Advanced Temporal Data Abstraction for Guideline Execution

Andreas SEYFANG and Silvia MIKSCH
Institute for Software Technology and Interactive Systems
Vienna University of Technology
Austria

Abstract. Temporal data abstraction bridges the gap between snap shot values delivered by monitoring devices and laboratory tests on one side and high-level medical concepts used in guidelines and by medical professionals on the other side. Within this field, the detection and abstraction of repeated patterns is a complex and important challenge. A repeated pattern is a combination of events or intervals which occur multiple times in a formally describable temporal relation.

While there are many approaches to detect patterns in time series without prior definition of target concepts, we describe the application of temporal data abstraction in the context of guideline execution. Here predefined concepts of temporal patterns must be compared with measurement series describing the patient state. We discuss the requirements coming from both high-frequency domains such as intensive care units and low-frequency domains such as diabetes monitoring and show our solution based on a new version of the Asgaard data abstraction unit. It interfaces the dynamically changing patient state to the guideline execution unit and features abstraction modules ranging from simple calculations to statistical measures calculated for sliding time windows.

1 Introduction

The need for data abstraction. Computerized clinical guidelines and protocols are more and more used today and their advantages have been demonstrated [8]. An important condition for the success of computer supported guideline execution is the integration of such a system into the flow of information at the site of application [7]. Beside the technical challenge of integrating manifold equipment, each with its own proprietaire data format, there is a huge gap to close between the low level measurements delivered by monitoring devices or laboratory tests and the high level concepts used by the medical staff. The first deliver snapshots in numerical style which appear exact but they are subject to various measuring errors. The latter describe the patient condition on a high level such as ”high glucose” or ”sufficient ventilation”. The first can be mapped to the latter using different mapping tables for different contexts and sometimes additional rules. This means that the same numeric reading can map to different qualitative concepts under different context. The context is formed by the disease of the patient and by the current treatment scheme among other things. This process is called (knowledge-based) data abstraction.

The need for temporal *data abstraction.* The field of temporal data abstraction deals with the abstraction of measurement series taken over a certain span of time. Analyzing the temporal dimension is important for both long term evaluation of a treatment and short term detection of changes in the patient’s state.

The role of repeated patterns. Within this field, repeated patterns play an important role. These cannot be described by simple formulas but only by a set of abstraction rules and constraints. Since most patterns can only be described in terms of high-level concepts, the abstraction of these concepts is an indispensable first step. Then the repetition of these constraints is described in terms of the relations between the instances of the repeated pattern. These relations can be quantitative, e.g., a certain amount of time lying between two occurrences, or they can be qualitative such as ”A happens during B” where B is an interval longer then A.

Data abstraction needs guideline execution. Not only guideline execution needs temporal data abstraction – data abstraction itself without guideline modeling is of limited use. At its best, the top-level abstractions map to treatment recommendations. But even in this case the actual treatment can only be recorded by means of these recommendations (and whether the physician accepted the recommendation). The temporal dimension of the treatment, its decomposition into different steps or actions, and any information associated with treatment steps is lost if not manually entered into the system (as qualitative input parameters). Therefore, the ideal system combines data abstraction with treatment planning [14].

The cooperation of data abstraction and guideline execution. The ultimate aim of the data abstraction process is to deliver information in a form suitable for the guideline execution during runtime. This information is used in different places in the guideline, most notable in conditions which influence the selection of parts of the treatment. This holds for most systems for computerized guideline execution, not only the Asgaard system. The latter is described in detail in Section 4.

Data-driven versus concept-driven abstraction. Two strategies for the abstraction of higher-level concepts from raw data can be distinguished. Either the abstraction process consists of matching data to a set of predefined concepts or scenario descriptions to see at which time which scenario is present. Or the abstraction is guided by features found in the data, e.g., looking for the most frequent or the most rare patterns.

In the context of data abstraction for guideline execution, we always deal with the first case: The guideline describes the data in high-level terms and the consequences of finding certain patterns or scenarios in the input data describing the patient state.

For this reason, data driven approaches which perform abstractions without considering the current mode of treatment or other aspects of the guideline execution, are problematic if applicable at all, since even such seemingly simple preprocessing steps as removal of obviously wrong measurements typically need complex context-information such as overall patient condition, age, or underlying diseases. For a patient in a life-threatening situation a measurement yielding a certain value may be correct but in a healthy person the same value must be an error.

Permanent monitoring versus occasional encounter. For the technical implementation, it makes a difference whether the patient is permanently monitored as in a intensive care unit, where the incoming data must be processed item by item and the abstractions (e.g., alarms) must be produced as soon as possible, or whether the patient meets the physician occasionally bringing his record with him such as in traditional glucose monitoring in diabetes. In such a case, the abstraction algorithm can process the whole record at once. This is an important advantage in matching high-level concepts to the data because analyzing the temporal dimension is easier if intervals are already completed.

However, the medical community strives to extend the scenario of continuous monitoring to more and more fields, e.g., using mobile phones to send the measured glucose readings to a central server after each measurement [12]. Therefore, the need to detect matching episodes as soon as possible is expanded beyond the domain of high-frequency into the domain of automatically monitored low-frequency data.

In practice this means, that if the temporal pattern reads ”high fever for at least 2 hours” then 2 hours after the value of the parameter ”fever” becomes ”high” (provided that is does not change meanwhile), this episode must be reported to the guideline execution unit which will initiate appropriate reaction. The much easier solution, to wait until the interval is closed (ended) and then see whether its duration exceeds two hours, is of very limited use in the context of guideline execution.

Since different teams use different names for the basic concepts of this fields and therefore different definitions are used in the literature, we define the following terms for clarity.

Definition 1. *Input variables are called parameters. They have numerical (quantitative) or symbolic (qualitative) values.*

Definition 2. *An episode is a period of time during which a certain parameter has a certain value.*

Definition 3. *A parameter proposition describes desired values for a parameter, a context and temporal constraints for the time during which the parameter should hold this (or these) value(s). It can match zero, one or more episodes. If it matches at least one episode, the parameter proposition is fulfilled.*

Definition 4. *A temporal pattern is a combination of parameter proposition or other temporal patterns.*

Definition 5. *The valid time of a data point is the time at which the measurement was taken. The transaction time is the time at which the measurement was entered into the system. In general, only the valid time is considered in the abstraction process.*

In this paper we first describe related projects in this field and then develop a list of challenges coming from both these publications and from our own experience using and developing data abstraction methods. Then we describe the Asgaard data abstraction unit and how it can be used to abstract repeated patterns.

2 Related Work

Bellazzi et al. [1] uses temporal data abstraction to gain higher level concepts from monitor dialysis sessions and searches dependencies of such patterns and patient outcome using machine learning technologies. They also abstract high level concepts from repeated patterns in diabetes monitoring to arrive at useful presentations of the data to the physicians [2]. In both cases, guideline execution is not integrated in the system.

Cukierman et al. [6] are investigating a formal representation of time units, calendars, and time unit instances as restricted temporal entities for reasoning about repeating activities. However, there is no provision for deviations from the typical cycle or in the recurrence pattern which is unavoidable in practical applications.

The RESUME-system [15] implements a knowledge-based temporal-abstraction method to derive high-level concepts from low-level data in the way as described above. This generic framework inspired the data abstraction in Asgaard from the beginning, e.g., the context-dependent abstraction of qualitative values. Based on abstractions delivered by RESUME, CAPSUL [4] abstracts repeated patterns from low-frequency data. Recently, CAPSUL and other knowledge-base temporal abstraction methods have been integrated into Alma, which is the default temporal abstraction module of the Idan framework [3]. The latter takes a modular, distributed approach to temporal data-abstraction while the Asgaard concept used a single (but modular) plan library to be executed on a single computer.

While there are many guideline modeling approaches (see [11] for a comparison), there are few to integrate strong data abstraction resources ([16], [10]). This may be caused by the fact that today's guidelines often leave the details of data abstraction open since such knowledge is rather found in textbooks or else assumed to be common-place. It is also caused by the fact that in most settings data is entered manually which allows to delegate the data abstraction task to the user by demanding qualitative high-level input instead of the original data. Another important aspect is the fact that most guidelines deal with low-frequency applications, many of which involve rather little data, and that the full integration of guideline execution into the clinical data flow is not realized today. We are convinced, that with the increasing availability of (reliable enough) data in the past and next decade, the importance of bridging this data to guidelines will increase.

3 Challenges

In this section we show practical problems from two different domains – artificial ventilation as an example for high-frequency domains, and diabetes as an example for a low-frequency domain. Based on the practical problems, we define a set of challenges. In section 4.3 we will show how these challenges are met. While the problems of artificial ventilation are taken from our own experience, those for diabetes come from publications on CAPSUL.

3.1 Example Domain Artificial Ventilation

In spite of significant progress, the field of artificial ventilation of premature neonates remains challenging. This is due to the fact that the lung of very young neonate is not mature and thus there are no suitable models for its behavior. Instead, the settings of the ventilation devices must repeatedly be adjusted according to a set of rules. While some of these rule are well known and thus can be formally encoded, others are not. In a long term project, we strive to gradually extend the amount of help that computer science (temporal data abstraction and therapy modeling) can offer. This comprises both data abstraction and treatment modeling. A model evaluated in a clinical trial [13] focused on the elimination of suboptimal changes in the treatment caused by faulty input data which was successful.

One next step in the evolution is the judgment of the patient state as a basis for adaptation of the ventilation strategy. This necessitate temporal data abstraction. The following are three representative examples.

- A parameter describing the patient state is the amount of hypoxic periods within a certain time period. These are intervals, during which SpO_2 is below 80 and which last for at least 4 seconds.
- Another measure is the fraction of time during which the SpO_2-readings were *normal* (within the last 15 minutes).

- A qualitative expression of the trend of a patient's state is: If either the amount of hypoxic episodes in the previous 30 minutes or their total duration increased by more then 10 % compared to the 30 minutes before then the trend is *worsening*. If both indicators (amount and total duration) improve, then the trend is *improving*. Otherwise, it is considered *stable*. Note that this concept is somewhat independent from the numerical trend by design.[1]

On the technical level this means that we must perform the tasks described as the following challenges. These challenges are specific to the described problems, but representative for a class of problems. Section 5 discusses features and limitations of our system on a more general level.
Challenge 1. *Map the numeric (quantitative) readings delivered by the pulsoximeter to qualitative values.*
Challenge 2. *Find episodes of SpO_2<80 lasting for at least 4 seconds.*
Finding episodes during which SpO_2 is normal is assumed to follow the same lines and not listed as a separate challenge.
Challenge 3. *Collect these episode for a sliding time window of 30 minutes length.*
Challenge 4. *Count the episodes in both time windows.*
Challenge 5. *Measure the total length of all episodes in a time window.*
Challenge 6. *Multiply previous values with 1.1 resp. 0.9 and compare them with the latest ones. This is done for both amount and duration of episodes.*
Challenge 7. *Implement the abstraction rule "if a or b then* worsening *else if c and d then* improving *else* stable*", where a stands for "previous amount * 1.1 < latest amount" etc.*

3.2 Diabetes

The domain of glucose monitoring in diabetes patients strongly differs in its setting and in its difficulties from the domain of artificial ventilation. While in the diabetes a few measurements per day are collected over the life span of a patient, in artificial ventilation several measurements per second are collected for (hopefully) a few days. And while in artificial ventilation signal quality is a main problem in data collection, in diabetes the patient's motivation is the key to consequent self-monitoring and thus "data quality".

A unique feature in low-frequency domains is the consideration of calendar date and day of time. E.g., the glucose measurement in the morning is interpreted different to that taken in the evening and differences between workday and the weekend are often observed (and important to notice).

An example (slightly adapted from [4]) is "normal blood glucose levels in the morning followed by high blood glucose levels in the evening that occur at least three times per week four times within a period of 3 month". Another one (taken from [5]) is the same combination of normal morning and high evening glucose, but this time "occurring during the weekend 5–7 times during the summer."

On the technical level, this means that aside from mapping numeric values (glucose) to qualitative values *high* and *low*, which is already part of the challenges above, we need to treat the measurements different depending on the time of day at which the measurement was taken, to implement *morning glucose* and *evening glucose*. This can be implemented by using two (or three, including noon) distinct parameters, one for each time of glucose measurement, so it is not a challenge at all.

[1]In practice, we use further parameters such as heart rate in addition, which are omitted here for brevity.

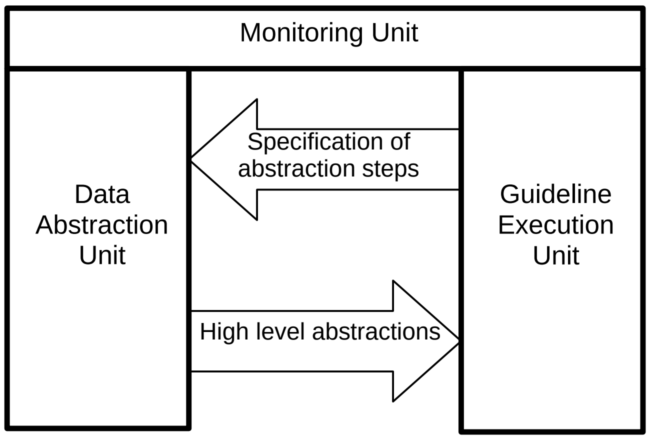

Figure 1: The Asgaard runtime system.

Challenge 8. *Find days on which normal morning glucose is followed by high evening glucose.*
Challenge 9. *Find periods of 3 months during which there are 4 or more weeks each containing at least 3 days defined by the previous challenge.*
While the grouping of day in a week is similar to that of hypoxic episodes in a 30-minute time window, the nested grouping need not be trivial. For the second question we need to know whether it is a day of the weekend. We also must abstract summer from the month of the measurement and previously establish consensus of domain experts concerning which months constitute summer.
Challenge 10. *Find those days of normal morning glucose followed by high evening glucose which lie on weekends in summer (where summer is defined as a fixed range of month).*
Finally, episodes (low-high combinations of glucose) are counted for certain periods of time and such counts again are analyzed further. Although the time scale is different, on the technical level this is very similar to counting the hypoxic episodes in an hour.

4 Implementation in Asgaard

4.1 System Architecture

The Asgaard[2] run-time system has two main parts to consider in this article: the data abstraction unit and the plan execution unit. In Asbru, a guideline or protocol is represented by a set of skeletal plans which form a hierarchy. Each plan stands for a part of the treatment, be it at a high level such as a group of other plans in a particular temporal order or on a detailed level such as a plan which directly maps to a therapeutic action. These plans are contained in a plan library together with the definition of the data abstraction setup necessary for these plans.

The plan execution unit reads the plan library, forwards the necessary parts to the data abstraction unit which itself configures a network of abstraction modules to implement the abstractions. Then the user connects sources of input (files, monitoring devices, user interface modules) to the data abstraction unit and invokes a top-level plan. In the whole process the monitoring unit mediates between data abstraction unit and guideline execution unit. Figure 1 describes this graphically.

[2]Asgaard is the heaven of the Nordic mythology and the name of the whole system presented here. Asbru is the rainbow bridge to Asgaard and the guideline representation language we use.

Each plan has a set of plan states (active, suspended, etc.). The transition between these states is governed by conditions which build on the abstractions delivered by the data abstraction unit. Thus the results from the abstraction process together with the guideline represented by plans drive the course of action.

4.2 Overview of Data Abstraction Capabilities

The data abstraction unit consists of a set of abstraction modules (i.e., classes of abstractions) which are instantiated and connected to each other as specified in the plan library. The data abstraction unit can also be used alone to analyze data via a graphical user interface. Each of the abstraction module represents one abstraction step in a potential complex abstraction hierarchy. The following is a brief overview of the available modules.

Calculations. Numeric parameters can be combined by arithmetic operators ranging from *add* to *minimum*. Likewise, Boolean-valued parameters can be combined by Boolean operators.

Sliding time windows. A time window is defined for a certain parameter and has a width, specified in terms of the valid time of the points within or by the number of the points to contain, and a step width, by which the window is advanced along the time axis. This means that if width is 1 hour and step width is 1 minute, then all values of the parameter measured for the previous hour are stored and once per minute the time window is evaluated by one of the modules described next.

Time window analysis. For a time window, statistical measures ranging from average and median to centiles with changing threshold and linear regression can be performed. A special abstraction based on linear regression is the *time to alarm*, i.e., the temporal distance between the current time point and the intersection of the regression line with a certain threshold, in other words, the time left until the parameter reaches the threshold if the current trend continues.

Spread-based abstractions. A *spread* is a band of varying width representing oscillating measurements in a more steady way [9]. Its margins are calculated based on linear regression and its standard deviation or centiles. The spread gives a robust basis for the abstraction of steady, qualitative values from oscillating or disturbed high-frequency input such as measurements delivered by monitoring devices.

Qualitative values. Qualitative values can be abstracted from a spread, directly from numeric values using context-specific mapping tables, or by arbitrary rules as in a case distinction in programming languages.

Episode detection. The basic concept for episode detection in Asbru is the *parameter proposition* it contains a parameter name, a value description, a context and a time annotation. The time annotation contains three intervals, one delimiting the start of the interval, during which the parameter should have the described value, one interval delimiting the end of the interval and one delimiting its duration. All of these 6 values are optional, so guideline designers can enter exactly that piece of knowledge the acquired. These 6 values are not absolute time points but shifts relative to a reference point also contained in the time annotation, which is a reference to a time point given in a variable or define by a plan state change or a parameter change.

Episode combination. A *temporal pattern* in Asbru contains either a combination of parameter propositions or other temporal patterns, a count constraint describing repetitions of another temporal pattern, or a temporal constraint describing the qualitative relation between temporal patterns using Allen's relations.

Explicit time references. The valid time of quantitative parameters and the duration during which qualitative parameters retained their values can be accessed just like numeric input by other abstraction modules allowing arbitrary calculations and comparisons.

Calendar references. For values representing time points, various attributes such as day of week, month, etc. can be extracted.

Delays. A parameter can also be delayed by a certain amount of time, which is useful to compare the most recent readings to older ones.

The strength of the Asgaard data abstraction system comes from the possibility to freely combine the above abstractions. This means, that it is not only possible to calculate the average of a certain parameter for the previous hour, it is also possible to calculated the difference between the average within the previous hour and that for the previous minute and compare the result to some other value. The next section gives a few illustrative examples for such combinations. Section 5 gives a more abstract view of the features and limitations of our approach.

4.3 Solutions to Example Problems

In this section we briefly repeat the challenges found above and then show how they can be met using Asbru to specify the solution based on temporal data abstraction which the data abstraction unit will perform during guideline execution.

4.3.1 Artificial Ventilation

Solution to challenge 1. *Map the numeric (quantitative) readings supplied by the pulsoximeter to qualitative values.*

This involves three definitions in Asbru. First, the raw data as delivered by the monitoring device is defined (*SpO2-raw*). Then the qualitative values SpO_2 can take are listed and assigned to a *qualitative scale* which corresponds to abstract data types in programming languages. Third, a qualitative parameter *SpO2-qualitative* is defined which produces qualitative values of the before defined type from numerical input in *SpO2-raw* using a list of limits which form a mapping table. Note that this mapping is valid only for the context of NICU (neonatal intensive care unit), we could specify mappings for other contexts, e.g., health neonates or adults, by adding another *limits* entry with another context.[3]

[3]The syntax shown in this paper is slightly beautified Asbru – closing XML-tags as well as the angle brackets are removed for better readability.

```
parameter-def name="SpO2-raw" type="percent"
  raw-data-def unit="%" mode="automatic" channel-name="SPO2"

qualitative-scale-def name="SpO2-qualitative"
  qualitative-entry entry="hypoxy"
  qualitative-entry entry="very-low"
  qualitative-entry entry="low"
  qualitative-entry entry="normal"
  qualitative-entry entry="high"
  qualitative-entry entry="very-high"

parameter-def name="SpO2-qualitative" type="SpO2-qualitative"
  qualitative-parameter-def
    limits unit="%" scale="SpO2-qualitative"
      context
        context-ref name="NICU"
      limit-entry value="0"
      limit-entry value="80"
      limit-entry value="86.5"
      limit-entry value="89.5"
      limit-entry value="93.5"
      limit-entry value="96.5"
      limit-entry value="100"
    parameter-ref name="SpO2-raw"
```

Solution to challenge 2. *Find episodes of SpO_2<80 lasting for at least 4 seconds.*

In Asbru, parameter propositions are used to describe episodes which should be found. Parameter *acute-hypoxy* is therefore defined using a parameter proposition prescribing that *SpO2-raw* must be less then 80 in the context of NICU for a minimum duration of 4 seconds. The parameter proposition is enclosed by a *boolean-def* abstraction, which transforms the abstract episode objects found by the parameter proposition matching into a series of intervals during which the boolean value is true if (and only if) an episode occurs at this particular moment in time. In other words, at the start of an episode a data point with value *true* is issued and at the end one with value *false*. Since the start of the episode is recognized with some delay (4 seconds in our example) the valid time differs from the transaction time by this amount of time.

```
parameter-def name="acute-hypoxy" type="boolean"
  boolean-def
    parameter-proposition parameter-name="SpO2-raw"
      value-description type="less-than"
        numerical-constant value="80" unit="%"[4]
      context
        context-ref name="NICU"
      time-annotation
        time-range
          duration
            minimum
              numerical-constant value="4" unit="s"
        now
```

Solution to challenge 3. *Collect these episode for a sliding time window of 30 minutes length.*

[4]Reusing the definition of SpO2-qualitative above, this could also be stated as `SpO2-qualitative equal hypoxy`.

We define a time window of 30 minutes which advances in steps of 1 minute along the time axis and which collects the intervals of *acute-hypoxy* during that time.

```
parameter-def name="30-minutes-window" type="time-window"
  time-window-def
    window-length
      numerical-constant value="30" unit="min"
    step-width
      numerical-constant value="1" unit="min"
    source
      parameter-ref name="acute-hypoxy"
```

Solution to challenge 4. *Count the episodes in both time windows.*

This abstraction is called time-window-analysis in Asbru, with operation *count*.

```
parameter-def name="hypoxy-count-in-last-30-minutes" type="amount"
  time-window-analysis-def operator="count"
    parameter-ref name="30-minutes-window"
```

Solution to challenge 5. *Measure the total length of all episodes in a time window.*

In contrast to the above, the total length of all episodes in a time window is calculated by a *total-duration-def* which receives the value of the parameter for which the durations are to be added. While this is trivially the value *true* here, it could also be any other qualitative value in appropriate cases.

```
parameter-def name="hypoxy-duration-in-last-30-minutes" type="time"
  total-duration-def
    value
      qualitative-constant value="true"
    source
      parameter-ref name="30-minutes-window"
```

Solution to challenge 6. *Multiply previous values with 1.1 resp. 0.9 and compare them with the latest ones. This is done for both amount and duration of episodes.*

To reduce the complexity of the solution to the next challenge we define four intermediate abstractions – *count-increase*, *count-decrease*, *duration-increase*, and *duration-decrease* – each based on the comparison of the value computed 30 minutes ago with the current one (considering the above factors). This is not implemented by a parameter proposition but by a comparison, since we are not interested in finding episodes but only in comparing instantaneous values. For space considerations, only the first of the four is shown.

```
parameter-def name="count-increase" type="boolean"
  comparison-def operator="greater-than"
    left-hand-parameter
      parameter-ref name="hypoxy-count-in-last-30-minutes"
    right-hand-parameter
      calculation-def operator="multiply"
        numerical-constant value="1.1"
        delay-def
          delay
            numerical-constant value="30" unit="min"
          source
            parameter-ref name="hypoxy-count-in-last-30-minutes"
```

Solution to challenge 7. *Implement the abstraction rule "if a or b then* worsening *else if c and d then* improving *else* stable*", where a stands for "previous amount * 1.1 < latest amount" etc.*

This is implemented by the *logical-dependency-def* in Asbru, which has the same rule implementation capabilities as if-statements in programming languages, with the great advantages that all abstractions are evaluated in parallel, i.e., the style of modelling is rather declarative then procedural.

Note that the code shown uses an abstract type *hypoxy-development* defined by a *qualitative-scale-def* before, but not shown in this paper.

```
parameter-def name="hypoxy-development" type="hypoxy-development"
  logical-dependency-def
    if
      logical-combination-def operator="or"
        parameter-ref name="count-increase"
        parameter-ref name="duration-increase"
    then
      qualitative-constant value="worsening"
    if
      logical-combination-def operator="and"
        parameter-ref name="count-decrease"
        parameter-ref name="duration-decrease"
    then
      qualitative-constant value="improving"
    if
      default
    then
      qualitative-constant value="stable"
```

4.3.2 Diabetes

To avoid repetitions we omit the abstraction of qualitative values from quantitative input here, which is similar as above and assume that the values for morning and evening glucose are entered with their qualitative values *high*, *normal*, and *low*.

Solution to challenge 8. *Find days on which normal morning glucose is followed by high evening glucose.*

Here we use a parameter proposition to match episodes of high evening glucose the temporal dimension of which is defined relative to a previous start of normal morning glucose. We defined morning glucose with a *trust period* shorter than 24 hours, i.e., these values loose their validity at the end of the day. Therefore, each normal value for morning glucose starts a new interval, since during the night the morning glucose was undefined.

The following defines a bad day to be one on which evening glucose is high 5 to 20 hours after morning glucose was normal.

```
parameter-def name="bad-day" type="boolean"
  boolean-def
    parameter-proposition parameter-name="evening-glucose"
      value-description type="equal"
        qualitative-constant value="high"
      context
        any
      time-annotation
        time-range
          starting-shift
            earliest
              numerical-constant value="-20" unit="h"
            latest
              numerical-constant value="-5" unit="h"
        parameter-change value="normal" direction="enter"
          parameter-ref name="morning-glucose"
```

Solution to challenge 9. *Find periods of 3 months during which there are 4 or more weeks each containing at least 3 days defined above.*

We first collect the instances of ”bad days” for each week and count them as *bad-week*s. Note that a week here is just an interval of 7 days, it is not aligned to any calendar date.

```
parameter-def name="bad-week" type="boolean"
  comparison-def operator="greater-or-equal"
    left-hand-parameter
      time-window-analysis-def operator="count"
        time-window-def
          window-length
            numerical-constant value="7" unit="d"
          step-width
            numerical-constant value="7" unit="d"
          source
            parameter-ref name="bad-day"
    right-hand-parameter
      numerical-constant value="3"
```

Then, we look at the previous 3 months and count the ”bad weeks”. This is implemented similar. The main difference is that the step width is smaller – one week – then the window size which is 3 month. This means that a single series of bad weeks will repeatedly cause *bad-period* to be *true* and that we detect such a series as soon as it is completed and not months latter as it would be the case with a larger step width.

```
parameter-def name="bad-period" type="boolean"
  comparison-def operator="greater-or-equal"
    left-hand-parameter
      time-window-analysis-def operator="count"
        time-window-def
          window-length
            numerical-constant value="3" unit="m"
          step-width
            numerical-constant value="7" unit="d"
          source
            parameter-ref name="bad-week"
    right-hand-parameter
      numerical-constant value="4"
```

Solution to challenge 10. *Find those days of normal morning glucose followed by high evening glucose which lie on weekends in summer (where summer is defined as a fixed range of month).*

To this end, we use both weekend and summer as necessary context for the abstraction of bad days. Both *weekend* and *summer* are abstractions themselves: *weekend* is true if the day referring to the current *valid time* is number 6 or 7. *summer* is true if the current month is one of number 6–9 (as it was suitable for 2003). Only if both are true, the context of the parameter proposition in *bad days* is matched, otherwise the glucose values are ignored.

For space considerations, the definition of summer must be omitted.

```
parameter-def name="weekend" type="boolean"
  comparison-def operator="greater-or-equal" use-as-context="yes"
    left-hand-parameter
      day-of-week
        now
    right-hand-parameter
      numerical-constant value="6"

parameter-def name="bad-summer-weekend" type="boolean"
  boolean-def
    parameter-proposition parameter-name="evening-glucose"
      value-description type="equal"
        qualitative-constant value="high"
      context
        context-combination operator="and"
          context-ref name="weekend"
          context-ref name="summer"
      time-annotation
        time-range
          starting-shift
            earliest
              numerical-constant value="-20" unit="h"
            latest
              numerical-constant value="-5" unit="h"
        parameter-change value="normal" direction="enter"
          parameter-ref name="morning-glucose"
```

5 Discussion

In this paper, we presented a set of challenges and their solutions using Asbru and the Asgaard data abstraction unit. The types of abstraction methods available are described in Section 4.2.

5.1 Features

- Parameters can be combined using algebraic and logical combinations.
- Statistical measures can be evaluated for moving time windows.
- Numeric parameters can be abstracted to qualitative parameters in a context-sensitive way.
- Both the value of the parameter and the temporal extend of an episode can be described by absolute values as well as relative to other episodes, parameters, and events.
- For each episode the time of start and end and its duration are abstracted parameters which can be used in further abstractions.
- Episodes can be combined to form arbitrarily complex temporal patterns.
- Repeated patterns can be described in a flexible way, allowing the relations between episodes to change over time.

5.2 Limitations

The following features are not included in the Asgaard abstraction system.

Fuzzy logic. While context-sensitive mapping quantitative values to qualitative concepts is implemented, there is no provision for fuzzy borders of qualitative regions.

Uncertainty of measurements. The validity of data and the propagation of this validity values through the abstraction network is modelled in a very basic way: Input is valid or not. Expiration of a measurement after a certain time and closing gaps between measurements is implemented, but gradual loss of trust in the measurement and extrapolation must be modelled explicitly.

Probability distributions. The distribution of data can also only explicitly and in a coarse manner be modelled, e.g., by specifying certain centiles and limits on the standard deviation.

Advanced statistical analysis. Methods such as logical regression or correlation analysis are not included. They must be applied off-line in the form of post processing the output of the data abstraction unit.

Complex queries for episodes. Constraints of episodes (expressed in a parameter proposition) can refer to features of other episodes to a very limited extend. References to features of other episodes implicitly refer to the most recently found instances. Conjunctions of constraints which should refer to the same instance of episode are not possible. E.g., ”find episodes of A between instances of B and C where the duration of B and C together is longer then that of A but each of them alone is shorter and there are not other instances of A, B, or C such a triplet” cannot be implemented. The constraints alone can be stated, but it cannot be guaranteed (without an incredible amount of tricks) that these constraints will finally match the same episodes and not a bigger group among which each episode confirms to one of the constraints.

Another, equally severe limitation is that of *knowledge acquisition*. As for guideline modelling, physicians cannot handle or understand the computer readable representation. A computer scientist familiar with the medical concepts is necessary to acquire the knowledge to encode in the system from the medical experts in a series of dialogs.

The interactive, graphical user interface serves to ease the effort of the knowledge engineer and to immediately produce results on recorded sample data, in order to speed up the knowledge acquisition process.

A main concept of the system is the abstraction and usage of medically meaningful concepts. This allows us to describe the abstractions in term of these concepts which greatly improves the communication between knowledge engineer and physician.

6 Conclusion

In order to integrate the computer supported execution of clinical guidelines and protocols into the data flow of the site of application, the integration of data abstraction and guideline execution is necessary.

While there are several implementations meeting several of the above demands, we believe that – in particular for high-frequency domains – hybrid systems such as the Asgaard data abstraction unit, which combine numerical and statistical calculations with qualitative reasoning about both values and temporal dimension are an optimal solution.

They allow the guideline designers to be precise and efficient in the description of abstractions where precise instructions to process large input streams are present, and to handle the vagueness of the medical knowledge where necessary.

References

[1] R. Bellazzi, C. Larizza, P. Magni, and R. Bellazzi. Quality assessment of hemodialysis services through temporal data mining. In M. Dojat et al., editor, *Artificial Intelligence in Medicine*, pages 10–20, Berlin, 2003. Springer.

[2] R. Bellazzi, C. Larizza, P. Magni, S. Montani, and G. De Nicolao. Intelligent analysis of clinical time series by combining structural filtering and temporal abstractions. In W. Horn et al., editor, *Artificial Intelligence in Medicine*, Berlin, 1999. Springer.

[3] D. Boaz and Y. Shahar. Idan: A distributed temporal-abstraction mediator for medical databases. In M. Dojat et al., editor, *Artificial Intelligence in Medicine*, pages 21–30, Berlin, 2003. Springer.

[4] S. Chakravarty and Y. Shahar. A constraint-based specification of periodic patterns in time-oriented data. In *Proceedings of the TIME-99*, pages 29–40. IEEE Comp. Soc., 1999.

[5] S. Chakravarty and Y. Shahar. Specification and detection of periodic patterns in clinical data. In *Fourth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-99)*, pages 20–31, 1999.

[6] D. Cukierman and J. Delgrande. A formalization of structured temporal objects and repetition. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pages 83–87, Berlin, 2000. IOS Press.

[7] W. Horn. AI in medicine on its way from knowledge-intensive to data-intensive systems. *Artificial Intelligence in Medicine*, 23(1):5–12, 2001.

[8] M. E. Johnston, K. B. Langton, R. B. Haynes, and A. Mathieu. Effects of computer-based clinical decision support systems on clinician performance and patient outcome: a critical appraisal of research. *Ann Intern Med*, 120:135–142, 1994.

[9] S. Miksch, A. Seyfang, W. Horn, and C. Popow. Abstracting steady qualitative descriptions over time from noisy, high-frequency data. In W. Horn et al., editor, *Artificial Intelligence in Medicine*, pages 281–290, Berlin, 1999. Springer.

[10] M. Peleg, A. Boxwala, S. Tu, R. Greenes, E. Shortliffe, and V. Patel. Handling expressiveness and comprehensibility requirements in GLIF3. In *Proceedings of the 10th World Congress on Medical Informatics (MedInfo 2001)*, pages 241–245, London, 2001.

[11] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and M. Stefanelli. Comparing computer-interpretable guideline models: A case-study approach. *JAMIA*, 10(1), 2003.

[12] C. Popow, W. Horn, B. Rami, and E. Schober. VIE-DIAB: a support program for telemedical glycaemic control. In M. Dojat et al., editor, *Artificial Intelligence in Medicine Proceedings of the 9th Conference on Artificial Intelligence in Medicine in Europe (AIME-2003)*, Berlin, 2003. Springer.

[13] A. Seyfang, S. Miksch, W. Horn, M. S. Urschitz, C. Popow, and C. F. Poets. Using time-oriented data abstraction methods to optimize oxygen supply for neonates. In S. Quaglini et al., editor, *Artificial Intelligence in Medicine*, pages 217–226, Berlin, 2001. Springer.

[14] A. Seyfang, S. Miksch, and M. Marcos. Combining diagnosis and treatment using Asbru. *International Journal of Medical Informatics*, 68((1-3)):49–57, 2002.

[15] Y. Shahar and M. A. Musen. Knowledge-based temporal abstraction in clinical domains. *Journal of Artificial Intelligence in Medicine*, 8(3):267–298, 1996.

[16] S. Tu and M. Musen. A flexible approach to guideline modeling. In *AMIA Annual Symposium*, pages 475–497, Washington D.C., 1999. Hanley & Belfus.