A Constraint-Based Specification of Periodic Patterns in Time-Oriented Data

Shubha Chakravarty and Yuval Shahar
Stanford Medical Informatics, Stanford Medical School
Stanford University, Stanford, CA, U.S.A.
{schakrav, shahar}@smi.stanford.edu

Abstract

We use a constraint-based language to specify periodic temporal patterns. The Constraint-based Pattern Specification Language (CAPSUL) is simple to use, but allows a wide variety of patterns to be expressed. CAPSUL solves problems such as (1) how to use calendar-based constraints to define repetition of a periodic event, (2) what temporal relations must exist between consecutive repeats of a pattern, and (3) how expressivity is limited if the same temporal relations must hold between each pair of intervals in the pattern. We implemented CAPSUL in a temporal-abstraction system called Résumé, and used it in a graphical knowledge-acquisition tool to acquire domain-specific knowledge from experts about patterns to be found in large databases. We summarize the results of preliminary experiments using the pattern-specification and pattern-detection tools on data about patients who have cancer and have been seen at the University of Chicago bone-marrow–transplantation center.

1. Introduction

Recognizing patterns in time-oriented data is useful in many domains, from medicine to astronomy. The task has been studied by computer scientists who specialize in artificial intelligence and in temporal databases. Some of the most interesting patterns are *periodic*: They consist of one or more elements that repeat over time. In large databases, which typically contain only raw data, it is often unwieldy to pick out elements that occur repeatedly, especially when those elements do not occur at regular intervals or are abstractions from several types of raw data that are not stored explicitly in the database. An example in the medical domain is a pattern of “normal” blood glucose levels in the “morning” and of “high” blood glucose levels in the “evening” (as these terms are defined in the context of a diabetic patient) that occurs at least three times per week four times per month, within a period of 3 months. Our system is designed to allow a domain expert to express patterns that range from simple to complex, which could then be recognized in large sets of raw time-oriented data and high-level abstractions. An example of the patterns with which we work is shown in **Figure 1**.

To express the patterns to be found, we had to define a language that characterizes a periodic pattern completely. We are developing such a language, and determining what kind of periodicity it can describe. CAPSUL does not allow the user to express every kind of periodic pattern possible; rather, it is simple enough to allow a user who has moderate mathematical and computer expertise to define complex patterns. For example, CAPSUL does not allow the definition of a pattern that repeats in a nonlinear fashion; that is, in CAPSUL, the relationship between each consecutive pair of repeating events must remain constant for the duration of the pattern. The user can express most useful types of patterns, yet neither CAPSUL nor the search algorithm is complex. CAPSUL comprises a set of different types and layers of constraints that together define completely a periodic pattern. Most of the constraints are not necessary to define a pattern; they can be used optionally to narrow the definition of the pattern.

The term *periodicity* may be misleading, in that the types of patterns that we studied do not necessarily repeat with any regularity. We can think of the periodic pattern as a series of intervals, during each of which exactly one instance of the repeating event occurs. In a truly periodic pattern, these intervals would be disjoint, with the end of one interval occurring before the start of the next interval. In the patterns that we studied, however, we found that such a constraint was not necessary. The intervals that compose a pattern may overlap, meet, or end together; two consecutive intervals may have any of the 13 binary relations in Allen’s temporal algebra for intervals [Allen, 1983]. It is convenient, therefore, to think of a periodic pattern as a set of intervals (without being concerned about what occurs during these intervals), to separate the temporal relations between events from the events themselves. Since there are no imposed restrictions on the

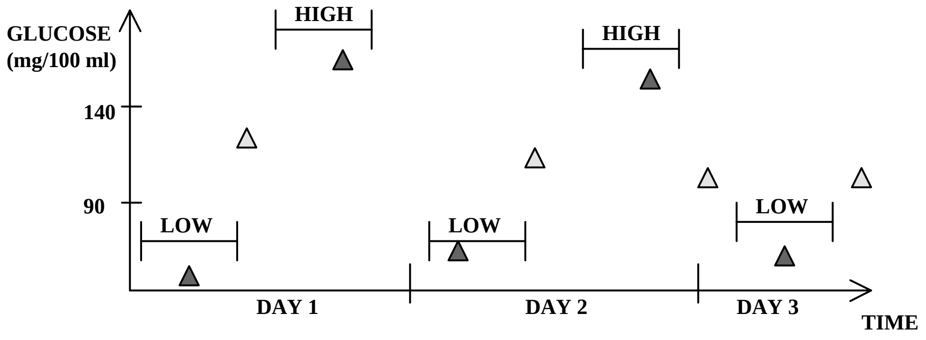

Figure 1. A pattern of recurring intervals of low glucose in the morning and high glucose in the evening. The triangles in the diagram represent individual blood glucose measurements. Dark triangles represent the data that is part of the pattern, while light triangles correspond to "normal" glucose values which are not part of the pattern. Note that the abstract terms "morning", "evening", "high", and "low" must be defined. In this example, a "high" level of glucose is defined to be above 140 mg/100 ml and a "low" level is below 90 mg/100 ml.

temporal relations between intervals, the patterns may be defined simply by cardinality (an event occurs twice), or by a complex combination of constraints (all the events start at the same time, but each successive event has a longer duration than its predecessor). Because we do not require that each interval end *before* the start of the next, we can extend the range of patterns that can be expressed in the language.

We have studied such patterns within the framework of the temporal-abstraction system **Résumé**, which is used primarily in medical domains, although it is domain independent. Résumé uses **knowledge-based temporal abstraction (KBTA)** [Shahar, 1997] to form useful abstractions from time-stamped raw data. The system can then use the Résumé abstractions to form additional higher-level abstractions. While developing such higher-level abstractions, we found it useful to be able to express and detect patterns. We are currently experimenting with the system in oncology; with the help of an expert in this area, we have tested the expressivity of CAPSUL as well as the effectiveness of our tools for acquiring and detecting periodic patterns in data.

In Section 2, we describe previous and related work in representing temporal repetition and calendar-based time. Section 3 provides a brief description of the Résumé temporal-abstraction system. We present the details of CAPSUL, our language for representing periodic patterns, in Section 4. Section 5 discusses the expressive power of CAPSUL and contrasts CAPSUL to previous work. The software system and the overall process used for acquiring domain-specific knowledge about periodic patterns are detailed in Section 6. Much of the design of CAPSUL followed directly from our experiences in designing a usable interface through which to acquire such knowledge. Section 7 describes what the mechanics of our interpretation process are and how we detect these periodic patterns in large databases. In Section 8, we present a preliminary experiment using our pattern-detection tool on oncology data gathered in the University of Chicago bone-marrow–transplantation center. The final section discusses our results and offers directions for future work.

2. Previous Work

Most work in the area of temporal reasoning involving interval-based representations of time is based on the work of Allen [1983], who defined an interval algebra of 13 basic binary temporal relations for convex time intervals. A **convex time interval** is an interval that contains no gaps. Ladkin [1986a] developed a taxonomy of binary relations between nonconvex intervals. His relations allow a degree of *fuzziness,* such as the relation *mostly meets* for nonconvex intervals *i, j*; that relation requires all the subintervals of *i* to have the relation *meet* with all the subintervals of j, but allows j to contain extra subintervals that do not correlate to any subintervals in *i*. In that work, Ladkin also showed that the number of relations between unions of convex intervals is at least exponential in the number of convex subintervals, a result that proved useful to us, as discussed in Section 5.

In another work, Ladkin [Ladkin 1986b] dealt with the representation of calendar-based time. He developed a formal representation of repeating intervals, such as *Mondays*, that we referred to when incorporating repeating events into our framework. In a paper dealing with the representation of events in calendar time, Clifford and Rao [Clifford and Rao, 1988] identified different types of elements in their temporal universe, all of which could be represented in different time units. Like us, they disallowed the problematic use of weeks as a time unit: Weeks can overlap with months and years and therefore do not fit into the natural subset chain of <seconds, minutes, hours, days, months, years>.

Several people have proposed frameworks similar to ours in which to define repeating events. Cukierman and Delgrande [1996] studied calendar-based repeating temporal objects. In this language, a series of n repeats is represented as follows: $<r_1, g_1, r_2, g_2, ..., r_n, g_n>$, where r refers to a repeat (an instance of the repeating event) and g refers to a gap. An important component of the Cukierman language is the pattern of gaps $< g_1, g_2, ..., g_n >$, which allows each gap to be the same, to be different, or to be defined probabilistically. More recently, Cukierman and Delgrande [1997] introduced a revised framework for representing repetition with the notion of a *time loop* that is made up of a cycle (or a repeat) and the relations between cycles. Furthermore, this loop may be *flattened* to form a *closed-view representation* of the sequence of instances that constitutes the loop's semantics. Thus, all repeats can be represented in only a single cycle, so the object that repeats does *not* vary with time. In the Cukierman–Delgrande language, loops are represented in closed time: A pattern of Mathematics class on Mondays and English class on Tuesdays is represented as the convex interval <Mathematics before English> that repeats weekly. Representing this same pattern in Ladkin's language involves correlating the two nonconvex intervals <Mathematics on Mondays> and <English on Tuesdays>. Cukierman's representation was useful to us; CAPSUL operates entirely under this closed-view scheme.

The temporal-repetition language presented by Terenziani [1997] contains many important distinctions that we used in CAPSUL. For example, Terenziani introduces a distinction between a *time frame* (which can be seen as a global "envelope" in which the entire pattern occurs) and a *period* (which could be a specific Monday on which one instance of the pattern occurs) that we use as well. Furthermore, Terenziani's constraint-based language distinguishes local and global constraints (where *local* constraints are applied to instances of the repeating event, and *global* constraints apply to the overall pattern) and between qualitative and quantitative constraints.

Morris and Khatib [1998b] discussed a set of temporal constraints in detail. In their representation, gaps can be specified either by a fixed length or by a probability distribution. They also discussed the idea that there is a set of possible constraints, but that only a subset is necessary to specify completely a repeating event. More recently, Morris and Khatib [1998a] have used a matrix of binary relations to represent the temporal relations between every pair of nonconvex intervals in a pattern, where the pattern consists of just two repeating elements. This matrix idea, which corresponds to Ladkin's argument that allowing temporal relations to differ between every pair of elements leads to an exponential number of possible ways to assign such relations to a pattern, helped us to think about the advantages and limitations of allowing only one relationship to be specified for all the pairs of consecutive intervals.

We have found that, in all these works, a *periodic event* generally is considered to be an occurrence that repeats at regular intervals, where each event occurs before the next. We modified this notion to define patterns that exhibit an *episodic*, less regular, pattern of repetition.

3. The Résumé Temporal-Abstraction System

Much of our work in creating a language for the representation of periodic patterns was made easier because we used the framework of the well-defined temporal-abstraction system called **Résumé** [Shahar and Musen, 1996]. The framework underlying the Résumé system [Shahar, 1997] is the knowledge-based temporal-abstraction (KBTA) method. Résumé is designed to create interval-based abstractions from time-stamped data. The components of Résumé include a time-stamped database (e.g., a database of patient records), a knowledge base containing the domain ontology (i.e., the knowledge required by the KBTA method), and an interpreter that forms abstractions by applying the knowledge to the time-stamped data.

The database used in the Résumé system includes time-stamped parameters (e.g., the patient's platelet count) and events (e.g., a surgical procedure). The ontology is organized as an IS-A class hierarchy that contains all the domain-specific knowledge that Résumé uses to create abstractions. The four major knowledge classes in the hierarchy are parameters (which include raw data, such as hemoglobin level, or their abstractions, such as the level of anemia), events (external interventions, such as administration of chemotherapy), contexts (induced by parameters or events) [Shahar, 1998], and patterns. Knowledge is acquired from a domain expert who, using a graphical tool, creates instances of these classes that pertain to the domain at hand. For example, an expert physician can create an abstract *anemia* parameter and supply the ontology with all the knowledge needed for Résumé to detect and classify the level of anemia as determined by the patient's hemoglobin levels. However, the expert would first have to define the parameter *hemoglobin* in the ontology. Hence, for any relation (in this case, abstracted-from), the ontology has many levels, and the expert must define not only the instances of classes, but also the relations between different instances.

The output of Résumé is a set of contexts and abstractions, where the abstractions are of type state (low, high), gradient (increasing, steady), rate (fast, normal), or pattern. The pattern class in the ontology has two subclasses corresponding to *linear* and *periodic* patterns. The distinction between linear and periodic patterns is that each input component of a linear pattern occurs only once, and the entire pattern is defined explicitly by the user at acquisition time, whereas a periodic pattern consists of any repeated occurrence (typically, a linear pattern); the instance is defined only once, with information about its repetition. The pattern-representation language guided our development of the acquisition interface for patterns, which allows the expert to fill in slots that define the constraints on and content of the patterns. Once the patterns (and the rest of the ontology) are acquired, the Résumé system applies the knowledge about patterns to the patient data to form abstractions that are relevant for the patient.

An important aspect of the Résumé system is the highly flexible representation of time. To work effectively in medical domains, Résumé can use absolute (3/7/92), relative (24 hours after the start of therapy), and calendar (the same *day*) representations of time. Résumé has an efficient set of functions that can translate among these different representations of time, so that it can work with a variety of time-stamped data. Résumé uses an interval-based time representation, in which each interval is defined by a starting time and ending time; thus, the time primitives are time stamps. Abstractions are interpreted only over intervals, with each interval denoting the period over which the abstraction was true. When a pattern is found, it is asserted in memory as an interval that represents the *convexification* [Ladkin 1986a] of all intervals contained in the pattern.

A fundamental concept in the Résumé system is that of *bootstrapping*. In Résumé, all raw data, primitive parameters, abstracted parameters, and events are contained together in a general pool of instances, so that further abstractions can be made from either low-level data or high-level abstractions. Thus, once an abstraction has been created (i.e., asserted in memory over some interval), it can be used as input data so that other higher-level abstractions can be formed from it. Using this feature, the expert defines the repeating component as a pattern, parameter, event, or context (unless it already exists in the ontology); to define a new periodic pattern, the expert just *selects* the appropriate repeating component, and then defines the temporal periodic constraints pertaining to it. In this way, patterns can be composed not only of raw data but also of abstractions and other patterns.

4. Language for Specification of Periodic Patterns

The basic structure of a periodic pattern consists of a repeating component and a set of periodic constraints that define how this component repeats. This structure corresponds to other languages that distinguish the temporal objects of a series from the relations among those objects. The repeating component defines the core of what is repeated through the pattern. As part of this component, all *interval* constraints that are specific to the component (i.e., that have nothing to do with how the component repeats) are defined. When the *periodic* constraints are defined, all the instances of the repeating component are treated as simple intervals, each of which have a start time, an end time, and a value. Thus, the constraint set consists of one gap relation, one value relation, and so on, which apply to every pair of components of the pattern.

So that we can understand the distinction between repeating components and periodic constraints, consider the following example: We want to define a pattern of weekly blood transfusions, each of which is followed by a high platelet count, then by moderate to severe anemia. The pattern occurs at least twice per month for 2 consecutive months, and the level of anemia does not decrease during this time. (This pattern occurs often after a patient undergoes a bone-marrow transplant and then receives frequent blood transfusions. This particular pattern would alert the physician that the patient has a problem, since the severity of anemia continues to increase although the patient should be recovering.) In this example, the repeating component is the linear pattern composed of blood transfusion, high platelet count, and moderate anemia. The period of the repeating component is 1 week: Each instance of the linear pattern must occur within the span of any 7 consecutive days (without regard to whether the week corresponds to a calendar week). The periodic constraints are used in this pattern to require that the repeating component occurs twice per month for 2 months.

The constraints that define both the repeating component and the cyclical constraints are divided into two main classes: value constraints and temporal constraints. **Value constraints** specify constraints on the value of the components in questions, both the values of the components (e.g., moderate to severe anemia) and the relations among different components (e.g., nondecreasing values over each pair of components). **Temporal constraints** define everything else about the components, both at the local level (defining the period in which the components occur and what their duration is) and in relation to one another (which occurs first, what the gap between consecutive events is).

When defining a temporal constraint, CAPSUL represents time in calendar units: seconds, minutes, hours, days, weeks, months, years. A period of time is either *fixed* or *floating*: The distinction is that a fixed time period is an amount of time (e.g., 7 to 9 A.M. on 1/1/98), while a floating time period is a certain interval of time (e.g., 10 months after surgery, whenever surgery occurred). Time can also be measured in *absolute* or *calendar* terms. An absolute day is 24 hours, even if those 24 hours occur from 4 P.M. to 4 A.M. By contrast, a calendar day is a 24-hour period that starts at midnight and ends at the next midnight.

A time period can be alternatively defined as a **repeating interval** such as *weekends*, which defines a fixed period of any particular week. Repeating intervals are defined using five ranges: minutes, hours, days of the week, months of the year, and years. For example, weekends are defined using the full range of all 5 time units except for days of the week, which is restricted to (Saturday, Sunday). To further narrow this repeating interval to *Weekends in Summer* we could select the range of months (June, August). We have not solved completely the problem of weeks, although we can specify such constraints as *within 2 to 4 days within the same week*, meaning that the interval of 2 to 4 days does not spread across more then one calendar week. As described in Clifford and Rao [1988], weeks are a separate class of time unit into which months cannot be evenly partitioned. If the repeating interval *Weekend* is the period of a pattern, the entire pattern falls into the same weekend. For a pattern in which each occurrence falls on a weekend, *Weekend* is the period of the repeating component rather than the period of the entire pattern.

Patterns—both linear and periodic—do not necessarily have one unique set of components that define the pattern entirely. In CAPSUL, the user can define the same pattern in several different ways, by using a logical disjunction at the top level of the pattern definition. All sets of components are defined as one clause of a disjunction, so the pattern specification is in **disjunctive normal form (DNF)**. For example, the pattern "Blood transfusion event followed by an alarming hematological state abstraction" could be defined as (transfusion AND low platelet count) OR (transfusion AND severe anemia). This disjunction exists in both linear and periodic patterns, and at the levels of the repeating component and the periodic constraints. Furthermore, each component set consists of not only a conjunction of components but also the corresponding constraint set, which contains local, global, and period constraints. Note that this DNF format limits the depth of the logical AND–OR tree that corresponds to the pattern specification to 2, thus considerably facilitating the acquisition and maintenance of patterns in the graphical KA tool. (We have found that an unlimited AND–OR tree is not easily understandable to users, in informal experiments).

The specification of the periodic pattern is divided into two stages: the repeating component and the periodic constraints. By separating the content of each interval from the temporal relations between the intervals, we can separate **local constraints,** which define the component internally, from the **global constraints,** which define the relations between the intervals. Once the user has defined the linear component, she can use it to define several patterns in which the temporal repetition of the component varies. While defining periodic constraints, she can think of each component as a convex interval, without worrying about the contents of the component. Space limitations prevent us from including a formal syntax of the language in this paper; instead we describe each type of constraint in detail.

4.1. Specification of the Repeating Component

The repeating component defines what occurs during each interval of the pattern. The component can be a single event (e.g., an administration of insulin), a primitive or abstract parameter (e.g., a *high* temperature), an entire linear pattern (e.g., a blood transfusion followed by high platelet count followed by a period of moderate anemia), or even another periodic pattern. As an example, we shall consider the **linear pattern**, which is a complicated type of repeating component. At the level of the linear pattern, several important constraints and components can be defined.

4.1.1. Linear components. A linear component defines the contents of a linear pattern. The set of linear components forms a logical conjunction. For example, the pattern “blood transfusion followed by high platelet count followed by moderate anemia” has the linear components blood transfusion (an event), platelet count (a primitive [raw-data] parameter), and anemia (an abstract parameter). The linear component can also be defined as a class variable; the user can specify a “hematological state abstraction,” which refers to the abstraction of platelet count, red-blood-cell count, or granulocyte count, rather than specifying “anemia” as part of the pattern.

4.1.2. Local constraints. A local constraint applies to a particular component of the linear pattern. The local constraint can specify either the value of a component (e.g., the level of anemia) or the time interval in which the component occurs (e.g., platelet count must be high for at least 6 hours). Value local constraints use relational operators (equal to, not equal to, greater than, etc.) to define a possible range for the value of a component. Temporal local constraints include a fuzzy start interval (7 to 8 A.M.), a fuzzy end interval (10 to 11 A.M.) and a fuzzy duration (2 to 3 hours). This method for describing fuzzy time intervals corresponds to the one described by Rit [1986]. The final type of local constraint is a negation operator, which specifies the absence of a component in the pattern.

4.1.3. Global constraints. A global constraint defines the relations between different components of the linear pattern. Like a local constraint, a global constraint defines either the temporal or value relations between components. Furthermore, the global constraint can be qualitative or quantitative. In the previous pattern, a global temporal quantitative constraint would specify that the high platelet count must begin within 6 hours after the transfusion, and a global temporal qualitative constraint would specify that the period of anemia must come after the period of high platelet count.

4.1.4. Period constraints. A period constraint defines a maximal and/or minimal period during which the pattern occurs. The period constraint can be used at any level of the pattern, so individual components as well as the whole pattern can be constrained to fit during some period. As discussed previously, a period can be expressed in calendar or absolute terms, and as a fixed or floating amount of time. In our example, the maximum period of the pattern would be defined as 7 absolute days, since we do not care whether it corresponds to a calendar week (Monday through Sunday). Note that this period defines the time interval in which the linear pattern must occur; it does not tell us how often or at what intervals the linear pattern repeats.

4.1.5. Statistical Constraints. A statistical constraint defines a function that is applied to the instances of the linear component. The result of applying this function is assigned as the value of the linear component. A few predefined functions are available to the user, such as average, standard deviation, minimum, and maximum. In addition, the expert may define her own function to be applied to the abstractions of any quantitative parameter. For example, one component of a linear pattern could be "average temperature of the patient during 1 week." In our previous example, we could change the "high platelet count" component to be "high *average* platelet count." Our use of the term *statistical constraint* does not imply that CAPSUL supports the use of statistical distributions in defining these constraints. For example, we cannot define constraints requiring that the pattern of platelet counts follows a uniform distribution.

4.1.6. Necessary contexts. A necessary context is one that must be present in order for each component or for the pattern to be true. Note that, in different contexts, the same value of the same component may have different definitions. The "normal" platelet count for a patient who has an oncological illness has a value lower than that for a patient who has diabetes. The necessary context constraint allows the user to define two separate patterns, one for each case, or to define just the pattern that is relevant for the context she has in mind [Shahar, 1998].

4.2. Specification of the Periodic Constraints

After defining the linear component, the user defines the periodic constraints that specify how the component repeats. At the level of periodic constraints, each instance of the linear component is handled as a convex time interval, and we are not concerned with its contents. Each type of periodic constraint adds a new dimension to the complexity of a periodic pattern. Users can define a pattern using all, any combination, or none of these constraints. A simple periodic pattern consists of a component that repeats a certain number of times: the only periodic constraint that is used in this case is the cardinality constraint, which defines a minimum number of times that the component repeats. More complicated patterns emerge when the constraints include clusters of consecutive events, subpatterns in the values of the repeating component, or global periods in which the patterns exist.

4.2.1. Global Start and End Time Constraints. The global start and end time constraints specify an envelope of time within which the pattern occurs. These times can be specified in absolute terms (a date and time) or in relative terms (a reference to a parameter or event defined previously in the knowledge base). These constraints may correspond not to the actual starting and ending times of the pattern, but rather to the earliest and latest points at which the pattern could start and end. For example, suppose that the global start and end times are June 1 and 30, for a pattern consisting of a certain event occurring on

Tuesdays. In this case, rather than starting on June 1, the pattern will start on the first Tuesday of the month.

4.2.2. Cardinality Constraints. Cardinality constraints define the number of times that the component repeats. The cardinality is expressed as a range to allow for fuzziness. For example, the transfusion example has a cardinality constraint that requires the component to repeat at least four but no more than eight times.

4.2.3. Local Gap Constraints. Local gap constraints define the quantitative distance between consecutive pairs of the repeating component. The gap is defined as an amount of time, and, just like a time period, can be measured in absolute or calendaric terms. The gap can be measured in four ways: beginning to beginning, beginning of first to end of second (which is often called the convex distance [Cukierman 1996]), end of first to beginning of second, and end to end. Also, the gap is defined as a range, allowing a degree of fuzziness in the actual distance between the intervals. This local gap constraint is defined once for the entire pattern; the user cannot define a separate gap constraint between each pair of intervals. This design decision, although it limits the scope of patterns that can be defined, reduces the time required to define a pattern. Also for the sake of simplicity, rather than allowing probability distributions, as suggested by Morris and Khatib [1998b] and by Cukierman [1996], CAPSUL specifies all gaps as a range of values.

4.2.4. Global Gap Constraints. Global gap constraints define a qualitative relation between gap *i* and gap *i+1* that must hold throughout the entire pattern. Note that the local gap constraints define the distance between pairs of intervals, whereas the global gap constraints specify the relation between the gaps themselves as increasing, decreasing, or constant. The user specifies at most one relation (or disjunction of relations) for the entire pattern.

4.2.5. Value Constraints. Value constraints define the relationships between the values of consecutive instances of the repeating component. The value depends on the component's type. For example, the value of the anemia might be on an ordinal scale from severe to normal. Value constraints are either qualitative or quantitative, just like gap constraints. Thus, a period of anemia could be less severe than the next (indicated by an *increasing* constraint), or the difference in the value of the hemoglobin level could be 2. Once again, only one value constraint can be defined for the entire pattern; it cannot be specified explicitly for each pair of intervals.

4.2.6. Cluster constraints. A cluster within a periodic pattern is a set of instances of the repeating component that occur without any regularity within a specific time period. The time period is defined in any of the ways discussed. Cluster constraints are one of the most useful periodic constraints because they allow the occurrences of the repeating component to be aggregated temporally into one or more sets. For example, the user can specify that the periodic pattern (whose envelope might span 6 months) contains at least three clusters, each over a period of up to 3 weeks, which obey all the constraints of the overall pattern. A cluster is defined as a period and a cardinality that represents the number of such clusters in the overall pattern. In the earlier transfusion example, we would define a cluster of repeating components in which the period was 1 (absolute, noncalendaric) month and the cardinality was 2.

Furthermore, each defined cluster may have one or more subconstraints associated with it; they describe the relationships among the events within the cluster. The subconstraints may specify the number of events per cluster, the gaps between the events in a cluster, or even a cluster that occurs within another cluster. Any of the constraints discussed in this section are valid subconstraints within a cluster. Clusters can also define a time period in which the entire pattern must occur, simply by selecting all the intervals of the pattern, rather than only a subset of them. In the earlier example, a cardinality subconstraint would require that at least two instances of the repeating component be found within each cluster.

4.3. The output of the pattern-matching process

In addition to the repeating component and the periodic constraints, which serve to define the constraints of the periodic pattern, the user may also specify what the pattern should look like when it has been found. First, she can specify the start and end times of the pattern. Within the Résumé system, each concluded parameter or pattern becomes another interval from which other parameters may be abstracted. The interval of a pattern consists of a start and an end point. The user may define the start and end of the pattern to be the start or end of any of intervals contained within the pattern. These slots are entered as the index of the desired instance (e.g., the second) and which part (beginning of, end of) of that instance is to be the boundary of the outputted pattern.

Second, the user may specify the value of the pattern. The default output value is TRUE, meaning simply that the pattern was found in the database. However, since patterns can be the components of other patterns, it is

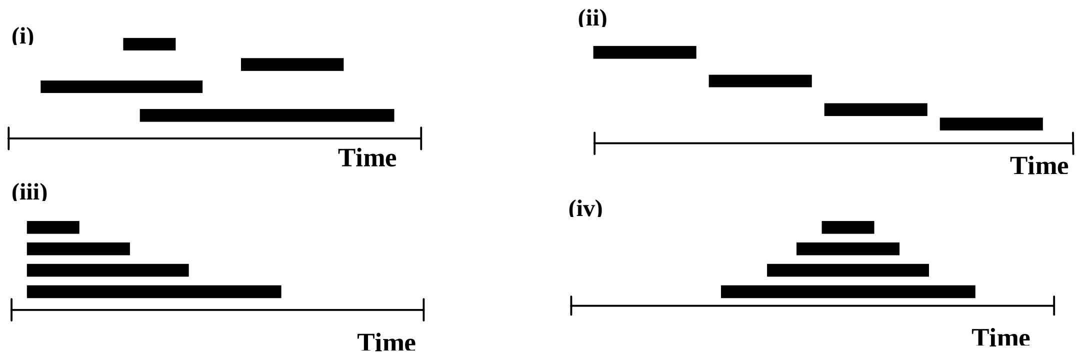

Figure 2. Diagrammatic representations of several patterns that can be specified using our language.

(i) A seemingly "random" assortment of intervals, using only the cardinality constraint.

(ii) A standard periodic pattern, where each intervals falls *before* its successor.

(iii) A pattern of intervals with increasing duration, where the relation between consecutive intervals is *starts*.

(iv) A pattern of intervals with increasing duration, where the relation between consecutive intervals is *within*.

often useful to assign values to the pattern, such as *severe*, thus allowing different instances of the same pattern to be compared. The output value could be simply a one-to-one mapping of the value of one of its components. It could also be a simple mathematical (or statistical) function (such as the sum of the values of all its components).

5. Expressivity of the Language

As mentioned earlier, CAPSUL allows users without much computer or mathematical expertise to define complicated patterns, yet it maintains reasonable computational complexity. In other words, we designed CAPSUL to be as expressive and tractable (for users and for program) as possible. What exactly is the class of patterns that can be expressed by CAPSUL?

There are several design decisions that extend CAPSUL, but that prevent it from expressing *all* possible patterns; for example, we cannot represent patterns that consist of a set of instances that repeat according to a Fibonacci series of temporal durations. The constraints described in Section 4 allow a good deal of fuzziness and thus help us to define patterns realistically. For example, we allow temporal gaps to be specified as a range of durations, rather than one fixed time period. In real-world patterns, it is difficult for an expert to provide a precise amount of time between components of the pattern; for example, the time between a blood transfusion and a detection of the resulting rise in the number of red blood cells could vary from 1 to several hours. By allowing the expert to define this as a range (1 to 6 hours), we can represent a more realistic relation between the two events.

Another feature that extends the expressive power of CAPSUL is that we allow temporal relations between consecutive instances to be any of the seven distinct Allen relations (we combine before and after into just one relation, *before*, and allow the user to specify which interval comes first). Because we do not restrict this choice to the *before* relation, as is standard in defining periodicity, CAPSUL can express many interesting types of patterns. The intervals can overlap in time, rather than having one interval always being placed completely before the next. Also, the intervals may all occur at the same time, and their values may be what distinguish them from one another. All the intervals can begin at the same time, but each have consecutively longer durations. (**Figure 2**).

On the other hand, the decision to limit periodic patterns to at most one repeating component restricts the set of patterns that can be expressed in CAPSUL. Recall that, in the definition of a periodic pattern, the user must specify everything about what repeats in a single repeating component, which can be a single parameter or a complex pattern. In essence, we are allowing into CAPSUL only those patterns that can be flattened into one component, as described by Cukierman [1998]. This component must be representable as a closed interval and cannot change over time. This limitation serves to disallow patterns in which the intervals vary in a nonlinear fashion. If, at the beginning of the pattern, the repeating component lasts 1 week and consists of a mathematics class followed by an English class, we cannot create a periodic pattern in which the mathematics class temporally approaches the English class and eventually comes *after* the English class. The latter pattern would require that the repeating component change over time and that the instances of the repeating component (which are treated as a single uniform instance) do not maintain the same temporal relation throughout.

CAPSUL further limits expressivity by requiring that gap and value relations that relate interval *i* to interval *i+1* remain constant for all pairs of intervals. As mentioned earlier, allowing these relations to vary would have introduced an exponential complexity in the number of possible relations between all the components of the pattern [Ladkin 1986]. Consider the periodic pattern of *English class before mathematics class every week*, where all occurrences of English classes are listed on the vertical axis of a matrix and all occurrences of mathematics classes are listed on the horizontal axis of a matrix (**Figure 3)**. Each slot in this matrix represents one instance of the repeating component of our pattern. The repeating component in this case is the linear pattern of *English before mathematics in 1 calendar week*. This same periodic pattern would be described by Ladkin [1986a] as a relationship between the two nonconvex intervals *English class* and *mathematics class*. For this pattern, we define the temporal constraints between each (*English*, *mathematics*) pair to be: (*before* and *during the same week*). We can then evaluate each slot *(i, j)* in this matrix as *true* if English class *i* and math class *j* satisfy the constraints of the pattern.

Because it limits the user to defining only *one* gap and value relation between interval *i* and *i+1*, CAPSUL is limited to the class of patterns for which there exists in the matrix one crucial diagonal (not necessarily a major one) along which the temporal and value constraints are true (see **Figure 3**). Thus, any representative instance pair (E_i, M_j) that lies along this crucial diagonal defines the entire periodic pattern. If no such diagonal exists for which the user can fill in the same set of constraints in each slot, then the pattern cannot be specified in CAPSUL. This definition can be extended similarly to *N*-dimensional matrices for *N* nonconvex intervals (e.g., a partial three-dimensional diagonal in a cube matrix).

Note that the diagonal does not need to contain consecutive *true* values along its entire length: If the pattern contains only the cardinality constraint, for example, then the diagonal that represents the pattern could have values of *false* interspersed with a correct number of true instances. Furthermore, a successive list of *true* values in the matrix does not guarantee the presence of the pattern. CAPSUL allows global constraints, which, within this matrix representation, function as constraints that must hold *between* slots of the matrix. Thus, this matrix representation helps us to understand the limitations involved in allowing only one repeating component, but the matrix does not provide a complete representation of the pattern language.

6. Acquisition of Pattern Specifications

Using CAPSUL, a domain expert can define patterns of interest that Résumé can then search for in a database. These patterns are acquired through a graphical **knowledge- acquisition (KA) tool.** The addition of tool using the Protégé set of tools [Musen, 1998] as an

		Occurrences of mathematics class						
		1	**2**	**3**	**4**	**5**	**6**	**…**
Occurrences of English Class	**1**	**False**	**False**	**False**	**False**	**False**	**False**	
	2	**True**	**False**	**False**	**False**	**False**	**False**	
	3	**False**	**True**	**False**	**False**	**False**	**False**	
	4	**False**	**False**	**True**	**False**	**False**	**False**	
	5	**False**	**False**	**False**	**True**	**False**	**False**	
	6	**False**	**False**	**False**	**False**	**False**	**False**	
	⋮							

Figure 3. A matrix representation of the periodic pattern "English class before Math Class every week". Note that the pattern begins at the second occurrence of English class, which corresponds to the first occurrence of math class. The highlighted diagonal represents the pairs of occurrences that satisfy the temporal and value constraints of this pattern. This matrix representation is an extension of the one developed by Khatib [1994].

extension of the Résumé KA tool [Shahar et. al., 1998].

In the graphical user interface of the KA tool, each class is represented as a separate window with slots to allow the user to specify the various values of the parameters, events, and patterns. As much as possible, the design of the user interface followed directly from the language of periodic patterns. Since each periodic pattern consists of a repeating component and a set of periodic constraints, the periodic-pattern windows in the KA tool contain slots for the repeating component and the set of periodic constraints. Each of the eight types of constraints is each maintained as its own subclass. The slot for periodic constraints contains a list of the constraints defined for that pattern, allowing the definition of the constraints to be separated from the definition of the pattern; the only information stored in the pattern is a list of references to the appropriate constraints. In this way, a single constraint can exist independently from any patterns and can be used in several different patterns.

Adding a graphical interface to the knowledge-acquisition process for periodic patterns greatly facilitates the expert's role in detecting periodic patterns. The knowledge can be entered and maintained by the expert with minimal help from a knowledge engineer. Also, the development of the KA tool within the already-existing Protégé framework (currently implemented on Windows/NT machines in the CLIPS language, and in the process of reimplementation in Java) enhances the portability and reusability of the acquired knowledge.

Using a filtering process, the system must convert the knowledge base defined in the KA tool into a standard format that can be used by the Résumé system. Ideally, the knowledge base as acquired would have had a one-to-one correspondence with the knowledge classes in the Résumé rule engine. However, we decided to create a slightly different hierarchy in the KA tool to facilitate the knowledge-acquisition process. The classes are organized in a way that is more intuitive to experts when they are entering knowledge about patterns. The filter also creates redundant back pointers and performs other useful postprocessing tasks to ensure that the knowledge base is in a usable form for Résumé.

7. Detection of Patterns in Résumé

The pattern-matching module of Résumé is currently implemented as a rule-based system in the CLIPS language. The rule-based system reduces the problem of searching for periodic patterns to the problem of building a rule that fires whenever a periodic pattern exists in the data. The left-hand side of such a rule describes a periodic pattern, including the repeating component and all cyclical constraints, using CAPSUL, as described in Section 4. The right-hand side of the rule generates an instance of the cyclical pattern with the appropriate start and end times, value, and contexts. Recall that, once generated, a Résumé abstraction is added to a general pool of from which further abstractions can be derived. Thus, the inputs to a periodic pattern consist of both raw data and high-level abstractions (such as linear patterns) that have been already generated.

The rule builder is a function that uses the specifications of periodic patterns filtered in from the knowledge base to generate rules that check for these patterns in the database. It works by concatenating a string that is the rule and then sending the string into the CLIPS *build* function that converts a string into a rule. This process is simplified in that, once the information has been filtered, the slots of the instances contain almost exactly the same text as is required by the CLIPS rule syntax. Once the rule is built, the interpreter relies on several outside functions to check for the correct values of all the input parameters. When a periodic-pattern rule fires, the resulting instance that is created contains the CAPSUL of the pattern, the start and end times of the pattern (in real time points, in the same units as the database), and the value according to the value defined in the knowledge base.

7.1. Checking value and gap constraints

The two types of gap constraints and value constraints can all be checked in a similar fashion. First, all instances of the repeating component are found and are stored together as a set. Then, we iterate over the set in sorted order (sorted temporally), and for each successive pair of intervals, the constraint is checked. If the constraint holds, a counter is incremented. After iterating through all intervals, we check the number found against the total number of intervals.

7.2. Checking for clusters

The cluster check has two steps. The first step determines whether a cluster of instances exists within the specified time period The interpreter iterates through the set of instances and finds the first pair whose gap falls within the range of the time period. Then, it iterates through all the rest of the instances, incrementing a counter for each one whose distance from this first pair also falls within the range of the time period. At the end of the iterations, the counter represents the number of instances which belong to this first cluster. These instances are then removed from the set of instances, and the process is repeated until any more clusters that may exist within the set of instances are found. If the total number of clusters falls into the range specified in the knowledge base, then the first part of the periodicity check passes. The second step checks all the subconstraints for each cluster found. As each cluster is found, its subconstraints are checked. This check short circuits: If one cluster fails to satisfy the subconstraints, the other clusters are not checked, and the cluster constraint fails.

8. Preliminary Experimentation

To test the expressiveness of our pattern language and the correctness of the acquisition and detection tools, we collaborated with an expert oncologist to design and conduct a test of the pattern language. Because the Résumé system has been tested extensively (see [Shahar and Musen 1996]), we tested how well the ability of CAPSUL to acquire and detect trends in patient data. The data were provided by the University of Chicago Center for Bone Marrow Transplantation; they consisted of the records of 150 patients who received bone-marrow transplants over 2 years (1990 and 1991). The records contained several hundred data points per patient of more than 200 parameters.

First, we translated the patient data into a more tractable form. We ported the database into a standard ODBC format, and then into the format expected by Résumé. With the help of the expert, we selected five representative patient files and eighty measured parameters that he thought would be most useful.

Then, we entered into the oncology domain ontology several linear and periodic patterns of interest. We coupled a knowledge engineer with our expert to demonstrate the use of the graphical KA tool, then asked the expert to enter new patterns. Given the straightforward setup of the graphical KA tool, this task proved not to be difficult for the expert. Furthermore, we found that most of the patterns for this domain and data set that our expert was interested in entering were expressible in our representation language. Thus, although CAPSUL cannot express every pattern that is theoretically possible, we are convinced that it allows an impressive array of useful patterns to be expressed.

Finally, we ran Résumé using as input the new domain ontology and the shortened patient databases. We found several of the defined patterns in the database, and we verified the results by manually finding the data corresponding to the patterns. We are currently extending this experiment by testing the pattern-detection tool on larger sets of data.

9. Conclusions

We have described CAPSUL, an expressive language for the representation of periodic patterns, and the implementation of CAPSUL in an acquisition and detection tool within a larger temporal-abstraction system. This work builds on previous work in defining temporal repetition and constraint-based representations of time to develop a more complete, cohesive, and tractable representation of periodicity. The need for such a representation language arose as we extended the existing Résumé temporal-abstraction system to include higher-level and more complicated abstractions. Physicians in clinical domains can use such a tool to garner invaluable information about trends in patient data that they cannot easily discern by viewing patient charts. Our work is, moreover, easily applicable and extensible to many other domains that use temporal or even spatial data, such as planning, scheduling, and traffic monitoring.

Our first task was to create a language in which to define periodic patterns. We believe that CAPSUL captures much of what is useful about interesting patterns without being too complex for domain experts or for the interpreter. The set of patterns that can be expressed in CAPSUL contains all patterns in which the temporal and value relationship between consecutive intervals is uniform over the entire pattern. If we were to extend CAPSUL by allowing these relationships to vary over time, the added complexity not only would make CAPSUL intractable in the number of constraints to be specified, but also would, we believe, add little utility for our users. CAPSUL differs from its predecessors in its separation of local and global constraints (which apply to the repeating component of the pattern) from periodic constraints (which define the repetition of the component). For example, the internal relations between two nonconvex intervals described by Khatib [1994] correspond to the global qualitative temporal constraints applied to a linear component in CAPSUL.

Another important aspect of our work was the process by which we acquired from experts the domain-specific knowledge about the patterns to be found. A simple yet interesting conclusion is that it is possible for and useful to physicians to detect such patterns in patient databases. An important measure of the success of CAPSUL was the ease with which we implemented CAPSUL in a graphical KA tool. We have also found that it is important to create in the KA tool a class hierarchy and organization of constraints slightly different from those in Résumé, so that we facilitate an expert's understanding of the temporal ontology. Furthermore, this slightly different organization is especially important for periodic patterns.

The final piece of the project was the implementation of an interpreter to detect periodic patterns in a time-oriented database. At this time, the task of searching for the pattern is handled by a rule-based system in which conclusions (patterns) are found in no particular order as new data are encountered. The constraint-based language fits naturally with this type of algorithm, in that rules are expressed as a set of constraints followed by an action to be performed after the constraints are met.

In the future we intend to test CAPSUL in additional clinical domains. A particularly challenging domain will be monitoring patients who have insulin-dependent diabetes, in which there exist large numbers of data per patient within which physicians are concerned to identify detailed periodic trends (in particular, diurnal and weekly trends).Such patterns are difficult to find with manual search. We also intend to extend CAPSUL to apply to nonmedical problems.

10. Acknowledgments

We thank Robert Morris and Lina Khatib for helpful conversations and insights about representations of periodicity. Dr. Herbert Kaizer and Dr. Lawrence Basso also helped as domain experts who provided many examples of periodic patterns on which to test the expressivity of CAPSUL. We thank the University of Chicago Bone Marrow Transplantation Center, which, through Dr. Kaizer, allowed us to use data collected on patients. Mr. Arthur Menaker assisted us in expanding and maintaining the Résumé system. This work has been supported by grants LM05708 and LM06245 from the National Library of Medicine and IRI- 9528444 from the National Science Foundation. Computing resources were provided by the CAMIS project, funded under grant No. LM05305 from the National Library of Medicine.

11. References

1. Allen, James. “Maintaining Knowledge about Temporal Intervals”. in *Communications of the ACM*. pgs. 832-843. 1983.

2. Clifford, James and Rao, Ahobala. “A simple, general structure for temporal domains”. *Temporal Aspects in Information Systems*, pgs. 17–28. IFIP: 1988.

3. Cukierman, Diana and James Delgrande. “Characterizing Temporal Repetition”. In *Proc. of the International Workshop on Temporal Reasoning and Representation (TIME) -96*, pgs. 80–87. IEEE Press. Key West, FL. May 1996.

4. Cukierman, Diana and Delgrande, James. “Towards a formal characterization of temporal repetition with closed time”. In *Proc. of the International Workshop on Temporal Reasoning and Representation (TIME)-98*, pgs. 140–147. IEEE Press. Sanibel Island, FL, May 1998.

5. Khatib, Lina. *Reasoning with Non-Convex Time Intervals*. Ph.D. Thesis. Florida Institute of Technology. September 1994.

6. Ladkin, P. “Time representation: A taxonomy of interval relations”. In *Proc. of the AAAI-86*, pgs. 360–366. 1986a.

7. Ladkin, P. “Primitives and Units for Time Specification”. In *Proc. of the AAAI-86*, pgs. 354–359. 1986b.

8. Morris, Robert and Lina Khatib. “Periodic and Repeating Events”. pgs. 1–17. In Vila, L., Van Beek, P., Fisher, M., Boddy, M., Galton, A., Morris, R., and Nebel, B.; *Handbook of Time and Temporal Reasoning in AI*. MIT Press (forthcoming). July 17, 1998a.

9. Morris, Robert and Lina Khatib. “Quantitative Structural Temporal Constraints on Repeating Events”. In *Proc. of the International Workshop on Temporal Reasoning and Representation (TIME)-98*, pgs. 74–79. IEEE Press. Sanibel Island, FL. May 1998b.

10. Musen, M.A. “Modern Architectures for Intelligent Systems: Reusable Ontologies and Problem-Solving Methods”. In *Proc. of 1998 AMIA Fall Symposium*, Orlando, FL. November 1998.

11. Rit, J. F. “Propagating Temporal Constraints for Scheduling”. *Proc. of the Fifth National Conference on Artificial Intelligence (AAAI-86).* Morgan Kaufmann, Los Altos, CA, pgs. 383–388, 1986.

12. Shahar, Y. and Mark Musen. “Knowledge-based temporal abstraction in clinical domains”. *Artificial Intelligence in Medicine*; 1996. 8(3): pgs. 267–298.

13. Shahar, Y. “A Framework for Knowledge-Based Temporal Abstraction”. *Artificial Intelligence* 90: pgs. 79–133, 1997.

14. Shahar, Y. “Dynamic Temporal Interpretation Contexts for Temporal Abstraction”. *Annals of Mathematics and Artificial Intelligence*, 1998; 22(1-2): pgs. 159–192.

15. Shahar, Y. & C. Cheng. “Intelligent Visualization and Exploration of Time-Oriented Clinical Data”. In *Proc. of 1998 AMIA Fall Symposium*. November 1998: pgs. 155–159.

16. Shahar, Y., H. Chen, D. P. Stites, L. Basso, H. Kaizer, D. M. Wilson, & M. A. Musen. “Semiautomated Acquisition of Clinical Temporal-abstraction Knowledge”. Stanford Medical Informatics Technical Report No. 0735, Stanford University, Stanford, CA.1998.

17. Terenziani, P. “Qualitative and Quantitative Temporal Constraints About Numerically Quantified Periodic Events”. In *Proc. of the International Workshop on Temporal Reasoning and Representation (TIME)-97*, pgs. 94–101. IEEE Press. Daytona Beach, FL. May 1997.

18. Terenziani, P. “Generating Instantiations of Contextual Scenarios of Periodic Events”. In *Proc. of the International Workshop on Temporal Reasoning and Representation (TIME) –98*, pgs. 61–68. IEEE Press. Sanibel Island, FL, May 1998.