infty$). Note also that the resultant context intervals do not have to span the same temporal scope over which the inducing proposition is interpreted. Section 2.3.1 discusses the advantages of the DIRC representation, which separates propositions from the contexts that they induce.

We now can clarify exactly which basic propositions and relations exist in the ontology of the KBTA theory and problem-solving method: abstraction goals, event propositions, parameter propositions, interpretation contexts, and DIRCs.

The set of all the relevant event schemata and propositions in the domain, their attributes, and their subevents forms the domain's *event ontology.* The set of all the potentially relevant contexts and subcontexts of the domain, whatever their inducing proposition, defines a *context ontology* for the domain. The set of all the relevant parameters and parameter propositions in the domain and their properties forms the domain's *parameter ontology.* These three ontologies, together with the set of abstraction goal propositions and the set of all DIRCs, define the domain's *TA ontology.*

To complete the definition of the TA task, we assume the existence of a set of *temporal queries*, expressed in a predefined *TA query language* that includes constraints on parameter values and on relations among start point and end point values among various time intervals and context intervals. That is, a temporal query is a set of constraints over the components of a set of parameter and context intervals, using the parameter and context ontologies. Intuitively, the TA language is used (a) to define the relationship between a pattern-type abstraction and its defining parameter intervals, and (b) to ask arbitrary queries about the result of the TA inference process.

The *TA task* solved by the KBTA method is defined as follows: given at least one abstraction goal interval $\langle \psi, I \rangle$, a set of event intervals $\langle e_j, I_j \rangle$, and a set of parameter intervals $\langle \pi_k, \nu_k, \xi_k, I_k \rangle$ (ξ_k might be the empty interpretation context in the case of primitive parameters), and the domain's temporal abstraction ontology, produce an *interpretation*—that is, a set of context intervals $\langle \xi_n, I_n \rangle$ and a set of (new) abstractions

$\langle \pi_m, \nu_m, \xi_m, I_m \rangle$ (see Fig. 1)—such that the interpretation can answer any temporal query about all the abstractions derivable from the transitive closure of the input data and the domain knowledge.

Note that, for output abstractions to be meaningful, the domain knowledge representing the relationship between primitive and abstract parameters, such as ABSTRACTED-INTO relations, must be part of the input.

Notation. In the rest of this paper we often use a *reified* representation [51], in which the proposition "the (gradient of the) Hemoglobin value is decreasing during interval I_1 in the context ξ" is stated formally as "True(I_1, DECREASING(Hemoglobin_gradient, ξ))", thus separating the temporal component of the sentence from the atemporal (parameter proposition) component. For simplicity, we often refer to this proposition as DECREASING(I_1, Hemoglobin_gradient, ξ), or even as DECREASING(Hemoglobin), when no confusion is likely. All should be read as "the value of the gradient abstraction of the Hemoglobin parameter (i.e., the Hemoglobin_gradient parameter) is DECREASING (during the relevant context and over the relevant time interval)".

2.2. The temporal abstraction tasks and mechanisms

As explained in Section 1.1, the KBTA *method* decomposes the TA *task* into five subtasks, each of which can be performed by a TA *mechanism* (see Fig. 2). Each mechanism has its own domain-specific knowledge requirements and its control structure. This structure is similar to the one employed in the PROTÉGÉ-II framework for development of knowledge-based systems [15]. Using the TA ontology of Section 2.1, we can present informally the five subtasks into which the KBTA method decomposes the TA task, and then describe in detail the reasoning elements of the KBTA framework—the five TA mechanisms that solve each of the TA subtasks. The five TA subtasks are:

(1) *Temporal context restriction*, which involves creation of *context intervals* from a set of input abstraction goal, parameter and event intervals, thus limiting the scope of inference.
(2) *Vertical temporal inference*, which involves creation of *abstractions* by inference from contemporaneous parameter propositions into abstract parameter propositions.
(3) *Horizontal temporal inference*, which involves creation of *parameter intervals* by performance of inference on parameter propositions of the same parameter (e.g., Hemoglobin_state) and interpretation context (e.g., AIDS therapy), interpreted over intervals that are not disjoint, but that differ in temporal span.
(4) *Temporal interpolation*, which involves creation of *parameter intervals* by joining of disjoint parameter points or parameter intervals over which are interpreted propositions of the same parameter and interpretation context.
(5) *Temporal pattern matching*, which involves creation of *abstraction intervals* by matching of patterns—constraining parameter values and time interval values and relations—over possibly disjoint parameter intervals, associated with parameter propositions of various parameters.

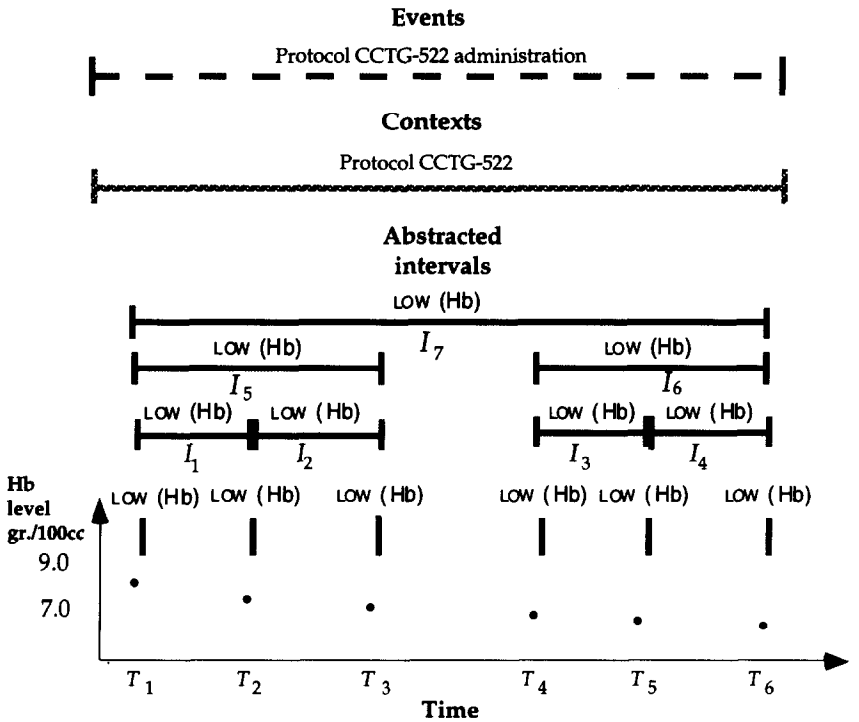

Fig. 3. Processing of parameter points and intervals by the temporal abstraction mechanisms. The (primitive) parameter points that hold at times T_1 and T_2 are abstracted into two abstraction points, over which a LOW(Hb) state abstraction is interpreted, by contemporaneous abstraction; these point abstractions are joined into a LOW(Hb) interval abstraction I_1 by temporal interpolation. Abstractions I_1 and I_2 are joined by temporal inference into the longer LOW(Hb) interval abstraction I_5, as are I_3 and I_4 into I_6. Interval abstractions I_5 and I_6 are joined into a LOW(Hb) interval abstraction I_7 by temporal interpolation. A DECREASING(Hb) gradient abstraction during interval I_7 can be computed similarly, if all steps in the computation are permitted by the domain's temporal abstraction properties. Hb = Hemoglobin; • = Hb levels; dashed interval = event interval; shaded interval = closed context interval; ⊢⊣ = closed abstraction interval. The interpretation context "Protocol CCTG-522" was induced by the event "Protocol CCTG-522 administration" and is contemporaneous with that particular event.

The five TA *mechanisms* perform the five TA subtasks, given the domain's TA ontology (see Fig. 2). We can define their input and output values using the TA ontology (Fig. 3):

(1) The *context-forming mechanism* creates context intervals, given abstraction goal intervals, event intervals, abstractions, existing context intervals, and the domain's TA ontology (in particular, DIRCs).

(2) The *contemporaneous abstraction mechanism* creates abstraction points and intervals of type STATE, given contemporaneous parameter points, parameter intervals, and context intervals and the parameter ontology (including, in particular, the ABSTRACTED-INTO relation).

(3) The *temporal inference mechanism* performs two subtasks. *Temporal horizontal inference* involves inferring a parameter proposition (e.g., NONDEC) from two given parameter propositions (e.g., SAME and INC) when the temporal elements over which the parameter propositions are interpreted are disjoint. *Temporal semantic inference* involves inferring a parameter proposition given a parameter interval and a time interval (e.g., a subinterval of that interval), interpreting a parameter proposition over the time interval, thus creating another parameter interval. The parameter and the interpretation context parts of the input propositions must be the same. The input also includes the parameter ontology.

(4) The *temporal interpolation mechanism* joins disjoint parameter points or parameter intervals, both with the same parameter and interpretation context. The input also includes the parameter ontology. The output is a parameter interval.

(5) The *temporal pattern matching mechanism* creates new abstraction points and intervals of type PATTERN, by matching patterns over disjoint parameter intervals or parameter points. The patterns include restrictions on parameter values and time interval values and relations. The parameters and interpretation contexts might differ among the involved parameter propositions. Pattern classifications are represented in the parameter ontology, which is included in the input. The input might also include a set of runtime temporal queries.

The TA mechanisms require knowledge (often, particular parameter properties) that is more specific than are the general knowledge categories used for defining the TA task. We distinguish among four domain *knowledge types* used by the TA mechanisms (see Fig. 2):

(1) *Structural knowledge* (e.g., IS-A, ABSTRACTED-INTO, and PART-OF relations in the domain; parameter properties such as measurement scale and units).

(2) *Classification knowledge* (e.g., classification of Hemoglobin value ranges into LOW, HIGH, VERY HIGH; classification of temporal patterns).

(3) *Temporal semantic knowledge* (e.g., the *downward hereditary* property [51] allows us to infer a parameter proposition over any subinterval of an interval over which it is true; the *concatenable* property allows us to concatenate adjacent equal parameter intervals).

(4) *Temporal dynamic knowledge* (e.g., persistence of the truth value of LOW(Hemoglobin) when the Hemoglobin parameter is not measured; minimal values of a significant change).

In Sections 2.3–2.7, we describe the five TA mechanisms, and point out the precise knowledge that each mechanism needs to be *instantiated* in (applied to) a new domain. This knowledge is usually a specialization and combination of the four knowledge types (see Fig. 2).

2.3. The context-forming mechanism

Abstractions are meaningful only within the span of a relevant *context interval*, such as administration of the drug AZT as part of a particular clinical protocol for therapy of AIDS. Context intervals create a relevant *frame of reference* for interpretation, and thus enable a TA mechanism both to compute the most specific abstractions for that context

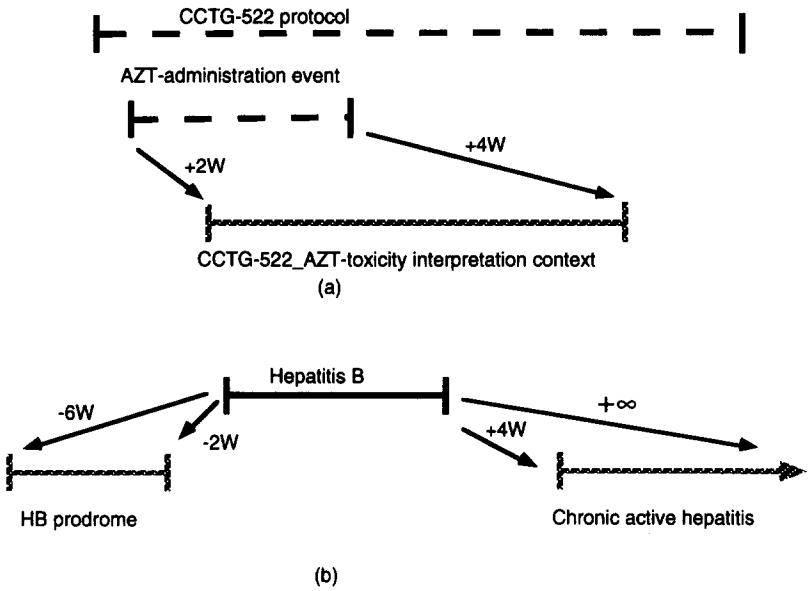

Fig. 4. Dynamic induction relations of context intervals (DIRCs). (a) An overlapping direct and prospective AZT toxicity interpretation context induced by the existence of an AZT administration event in the context of the CCTG-522 AIDS treatment experimental protocol. The interpretation context starts 2 weeks after the start of the inducing event, and ends 4 weeks after the end of the inducing event. Note that structural knowledge about the PART-OF relationship between the AZT administration subevent and the CCTG-522 protocol event and the SUBCONTEXT relationship between the two interpretation contexts induced by these events is also required to create the interpretation context of "AZT therapy toxicity within the CCTG-522 protocol". (b) Prospective (chronic active hepatitis complication) and retrospective (hepatitis B prodrome) interpretation contexts, induced by the external assertion or internal conclusion of a hepatitis B abstraction interval, a context-forming abstraction. Note the temporal constraints. Dashed interval = event interval; shaded interval = closed context interval; shaded arrow = open context interval; $\vdash\!\dashv$ = closed abstraction interval.

and to avoid computing abstractions for irrelevant contexts. Context intervals are created by the *context-forming mechanism.*

As intervals and several types of propositions that can *induce* these intervals (Fig. 4). Context intervals might be induced by the existence of an *abstraction goal interval*, such as "therapy of insulin-dependent diabetes", or by the existence of an *event interval*—that is, an external process or action, such as treatment in accordance with a particular clinical protocol. A context interval can also be induced by the existence of a *parameter interval* that includes a *context-forming* (see Section 2.1) parameter proposition $\langle \pi, \nu, \xi \rangle$; namely, the value ν of the parameter π, in the context ξ, is sufficiently important to change the frame of reference for one or more other parameters (e.g., the LOW value of the Hemoglobin_state abstract parameter in the context of protocol CCTG-522 might affect the interpretation of values of the Platelet parameter).

A *composite interpretation context* (see Section 2.1) can be composed by the context-forming mechanism from a conjunction of existing *basic* interpretation contexts that have a SUBCONTEXT relation. The composite interpretation context would be interpreted

over a context interval formed from a *temporal intersection* of two or more context intervals. For instance, a composite interpretation context is often induced by an *event chain*—a connected series of events $\langle e_1, e_2, \ldots, e_n \rangle$, where e_{i+1} is a subevent of e_i. In that case, the composite interpretation context would denote an interpretation context induced by the most specific subevent, such as administration of a particular drug as part of a certain protocol. The reason is that subevents of an event typically induce interpretation contexts that have a SUBCONTEXT relation to the interpretation context induced by the event. This knowledge can be used as a default in the context ontology, and can also be exploited during manual or automated knowledge acquisition, either for knowledge elicitation or for knowledge verification and cross-validation. In addition, interpretation contexts can also be formed by concatenation of two adjacent, or *meeting* [2] equal context intervals. (Interpretation contexts are assumed to be *concatenable*; see Section 2.5.)

Dynamic induction of context intervals by parameter propositions might lead to new interpretations of existing parameter intervals, thus potentially inducing new context intervals within which another or even the original parameter value (the input datum) might have new interpretations. However, we can show [43] that no contradictions or infinite loops can be generated by the context-forming process.

Observation 1. *The context-forming process has no oscillation cycles among different interpretations of the same parameter: the same parameter proposition can never be retracted and eventually reasserted.*

Proof. Parameter propositions are never retracted by the addition of a new interpretation context. Rather, a new interpretation is added to the set of true parameter propositions. This addition creates no contradiction, since the same parameter has two different interpretations within two different contemporaneous contexts. Thus, if a parameter proposition $\langle \pi, \nu_1, \xi_1 \rangle$ induces a new interpretation context ξ_2 over some interval, and within the scope of that interval the parameter π is interpreted to have another value, a new parameter proposition $\langle \pi, \nu_2, \xi_2 \rangle$ would simply be inferred and added to the set of true propositions. The existence of two contemporaneous parameter propositions different in parameter value creates no contradictions, since the parameter π—or some abstraction of π (say, state(π))—is interpreted within two different contexts and can thus have two different values at the same time. □

Observation 2. *The context-forming process is finite.*

Proof. The total number of different interpretation contexts that, potentially, can be inferred (including composite ones) is limited by an existing upper bound: the size of the context ontology and the number of potential *subcontext chains* (which can form composite contexts) of interpretation contexts that have SUBCONTEXT relations. Furthermore, for each parameter π, the number of possible induced context intervals is bound by the number of DIRCs in which a parameter proposition including π is an inducing proposition. Since Observation 1 ascertained that there are no loops either, the process must end for any finite number of input (interval-based) propositions. □

2.3.1. Advantages of explicit interpretation contexts and DIRCs

Explicit interpretation contexts, separate from the propositions inducing them and from abstractions using them, have significant conceptual and computational advantages for context-specific interpretation of time-stamped data.

(1) Since the four temporal measures of a DIRC, representing temporal constraints over an induced context interval with respect to the start time and the end time of the inducing proposition, can be positive, negative, or infinite time measures, the context interval induced by a context-forming proposition can have any one of Allen's [2] 13 binary temporal relations (e.g., BEFORE, AFTER, or OVERLAPS) to the time interval over which the inducing proposition is interpreted (see Fig. 4). Thus, a context-forming proposition interval can create a *context envelope* that might include, in addition to a *direct* context (concurrent with the temporal scope of the proposition's interval), *retrospective* context intervals (e.g., potential preceding symptoms of a disease), *prospective* context intervals (e.g., potential complications of a disease), or both (see Fig. 4). Intuitively, retrospective interpretation contexts represent a form of *abductive* reasoning (e.g., from effects to causes, such as preceding events), whereas prospective interpretation contexts represent a form of *deductive* reasoning (e.g., from an event to potential complications). (Note, however, that we infer only an interpretation context, rather than the actual abstraction.) The context-forming mechanism creates retrospective and prospective contexts mainly to enable the use of context-specific TA functions, such as the correct mapping functions related to ABSTRACTED-INTO relations and the relevant temporal persistence functions [43], that should not be considered in other contexts. Creation of explicit contexts enables the TA mechanisms to focus on the abstractions appropriate for particular contexts, such as potential consequences of a certain event, and to avoid unnecessary computations in other contexts. In addition, the ability to create dynamically retrospective contexts enables a form of *hindsight* [42], since the interpretation of *present* data can induce new interpretation contexts for the past and thus shed new light on *old* data.

(2) Since a context-forming proposition can be an inducing proposition in more than one DIRC, the *same proposition* can *induce* dynamically *several interpretation contexts*, in the past, the present, or the future, relative to the temporal scope of the interval over which it is interpreted. Thus, we can model, for instance, several potential effects of the same action, each of which creates a different interpretation context, or several inferences from the same temporal pattern, once detected.

(3) The *same interpretation context* (e.g., potential Bone-marrow toxicity) might be induced by *different propositions*, possibly even of different types and occurring over different periods (e.g., different types of chemotherapy and radiotherapy events). The domain's TA ontology would then be representing the fact that, within the particular interpretation context induced by any of these propositions (perhaps with different temporal constraints for each proposition), certain parameters would be interpreted in the same way (e.g., we can represent the properties of the Hemoglobin state parameter within the scope of a Bone-marrow toxicity context interval, without the need to list all the events that can lead to the creation of such a context interval). Thus, the separation of interpretation contexts from their inducing propositions also facilitates maintenance and reusability of the TA knowledge base.

(4) Since several context intervals, during which different interpretation contexts hold, can exist contemporaneously (that is, concurrently), it is possible to represent several abstraction intervals in which the *same abstract parameter* (e.g., the state abstraction of the Hemoglobin level parameter) has *different values* at the *same time*—one for each valid and relevant context (e.g., two contemporaneous abstractions, "LOW Hemoglobin state" in the context of having AIDS without complications, and "NORMAL Hemoglobin state" in the context of being treated by the drug AZT, which has expected side effects). Thus, the context-forming mechanism supports maintenance of several *concurrent views* of the abstractions in the resultant abstraction database, denoting several possible interpretations of the same data. Context-specific views are one reason that parameter propositions (including temporal pattern queries to the abstraction database) must include an interpretation context: a parameter value by itself (e.g., LOW) might be meaningless.

2.3.2. *Generalized and nonconvex interpretation contexts*

Additional distinctions important for the TA task are enabled by the explicit use of interpretation contexts and DIRCs. Usually, abstractions are specific to a particular interpretation context, and *cannot* be joined (by the temporal inference or temporal interpolation mechanisms) to abstractions in other interpretation contexts (e.g., two LOW(Hemoglobin_state) abstractions might denote different ranges in two different subcontexts of the same interpretation context induced by a chemotherapy protocol event). This restriction is reasonable, since the primary reason for having contexts is to *limit* the scope of reasoning and of the applicability of certain types of knowledge. A *simple interpretation context* is a *basic* (single) or a *composite* interpretation context. Our discussions till now concerned simple interpretation contexts.

It is both desirable and possible, however, to denote that, for certain classes of parameters, contexts, and subcontexts, the abstractions are *sharable* among two meeting context intervals (with different interpretation contexts). Such abstractions denote the same state, with respect to the task-related implications of the state, in all sharing contexts. For instance, two meeting LOW(Hemoglobin) abstractions might indeed denote different ranges in two different contexts, and the Hemoglobin_state parameter might even have only two possible values in one context, and three in the other, but the domain expert still might want to express that parameter intervals with a LOW value of the Hemoglobin_state abstraction can be joined meaningfully to summarize a particular hematological state of the patient. The sharable abstraction values would be defined within a new *generalized interpretation context* that is equivalent to neither of the two shared subcontexts (e.g., those induced by two different parts of the same clinical protocol), nor their parent context (e.g., the one induced by the clinical protocol itself, within which the Hemoglobin_state parameter might have yet another, default, LOW(Hemoglobin) range). This generalized context can be viewed as a generalization of two or more subcontexts of the parent interpretation context. For instance, Fagan's [17] *VM* system provided advice for managing patients dependent on a ventilator machine. VM created context-specific abstractions of clinical parameters by mapping the values of these parameters into the same set of values (e.g., ACCEPTABLE). These values could be aggregated regardless of the context

in which they were formed. In our framework, we would describe such a system as an extreme case in which *all* parameter values are sharable among *all* subcontexts.

Finally, note that we might want to abstract the state of a parameter such as Glucose in the preprandial (before meals) interpretation context, over two or more temporally *disjoint*, but semantically equivalent, preprandial interpretation contexts (e.g., the PRELUNCH and PRESUPPER interpretation contexts are both PREPRANDIAL interpretation contexts). We might even want to create such an abstraction within only a particular preprandial context (e.g., a PRESUPPER interpretation context), skipping intermediate preprandial contexts (e.g., PREBREAKFAST and PRELUNCH interpretation contexts). Making this interpolation is different from sharing abstractions in a generalized interpretation context, since the abstractions in this case were created within the *same* interpretation contexts, but the interpolation operation joining them needs to skip temporal gaps, including possibly context intervals over which different interpretation contexts hold. The output is a new type of a parameter interval—a *nonconvex interval*, as defined by Ladkin [29]. A LOW(Glucose) abstraction would be defined, therefore, within the *nonconvex context* of "prebreakfast episodes". Note that parameter propositions within such a nonconvex context will have different temporal semantic inference properties (see Section 2.5) than when the same parameter propositions are created within a simple, convex context. For instance, propositions will usually not be *downward hereditary* ([51]; see Section 2.5) in the usual sense, unless subintervals are confined to only the convex or nonconvex intervals that the nonconvex superinterval comprises (e.g., only morning times).

Thus, the interpretation context of a parameter proposition is a combination of simple, generalized, and nonconvex interpretation contexts. Assume that a Gen (generalize) operator returns the generalizing context parent (if it exists) of a parameter proposition in the parameter properties ontology. Assume that a Gen* operator, which generalizes the Gen operator, returns the least common generalizing context ancestor (if one exists) $\langle \pi, \nu, \xi_g \rangle$ of two parameter propositions $\langle \pi, \nu, \xi_1 \rangle$, $\langle \pi, \nu, \xi_2 \rangle$, in which the parameter π and the value ν are the same, but the interpretation context is different. Assume that an NC (nonconvex) operator returns the nonconvex context extension of a parameter proposition (if one exists). Then, the parameter proposition that represents the nonconvex join (over disjoint temporal spans) of two parameter propositions in which only the interpretation context is different can be represented as

$$\mathrm{NC}(\mathrm{Gen}^*(\langle \pi, \nu, \xi_1 \rangle, \langle \pi, \nu, \xi_2 \rangle)).$$

For instance, we first look for a generalizing interpretation context for Glucose_state abstractions in the PRELUNCH and PRESUPPER interpretation contexts—in this case the PREPRANDIAL one. Then, we represent the parameter proposition "low preprandial glucose state values" as the LOW value of the Glucose_state parameter in the nonconvex extension of the PREPRANDIAL interpretation context. We would interpret this proposition over some time interval to form a parameter interval. (Generalized and nonconvex interpretation contexts are a part of the context ontology; the corresponding extended parameter propositions are a part of the parameter ontology.)

2.3.3. Formation of contexts and the event and context ontologies

The context-forming mechanism assumes that the domain's ontology includes an *event ontology*—that is, a theory that represents the external events in that domain (e.g., protocols, medications, specific drugs), the relationships among them (e.g., a medication might have a PART-OF relationship with a protocol), and any DIRCs in which event schemata and propositions are the inducing proposition (see Section 2.1). The event ontology can be represented as a frame hierarchy with IS-A and PART-OF relations among frames.

The event ontology includes all relevant events, subevents, and event DIRCs, which the context-forming mechanism can use at runtime. The default event DIRC comprises a single DIRC representing a *direct* (i.e., contemporaneous) interpretation context whose name is the event's name—that is, the DIRC $\langle \text{event name}, \text{event name}, 0, *, 0, * \rangle$. In addition, the event ontology is used by the context-forming mechanism to disambiguate the PART-OF relationship of several coexisting events, and thus to decide when it is reasonable to form a new, more specialized, context when two or more events coexist (e.g., when both a CCTG-522 protocol event and an AZT event are detected), and which event is the subevent (and thus might induce a subcontext) of the other. Similar reasoning is useful also for acquiring a context ontology and for maintenance of a TA ontology.

The set of all the potentially relevant interpretation contexts and subcontexts of the domain and their properties defines a *context ontology* for the domain. The context ontology, like the parameter and event ontologies, can be represented as a frame hierarchy. The types of semantic links among context nodes in the context ontology include IS-A and SUBCONTEXT relations. The knowledge represented in the context ontology complements the knowledge represented in the parameter ontology and the event ontology and assists the context-forming mechanism in correctly forming composite context intervals from several contemporaneous context intervals. For instance, the interpretation context induced by an event or by one of the subevents does not necessarily bear the name of its inducing event, nor does it necessarily have the same temporal scope; the only indication for a SUBCONTEXT relationship exists in that case in the context ontology. Thus, an AZT drug administration subevent of the CCTG-522 clinical protocol event induces a AZT-TOXICITY interpretation context that has a different name and a different temporal scope (see Fig. 4), but that has a SUBCONTEXT relationship to the CCTG-522 interpretation context induced by the CCTG-522 protocol event.

The context ontology is also important for representing other types of knowledge. For instance, in the domain of monitoring therapy for insulin-dependent diabetes, the PREBREAKFAST interpretation context (induced by a morning meal) has an IS-A relationship to the more general PREPRANDIAL interpretation context (induced by any meal). This relation is represented explicitly only in the context ontology of the diabetes monitoring domain [48]. Finally, special interpretation contexts, such as *generalized contexts* and *nonconvex contexts*, appear explicitly in the context ontology.

The rest of the TA mechanisms *do not depend on the event and context ontologies*. These mechanisms assume the existence of context intervals and of interpretation con-

texts as part of the parameter propositions. The context-forming mechanism is thus the only *interface* with the representation of the domain's events and DIRCs, and shields the rest of the TA mechanisms from any need to know about external events, such events' structures, or the way such events induce interpretation contexts.

In summary, the knowledge required for forming context intervals is mainly *structural knowledge*, which comprises an ontology of event schemata and propositions, including the PART-OF relation; an ontology of parameter schemata and propositions; an ontology of interpretation contexts, including the SUBCONTEXT relation; and the set of abstraction goal propositions; and the set of all DIRCs, a special type of structural knowledge, based on an extension of an INDUCED-BY relation.

The rest of the TA mechanisms described in this section require various types of knowledge represented in the parameter ontology (see Section 3 for a particular representation scheme of the parameter ontology).

2.4. Contemporaneous abstraction

The contemporaneous abstraction mechanism accepts as input one or more parameter points or intervals and returns as output an abstraction point or interval of abstraction-type STATE (see Fig. 1). The resulting conclusion holds for the intersection of the input intervals. The time stamps of all the time intervals of the input parameter points or intervals must be equivalent (a domain- and task-specific definition). For instance, if the task has DAY as the lowest granularity level, parameters measured at different hours of the same day might be considered to have the same time stamp.

One type of knowledge required by the contemporaneous abstraction mechanism is *structural knowledge*, such as the ABSTRACTED-INTO relation (see Section 2.1). In addition, *temporal dynamic knowledge* about the dynamic behavior of the parameters involved might be required, such as expected *persistence* (validity) of a measurement both *before* and *after* that measurement was taken (see Section 2.6.2). Such temporal dynamic knowledge enables abstraction of primitive or abstract parameter values that have been recorded as valid at different times (e.g., the Hemoglobin level was measured on Tuesday, and the Platelet count is not known until Thursday, but a Bone-marrow state abstraction requires both).

Contemporaneous temporal abstraction is used for two related subtasks. The first is *classification* of the value of the parameter of an input parameter point or parameter interval. Examples include creation of HIGH, MEDIUM, or LOW Hemoglobin value state abstractions. The knowledge requirements include a list of allowed values of the resultant state abstraction, and range tables that map primitive or abstract parameter values into discrete state values. The classification is sensitive to the relevant interpretation context (e.g., a chemotherapy protocol event). The second subtask, *computational transformation*, is an extension of classification; it assumes a function that maps the values of one or more parameters into the value of another, abstract, parameter. For example, a transformation function might compute the percentage of white blood cells that are lymphocytes at a specific time point, and map that value into a predefined category. This category might depend on the context. Thus, computational transformation might use

additional parameters, might involve additional computation, and might require special function tables that are specific for the domain, the task, and the context.[3]

Both subtasks need *vertical classification knowledge*, a subtype of the classification knowledge type that is relevant for mapping contemporaneous parameter values into another, contemporaneous value of a new parameter. For either subtask, we can distinguish between *single parameter* contemporaneous abstraction, which maps a single parameter directly into its state abstraction values, and *multiple parameter* contemporaneous abstraction, which maps several parameters into one abstract value. (Section 2.6.1 discusses the significance of this distinction.)

In summary, the knowledge required for forming contemporaneous abstractions is an ontology of parameter schemata and propositions that includes

(1) *structural knowledge*, which includes the ABSTRACTED-INTO relation and appropriate interpretation contexts for applying the vertical classification knowledge;
(2) *classification knowledge*, consisting of vertical classification knowledge (i.e., the range classification tables and the computational transformation functions); and
(3) *temporal dynamic knowledge*, which comprises local persistence functions.

2.5. Temporal inference

The temporal inference mechanism involves logically sound inference regarding interpretation of a proposition or a combination of propositions over one or more temporal intervals. The temporal inference mechanism performs two related, well-defined, inference subtasks: *temporal semantic inference* and *temporal horizontal inference.*

2.5.1. Temporal semantic inference

Temporal semantic inference employs a set Φ of domain-specific temporal semantic properties of parameter propositions. These properties are an extension of Shoham's [51] temporal propositional properties and are exploited by a set of *temporal inference actions* performed by the temporal inference mechanism (actions can be seen as "minimechanisms", whose task is implicit). The effects of temporal inference actions depend on the precise input arguments and on the temporal semantic properties that apply to the input parameter propositions. The actions accept as input either one abstraction and a time (or context) interval, or two abstraction intervals. They return as output an abstraction interpreted over either the time (or context) interval, or a newly defined superinterval of the two abstraction intervals. The time intervals of the two input abstractions can have, in general, any of Allen's [2] 13 binary temporal relations except BEFORE and AFTER (i.e., there must be no gap between the intervals; the case where a temporal gap exists is discussed in Section 2.6—it is handled by the interpolation

[3] The computational transformation subtask is, in theory, a very general one, and might involve computing a given black-box function. Several measures are used in the RÉSUMÉ system to reduce the ambiguity of such functions and to increase the amount of potential automated reasoning possible regarding such functions [43]. Examples include the disciplined use, when possible, of classification tables with predefined semantics, and the explicit representation of ABSTRACTED-INTO relations and qualitative dependencies relations in the parameter ontology.

mechanism). The resulting conclusion holds for the intersection or the union of the intervals. In the case of one abstraction and a time interval, the relations include STARTS, FINISHES, DURING, and the inverses of these relations; the inferred conclusion would hold for the duration of the input time interval.

For instance, certain propositions, when known at interval I_1, can be inferred for every subinterval I_2 that is contained in I_1 (e.g., the characterization "patient is in coma"). These propositions have the *downward hereditary* temporal semantic property [51]. That is, the *downward hereditary* property of these propositions has the value TRUE. This property, however, is *not* true, for instance, in the case of the NONMONOTONIC gradient abstraction, for any parameter. Using the TA ontology, the downward hereditary temporal semantic inference action can be expressed as follows (universal quantifiers have been dropped):

Input: abstraction $\langle \pi, \nu_1, \xi, I_1 \rangle$, time interval I_2.
Conditions: I_2 during I_1;
$\langle \pi, \nu_1, \xi \rangle$ is DOWNWARD HEREDITARY;
$\neg[\langle \pi, \nu_2, \xi, I_2 \rangle, \nu_1 \neq \nu_2]$.
Conclude: abstraction $\langle \pi, \nu_1, \xi, I_2 \rangle$.

The interval I_2 is the temporal element of some other entity (e.g., an existing abstraction). The negative condition is checked to ensure consistency; if all other conditions succeed but this condition fails, the inference returns FALSE, an indication for a *contradiction* (this situation will be discussed later in this section).

To join two meeting abstraction intervals (e.g., two intervals previously classified as DECREASING for the gradient abstraction of some parameter) into one superinterval (e.g., a longer DECREASING abstraction of that parameter), we need another inferential property—namely, the *concatenable* property [51]. That is, we need a TRUE value for the *concatenable* property of the DECREASING value of the gradient abstraction of that parameter in that context. A parameter proposition is *concatenable* if, whenever it holds over two consecutive time intervals, it holds also over their union. The input for the inference action in this case includes two abstraction intervals over which the same parameter proposition holds; the output, if the inference is relevant, is the abstraction superinterval (see Fig. 3). (When the values of the same parameter are different for the two parameter propositions, we need *horizontal classification knowledge*, discussed later in this section.) For instance, when the proposition "the patient had HIGH blood pressure" is true over some interval as well as over another interval that that interval meets, then that proposition is true over the interval representing the union of the two intervals. Note that two meeting, but different, 9-month pregnancy episodes are *not* concatenable, since it is *not* true that the patient had an 18-month pregnancy episode. (The two *interpretation contexts* induced by the two separate episodes would be concatenable.)

Other common temporal semantic properties for parameter propositions, originally defined by Shoham [51] for general propositions, are *gestalt* and *solid*. A parameter proposition is *gestalt* if it never holds over two intervals, one of which properly contains the other (e.g., the interval over which the proposition "the patient was in a coma for exactly 2 weeks" is true cannot contain any subinterval over which that proposition is also true). A parameter proposition is *solid* if it never holds over two properly over-

lapping intervals (e.g., "the patient received a *full* course of the current chemotherapy protocol, *from start to end*" cannot hold over two different, but overlapping intervals). In both cases, the input to the inference action includes two abstraction intervals, and the conclusion is simply FALSE if a contradiction is detected.

We can define additional useful temporal semantic properties (and implied temporal inference actions). One such property is *universally diffusive*: the parameter proposition can be inferred for any superinterval of the proposition interval, in a particular context. This property is assumed to have the value FALSE as a default. (However, the NONMONOTONIC value of the gradient abstraction proposition for all parameters and contexts has the *universally diffusive* inferential property as default, since any interval that includes a NONMONOTONIC subinterval is NONMONOTONIC for that parameter.) *Constant* parameters, however (e.g., gender), can be assumed, as a default, to have the *universally diffusive* property for all contexts, once their value is set for some interval. Constant parameters, although usually considered atemporal, can therefore be modeled as *temporal* parameters that have the *universally diffusive* inferential property and the *downward hereditary* property. They can also induce corresponding interpretation context intervals.

The *universally diffusive* property can be refined further. Some propositions are naturally only *backward diffusive*—that is, they are true as a default only from $-\infty$ (or at least, from the zero point time stamp) up to the time in which they are known to be valid (e.g., "the patient was living in England until now", and other propositions of the type "P is true until *now*"; when asserted at any time $[t,t]$, they hold over the interval $[-\infty,t]$, or at least over $[0,t]$). Other propositions are naturally *forward diffusive*; that is, they are true as a default only from the present until $+\infty$ (e.g., "the patient had a liver transplant" and other propositions of the type "P was true in the past"). Note that the input for the temporal inference action in the case of the three diffusive properties includes an abstraction interval and a *context interval*; the output, if the inference holds, is an abstraction interval where the scope of the temporal element varies, dependent on the scope of the input abstraction and context intervals.

Defaults for the temporal semantic properties of propositions exist and can be overridden for any specific parameter, value, and context by the knowledge engineer defining the basic concepts and entities of the task, or by the domain expert elaborating the temporal abstraction knowledge. The *temporal semantic knowledge* for a domain can be summarized as a relation that we call an *inference properties table*, where every tuple (π,ν,ϕ,τ,ξ) in the relation represents that the temporal semantic property $\phi \in \Phi$, for value ν, of parameter π, in the context ξ, has the truth value τ ($\tau \in \{\text{TRUE},\text{FALSE}\}$). The parameter π is assumed here to include its abstraction type.

Most temporal semantic properties have, as a default, values that are stable over many domains; thus, the inference properties table can be defined at a high, domain-independent level and modified for particular parameters only when necessary. Both the *downward hereditary* property and the *concatenable* property can be assumed to have the value TRUE as a default for most abstractions, as well as for interpretation contexts, unless specified otherwise. The *solid* and *gestalt* properties can be assumed to be TRUE by default, although the values of these properties can vary. The three diffusive properties are assumed to have the value FALSE by default. External *events* are considered, as a

default, to be *nonconcatenable* and *solid*, but *interpretation contexts* are assumed, as a default, to be *downward hereditary* and *concatenable*.

The temporal semantic inference subtask uses the temporal semantic properties for two types of conclusions:

(1) *Positive inference*: creation of new abstractions, such as when the *downward hereditary* property is used to interpret a parameter proposition over a subinterval of the input abstraction interval.

(2) *Negative inference*: detection of contradictions, such as when a parameter proposition for which the *gestalt* temporal semantic property has the value TRUE is detected within the temporal scope of an identical parameter proposition. (In that case, following the contradictory inference, we need to decide which parameter proposition, if any, should be retracted, and how to propagate the results of such a retraction to the rest of the abstraction database; see Section 3.3.)

Performing a contemporaneous abstraction of several parameter propositions into another one often involves performing a preliminary step of temporal semantic inference. If two abstractions containing two propositions ϕ_1, ϕ_2 overlap with respect to temporal scope, and there is a vertical classification rule ϕ_1 and $\phi_2 \Rightarrow \phi_3$, we cannot immediately infer that ϕ_3 holds over the intersection interval I. Both ϕ_1 and ϕ_2 must be *downward hereditary*. In that case, ϕ_3 is *necessarily true* over I at our current state of information (unless new data invalidate that conclusion). If ϕ_1 is *gestalt*, we cannot infer ϕ_3; we can infer, however, that ϕ_1 is *necessarily false* over I. Finally, if ϕ_1 is not *downward hereditary* and is not *gestalt* (e.g., the NONMONOTONIC(blood pressure) parameter proposition), we can conclude only that ϕ_1 is *possibly true* over I, or, equivalently, is *possibly false*. New data can change the truth value of the proposition ϕ_1 over I to either necessarily true or necessarily false. Similar analysis holds for other temporal relations between interval-based propositions, leading to a *modal logic* [24] of combining propositions over time. Thus, answering a temporal query involving one or more inferred parameter propositions might generate, in principle, answers such as "the value ν for parameter π is possibly true during the period denoted by interval I".

2.5.2. *Temporal horizontal inference*

The second temporal inference subtask is *temporal horizontal inference*. This subtask determines the value of the join operation ($\oplus$) of two meeting abstraction intervals in the same context, where the same parameter has equal or different values (e.g., one over which the parameter is abstracted as INCREASING, and one over which the parameter is abstracted as DECREASING), into an abstraction superinterval (e.g., with a parameter value NONMONOTONIC). This inference action uses, apart from temporal semantic knowledge (i.e., whether the concatenable property of the parameter has the value TRUE), *horizontal classification knowledge*. Note that such knowledge is a classification-type knowledge. In general, a horizontal inference table should be constructed for all the task-relevant abstraction combinations, possibly specialized also for the context. A *horizontal inference table* is a relation that includes tuples of the form $(\pi, \nu_1, \nu_2, \nu_3, \xi)$, meaning that, for parameter π (assuming that π includes its abstraction type), when an abstraction interval with parameter value ν_1 meets an abstraction interval with parameter value ν_2, in the context ξ, the value of the parameter of the joined abstraction interval

should be v_3. That is, $v_1 \oplus v_2 = v_3$. In a horizontal inference table, it is assumed that concatenated abstractions are of the same type—for instance, a state abstraction type (e.g., HIGH or LOW) or a gradient abstraction type (e.g., INCREASING or SAME). The $\oplus$ operator is the *horizontal join* operator.

In the case of the gradient abstraction type, we define a default, domain-independent, horizontal inference table, denoting a qualitative $\oplus$ operation for joining gradient-abstracted intervals [50]. The default $\oplus$ operation assumes that the set of allowed gradient abstraction values is {DECREASING, INCREASING, SAME, NONINCREASING, NONDECREASING, NONMONOTONIC}. The abstractions SAME, DECREASING, and INCREASING are the results of primary interpolation (see Section 2.6) from two parameter points and require only standard algebra; the rest are concluded only as the result of secondary abstractions of abstraction intervals (e.g., DECREASING $\oplus$ SAME = NONINCREASING). The default $\oplus$ operation can be viewed as a modification of the $\oplus$ operation in the Q1 algebra [57], if the sign of the derivative of the parameter, whose gradient is abstracted in the Q1 algebra, is mapped in the following fashion: $\{0\} \rightarrow$ SAME, $\{+\} \rightarrow$ INCREASING, $\{-\} \rightarrow$ DECREASING, $\{?\} \rightarrow$ NONMONOTONIC. We have added the NONINCREASING symbol for the set $\{0, -\}$, and the NONDECREASING symbol for the set $\{0, +\}$. It can be easily proved that this extension still preserves associativity (i.e., $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$) and commutativity (i.e., $X \oplus Y = Y \oplus X$). As in other qualitative algebras, there is no additive inverse, although the values of the original input parameter propositions remain in the knowledge base.

Most of the entries in the horizontal inference table, for a particular application task, can exist as domain-independent (or at least as context-independent) defaults. Otherwise, these entries can be filled in by the knowledge engineer; the domain expert can add more abstraction values, and then can define the result of concatenating meeting intervals already abstracted by temporal abstractions. As Section 2.6 shows, the horizontal inference table also is used by the temporal interpolation mechanism.

In summary, the temporal inference mechanism requires an ontology of parameter propositions that includes

(1) *classification knowledge*, which comprises horizontal classification knowledge represented as a horizontal inference table, and
(2) *temporal semantic knowledge*, which comprises temporal semantic properties represented as an inference properties table.

2.6. Temporal interpolation

The temporal interpolation mechanism accepts as input two parameter points, two parameter intervals, or a parameter interval and a parameter point, and returns as output an abstraction, interpreted over a superinterval of the input's time points or intervals, interpolating over the gap between these time intervals (see Fig. 3). For instance, we need temporal interpolation to create a DECREASING gradient abstraction for a parameter over an interval, when the start point parameter value in that interval is greater than the end point parameter value (and no parameter values of intermediate time points are known), or when we are joining two sufficiently (temporally) close interval abstractions of LOW(Hemoglobin) into a longer one.

We distinguish between two types of temporal interpolation, depending on the temporal component of the input:

(1) *primary interpolation*, which accepts two parameter points and returns an abstraction interval, and

(2) *secondary interpolation*, which accepts two parameter intervals (or a parameter interval and a parameter point), and returns an abstraction superinterval.

There are three basic abstraction types (STATE, GRADIENT, RATE); thus, there are six interpolation subtasks, or inference actions. The following presentation of the six temporal interpolation subtasks uses the default gradient abstraction values mentioned in Section 2.5. (Even these essentially domain-independent values can be modified.)

- *Primary state interpolation* accepts two disjoint state abstraction points T_1 and T_2 for the same parameter π (e.g., a state abstraction of π with the value LOW), and bridges the gap between them to infer a state abstraction of π interpreted over the interval $[T_1, T_2]$.
- *Primary gradient interpolation* accepts two parameter points T_1 and T_2 of the same parameter π and, if certain conditions hold, infers a gradient abstraction of π interpreted over the interval $[T_1, T_2]$ whose value is DECREASING, INCREASING, or SAME with respect to the change in π.
- *Primary rate interpolation* infers rate abstractions; it classifies rates of change in π (e.g., STABLE, SLOW, or other domain-specific values) between two parameter points of the same parameter.
- *Secondary state interpolation* occurs when, from two state abstraction intervals I_1 and I_2 of parameter π, a state abstraction of π is inferred, interpreted over a new interval $I_j = [I_1.\text{start}, I_2.\text{end}]$. An example is joining two sufficiently (temporally) close intervals over which the state abstraction of the Hemoglobin level parameter has the value LOW.
- *Secondary gradient interpolation* infers, from two gradient abstraction intervals of parameter π, a gradient abstraction superinterval of π whose default values include INCREASING, DECREASING, SAME, NONDECREASING, NONINCREASING, and NONMONOTONIC.
- *Secondary rate interpolation* infers, from two rate abstraction intervals of parameter π, a rate abstraction superinterval of π.

Note that a parameter, such as the Hemoglobin level, might be abstracted as INCREASING in a certain interval with respect to the gradient abstraction, might be abstracted as SLOW with respect to the rate abstraction in that interval, and might be abstracted as NORMAL with respect to the state abstraction type. The three abstraction types are thus mostly independent, except for a possible correlation between the gradient and the rate (e.g., a value of the rate abstraction other than STABLE, or its domain-specific equivalent, might imply that the value of the gradient abstraction was other than SAME).

Temporal interpolation requires that the temporal distance between the two time points or intervals be less than a certain time *gap*. Within that time gap, the *characterization* of the parameter, as defined by the specific value of the given abstraction (e.g., LOW or INCREASING), can then be assumed to hold. The maximal allowed gap must be a domain-, task-, and context-dependent function (e.g., the maximal allowed gap for

LOW(Hemoglobin) in the domain of oncology, the task of caring for patients using certain protocols, and the interpretation context of patients receiving X-ray therapy). The interpretation context is common to both joined intervals. The arguments of the maximal gap function for each specified context include the parameter that we are abstracting and the specific abstraction that we have in mind. They also include a measure of the rate of change of the parameter before and after the time gap. As an approximation of the arguments affecting the rate of change, we use the length of the intervals before and after the gap. Thus, for every context ξ, we denote the maximal gap function Δ as $\Delta(\pi, \nu, L(I_1), L(I_2), \xi)$ of the specific abstracted parameter π (assuming that π includes its abstraction type) and the lengths $L(I_1)$, $L(I_2)$ of the intervals I_1 and I_2, to be joined in the context ξ into an interval with an abstraction value ν. The Δ function returns the length of the maximal temporal gap that still allows interpolation between I_1 and I_2. For instance, in any context, joining two intervals where the Hemoglobin level state abstraction was classified as LOW into a longer interval whose Hemoglobin level state abstraction is classified as LOW depends on the time gap separating the two intervals, on the properties of the Hemoglobin level state abstraction for the value LOW in that context, and on the length of time in which the LOW property was known both before and after the time gap (see Fig. 3).

A prerequisite to an interpolation operation is that the value ν of the parameter π has the value TRUE for the *concatenable* inferential property in the context ξ. This prerequisite involves *temporal semantic knowledge* represented in the *inference properties table* (see Section 2.5).

Similarly, deciding *what* is the value of the resulting abstraction when joining two abstraction intervals with different values ν_1 and ν_2 of the same parameter π requires using *horizontal classification knowledge* as represented in the *horizontal inference table* (see Section 2.5). In this case, both the temporal semantic knowledge (inferential property) and the temporal dynamic knowledge (Δ function) that are used for interpolation are those specific to the value ν_3 that is the result of joining ν_1 and ν_2, $\nu_1 \oplus \nu_2$.

In the following discussion, we often omit the context argument of the Δ function. In addition, the value of an abstraction (e.g., INCREASING) usually implies the abstraction type (e.g., gradient), so the type of the abstraction usually will be omitted as well.

Primary interpolation is the initial *constructor* of abstraction intervals from parameter points. A necessary requirement for primary interpolation is that $L([T_1, T_2]) \leqslant \Delta(\pi, \nu, 0, 0)$, where $L(I)$ is the length of I. Primary interpolation also requires that the parameter π can be measured at least on an ordinal scale (for gradient abstractions) or on an interval scale (for rate abstractions).

Secondary state, gradient, and rate interpolation require additional conditions, apart from an upper bound on the gap between the intervals, to preserve consistency. These conditions, as well as the maximal gap specified by the maximal gap function, can be summarized by an extension of the horizontal inference table, an *interpolation inference table*, which defines the interpolation operation for every abstract parameter (e.g., state(Hemoglobin)) and combination of abstraction values (e.g., INCREASING and SAME). An interpolation inference table represents both the horizontal classification knowledge and the special temporal conditions that should hold between the temporal elements of the involved abstractions.

For example, we need to check that, when we use secondary temporal interpolation to join two DECREASING abstractions for π that are true over two intervals I_1, I_2 into a DECREASING abstraction for π over a superinterval I_j, the value of π has indeed decreased, or at least has not increased above a certain predefined threshold during the time gap $[I_1.\text{end}, I_2.\text{start}]$ (see Fig. 3). In other words, we have to check that $I_1.\text{end}.\pi \geqslant I_2.\text{start}.\pi - C_\pi$, where C_π represents a measurement variation for π—the maximal increment in parameter π, below which a change in π will not be considered as an increase. C_π can be interpreted as a measurement error of π, or as a natural random variation of π over time, or as a clinically significant change of π, for a particular task, depending on the context. C_π is a function of π, $f_c(\pi)$, that is defined either by the domain expert or through analysis of the distribution of π. In general, $f_c(\pi)$ might also use a context argument ξ and the initial value of π, $I_1.\text{end}.\pi$ (e.g., what is considered as a significant variation in the value of the Hemoglobin value parameter might be different within the interpretation context BONE-MARROW DEPRESSION, and might be even more different when the most recent Hemoglobin value known is abstracted as VERY LOW).

Using the C_π property, we can ignore minor absolute changes in the value of π that are less than a certain threshold when we wish to identify general qualitative trends. Furthermore, in the case of primary temporal interpolation for the INCREASING gradient abstraction, we require that $T_2.\pi - T_1.\pi \geqslant C_\pi$. Similarly, in the case of primary temporal interpolation for the DECREASING gradient abstraction, we require that $T_1.\pi - T_2.\pi \geqslant C_\pi$; in the case of primary temporal interpolation for the SAME gradient abstraction, we require that $|T_2.\pi - T_1.\pi| \leqslant C_\pi$. In the case of secondary temporal interpolation for two DECREASING gradient abstractions, we require the value constraint $I_1.\text{end}.\pi \geqslant I_2.\text{start}.\pi - C_\pi$, apart from the gap requirement $I_2.\text{start} - I_1.\text{end} \leqslant \Delta(\pi, \text{DEC}, L(I_1), L(I_2))$. Similar constraints exist for other value combinations [50].

In certain cases, the secondary interpolation operation also can be interpreted as follows: if parameter π can be interpolated by primary interpolation as, for example, DECREASING over the gap interval I_g between the two abstractions I_1 and I_2 ($I_g = [I_1.\text{end}, I_2.\text{start}]$), then we infer a DECREASING abstraction of π over I_g. That is, we infer by interpolation that DECREASING$([I_1.\text{end}, I_2.\text{start}], \pi)$. At this point, the *temporal inference* mechanism suffices to infer DECREASING(I_j, π) by iterative joining of I_1, I_g, and I_2. However, in general, an overall INCREASING or DECREASING abstraction might be created even when the gap interval could be abstracted as SAME, since the secondary interpolation operation as defined here also considers the value and the temporal scope of the abstractions both before and after the gap.

The temporal interpolation mechanism can be seen as a heuristic extension of the logically sound temporal inference mechanism, in which the temporal semantic *concatenate* inference action joins disjoint, but concatenable, abstractions (after horizontal classification knowledge has been used for computing the value of the resultant joined abstraction, to check whether that value is concatenable). Alternatively, the temporal semantic *concatenate* inference action can be modeled as a private case of temporal interpolation. The conclusions of the temporal interpolation mechanism often override the somewhat weaker conclusions of the temporal inference mechanism. For example,

a stronger conclusion of a DECREASING abstraction might be formed for a joined interval, ignoring a potentially SAME intermediate gap interval, instead of the weaker NONINCREASING abstraction that would be formed if the gap interval were abstracted only as SAME (see Fig. 3, for the case of the creation of interval I_7 from intervals I_5 and I_6).

2.6.1. Multiple parameter interpolation

Primary gradient and rate interpolations for abstract parameters whose values are inferred by a multiple parameter contemporaneous abstraction are special. There are two options regarding the semantics of the operation. The *direct* option is to consider only the values of the abstract parameter, assuming that these values are ranked (for gradient abstractions) on an ordinal scale from lowest to highest (e.g., LOW, MODERATE, HIGH, EXTREME). An INCREASING value for the gradient abstraction of π during interval I can then be equivalent to a positive change in the rank of the values $I.\text{start}.\pi$, $I.\text{end}.\pi$. An equivalent definition can apply to rate abstractions. The *indirect* option, which we call a *trend* abstraction, defines the meaning of, for instance, an INCREASING primary interpolation for parameter π as the case when all (or at least one) of the parameters qualitatively positively proportional to π are INCREASING, and all of the parameters qualitatively negatively to π are DECREASING (or at least are not INCREASING), even if the value of π's state over the interval I has not changed. We define the indirect (trend) abstraction recursively [50] using the *qualitatively proportional* notation introduced by Forbus [18]. Either option can be the default for a particular domain and task [50]. The default semantics, however, might be overridden by the domain expert at knowledge acquisition time.

The qualitative dependency relationships (e.g., POSITIVELY PROPORTIONAL and NEGATIVELY PROPORTIONAL) between an abstract parameter and the primitive or abstract parameters defining that parameter are part of the domain's *structural knowledge*. These relationships are an additional aspect of the ABSTRACTED-INTO relation over the domain's parameters.

2.6.2. Local and global persistence functions and their meaning

We distinguish between two types of persistence functions, from the aspect of temporal scope: *global persistence* (Δ), or *maximal gap*, *functions* and *local persistence* (ρ) *functions*. Maximal gap (Δ) functions allow interpolation between parameter points and intervals by creation of a default abstraction during the maximal gap interval. They represent domain- and task-dependent knowledge regarding the rate of change of a parameter proposition $\langle \pi, \nu, \xi \rangle$ over time, or the *persistence* of the truth of that proposition over a temporal gap. Local persistence functions represent the local persistence of the truth of a parameter proposition, given a single parameter point or interval, *before or after* the temporal scope of the parameter proposition. In some cases, global persistence functions can be inferred from knowledge of local ones.

For the purpose of the following discussion, we assume that the context ξ and the value of parameter π, unless mentioned explicitly, are known and fixed (these arguments can serve as indices to a function with fewer arguments).

Local persistence functions

Local persistence (ρ) functions represent the local persistence of the truth of a parameter proposition, given a single parameter point or interval: $\rho(\pi, L(I), t)$, where $L(I)$ is the length of the interval I over which the parameter proposition is known to be true, and t is the time since that proposition was true. The ρ function returns a degree of belief—a probability distribution—in the proposition $\langle \pi, \nu \rangle$ being true at time $t_0 + t$, given that $\langle \pi, \nu \rangle$ was true at time t_0. The ρ function is an extension of McDermott's [35] persistence assumption and of McCarthy's [33] inertia principle, both of which include infinite persistence as a default. The ρ function model is similar to Dean and Kanazawa's [12] model of propositions that decay over time, and to de Zegher-Geets' [13] time-oriented probabilistic functions (TOPFs) in the IDEFIX system (a TOPF represents the probability of a state or disease given a previous identical state). However, ρ functions are more general, in the sense that they extend temporally in *both* directions: to the *future* and also to the *past*. Assuming that time t_0 is a random time in which the proposition was measured, there is no particular reason to assume that a parameter proposition was not true before that time. Thus, t might actually have a *negative* value. We need this extension if we are to include an approximation of the past value of a parameter, for purposes of interpretation, as opposed to forecasting a future value of the parameter. Thus, we can include a model of *forward decay* and *backward decay* in belief. The function describing this decay is equivalent to a statistical *survival function*.

In practice, the important question for performing an interpolation using a local persistence function is how long t can be before the belief in the parameter proposition $\phi \in P$ drops below a certain threshold ϕ_{th} (Fig. 5). The threshold belief ϕ_{th} can be interpreted as the point when the probability of the truth value of the proposition ϕ in which we are interested falls below a certain threshold (say, 0.9). The threshold has a task- and context-specific value.

Using a threshold creates a value- and context-specific *validity* time for a parameter proposition, similar to McDermott's [35] *lifetime* of a fact and to the *expected length* attribute of states in de Zegher-Geets' IDEFIX system. (That attribute, however, was used by IDEFIX mainly for time-oriented display purposes, and was not part of the interpretation framework [13].)

Global persistence functions

Global persistence (Δ), or maximal gap, functions bridge the gap between two propositions. Δ functions are an extension of ρ functions; in special cases, as we mention in this section, they can be constructed from the latter functions. The Δ function returns the maximal time gap that still allows us to join the propositions into an abstraction that is believed to be true—with a sufficient, task-specific, predefined degree of belief in the proposition—during the gap (and thus over a superinterval of the input propositions, given that both were true for some time before and after the gap). Δ functions are a global extension of the ρ persistence functions, since they implicitly assume both forward and backward decay of the propositions involved.

Fig. 5 presents a graphic view of the Δ function as an interpretation of a decay in the belief in the truth of a proposition. For instance, in the case that the abstractions'

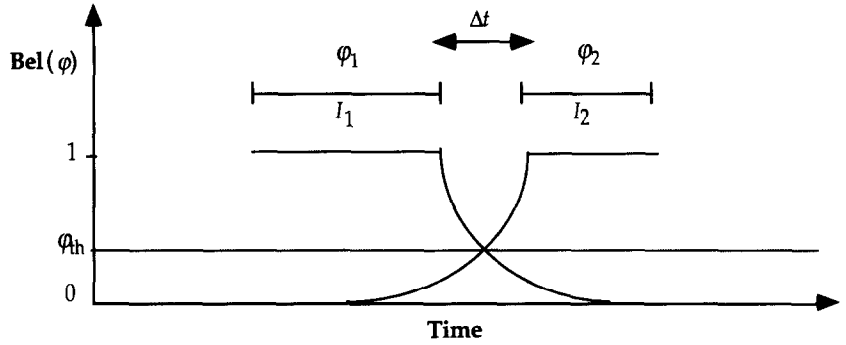

Fig. 5. Local and global persistence functions. We use the maximal time gap Δt returned by a global Δ function to decide whether the parameter propositions ϕ_1 and ϕ_2, interpreted over intervals I_1 and I_2, can be joined (possibly, if these propositions do not denote the same value of the relevant parameter, into a new proposition ϕ_3 with a parameter value $v_3 = v_1 \oplus v_2$). The time gap Δt can be interpreted—if $\phi_1 \equiv \phi_2$, and the truth values of the propositions are relatively independent—as the maximal time gap in which the belief produced by either the local forward or backward decay (represented by a local persistence (ρ) function) stays above the predefined confidence threshold ϕ_{th}. Bel(ϕ) = degree of belief in ϕ; ϕ_{th} = the task- and context-specific belief threshold value.

parameter values are identical—that is, the propositions are the same before and after the gap interval—and the forward and backward decay times are relatively independent, we are interested in whether, at all points inside the gap interval, *either* of the values, approximated by the forward belief decay in proposition ϕ, $\text{BEL}_{\text{forward}}(\phi)$, or by the backward belief decay, $\text{BEL}_{\text{backward}}(\phi)$, is true with a probability $p \geqslant \phi_{th}$. As the time gap Δt between the two abstractions increases, the belief that either the backward or forward decay value is true will eventually fall below the predefined threshold value ϕ_{th} (see Fig. 5).

If the ρ function is an exponential decay survivor function, such as used in several of de Zegher-Geets' TOPFs, and the backward and forward decay rates are independent, then we can compute the Δ function. Assume that the probability $p(t)$ of the parameter proposition ϕ being true is $e^{-\lambda t}$—a function of the time t since the reference time in which P was true, is regardless of the length of the time interval I during which ϕ was true. Let the forward decay rate be λ_1, and the backward decay rate be λ_2. It can be shown [43] that

$$\Delta t \leqslant [(\lambda_1 + \lambda_2)/(\lambda_1 * \lambda_2)]K, \quad K = -\ln \phi_{th}.$$

In other words, the Δ function for two parameter points, $\Delta(\pi, 0, 0)$, or for two parameter intervals when the duration of the intervals has no effect on the persistence of the propositions, is a constant determined by the forward and backward decay rates and by the desired level of confidence.

This analysis can be generalized. Assume that, the longer ϕ is known to be true in the past or in future, the longer we are likely to keep believing it or to believe that it already existed in the past, before we measured it (this assumption will be discussed in Section 2.6.3). One (not necessarily the only) way to represent that assumption would be to

modify the decay rate for each proposition, λ_i, by assuming that the rate is inversely proportional to the length of the relevant intervals, $L(I_i)$.

It can then be shown [43] that, if exponential decay rates decrease (equally) linearly forward and backward as a function of the duration of the proposition, then the maximal time gap allowing us to join equal length abstractions would be proportional to a linear function of the length of either interval, with the rest of the factors kept constant. The duration of the gap would be inversely proportional to the uniform decay rate. For instance, if $\lambda_1 = \lambda_2 = \lambda$ and $L(I_1) = L(I_2) = L$, then

$$\Delta t \leqslant [2L/\lambda]K, \quad K = -\ln \phi_{\text{th}}.$$

These simplified examples serve to show that, even though the decay rates λ_i are in general unknown, and the decay function is perhaps difficult to compute, the resulting global Δ function (using a belief threshold) might be a simple constant or polynomial, and thus can be more easily described, computed, or acquired from domain experts than can the underlying local persistence function.

Furthermore, if there is evidence for a particular type of decay function (e.g., logarithmic), we can compute the latter's coefficients by acquiring from the domain expert a few maximal gap values—that is, several examples of Δt. We might even check the expert's consistency (or the adequacy of the decay function) by repeating the calculation for several other examples. Alternatively, we can simply acquire a table of typical Δt values for various common $L(I_1)$ and $L(I_2)$ values, and can interpolate between these values, or extrapolate from them, when necessary.

Note that, due to the lack of independence between the forward decay of a parameter proposition attached to one time point and the backward decay of that proposition at a later time point, and, therefore, an implied *joint distribution* of the forward and backward belief values, we usually need the actual global (Δ) function, in addition to (or instead of) the local (ρ) function. In practice, the domain expert often knows several Δ function values (such as what is the maximal time gap allowed for her to join two parameter points for several parameter values in each context), even if she cannot define any particular, precise, ρ function (except, possibly, for specifying the forward and backward decay times Δt corresponding to reaching the local threshold value ϕ_{th}). Knowing only the Δ function still enables interpolation between two point- or interval-based parameter propositions. In view of the preceding discussion, in many domains, knowing only the values needed to maintain $\text{Bel}(\phi)$ above the threshold value ϕ_{th}—that is, the (simpler) Δ function—would be common.

2.6.3. *A typology of persistence functions*

Global persistence (Δ) functions can have four *qualitative types*, depending on whether the Δ function is either positive monotonic or negative monotonic, with respect to the length of the first parameter interval $L(I_1)$, or the length of the second parameter interval $L(I_2)$ (see Fig. 5). Thus, theoretically there are *positive-positive* (*PP*), *positive-negative* (*PN*), *negative-positive* (*NP*), and *negative-negative* (*NN*) monotonic Δ functions. We refer to these categories as *qualitative persistence function types*.

Formally, PP Δ functions are functions such that

$$L(I') > L(I) \Rightarrow \forall i[\Delta(I',i) \geqslant \Delta(I,i) \wedge \Delta(i,I') \geqslant \Delta(i,I)];$$

NN Δ functions are functions such that

$$L(I') > L(I) \Rightarrow \forall i[\Delta(I',i) \leqslant \Delta(I,i) \wedge \Delta(i,I') \leqslant \Delta(i,I)],$$

where $L(I)$ is the length of interval I, and $\Delta(I,i)$ stands for $\Delta(L(I),L(i))$.

In the case of ρ functions, whether representing backward or forward local persistence, we can categorize functions qualitatively into *positive* (P) and *negative* (N) categories with similar meaning (i.e., whether, the longer I, the longer or shorter the relevant validity interval, before or after I).

Most Δ functions, in practice, are of the PP type. In other words, the longer we know that a parameter proposition was true either before or after a time gap, the longer we would allow that gap to be while we maintained our belief that the parameter proposition stayed true throughout that gap (i.e., that its probability was always above a certain threshold). (For instance, the proposition denoting the MODERATE-ANEMIA value of the Hemoglobin_state (abstract) parameter usually would be associated with a PP Δ function, as would be the proposition denoting the DEEP COMA value of the Consciousness parameter).

Negative monotonic Δ functions occur when a longer duration either of I_1 or of I_2 lowers the probability that the abstraction was true during the gap, and the longer the lengths, the shorter the allowed Δt. For instance, knowing about a longer I_1 interval of an almost fatal cardiac arrhythmia (say, ventricular fibrillation) actually *lowers* the probability that the (following) gap interval had the same property, given the same I_2 interval and assuming that the patient is alive. Most of the negative monotonic functions emerge from a total length constraint on the time span of the abstraction (or an analogous probabilistic distribution on the expected total time), or from a total cardinality constraint on the number of events.

The common PP Δ functions often can be assumed as a default. Note that, the exponential decay ρ functions that were given as an example in Section 2.6.2 for local persistence functions dependent on the length of the parameter interval, imply, with the independence assumption, a PP-type Δ function. Moreover, we can prove [43] that PP Δ functions imply an important computational advantage.

Proposition 1. *PP Δ functions are associative (i.e., the order of joining intervals and points cannot change the resulting set of abstractions). The associativity property is important to ensure, for data-driven systems, that the final abstractions do not depend on the order of arrival of the input data.*

Proof. Assume parameter points T_1, T_2, and T_3, in that order. If both parameter interval $[T_1,T_2]$ and parameter interval $[T_2,T_3]$ can be formed, then, if we can eventually form the interval $[T_1,T_3]$, we can do so by forming initially either subinterval, since the Δ function is PP. That is, if we can join one point to another, we can certainly join that point—forward or backward, as necessary—to an interval starting or ending, respectively, with the other point. For instance,

$$L([T_1,T_2]) \leqslant \Delta(0,0) \Rightarrow L([T_1,T_2]) \leqslant \Delta(0,L([T_2,T_3])),$$

since the Δ function is PP, and therefore $\Delta(0,0) \leqslant \Delta(0,L([T_2,T_3]))$.

A similar argument holds for any four consecutive points.

Thus, the claim is true for any sequence of primary or secondary interpolations, since Δ functions are applied only when there are no intervening points between the two intervals or points to be joined. □

Proposition 2. *NN Δ functions are not associative.*

Proof. It is easy to construct a case for consecutive parameter points T_1, T_2, and T_3, where, if we create the interval $[T_1,T_2]$, we no longer can join it to T_3, and, if we create the interval $[T_2,T_3]$, the Δ function value will prevent our joining it to T_1 (e.g., a total-sum constraint does not allow us to create the interval $[T_1,T_3]$ with sufficiently high probability). □

NP and PN functions are not associative either. Whether such functions can even *exist* with appropriate semantic restrictions on the nature of Δ functions is doubtful, and we leave it as an open question.

Finally, another useful semantic distinction is *context continuity*: use of the Δ function within a *simple* versus a *nonconvex* interpretation context (see Section 2.3.2). Note that the maximal gap Δ function interpolating between two propositions within a *nonconvex* interpretation context might be different from the Δ function used for interpolation within each *convex* (or *simple*) segment of the nonconvex interpretation context. In the first case, the Δ function would be an *interphase* Δ function (e.g., between Blood-glucose abstractions measured over different mornings—i.e., between abstractions created within disjoint, although similar, interpretation contexts). In the second case, the Δ function would be an *intraphase* Δ function (e.g., between Blood-glucose values measured within the same continuous prebreakfast interpretation context).

In summary, the temporal interpolation mechanism requires an ontology of parameter propositions that includes

(1) *structural knowledge*, which consists of the qualitative dependencies aspect of the ABSTRACTED-INTO relation, and the domain-specific time units;
(2) *classification knowledge*, which comprises classification of domain-specific gradient and rate abstraction values, and horizontal classification knowledge—the horizontal inference table;
(3) *temporal dynamic knowledge*, which includes the maximal gap (Δ) functions and the local persistence (ρ) functions, the significant change values C_π, or functions $f_c(\pi)$, for the relevant parameters in various contexts, and additional temporal constraints for completing the interpolation inference table; and
(4) *temporal semantic knowledge*, which consists of the truth values of the *concatenable* property for the relevant input and inferred parameters.

The temporal dynamic knowledge about the domain does not necessarily need to include complete, closed definitions of Δ functions—partial tables may suffice, or actual functions might be approximated. Knowing whether a Δ function is PP or NN is impor-

tant for estimating the value of that function from a few examples, or for interpolating that value from several discrete entries in a table. This qualitative persistence type is easy to acquire, since domain experts usually have an excellent intuition about whether, qualitatively, a longer duration of a parameter proposition before or after a gap increases or decreases the probability of the proposition being true during a longer gap, even if precise probabilities are unknown.

2.7. Temporal pattern matching

In addition to the context-forming mechanism and the three basic TA mechanisms, a temporal pattern matching mechanism is required for abstracting more complex data patterns over time. For instance, such a mechanism is required for abstraction of an episode of drug toxicity from a state of a LOW(White blood cell) count *lasting more than* 2 weeks and starting *within* 0 to 4 weeks of a state of LOW(Hemoglobin) *lasting more than* 3 weeks, in a patient who is receiving certain drugs. Another example is recognizing a *quiescent-onset* pattern of chronic GVHD (see Fig. 1): a Chronic GVHD abstraction starting at least 100 days after a bone-marrow transplantation event, but within 1 month of the end of a preceding Acute GVHD abstraction. The temporal pattern matching mechanism extends the temporal inference and temporal interpolation mechanisms by abstracting over multiple intervals and parameters, and typically reflects heuristic domain- and task-specific knowledge. This mechanism solves the temporal pattern matching task. Output patterns are parameters interpreted over the respective time intervals.

Typically, when the temporal pattern matching mechanism is used, a considerable part of the TA task, depending on domain-specific knowledge, has already been solved by the three basic TA mechanisms and by the context-forming mechanism. These preliminary patterns include contemporaneous abstractions of different parameters, as well as horizontal inferences and interpolations, at various abstraction levels. Thus, the pattern matching mechanism does not have to create abstractions, such as a significantly DECREASING blood pressure, or to decide in what contexts the Hemoglobin level should be considered as LOW. The pattern matching process can use interval-based abstractions, possibly with their end point values. Furthermore, the pattern matching mechanism assumes the TA ontology semantics (e.g., it takes advantage of the small, finite number of abstraction types, such as STATE, and the way that these abstractions are derived, including relations such as ABSTRACTED-INTO). Thus, the pattern matching mechanism employs

(1) a *temporal abstraction query language* that allows the expression of value and time constraints involving parameter intervals, and
(2) a *temporal abstraction pattern matcher* that returns abstractions (or the values TRUE or FALSE) in response to queries in that language.

The knowledge used by the temporal pattern matching mechanism is mainly classification knowledge. The input is one or more parameter points and intervals, possibly including different parameters, different abstraction types, and different interpretation contexts. The output is an abstraction of type PATTERN. The constraints on the abstractions include constraints on the parameter values of the parameter propositions involved,

and constraints on the temporal attributes of the abstractions involved. Typically, the temporal span of the output abstraction is the union of the temporal span of the input abstractions.

In summary, temporal pattern matching requires

(1) *structural knowledge*, comprising ABSTRACTED-INTO relations from parameters to patterns, and

(2) *classification knowledge*, comprising classification of sets of abstraction points and intervals into PATTERN-type abstract parameters, using value and time constraints.

3. The RÉSUMÉ system

The KBTA method and TA mechanisms described in Section 2 have been implemented as a computer program: the *RÉSUMÉ* system [43,46,48]. The RÉSUMÉ system generates temporal abstractions, given time-stamped data, events, and the domain's TA ontology of parameters, events, and contexts. The RÉSUMÉ system is composed of a temporal reasoning module (the five TA mechanisms), a static domain knowledge base (the domain's TA ontology), and a dynamic temporal fact base (containing the input and output parameter points and intervals, event intervals, and context intervals). The temporal fact base is loosely coupled to an external database, where primitive time-stamped patient data and clinical events are stored and updated, and where abstractions can be stored by the RÉSUMÉ system for additional analysis or for use by other users. The TA mechanisms iterate alternately, activated by the currently available data and by the previously derived abstractions.

3.1. The parameter properties ontology

Most of the domain-specific knowledge required by the TA mechanisms is represented in an implementation of the parameter ontology called the domain's *parameter properties ontology*. The parameter properties ontology represents the parameter entities in the domain (e.g., Hemoglobin, Granulocyte state), their properties (e.g., temporal semantic properties, such as *concatenable*), and the relations among them (e.g., ABSTRACTED-INTO). Fig. 6 shows a small part of the parameter properties ontology used for the task of managing patients who are being treated by clinical protocols.

The parameter properties ontology is an IS-A frame hierarchy that specializes parameters mainly by increasingly specific interpretation contexts (e.g., classification tables for the Hemoglobin level parameter might be different during administration of a certain protocol or medication). The four knowledge types (including structural relations) are represented as parameter properties (see Section 2.1).

An important feature of the representation scheme shown in Fig. 6 is organization of abstract parameters by four output abstraction types (STATE, GRADIENT, RATE, and PATTERN). Thus, the Granulocyte_gradient_abstraction parameter is a gradient abstraction and inherits slots such as the default set of values and interpolation inference table for gradient abstractions, whereas state abstractions include properties such as

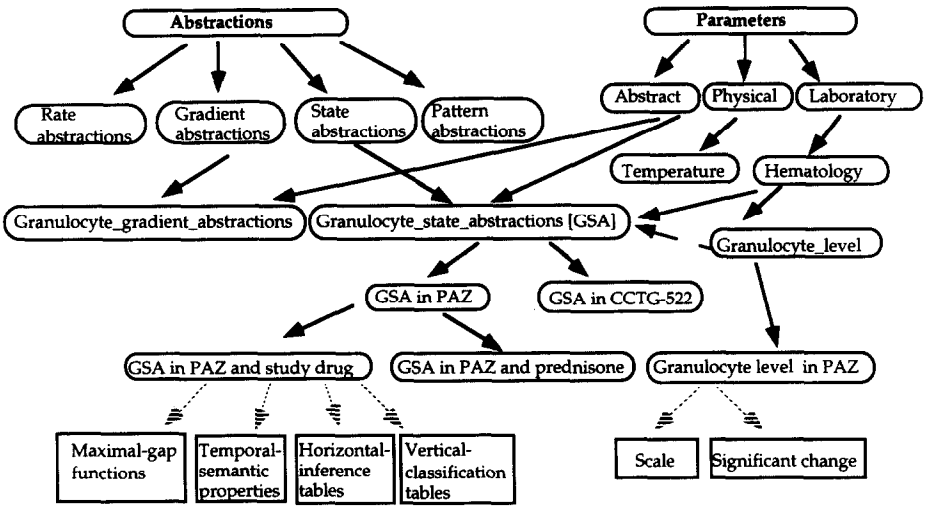

Fig. 6. A portion of the parameter properties ontology in the domain of protocol-based care. Shown in the figure is a specialization of the temporal abstraction properties for the Granulocyte state abstraction (GSA) parameter in the context of the prednisone azathioprine (PAZ) experimental protocol for treating chronic graft-versus-host disease, and in the context of each part of that therapy protocol. Ellipse = class; rectangle = property; arrow = IS-A relation; shaded arrow = PROPERTY-OF relation; dashed arrow = ABSTRACTED-INTO relation.

their defining classification functions (mapping tables). This structure proved flexible for representation and modification of TA knowledge in several domains.

Pattern parameters are represented as first-class entities in the parameter ontology. This uniformity allows the TA mechanisms, at runtime, to perform further temporal reasoning using the derived pattern intervals, and preserves the logical dependencies of these pattern intervals on the other parameters (and contexts) from which they were derived. Maintaining these dependencies allows updates to the past or present data to be propagated to all abstractions, including the temporal patterns. Furthermore, representing patterns as first-class entities in the ontology of the domain permits the use of uniform methods for acquisition, maintenance, sharing, and reuse of knowledge.

3.2. *Evaluation of the RÉSUMÉ problem solver*

We tested various aspects of the RÉSUMÉ system in several different clinical and engineering domains: protocol-based care (and three of its subdomains) [43,46], monitoring of children's growth [28,43], therapy of insulin-dependent diabetes patients [48], and monitoring of traffic-control actions [45]. We applied the RÉSUMÉ methodology to each domain in varying degrees. Sometimes, our focus was evaluating the feasibility of knowledge acquisition (including acquisition by another knowledge engineer), knowledge representation, and knowledge maintenance (i.e., modifications to the resultant knowledge base). In other cases, we emphasized application of the resultant instantiated temporal abstraction mechanisms to several clinical test cases. In one domain, we ap-

plied the RÉSUMÉ system, instantiated by the proper domain ontology, to a larger set of clinical data. We have therefore demonstrated most of the expected lifecycle in the development and maintenance of a TA system.

In the subdomains of protocol-based care, we have focused mainly on the representation of knowledge relevant for experimental therapy of patients who have AIDS, for therapy of patients who have graft-versus-host disease, and for prevention of AIDS-related complications, in each case, working with domain experts [43,46]. Typical abstractions included patterns such as "the second episode of Bone-marrow toxicity grade II that lasts more than 3 weeks" (see Fig. 1). As we expected, we were able to reuse easily both general TA knowledge (e.g., gradient abstractions) and parameter-specific TA knowledge (e.g., Hemoglobin level abstractions) using the explicit parameter properties ontology. Maintenance involved mainly adding or modifying classes that represent abstractions specialized by interpretation contexts.

In the domain of monitoring children's growth, we collaborated with a pediatric endocrinologist to form and maintain a growth monitoring TA ontology [28]. Running RÉSUMÉ on a few clinical test cases produced most of the relevant abstractions [43]. The goal in the growth monitoring domain was not to reach a final diagnosis, but rather was only to decide whether there was any abnormality in the growth chart that might fit a set of predefined internal patterns, or to answer multiple user-defined external temporal queries regarding relevant intermediate level abstractions. An abnormality detected by an internal or external query can call the physician's attention to the need for further monitoring of the particular child.

In the diabetes domain, we collaborated with two endocrinologists, acquiring within several meetings a TA ontology from one of the experts [48]. The two experts formed (independently) temporal abstractions from more than 800 points of data, representing two weeks of glucose and insulin data from each of eight patients. The RÉSUMÉ system created 132 (80.4%) of the 164 temporal abstractions noted by both experts. Examination of the output for the first three cases by one of the experts showed that the expert agreed with almost all (97%) of the produced abstractions—a result similar to the one we found in the domain of growth monitoring [43]. We expected this high predictive value, since the domain's TA ontology directly reflected that expert's knowledge about these low- and intermediate-level abstractions. Although these results are encouraging, we noted several difficulties in the representation and detection of cyclical (e.g., diurnal) patterns; in the matching of absolute time to task-specific time; in the integration of statistical and temporal queries; and in querying for patterns of events rather than for patterns of parameters [43]. Most difficulties can be solved by extensions to the language of the temporal pattern matching mechanism.

In the traffic-control domain, the RÉSUMÉ system was used to model the task of monitoring traffic-control actions, and to create a prototype for solving that task [45]. The task of monitoring traffic-control actions receives as input recent values of different road parameters (speed, flow, and occupancy) measured by sensors located along several highways, and a set of recent control actions (e.g., traffic diversion) undertaken by traffic controllers. It returns an evaluation of the adequacy of the control actions. Performance of this task requires both *temporal* reasoning (e.g., about durations, rates, and trends

of traffic parameters over time, for a given location) and linear *spatial* reasoning (e.g., about queue lengths along the highway, at a given time). We used the RÉSUMÉ problem solver to model and solve *both* tasks. First, we defined a *spatial abstraction ontology*, using the TA ontology knowledge structures, to describe properties of spatial parameters, such as Congestion, along the highway distance dimension. We used this ontology to create linear spatial abstractions in each highway zone, such as Saturation_level. Second, we created a TA ontology to decribe properties of spatial abstractions of each location or highway zone over time. We used this ontology to form and detect crucial traffic-control spatiotemporal patterns.

3.2.1. The work involved in acquisition of temporal abstraction knowledge

In general, the minimal amount of knowledge that we needed to acquire in a new domain included

(1) the relevant primitive and abstract parameters (the latter classified into the four abstraction types: state, gradient, rate, and pattern) and their structural relations (in particular, IS-A and ABSTRACTED-INTO relations);
(2) a task-specific significant change for each relevant parameter, if gradient abstractions are required;
(3) the list of potential state and rate abstraction values for all parameters relevant to the task for which these abstraction types are required; and
(4) the maximal gap Δ functions, when interpolation is required in the task, for each relevant parameter and context.

Temporal semantic properties, gradient abstraction horizontal inference values, and interpolation inference tables are more stable than are the knowledge types listed, and are less dependent on the interpretation context. Default values for these types of TA knowledge either can be inherited through the appropriate abstraction class (e.g., gradient abstractions), or can be acquired for only the most general parameter or extended parameter class (e.g., Hemoglobin_state abstractions in the interpretation context of the overall task). As additional applications were designed and knowledge was shared and reused, the gain in development time was apparent (e.g., in the protocol-based care domains).

Instantiating the TA mechanisms for a new domain would appear superficially to involve significant amounts of knowledge acquisition. However, the major knowledge acquisition effort in the nontrivial domains evaluated usually required only two to four meetings (each about 2 hours long) with the expert—a tenable amount of time. The size of the resultant TA ontologies was manageable. Maintenance of the resultant knowledge base by the knowledge engineer required significantly less time than would be needed to create or modify pure programming code. Furthermore, the knowledge acquisition process was *driven* by the KBTA method, so a methodical structure was added to the process. For instance, the knowledge engineer could ask if a gradient abstraction exists for a state abstraction of a known parameter. Thus, a certain measure of *completeness* was guaranteed. Furthermore, the process can be automated by the use of automated knowledge acquisition tools, such as those generated automatically by the PROTÉGÉ-II system [15,39]. We have recently developed such a tool [54]. Finally, most important parameters and their values *must* be represented in *some* fashion, if we are to perform

the TA task at hand (e.g., hematological toxicity tables *must* be acquired and represented in protocols for treatment of AIDS). Thus, most of the knowledge acquisition work was done on organizing in a useful architecture a significant amount of knowledge that had to be represented, implicitly or explicitly, to perform the task. Explicit representation has multiple additional benefits; in the experiments that we have conducted, its cost was not prohibitive, and the results justified the effort.

3.3. Nonmonotonicity of temporal abstractions

The five TA mechanisms create state, gradient, rate, and pattern abstractions. Unlike input data, however, the inferred parameter intervals are potentially refutable by any modification or addition to the known data points or events. The need to update past or present abstractions when older (but formerly unavailable) data arrive was noted by Long and Russ [31,42]. We refer to this activity as a *view update*. The activity of updating former conclusions and of revising assessments of former decisions given new, present time data is referred to as *hindsight* by Russ [42]. In the first case, we need to evaluate precisely that part of our interpretation and abstraction of the former data that is affected by the newly added old data, such as laboratory reports that are returned a few days after the initial assessment was done. Thus, the *past* influences the interpretation of the *present*. In the second case, we need to bring to bear our current understanding of the situation on the interpretation of the former events, even if no direct additional information about the past has been uncovered; usually, that understanding is a function of current developments, such as the effect of a therapy plan. Thus, the *present* influences the interpretation of the *past*.

The effects of updates to input parameter and event intervals, which might cause deletion of existing, previously concluded contexts and abstractions, are mediated in the RÉSUMÉ system through a truth maintenance system [46]. (The dynamic temporal fact base is thus essentially a *historic* database [52].) In addition, the temporal semantic properties of parameter propositions are used not only for the task of deriving further conclusions, but also for the task of detecting contradictions in existing conclusions. For instance, using the *downward hereditary* semantic property, the temporal inference mechanism not only can create new conclusions for subintervals of parameter intervals, but also can notice that parameter values for similar type parameter propositions within those subintervals (e.g., LOW(Hemoglobin)) actually differ from the parameter value of the parameter superinterval (e.g., HIGH(Hemoglobin)—a longer interval that was created when the included interval was unknown). Such a difference is a contradiction (due to an explicit assumption of a mutually exclusive set of values for the same parameter, time, and interpretation context) and requires retraction of at least one of the intervals. When a contradiction is detected (i.e., the result of the inference is FALSE), the RÉSUMÉ system uses several heuristics to decide *which* of the parameter intervals, if any, should be retracted (e.g., primitive input data might never be retracted, versus abstract conclusions, which might no longer be true). Finally, the results of retracting one or more parameter intervals should be propagated to previous conclusions that are logically dependent on the retracted conclusions. Similarly, when an event interval is modified, or a context interval that depended on a no longer valid event or abstraction is

retracted, the modification is propagated to the rest of the temporal factbase. Conclusions that are no longer valid are retracted and new conclusions are asserted.

3.4. A computational note

The parameter properties ontology does not contain a context specialized node corresponding to *every* interpretation context. The *nonexistence* of a specialization signifies that, for that particular context, the abstraction is *not* relevant, thereby cutting down on unnecessary inferences. Furthermore, types of desired output parameters and the type of inferences to be used can be prespecified. Thus, several goal-oriented mechanisms reduce the amount of abstractions generated by RÉSUMÉ [46].

Both horizontal classification and temporal interpolation are limited by reference to (two) parameter propositions of the same parameter type, context, and (in the case of interpolation) value. The upper bound is thus $O(N^2)$ comparisons for N relevant parameter propositions. Furtheremore, no more than $O(N)$ basic abstraction intervals (i.e., those generated by the contemporaneous abstraction, temporal inference, or temporal interpolation mechanisms) can be generated for any given parameter, context, and abstraction type, given N data points, since basic abstraction intervals are convex. Even if we also count all possible intermediate abstractions of the same value that can be joined up recursively to one interval, there cannot be more than $O(N)$ intervals, because once a parameter point is made part of a parameter interval, it cannot be used as part of another parameter interval for the same parameter and abstraction type (i.e., new parameter intervals for the same parameter are longer than are those from which they are derived). These observations lead to an overall $O(N^3)$ complexity for a complete application of each basic TA mechanism.

Temporal pattern matching can be, in the worst case, exponential in the number of parameter propositions, event propositions, and context intervals, but most patterns, in practice, specify constraints for only two or three intervals, and the pattern matching is highly constrained by parameter types, leading again to a polynomial complexity. In addition, the RÉSUMÉ system exploits the efficient RETE pattern matching algorithm [19]. This algorithm is employed by the CLIPS shell that we are using [21] and fits especially well with the task of matching temporal patterns in data arriving incrementally in some temporal order. The RETE algorithm incorporates each new datum (in this case, usually a parameter interval) into a network of *tokens* in a continuous manner. Although complexity is still exponential in the worst case, typical cases have linear time complexity, since patterns are being matched partially in a continuous manner. Data often arrive in incremental fashion and intermediate abstractions are always cached. Anwering queries typically involves the use of previously cached (asserted) abstractions at several levels.

4. Summary and discussion

The *temporal abstraction* (*TA*) *task*—formation of meaningful, context-sensitive, interval-based abstractions from time-stamped data—is important in many time-oriented domains. The interval-based abstractions that are the output of the TA task can be used

for selection and instantiation of plans; for monitoring plans during execution; for creation of high-level summaries of time-stamped data, for explanation purposes; and for critiquing the execution of a plan by one agent when the plan's overall and intermediate goals can be described in terms of creating, maintaining, or avoiding certain temporal patterns.

The knowledge requirements for performance of the TA task are implicit in traditional domain-specific applications. This lack of explicit representation prevents using general principles common to performance of that task in different domains, and sharing of knowledge common to several tasks in the same domain.

Our approach embodies a *knowledge level* view of the TA task We emphasize the importance of enabling *reuse*, *sharing*, *maintenance*, and *acquisition* of TA knowledge for sizable knowledge-based systems that are applied to time-oriented domains. Our goal has been to elucidate the nature of the knowledge that is required for solving the TA task by a knowledge-based method.

We presented a specific knowledge-based approach to the TA task: the *knowledge-based temporal abstraction* (*KBTA*) *method*. The KBTA *method* decomposes the TA *task* into five *subtasks*, each of which is solved by a different TA *mechanism* (see Fig. 2): temporal context restriction (creation of relevant interpretation contexts crucial for focusing and limiting the scope of the inference); vertical temporal inference (inference from contemporaneous propositions into higher-level concepts); horizontal temporal inference (inference from similar type propositions attached to intervals that span different periods); temporal interpolation (joining of disjoint points or intervals, associated with propositions of similar type); and temporal pattern matching (creation of intervals by matching of patterns over disjoint intervals, associated with propositions of various types). Four *knowledge types* are required for instantiating the temporal abstraction mechanisms in any particular domain (see Fig. 2):

(1) structural knowledge (e.g., IS-A and PART-OF relations in the domain, QUALITATIVE DEPENDENCY relations, SUBCONTEXT relations);
(2) classification knowledge, mostly functional (e.g., mapping of Hemoglobin level ranges into LOW, HIGH, VERY HIGH; joining of INC and SAME into NONDEC; matching of temporal patterns);
(3) temporal semantic, mostly logical, knowledge (e.g., relations among propositions attached to intervals, and propositions attached to their subintervals, such as the *downward hereditary* property); and
(4) temporal dynamic knowledge, mostly probabilistic (e.g., local forward and backward persistence (ρ) functions; global, maximal gap (Δ) functions; significant change functions).

The four knowledge types are sufficient to instantiate the five domain-independent TA mechanisms for any particular application area. The knowledge types form a declarative interface for a knowledge engineer developing a new TA system, and support automated acquisition of knowledge from domain experts.

The *RÉSUMÉ* system implements the KBTA method's inference structure and, in particular, the five TA mechanisms. We used the RÉSUMÉ system to model TA knowledge in several different clinical and engineering domains, and to form in those domains temporal abstractions comparabe to those of experts.

4.1. Related work

Researchers in philosophy, in general computer science, and in artificial intelligence have applied several different approaches to tasks that are at least comparable to the TA task, as we defined that task in Section 2.1. A comprehensive review of temporal reasoning approaches in general, and of their application to clinical domains in particular, and a comparison of frameworks applied in clinical domains to the KBTA method and to that method's implementation in the RÉSUMÉ system, is presented elsewhere [43]. Examples of systems performing a TA task that were implemented mainly for clinical domains include Fagan's [17] ventilation management (VM) system, Blum's [3] Rx system for knowledge discovery from time-oriented clinical databases; Downs' [14] summarization program for medical records; Kohane's [27] temporal utilities package (TUP); de Zegher-Geets' [13] IDEFIX system for summarizing patient visits; Russ' [42] temporal control structure system; Kahn's [25] TOPAZ system; the Guardian project [23]; Haimowitz and Kohane's [22] TrenDx system; and Larizza's [30] TA module in the M-HTP project.

Despite major differences among these systems, on close inspection most of them turn out to be solving tasks similar to the five fundamental subtasks into which the TA task is decomposed by the KBTA method. This decomposition, however, is never represented explictly, since it requires a task-specific approach. Furthermore, none of the previous approaches represents explicitly and declaratively the types of domain-specific knowledge on which the particular methods used to solve these tasks implicitly rely. The lack of explicit representation severely limits the reusability of these systems. Furthermore, most of these systems require the developer to encode the TA task as complex patterns (e.g., trend templates [22]) or programming language procedures (e.g., user-defined abstraction modules [42]). These approaches leave the tasks of both domain-independent and domain-specific TA to the developer. Unlike these approaches, The KBTA method and its implementation as the RÉSUMÉ system predefine ahead of time all task-specific, domain-independent TA inferences (i.e., the TA tasks and mechanisms), leaving the developer (possibly a domain expert) only the task of modification of arguments for these predefined, limited inferences (i.e., editing of the domain's mostly declarative TA ontology).

Thus, the KBTA method can also be viewed as a general *inference structure*, not unlike Clancey's [8] heuristic classification inference structure. The KBTA method makes explicit the subtasks that need to be solved for most of the variations of the TA interpretation task. These subtasks have to be solved, explicitly or implicitly, by any system whose goal is to generate meaningful interval-based abstractions from time-oriented data. The TA mechanisms that we presented for solution of these subtasks make explicit both the *tasks* that they solve and the *knowledge* that they require to solve these tasks. None of the approaches that we examined [43] focuses on the knowledge acquisition, knowledge maintenance, knowledge reuse, or knowledge sharing aspects of designing and building large knowledge-based TA systems. These approaches do not represent their inference strategy at the *knowledge level* [40], and thus might encounter several design and maintenance problems typical of knowledge-based systems. In particular, we would expect difficulties when we attempt

(1) to apply these approaches to TA tasks in new domains,
(2) to reuse them for related tasks in the same domain,
(3) to maintain the soundness and completeness of their associated knowledge base and its interrelated components, and
(4) to acquire the knowledge required to instantiate them in a particular domain and task in a disciplined and even automated manner.

4.1.1. Relationship to other knowledge-based problem-solving frameworks

It is interesting to compare the inference actions implied by the TA mechanisms with the basic inferential components existing in other knowledge-based problem-solving frameworks. Such a comparison would enable us to appreciate more to what extent the KBTA method is general and potentially sharable with frameworks substantially different from PROTÉGÉ-II. We collaborated with researchers in the European KADS-II project, a newer version of the KADS methodology for development of knowledge-based systems [56]. The TA mechanisms were compared with the KADS-II *primitive inference actions* (*PIAs*) [1]. A logical inference structure was constructed in KADS-II terms, which represented the essence of the contemporaneous abstraction, temporal inference, and temporal interpolation mechanisms. One of the main insights that we gained was that the TA mechanisms operate at a much higher level of inference granularity, as opposed to low-level, highly nonspecific PIAs such as SELECT or GENERALIZE. Thus, the KADS-II architecture has highly generalizable, low-level components; however, the tradeoff in using such highly modular components, and in representing all classification functions as, for instance, a SELECT operation, is that these components are not sufficiently task specific. For example, representing the temporal interpolation mechanism as a set of PIAs tends to obscure the way that the domain-specific TA knowledge is used by that mechanism. Also, a neutral representation volunteers no clue to what several researchers call the *knowledge use level* [53]. The knowledge use level is an intermediate level between Newell's [40] *knowledge level* and *symbol level* , and is an important step towards implementing methods presented at the knowledge level. This level seems highly useful for bridging the gap between theoretical problem-solving methods and working applications [53]. The knowledge use level is represented explicitly by the TA mechanisms' ontology of parameters, events, and contexts and the RÉSUMÉ system's knowledge structures.

4.2. Advantages of the RÉSUMÉ system for the temporal abstraction task

The KBTA method and its implementation as RÉSUMÉ introduce several conceptual and computational advantageous features. Advantages can be categorized by the following aspects:

(1) increase of flexibility in the representation and use of input and output data;
(2) enablement of nonmonotonic behavior;
(3) emphasis of context-sensitive interpretation;
(4) support for automated planning; and
(5) facilitation of acquisition, representation, sharing, and reuse of TA knowledge.

4.2.1. Input and output data of multiple types and at multiple abstraction levels

The parameter ontology represents, in uniform fashion, all clinical parameters (primitive or abstract) of all types—numeric and symbolic. The input data can be at different levels of abstraction (e.g., both raw data, such as height measurements, and higher-level concepts, such as the height standard deviation score abstraction). Thus, some input data can be partially abstracted by the user, or by another computational module, before they are entered into the RÉSUMÉ system. Important temporal patterns can be predefined at knowledge acquisition time, but the user of the application system can still query the temporal fact base at runtime for arbitrary temporal patterns and for parameter propositions at any level of abstraction.

4.2.2. Acceptance of input data out of temporal order

The inherent nonmonotonicity of the TA mechanisms is implemented in the RÉSUMÉ system by the truth maintenance system underlying the temporal reasoning process (see Section 3.3). The RÉSUMÉ system detects contradictions by using TA task-specific semantics (e.g., temporal semantic properties of parameter propositions). The truth maintenance system retracts conclusions that are no longer true, and propagates new abstractions to the rest of the temporal fact base. Thus, the past can change our view of the present; we call that process a *view update*. Furthermore, new data enable the RÉSUMÉ system to modify past interpretations; thus, the present (or future) can change our interpretation of the past, a process referred to as *hindsight* [42]. The hindsight task is performed by several components of the RÉSUMÉ system's architecture:

(1) the truth maintenance system;
(2) the context-forming mechanism, which can create both *prospective* and *retrospective* contexts dynamically; and
(3) the temporal interpolation mechanism, which has the ability to reason about both forward and backward persistence of belief in the truth of parameter propositions.

4.2.3. Context-sensitive interpretation

Interpretation contexts are separated logically from the propositions inducing them by DIRCs (Section 2.1). Abstractions are specialized in the parameter ontology by interpretation contexts. Interpretation contexts both reduce the computational burden and specialize the abstraction process for particular contexts, by enabling within their temporal context the use of only the temporal abstraction knowledge (e.g., mapping functions) specific to the interpretation context.

Explicit interpretation contexts, separate from the propositions inducing them and from the abstractions using them, have four significant conceptual and computational advantages for context-specific interpretation of time-stamped data. First, any temporal relation can hold between a context interval and the latter's inducing proposition. Interpretation contexts might be induced concurrently, in the future, and in the past, enabling a form of foresight and hindsight. Second, the *same context-forming proposition* can *induce* one or more context intervals (e.g., several potential effects). Third, the *same interpretation context* might be induced by *different* propositions. The separation of interpretation contexts from their inducing propositions facilitates maintenance and reusability of temporal abstraction knowledge. Fourth, parameter propositions include an explicit interpretation

context, thus enabling a representation of several abstractions in which the *same abstract parameter* (e.g., the Hemoglobin_state) has *different values* at the *same time*—one for each of the context intervals that holds during the relevant period. Thus, interpretation contexts support maintenance of several *concurrent* interpretations of the same data.

In addition, *generalized interpretation contexts* (a union of different, temporally meeting contexts) and *nonconvex interpretation contexts* (interpolation between similar, temporally disjoint contexts) enable sharing of abstractions of the same parameter among different contexts and temporal phases.

4.2.4. Separation of interpretation from planning

The KBTA method performs an interpretation task, separate from tasks such as planning actions or executing these actions. Such separation is useful for knowledge maintenance reasons and for task-specific reasons. For instance, although the *abstractions* identified by the two endocrinologists in the diabetes therapy domain (see Section 3.2) were remarkably similar, the *therapy recommendations* suggested by the experts differed significantly [43]. This observation validates a basic reason for performing the temporal abstraction task—namely, that intermediate conclusions from the data (the interval-based abstract concepts and patterns) are significantly more stable than are specific therapy rules predicated on these conclusions. Thus, knowledge about forming the intermediate patterns should be represented explicitly and separately from knowledge about appropriate action. The separation also allows a TA system to reason about the data offline, accessing data directly from a temporal database (e.g., an electronic patient record).

4.2.5. Facilitation of development, acquisition, maintenance, sharing, and reuse of TA knowledge

One of the major goals in constructing the KBTA model, and for specifying formally and explicitly the knowledge required by each of the TA mechanisms solving the subtasks of that method, was to facilitate, eventually, both manual and automated acquisition of that knowledge. The explicit representation supports acquisition of the knowledge necessary for applying the method to other domains, maintenance of that knowledge, reuse of the domain-independent TA mechanisms, and sharing of the domain-specific TA knowledge with other applications in the same domain. The organization of the TA knowledge in the RÉSUMÉ system into parameter, event, and context ontologies plays a major role in accomplishing these goals. The organization of the knowledge in the parameter ontology as subclasses of the four general abstraction types (state, gradient, rate, and pattern), with frame-based inheritance of general abstraction-type and domain-specific properties, further enhances the ease of designing new systems, acquiring the necessary knowledge, and maintaining the TA knowledge base.

An explicit representation of TA knowledge enables a designer to construct an automated knowledge acquisition tool that can be used directly by domain experts to augment the domain's TA ontology. Constructing such tools, when possible, has major benefits, mainly in facilitating the acquisition of knowledge without the intervention of a knowledge engineer. We have automated the acquisition of TA knowledge through the use of tools from the PROTÉGÉ-II project [54]. PROTÉGÉ-II is a framework for the design of knowledge-based systems [15,39] and includes tools that generate auto-

matically (and assist in custom tailoring) a knowledge acquisition tool. The input to these tools is the ontology of a problem-solving method mapped to the ontology of an application domain (in this case, the ontology of the KBTA method and the domain's general ontology).

4.3. Additional implications and extensions of the work

Many intriguing issues are raised by this investigation into the fundamental nature of temporal abstraction knowledge. In this section, we describe briefly several of the most interesting practical and theoretical issues.

4.3.1. Implications of the nonmonotonicity of temporal abstraction

Integrating a TA system with an external database creates several computational problems. An especially intriguing one is the inherent nonmonotonicity of temporal abstractions. This problem is solved in the RÉSUMÉ system by the use of a truth maintenance system (see Section 3.3). However, using a TA system, such as RÉSUMÉ, in conjunction with an external database might create inconsistency problems: the RÉSUMÉ system will update old conclusions in its temporal fact base as new data arrive; but the database system, not having access to the dependency links and the truth maintenance system, will also keep the old, incorrect conclusions. In addition, arrival of new data to the external database should be reported to RÉSUMÉ temporal fact base. Thus, we need to investigate whether the temporal fact base and the external database should be *tightly coupled* (each update is reflected immediately in the other database), *loosely coupled* (updates are sent intermittently to the other database), or not coupled at all. The choice might depend on the properties of the domain and on the capabilities of the external database (e.g., object-oriented databases handle links among entities more flexibly). The problem deserves further research.

A closely related issue is whether some, all, or none of the temporal abstractions should be saved in the database. Given that some of these abstractions are only intermediate, whereas other abstractions might be changed by future data (possibly with a past time stamp or having some influence on the interpretation of that past), it might be advisable not to save any abstractions, due to their logically defeasible nature. However, it is obviously useful, from an efficiency point of view, to cache several key conclusions for easy future use, either to respond to a direct query or to support another TA process. The caching is especially important for saving high-level abstractions, such as a significant pattern, that have occurred in the past, are unlikely to change, and are useful for interpreting the present. Such abstractions might be available for querying by other users (including programs), who do not necessarily have access to the RÉSUMÉ problem solver or to the domain's full TA ontology. One option is an episodic use of "temporal checkpoints" beyond which past abstractions are cached, available for querying but not for modification.

4.3.2. Implications for semantics of temporal database queries

One of the two main types of knowledge used by the temporal inference mechanism is *temporal semantic knowledge*, an extension of Shoham's [51] classification of the re-

lationship of predicates interpreted over one time interval to predicates interpreted over other time intervals. The temporal semantic properties of parameter propositions are valuable both for inferring further abstractions and for detecting contradictions among existing ones, leading to retraction of potentially incorrect conclusions and to propagation of the new conclusions by the truth maintenance system. The temporal semantic properties can be useful for performing similar functions in temporal databases. For instance, two tuples representing a certain predicate, whose temporal attributes refer to meeting time intervals, should not necessarily be concatenated for purposes of summarization. In addition, queries such as "was p necessarily true during time interval I?" or "is it *possibly true* that p held over a certain time interval?" when I has some temporal relation to an interval I_1 over which p is true, might be answered, among other means, by the use of temporal semantic properties. As mentioned in Section 2.5.1, the full use of the inferences implied by the temporal semantic knowledge might be viewed as a *modal logic* of combining propositions over time, and representing what is *possibly* true and what is *necessarily* true. Further research would be useful and can elucidate several issues in the semantics of queries referred to temporal databases.

4.3.3. Implications for visualization of time-oriented data

The ability to create meaningful, context-specific, interval-based abstractions from time-stamped data is a prerequisite for useful *graphic presentation* and *visualization* of those data. Such a presentation requires knowledge about temporal properties of the visualized primitive and abstract parameters. For instance, knowledge of typical persistence of parameter propositions, when data are not obtained continuously, coupled with temporal semantic properties, enables the creation of meaningful intervals and avoids creation of illegitimate or simply unrealistically long intervals.

Thus, the KBTA method can provide the semantics for a *temporal abstraction graphical user interface* to a set of parameter intervals. This interface will enable the user to manipulate the results of a TA query. Examples include highlighting or hiding of particular parts of the output, *zooming in* and *zooming out* operators whose predefined semantics employ relations in the domain's TA ontology (e.g., IS-A, ABSTRACTED-INTO, SUBCONTEXT), changing the temporal granularity level (e.g., from days to months), and focusing on particular time periods, interpretation contexts, abstraction types, or parameter classes. In addition, given those terms, a model specific to the user's preferences can be defined. Thus, the TA ontology can provide semantic operators that augment the syntactic time-line visualization operators suggested by Cousins and Kahn [9].

4.3.4. Implications for plan recognition and critiquing

An additional potential use for the KBTA framework is the enablement of an intelligent dialog between a human executing a plan and an automated assistant. We can facilitate such a dialog substantially by representing the *intentions* of the original designer of the plan as *TA patterns* (of both execution actions and world states) that should be created, maintained, or avoided [47]. For instance, a common format in clinical domains is a clinical guideline, which embodies a set of plans for managing patients of a particular category over time. For enhanced support, the automated

assistant needs to know what were the intentions of the guideline's author and to recognize the plans of the physician executing the guideline, assuming that the physician is aware of these intentions. To recognize and support the physician's plan, the automated assistant needs a library of plans represented in a uniform execution language that includes an expressive temporal abstraction syntax and semantics for representation of intentions, preferences, conditions, actions, and effects of plans and actions. We are developing such a language [44]. The automated assistant also needs a library of generic execution time plan revision strategies [47]. Such a representation enables, in principle, a *critiquing* approach to the support of planning [38] that realizes, given the physician's plan or actual actions, whether or not the physician is following the guideline's higher-level intentions and policies, even though she has not followed the prescribed actions literally. Thus, the system will still be able to support intelligently the decisions and plans of the human user. The ability for an intelligent, goal-oriented dialog with the user seems crucial for many knowledge-based decision support systems.

4.3.5. Implications for abstraction of data over other linear distance measures

The experiment in the traffic domain [45] (see Section 3.2) demonstrates the high level of reusability of the KBTA method for markedly different domains, and even for different distance measures (space and time), as long as these are linear. The results suggest that the KBTA method might be generalized into a *knowledge-based linear abstraction method*, which has an additional *dimension* argument. Performing the task of monitoring traffic-control actions required, in effect, two versions of this more general method: one for reasoning about time, and another for reasoning about space. Alternatively, using the PROTÉGÉ-II terminology, we might say that the knowledge-based linear abstraction method was *mapped* twice to the traffic domain: one time to the TA properties of the traffic parameters, thus creating a TA ontology of the traffic domain, and yet another time to the spatial properties of the traffic parameters, thus creating a spatial abstraction ontology for that domain. These two versions were then assembled to create a *knowledge-based spatiotemporal abstraction method.*

The linear distance measure does not need to be limited to time or space. In a separate study, we had shown that the TA ontology can model the task of context-sensitive information retrieval [49]. In that task, the distance measure is position of text within a document. Other potential tasks exist in the biomolecular domain—for example, matching of patterns, at several levels of abstraction, in DNA sequences.

5. Concluding remarks

One or more of the TA mechanisms that perform the five tasks into which the KBTA method decomposes the TA task can in principle be modified, or even replaced, without changing that method. (For instance, temporal interpolation might incorporate statistical methodologies.) Thus, the KBTA method can be viewed as an *inference structure* in the sense of Clancey's [8] heuristic classification method. The KBTA method and the RÉSUMÉ system implementing it embody a philosophy of making explicit the

subtasks involved in performance of the TA task, the mechanisms performing each subtask, and the domain-specific knowledge required by each mechanism to perform its task.

The knowledge used by domain experts to extract meaningful temporal intervals from a set of data is intricate and is largely implicit. This intricacy is reflected in the complexity of the TA ontology, which includes parameters, events, contexts, abstraction goals, and DIRCs; by the five domain-independent TA computational mechanisms; and by the four types of domain-specific knowledge these mechanisms require. Designers of knowledge-based systems cannot escape this complexity if they wish to support and maintain tasks that involve significant amounts of reasoning about and abstraction of time-stamped data. However, the well-defined knowledge roles of the KBTA method, and the use of automated knowledge acquisition tools, greatly facilitate the maintenance of the domain's or the particular application's TA ontology.

Our discussion of the KBTA method and of the RÉSUMÉ system suggests that both the TA task and the methodology proposed for solving it are relevant to many application domains other than clinical medicine. These domains are those in which interval-based abstraction of concepts from time-stamped input data is needed, and in which most of the features described in Section 4.2 are desired. The methodology that we presented is especially useful when several abstraction levels and data types exist as inputs or outputs of the TA task, when data might arrive out of temporal order, when several context-specific interpretations might need to be monitored in parallel, and when knowledge maintenance and reuse is relevant.

Furthermore, the example of the traffic-control task suggests that the KBTA method might be generalized to a knowledge-based *linear* abstraction method, whose task-specific knowledge might be mapped to a domain-specific (linear) distance measure (e.g., time, space). Many of the advantages and implications discussed in Sections 4.2 and 4.3 would then apply too.

Since the knowledge requirements of the TA mechanisms are well defined, the knowledge acquisition process can use either a manual methodology driven by the knowledge roles defined by the KBTA method and the TA mechanisms, or automatically generated knowledge acquisition tools, tailored to the domain and to the task, such as the knowledge acquisition tools generated by the PROTÉGÉ-II system. These tools can be used by a developer of a new TA system or by a domain expert maintaining that system.

Whatever the knowledge acquisition methodology chosen, however, understanding the knowledge required for abstracting data over time (and perhaps over other linear distance measures) in any particular domain is a useful undertaking. A clear specification of that knowledge, and its representation in an ontology specific to the task of abstracting concepts over time, as was done in the architecture of the RÉSUMÉ system, supports the design of knowledge-based systems that perform a TA task. The formal specification of the TA knowledge also supports acquisition of that knowledge from domain experts, maintenance of that knowledge once acquired, reuse of the problem-solving knowledge for TA tasks in other domains, and sharing of the domain-specific knowledge with other problem solvers that might need access to the domain's temporal reasoning knowledge.

Acknowledgements

This work has been supported in part by grant HS06330 from the Agency for Health Care Policy and Research, by grants LM05157, LM05305, LM05708, and LM06245 from the National Library of Medicine, and by grants IRI-9257578 and IRI-9528444 from the National Science Foundation. I thank Mark Musen, Richard Fikes, and Barbara Hayes-Roth for technical advice and support. I had many useful discussions with Samson Tu, Amar Das, and Michael Kahn.

References

[1] M. Aben, Y. Shahar and M.A. Musen, Temporal abstraction mechanisms as KADS inferences, in: B.R. Gaines and M.A. Musen, eds., *Proceedings of the Eighth Banff Knowledge-Acquisition for Knowledge-Based Systems Workshop* Vol. 2, SRDG Publications, Department of Computer Science, University of Calgary (1994) 28-1–28-22.

[2] J.F. Allen, Towards a general theory of action and time, *Artif. Intell.* **23** (1984) 123–154.

[3] R.L. Blum, Discovery and representation of causal relationships from a large time-oriented clinical database: the RX project, in: D.A. Lindberg and P.L. Reichartz, eds., Lecture Notes in Medical Informatics **19** (Springer, New York, 1982).

[4] B. Chandrasekaran, Towards a taxonomy of problem-solving types, *AI Magazine* **4** (1) (1983) 9–17.

[5] B. Chandrasekaran, Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design, *IEEE Expert* **1** (3) (1986) 23–30.

[6] B. Chandrasekaran, Design problem solving: a task analysis, *AI Magazine* **11** (4) (1990) 59–71.

[7] B. Chandrasekaran and S. Mittal, Deep versus compiled approaches to diagnostic problem solving, *Int. J. Man-Mach. Stud.* **19** (1983) 425–436.

[8] W.J. Clancey, Heuristic classification, *Artif. Intell.* **27** (1985) 289–350.

[9] S.B. Cousins and M.G. Kahn, The visual display of temporal information, *Artif. Intell. Med.* **3** (1991) 341–357.

[10] A.K. Das and M.A. Musen, A temporal query system for protocol-directed decision support, *Meth. Inform. Med.* **33** (1994) 358–370.

[11] A.K. Das, Y. Shahar, S.W. Tu and M.A. Musen, A temporal-abstraction mediator for protocol-based decision support, in: *Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care*, Washington, DC (1994) 320–324.

[12] T. Dean and K. Kanazawa, Probabilistic temporal reasoning, in: *Proceedings AAAI-88*, Minneapolis, MN (1988) 524–528.

[13] I.M. De Zegher-Geets, IDEFIX: intelligent summarization of a time-oriented medical database, M.S. Dissertation, Program in Medical Information Sciences, Stanford University School of Medicine, Stanford, CA (1987); also: Knowledge Systems Laboratory Tech. Rept. KSL-88-34, Department of Computer Science, Stanford University, Stanford, CA (1988).

[14] S.M. Downs, M.G. Walker and R.L. Blum, Automated summarization of on-line medical records, in: R. Salamon, B. Blum and M. Jorgensen, eds., *MEDINFO '86: Proceedings of the Fifth Conference on Medical Informatics* (North-Holland, Amsterdam, 1986) 800–804.

[15] H. Eriksson, Y. Shahar, S.W. Tu, A.R. Puerta and M.A. Musen, Task modeling with reusable problem-solving methods, *Artif. Intell.* **79** (1995) 293–326.

[16] L. Eshelman, MOLE: a knowledge-acquisition tool for cover-and-differentiate systems, in: S. Marcus, ed., *Automating Knowledge-Acquisition for Expert Systems* (Kluwer Academic Publishers, Boston, MA, 1988).

[17] L.M. Fagan, VM: representing time dependent relations in a medical setting, Ph.D. dissertation, Department of Computer Science, Stanford University, Stanford, CA (1980).

[18] K.D. Forbus, Qualitative process theory, *Artif. Intell.* **24** (1984) 85–168.

[19] C. Forgy, RETE: a fast algorithm for the many pattern/many object pattern-match problem, *Artif. Intell.* **19** (1982) 17–38.
[20] J.H. Gennari, S.W. Tu, T.E. Rothenfluh and M.A. Musen, Mapping domains to methods in support of reuse, *Int. J. Human-Comput. Stud.* **41** (1994) 399–424.
[21] J. Giarratano and G. Riley, *Expert Systems: Principles and Programming* (PWS Publishing Company, Boston, MA, 1994).
[22] I.J. Haimowitz and I.S. Kohane, Automated trend detection with alternate temporal hypotheses, in: *Proceedings IJCAI-93*, Chambery (1993) 146–151.
[23] B. Hayes-Roth, R. Washington, D. Ash, R. Hewett, A. Collinot, A. Vina and A. Seiver, Guardian: a prototype intelligent agent for intensive-care monitoring, *Artif. Intell. Med.* **4** (1992) 165–185.
[24] G.E. Hughes and M.J. Cresswell, *An Introduction to Modal Logic* (Methuen, London, 1968).
[25] M.G. Kahn, Combining physiologic models and symbolic methods to interpret time-varying patient data, *Meth. Inform. Med.* **30** (1991) 167–178.
[26] K. Kahn and G.A. Gorry, Mechanizing temporal knowledge, *Artif. Intell.* **9** (1977) 87–108.
[27] I.S. Kohane, Temporal reasoning in medical expert systems, Tech. Rept. 389, Laboratory of Computer Science, Massachusetts Institute of Technology, Cambridge, MA (1987).
[28] M.M. Kuilboer, Y. Shahar, D.M. Wilson and M.A. Musen, Knowledge reuse: temporal-abstraction mechanisms for the assessment of children's growth, in: C. Safran, ed., *Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care* (McGraw-Hill, New York, 1993) 449–453.
[29] P. Ladkin, Time representation: a taxonomy of interval relations, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 360–366.
[30] C. Larizza, A. Moglia and M. Stefanelli, M-HTP: a system for monitoring heart-transplant patients, *Artif. Intell. Med.* **4** (1992) 111–126.
[31] W.J. Long and T.A. Russ, A control structure for time dependent reasoning, in: *Proceedings IJCAI-83*, Karlsruhe, Germany (1983) 230–232.
[32] S. Marcus, SALT: a knowledge-acquisition tool for propose-and-revise systems, in: S. Marcus, ed., *Automating Knowledge Acquisition for Expert Systems* (Kluwer Academic Publishers, Boston, MA, 1988).
[33] J. McCarthy, Applications of circumscription to formalizing commonsense knowledge, *Artif. Intell.* **28** (1986) 89–116.
[34] J. McCarthy and P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer and D. Michie, eds., Machine Intelligence **4** (Edinburgh University Press, Edinburgh, 1969) 463– 502; also in: B.L. Webber and N.J. Nilsson, eds., *Readings in Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1981).
[35] D.V. McDermott, A temporal logic for reasoning about processes and plans, *Cognit. Sci.* **6** (1982) 101–155.
[36] J. McDermott, Preliminary steps toward a taxonomy of problem-solving methods, in: S. Marcus, ed., *Automating Knowledge Acquisition for Expert Systems* (Kluwer Academic Publishers, Boston, MA, 1988).
[37] M.A. Musen, Dimensions of knowledge sharing and reuse, *Comput. Biomed. Res.* **25** (1992) 435–467.
[38] P.L. Miller, *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer* (Springer-Verlag, New York, 1986).
[39] M.A. Musen, J.H. Gennari, H. Eriksson, S.W. Tu and A.R. Puerta, PROTÉGÉ-II: computer support for development of intelligent systems from libraries of components, in: *MEDINFO '95: Proceedings Eighth World Congress on Medical Informatics*, Vancouver, BC (1995) 766–760.
[40] A. Newell, The knowledge level, *Artif. Intell.* **18** (1982) 87–127.
[41] A.R. Puerta, J.W. Egar, S.W. Tu and M.A. Musen, A multiple method knowledge acquisition shell for the automatic generation of knowledge acquisition tools, *Knowledge Acquisition* **4** (1992) 171–196.
[42] T.A. Russ, Using hindsight in medical decision making, in: L.C. Kingsland, ed., *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care* (IEEE Computing Society Press, Washington, 1989) 38-44.
[43] Y. Shahar, A knowledge-based method for temporal abstraction of clinical data, Ph.D. Dissertation, Program in Medical Information Sciences, Stanford University School of Medicine, Stanford, CA

(1994); also: Knowledge Systems Laboratory Report No. KSL-94-64 (1994); also: Department of Computer Science Report No. STAN-CS-TR-94-1529, Stanford University, Stanford, CA (1994). URL: http://elib.stanford.edu:80/TR/STAN:CS-TR-94-1529?abstract=

[44] Y. Shahar, S. Miksch and P.D. Johnson, An intention-based language for sharing clinical guidelines, in: *Proceedings of the 1996 AMIA Annual Fall Symposium (formerly the Symposium on Computer Applications in Medical Care)* (Hanley and Belfus, Philadelphia, PA, 1996) 592–596.

[45] Y. Shahar and M. Molina, Knowledge-based spatiotemporal abstraction, in: *Proceedings AAAI-96 Workshop on Spatial and Temporal Reasoning*, Portland, OR (1996) 21–29.

[46] Y. Shahar and M.A. Musen, RÉSUMÉ: a temporal-abstraction system for patient monitoring, *Comput. Biomed. Res.* **26** (1993) 255–273; reprinted in: J.H. van Bemmel and T. McRay, eds., *Yearbook of Medical Informatics 1994* (F.K. Schattauer and the International Medical Informatics Association, Stuttgart, 1994) 443–461.

[47] Y. Shahar and M.A. Musen, Plan recognition and revision in support of guideline-based care, in: *Proceedings AAAI Symposium on Representing Mental States and Mechanisms*, Stanford, CA (1995) 118–126.

[48] Y. Shahar and M.A. Musen, Knowledge-based temporal abstraction in clinical domains, *Artif. Intell. Med.* **8** (3) (1996) 267–298.

[49] Y. Shahar and G. Purcell, The context-sensitive pattern-matching task, in: *Proceedings IJCAI-95*, Montreal, Que. (1995) 133–143.

[50] Y. Shahar, S.W. Tu and M.A. Musen, Knowledge acquisition for temporal-abstraction mechanisms, *Knowledge Acquisition* **4** (1992) 217–236.

[51] Y. Shoham, Temporal logics in AI: semantical and ontological considerations, *Artif. Intell.* **33** (1987) 89–104.

[52] R. Snodgrass and I. Ahn, Temporal databases, *IEEE Comput.* **19** (9) (1986) 35–42.

[53] L. Steels, Components of expertise, *AI Magazine* **29** (2) (1990) 28–49.

[54] A. Stein, M.A. Musen and Y. Shahar, A knowledge-acquisition tool for temporal abstraction, in: *Proceedings of the 1996 AMIA Annual Fall Symposium (formerly the Symposium on Computer Applications in Medical Care)* (Hanley and Belfus, Philadelphia, PA, 1996) 204–208.

[55] S.W. Tu, H. Eriksson, J.G. Gennari, Y. Shahar and M.A. Musen, Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: application of PROTÉGÉ-II to protocol-based decision support, *Artif. Intell. Med.* **7** (1995) 257–289.

[56] B. Wielinga, A.T. Schreiber and J. Breuker, KADS: a modeling approach to knowledge engineering, *Knowledge Acquisition* **4** (1992) 5–53.

[57] B.C. Williams, MINIMA: a symbolic approach to qualitative algebraic reasoning, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 94–99.