Explicit representation of temporal aspects in a medical monitoring system using CEC

L. Chittaro\textsuperscript{a}, M. Del Rosso\textsuperscript{a}, M. Dojat\textsuperscript{b}

\textsuperscript{a}Dipartimento di Matematica ed Informatica, Università di Udine, Via dell Scienze, 260, 33100 Udine, Italy
\textsuperscript{b}INSERM U. 296, Faculté de Médecine, 8 Rue du Général-Sarrail 94010 Créteil Cedex, France

Abstract

We show how the CEC temporal reasoning tool (presented by [1] in this volume) can be used to represent and reason about temporal aspects in medical monitoring systems. In particular, we have explored the application of CEC to the management of mechanical ventilation, exploiting it to interpret change in data over time, assess patient status and its evolution, and choose the proper level of assistance.

1 Introduction

In data-rich clinical environments such as Intensive Care Units (ICUs) or operating rooms, there is a crucial need for intelligent monitoring systems that can help the clinician to deal with the massive flux of information. A typical clinical application of such systems is the surveillance of the mechanical respiratory assistance provided to patients who suffer from a lung disease and are hospitalised in ICUs. Ideally, the advantages of a knowledge-based system for the management of ventilator therapy are: (i) to function on a 24 hours per day basis, allowing a continuous adaptation of the level of the assistance and a reduction of total duration of ventilation, and (ii) to develop specific strategies for withdrawing the mechanical support (called weaning procedure), including a gradual decrease of the mechanical support, difficult to obtain in clinical practice without the assistance of a computerised system. Such a system must work in a closed-loop to be useful to the clinical staff. Recently, clinical studies have validated this approach [3].

Time is a central factor in intelligent monitoring systems that are supposed to interact with real dynamic environments. The need for time representation covers two major aspects:
Temporal Reasoning: the physician i) builds dynamically an interpretation of the evolution of the patient's ventilation, ii) predicts the patient's evolution with regard to previous states and iii) constructs and executes a plan of actions to drive the patient to an expected state. The physician adapts his/her strategy to the history of the patient's ventilation and to the time the patient spent in a given respiratory state.

Respect of real-time constraints: the system should be able i) to acquire physiological data provided by several monitors, ii) to plan the ordered sequencing of the three fundamental tasks in medical reasoning [8] - observation, diagnosis and then therapy - each task being in turn decomposable in several sub-tasks and iii) to have a prompt reaction in alarming situations, which imposes to short-cut some sub-tasks.

The need for an explicit representation of time in medical diagnosis has been advocated by several researchers, but very few clinical decision support systems incorporate clear formalisms for temporal reasoning [7]. In patient monitoring we need to reason about disease evolution (e.g., "Has the patient's ventilation been stable enough to envisage a decrease in mechanical assistance?") or to judge a patient's response to therapy (e.g., "Is the increase of mechanical assistance effective?"). A model of the world based on events, whose occurrence modifies the state of the world and properties that have a tendency to persist during time, is well adapted to our applications [4]. Thus, we explore the Event Calculus (EC) [5] formalism to interpret change in data over time in the context of mechanical ventilation management. In particular, we choose as our temporal reasoning tool the Cached Event Calculus (CEC), an efficient implementation of the Event Calculus with preconditions. A brief introduction to CEC is provided by [1] in this volume, while [2] describes in detail the architecture and the implementation of CEC and formally proves the computational complexity of EC and CEC, showing the improvements obtained by CEC with a worst-case analysis. Experimental results show that the efficiency of CEC further increases on the average case and that the chronological acquisition of events (as it is the case of the mechanical ventilation domain we considered) further increases performance.

2 Interfacing CEC with the Application Domain

We used CEC to model time and change in an intelligent monitoring system for ventilator therapy management. The medical expertise was obtained from clinicians at the Henri Mondor Hospital, in France. Clinicians provided about 50 rules covering both interpretation of data, prediction of patient's evolution and therapeutic decisions. This domain knowledge has been modeled using events and properties. Temporal reasoning is performed by CEC to interpret the status of the patient, the evolution of patient's ventilation and the evolution of the therapy.
An application specific module has been added for interface and data acquisition purposes. This module allows clinical staff to initialize the system according to information about the patient's (weight, type of pathology, ...). It receives incoming data from a gas analyzer (for expiratory partial pressure of CO2: PCO2) and from the ventilator (for expired volume VT and respiratory rate RR) and provides them to CEC in a proper format. CEC performs all the inferential activity needed to interpret the data received by the interface module and take the proper therapeutic decisions. The interface modifies the ventilator settings (level of PS, mode of ventilation) according to CEC decisions, carries out the therapeutic decisions taken by CEC, and presents the inferences done by CEC to the clinical staff.

The interface module reads the values of the clinical parameters with a frequency of 0.1 Hz, for a period of time whose duration can be changed (by CEC) according to the current clinical situation. At the end of this observation period, the interface module sends to CEC the averages of the clinical parameters read all over the period. As a result, CEC performs its inferential activity to update its database in response to the new information available about the patient. When the database update ends, the interface module reads the current value of those properties that prescribe the current level of assistance and the duration of the next observation period, respectively. The former is sent to the ventilator, and the new observation period is started. In some cases, CEC may also determine that an alarm should be raised for the clinical staff, because some specific situation is occurring (for example, patient is hypoventilated and the mode of ventilation should be changed to Control Mechanical Ventilation, or despite successive increases in assistance, the ventilation remains incorrect).

3 Exploiting CEC for Mechanical Ventilation Management

In this section, we present how the considered domain, especially the temporal part of the expertise, was modelled with CEC in terms of events and properties. Some examples, extracted from the knowledge base of our prototype, are used to illustrate specific points.

3.1 Events
The sequencing of the three main tasks (observation, diagnosis and therapy) is represented using three types of events that control the interactions between CEC and the interface module. The first type of event models the receipt of new values for the observed clinical parameters and has the form receive(VT, PCO2, RR), where VT, PCO2 and RR are numerical values for the total inspired volume (in ml), expiratory partial pressure of CO2 (in mmHg) and respiratory rate (in cycles/min), respectively. The second and the third type of event model requests for the assessment of the evolution of the patient status (evolutionRequest event), and for the determination of therapy (therapyRequest event).
The reasoning steps take place as follows. When an observation period ends (observation task), the interface module sends to CEC a receive(VT, PC02, RR) event containing the averages of the observed clinical parameters. This results in the derivation of the patient classification (first step of diagnosis task). Then, the interface module sends an evolutionRequest event (second step of diagnosis task), and finally it sends a therapyRequest event (therapy task). The time of occurrence of each of these three subsequent events is read from a system clock and sent to CEC together with the event (this allows to keep track of the precise times at which each activity started). After the third event is processed, the interface module reads the current suggested therapy from CEC and is ready to send commands to the ventilator and to start a new observation period, or to warn the clinical staff that some special case is taking place.

3.2 Properties
Table 1 summarizes some of the most relevant properties that have been defined to model the domain. As it can be noticed, all these properties have one parameter (denoted as X in the Table). Moreover, the values each parameter can take are mutually exclusive. This allows us to assume that when a property with a certain parameter value starts to hold, the previously holding property with the same name but with a different value in the parameter is terminated. Thus, we expressed a termination condition valid for all the properties of this specific domain in a very compact and practical way, by using just one npTerminates_at clause. This clause states that a generic property Property (characterized by a value Value for its parameter) terminates at instant T whenever an event Event, that initiates the same property but with a different parameter value (NewValue), happens at instant T. All the other clauses in the knowledge base will be wInitiates_at clauses. Thanks to the weak interpretation [1] provided by wInitiates_at clauses, all properties are thus concatenable properties [6]: if whenever they hold over two consecutive intervals (i.e. two consecutive period of observation) they hold over their union.

We will now provide examples of properties characterizing patient classification, and evolution of patient status.

Depending on the values of the three clinical parameters (respiratory rate rr, inspired volume vt, and expiratory partial pressure of CO2 pC02), the patient's ventilatory state can be classified as normal or abnormal. In particular, we distinguish six different abnormal states: insufficientVentilation, tachypnea, severeTachypnea, hyperventilation, hypoventilation, inexplicable Hypoventilation.

The patient's ventilatory state is classified by the patientStatus(X) property, where the parameter X can assume the value normal or can be one of the six abnormal states. The regions of parameter values characterizing ventilatory states are not unique, but depend on patient's morphology. As an example, if the patient's weight is greater than or equal to 55 kg, the lower
bound ($v_{\text{tmin}}$) for his/her inspired volume in a normal ventilatory state is 300 ml, otherwise 250 ml can be sufficient. Analogously, upper bounds for the expiratory pressure of CO$_2$ ($p_{CO2_{\text{max}}}$) and the respiratory rate ($r_{\text{rmax}}$) depend on patient's pathology. For example, $p_{CO2_{\text{max}}}$ is 55 mmHg or 65 mmHg for BPCO patients, and $r_{\text{rmax}}$ is 28 or 32 cycles/min for patients with neurologic disorders. These upper or lower bounds for parameters are set during system initialization. The acquisition of new information about parameter values is modeled by means of the receive event type. For example, the update receive($420,36,17$), happening at a given time, reports that VT is 420 ml, PCO2 is 36 mmHg and RR is 17 cycles/min.

Table 1. A selected subset of relevant properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_t(X)$</td>
<td>average value of total inspired volume observations taken over the last observation period</td>
</tr>
<tr>
<td>$p_{CO2}(X)$</td>
<td>average value of expiratory partial pressure of CO$_2$ observations taken over the last observation period</td>
</tr>
<tr>
<td>$r_r(X)$</td>
<td>average value of respiratory rate observations taken over the last observation period</td>
</tr>
<tr>
<td>patientStatus($X$)</td>
<td>classification of patient's state</td>
</tr>
<tr>
<td>evolutionStatus($X$)</td>
<td>classification of the evolution of patient's state over previous observation periods</td>
</tr>
<tr>
<td>weaningStatus($X$)</td>
<td>classification of the progress of the weaning procedure</td>
</tr>
<tr>
<td>weaningDuration($X$)</td>
<td>planned duration for the weaning procedure</td>
</tr>
<tr>
<td>assistanceLevel($X$)</td>
<td>prescribed level of mechanical ventilatory assistance</td>
</tr>
<tr>
<td>minimumAssistance($X$)</td>
<td>minimum allowed level of assistance</td>
</tr>
</tbody>
</table>

In response to receive events, a set of wInitiates at clauses updates recorded parameter values, while another set handles the patient ventilatory state (the state is normal in this example). When a new value VT for the average of the inspired volume parameter is received, property $v_t(VT)$ initiates. The npTerminates at clause previously discussed and the temporal aggregation mechanism [1] guarantee that the already holding MVI for $v_t$ is clipped only if the value in it is different from VT.

A sample clause from the latter set is the following (according to PROLOG's conventions, variables' names begin with an uppercase letter and constants with a lowercase letter):

```prolog
wInitiates_at(receive(VT,PCO2,RR),patientStatus(tachypnea),T):-
  happens_at(receive(VT,PCO2,RR),T),
  holds_at(vtmin(Vm),T), VT>=Vm,
  (PCO2 < 5; PCO2 > 20),
  holds_at(rrmax(Rm),T), RR<35, RR>=Rm.
```

That is, whenever an event of type receive($VT,PCO2,RR$) happens, and
property vtmin (Vm) holds at that time (indicating the minimum allowed value Vm for the inspired volume), and the observed value VT is greater or equal to Vm, and the observed value PCO2 is lower than 5 or greater than 20, and property rrmax (Rm) holds (indicating the maximum allowed value Rm for the respiratory rate parameter), and the observed value RR is lower than 35 and greater than Rm, then the property patientStatus (tachypnea) starts to hold (if it was not already holding), and whatever property with the same name but a different parameter value (i.e. patientStatus (X) with X not equal to normal) was holding previously, terminates in the same instant.

Properties concerning the evolution of the patient are initiated by events of the form evolutionRequest. The main property in this group is the evolutionStatus (ES) property, which gives an assessment of the evolution of the patient state over previous periods of observation. For example, evolutionStatus (s0) indicates that the patient is actually normal and he/she has not been abnormally ventilated (tachypnea or insufficient ventilation) since evolutionStatus (s0) was initiated.

An example of a clause describing a change in evolutionStatus is the following:

\[
\text{wInitiates} \_\text{at(evolutionRequest, evolutionStatus(s2), T):}\]

\[
\text{happens} \_\text{at(evolutionRequest, T),}
\]

\[
\text{holds} \_\text{at(evolutionStatus(s0), T),}
\]

\[
\text{holds} \_\text{at(patientStatus(PtnSts), T),}
\]

\[
(PtnSts==\text{severeTachypnea}; \ PtnSts==\text{hyperVentilation}).
\]

This clause states that whenever an evolutionRequest event happens, and evolutionStatus (s0) holds, and patientStatus (PtnSts) holds with PtnSts equal to severeTachypnea or hyperVentilation, then the property evolutionStatus (s2) is initiated. The property evolutionStatus (s2) indicates that the patient is in severe tachypnea or hyperventilation for the first time since the instant when s0 was started.

The evolutionStatus property is crucial in the therapy activity, because it is often necessary to consider more than one single observation period to determine the appropriate assistance level. For example, in case of tachypnea, the level of assistance is raised as soon as the tachypneic status is observed. But in order to tolerate short instabilities without modifying the global therapy, the initial level of assistance is restored if in the next two observations the patient status returns normal. On the contrary, the new level is maintained if another tachypneic status is observed in one of the next two observation periods.

The assistanceLevel (X) property indicates which level of assistance the ventilator has to deliver to the patient during the next observation period. As a response to therapyRequest events, the previous value of the assistance can be incremented or decremented by a specified step.
4 A Reasoning Example

The application of CEC to the patient monitoring problem presented in this paper has been extensively tested on patient's data reflecting several real clinical situations. We provide here one example (illustrated in Figure 1) taken from prototype operation over a real clinical situation. For clarity purposes, the figure depicts only the properties in the database which are most relevant to the considered case. All initiations and terminations of properties are graphically linked with their causing events. In the considered situation, a patient is initially in a normal state, then it becomes tachypneic leading to a temporary increase in assistance. Assistance is later decremented because the patient is again normal, but then it is incremented permanently, because the patient becomes tachypneic a second time. That is, a short ventilation instability does not lead to modification in the global therapy, while a successive inadaptation does.

![Figure 1. Trace of prototype operation (partial).](image)

5 Conclusions

CEC allows to introduce an explicit time and change representation into real-time patient monitoring systems while ensuring computational tractability. The response time obtained with the prototype is within the bounds required by the application and it is equal to few seconds on a portable 80386 computer and fractions of a second on a Sun SparcStation 2. Although the execution time for a CEC update increases polynomially with the number of recorded events [2], response within a given specified time can be guaranteed in these applications, by regularly moving the oldest, no more necessary events and MVIs from the active CEC database to a separate database when new events arrive (forgetting mechanism); the separate database is used for historical purposes, and is queried with CEC.

We modelled the patient's evolution as a state transition problem and change as a discrete process. Events introduce an abrupt change in state, and states are bounded by events. Similar considerations have been adopted in other
medical contexts, but instead of adopting custom-tailored temporal-reasoning methods, our approach is based upon CEC, that is, a general temporal reasoning tool that can be extended and adapted to the purposes of different applications.

An interesting feature, which comes at no additional cost when using CEC, is the ability of not only to monitor and control in real-time the therapy, but also to reason non-monotonically about the data that were recorded in its database. For example, episodic information obtained from laboratory tests reflecting the quality of blood gas exchange, can lead to a new interpretation of the expired CO₂ pressure and to revise the assumption about the quality of the patient's ventilation and of the adequacy of the past and current therapy.

CEC is also an interesting tool to maintain and query the temporal database dynamically built. Besides existing query capabilities that allow to retrieve important information about the history of a patient's case, the addition of new functions to allow the comparison of histories of different patients can contribute to the improvement of the actual system. We also pursue our effort to integrate CEC into a prototype working at the patient's bedside [3] and to test it in a clinical environment.

References