A Treatment Validation Protocol for Cyber-Physical-Human Medical Systems

Po-Liang Wu, Dhashrath Raguraman, Lui Sha
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: {wu87, raguram2, lrs}@illinois.edu

Julian M. Goldman
Mass. General Hospital and CIMIT
Cambridge, MA
Email: jgoldman@mdpnp.org

Richard B. Berlin Jr.
Department of Surgery, College of Medicine and Department of Computer Science
University of Illinois at Urbana-Champaign
Email: rberlin@illinois.edu

***Abstract*—In cyber-physical-human medical environments, coordinating supervisory medical systems and medical staff to perform treatments in in accordance with best practice is essential for patient safety. However, the dynamics of patient conditions and the non-deterministic nature of potential side effects of treatments pose significant challenges. In this paper, we propose a validation protocol to enforce the correct execution sequence of performing treatment, regarding preconditions validation, side effects monitoring, and expected responses checking based on the pathophysiological models. The proposed protocol organizes the medical information concisely and comprehensively to help medical staff validate treatments. Unlike traditional validation mechanism for cyber systems, the medical system cannot lock or rollback the states of physical components, such as patient conditions. Therefore, the proposed protocol dynamically adapts to the patient conditions and side effects of treatments. Moreover, a cardiac arrest scenario is used as a case study to verify the safety and correctness properties of the proposed protocol.**

I. Introduction

Improving effectiveness and safety of healthcare infrastructure is an important issue for cyber-physical medical systems. Several artificial intelligence techniques, such as expert systems [8], fuzzy logic [9], Bayesian statistics/inference [1], have been proposed to many medical applications, for example, diseases diagnosis, treatment suggestion, and image classification, etc. Unlike those diagnosis support and treatment advice systems, in this work, we focus on reducing preventable medical errors by validating treatments and monitoring patient's physiological responses based on the pathophysiological models[1].

Statistics indicates that preventable medical error rate is highest in ICU as compared to other hospital units [14]. Many preventable medical errors are result from unintended deviation[2] from the best practice medical guidelines, because medical staff are under tremendous pressure and overloaded by the great amount of unorganized information [12]. In order to reduce preventable medical errors, treatment validation is an important aspect. In this work, validation includes checking preconditions of the treatments, continuously monitoring potential side effects, and checking patients' physiological responses. The preconditions and side effects are obtained from the pathophysiological model, which will be detailed in Section II, in collaboration with medical staff. The following examples illustrate the concept of treatment validation.

Example 1: In a cardiac arrest scenario, medical staff intend to activate a defibrillator to deliver a therapeutic level of electrical shock that can correct certain types of deadly irregular heart-beats such as ventricular fibrillation. The medical staff need to check two preconditions: 1)patient's airway and breathing are under control and 2)the EKG monitor shows a shockable rhythm[3]. Suppose the patient's airway is open and breathing is under control. However, the EKG monitor shows a non-shockable rhythm[4]. In order to induce a shockable rhythm, a drug, called epinephrine, is commonly given to increase cardiac output. Giving epinephrine, nevertheless, also has two preconditions: patient's blood pH value should be larger than 7.4.[5], and urine flow rate should be greater than 12 mL/s[6]. In order to correct these two preconditions, sodium bicarbonate should be given to raise blood pH value, and intravenous fluid should be increased to improve urine flow rate.

The cascading relations between preconditions and corrective treatments can be captured by a tree structure, as shown in Figure 1. It seems that the satisfaction of preconditions can be achieved by the well-known post-order tree traversal. The tree structure will be detailed in Section III-A. However, in medical environment, a treatment may not be effective, and the side effects of a treatment may invalidate the previously satisfied preconditions of any tree nodes at any time.

[1]Pathophysiology is a convergence of pathology with physiology to describe the physiology changes associated with a disease or syndrome.

[2]Unintended deviation is the medical staff unintentionally perform treatments noncompliance with the guidelines

[3]The shockable rhythms are ventricular fibrillation and ventricular tachycardia [5].

[4]Non-shockable rhythms are asystole and pulseless electrical activity

[5]Severe acidosis, which is an increased acidity in the blood and other body tissue, will significantly reduce the effectiveness of epinephrine [5]

[6]If a patient suffers from kidney insufficiency, giving epinephrine may worsen the kidney function and cause acute renal failure [5].

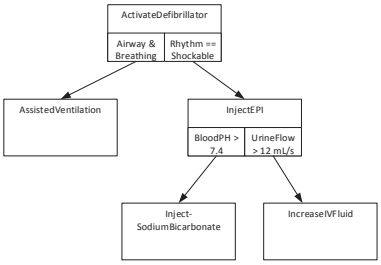


Fig. 1. Treatments and preconditions tree

Example 2: One potential side effect of sodium bicarbonate is suppressed respiratory drive[7], which adversely affect patient breathing. Since the precondition is invalidated, the tree should be expanded to include the corrective treatment, such as provide assisted ventilation. In addition, increasing IV fluid volume may not successfully improve patient's urine flow rate. In this case, diuretics, such as Lasix, should be given, which leads to a different tree structure.

As illustrated in Example 2, the dynamics of patient conditions and the non-deterministic behavior of treatments pose significant challenges. The post order tree traversal alone may not be able to address these challenges. Similar situation can be found in many other medical scenarios, for instance, laparoscopic abdominal surgery [16] and airway laser surgery [10]. In order to reduce preventable medical errors, we develop a validation protocol in accordance with pathophysiological model and best practice guidelines. From a system verification perspective, we use a model checking tool, UPPAAL [3], to formally verify the correctness of the proposed protocol. The reason we choose UPPAAL is that it provides nice user interface for the physicians to review the developed models. However, the proposed protocol can be modeled and verified by other model checking tools, such as NuSMV [4]. The contributions of this paper can be summarized as follows:

- We propose a validation protocol to structure preconditions and treatments and enforce the correct sequence of performing treatments.
- The proposed protocol monitors the dynamics of patient conditions and the side effects of the treatments and provides feedback to the medical staff.
- We use resuscitation as a case study to verify the safety and correctness properties of the proposed protocol.

II. System Overview

In this section, we describe the technical challenges of developing a treatment validation protocol and present the system and pathophysiological models used in the paper.

A. Treatment Validation and Technical Challenges

From the Example 1 and Example 2 described in the previous section, we understand that validating treatments is essential for patient safety. We develop a validation protocol to address the following three essential aspects.

Precondition: A treatment can be performed only if the preconditions are satisfied. Unlike traditional cyber system preconditions, the medical system cannot lock or rollback the states of physical components, such as patient conditions. The system should request corrective treatments from the medical staff if certain preconditions are not satisfied. Nevertheless, the corrective treatments may have preconditions as well, which result in cascading of preconditions and treatments. Therefore, the system must organize preconditions and treatment to help medical staff keep track of preconditions and performed treatments.

Potential side effect: The side effects of a treatment may adversely affect other treatments or invalidate previously satisfied preconditions. The system should continuously monitor the potential side effects and alert medical staff to adjust the treatments. Moreover, the system must dynamically change the structure of preconditions and treatments to reflect the adjustments from medical staff.

Expected response: The patient response must be checked after a treatment is performed. If patient response is not as expected, the system must alert the medical staff to issue an alternative treatment.

B. System and Pathophysiological Models

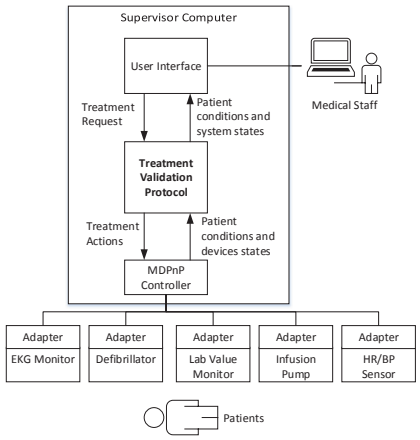


Fig. 2. The System Architecture

Figure 2 shows the system architecture. We envision that Medical Device Plug and Play (MDPnP) [6] will provide a centralized supervisory framework for integrating networked medical devices. We take advantage of such an integrated framework and propose a *Treatment validation protocol* on top of MDPnP. Medical staff issue a treatment request through user interface. The proposed *Treatment validation protocol* validates the preconditions, monitors potential side effects, and checks expected responses. If the treatments are approved by the medical staff, the protocol requests the medical devices to perform treatments through MDPnP controller. On the other hand, based on the feedback of patient conditions, the proposed protocol dynamically checks the side effects and

[7]Respiratory drive is the control of respiration, which involves the exchange of oxygen and carbon dioxide

requests the medical staff to adjust the treatments. The details of the protocol will be presented in Section III.

Let us first formally define the pathophysiological model, which consists of physiological condition and treatments.

Definition 1. ***PhysiologicalCondition (PC)*** *is defined as a tuple $<$ Checker, PM, Operator, RV $>$, where*
Checker is the entity capable of checking the condition[8]*,* *Checker* $\in$ *{System, MedicalStaff},*
PM (physiological measurement) $\in$ *{EKGRhyhtm, HeartRate, BloodPressure...},*
Operator $\in \{>, <, =, \leqslant, \geqslant\}$*,*
RV is the reference value, which can be threshold, trend, or pattern of the physiological measurements.

Definition 2. ***PhysiologicalConditionSpace (PCS)*** *is defined as the union of all known physiological conditions.*

In this work, we assume *PCS* is correctly specified by the medical staff. Verification and validation of *PCS* is out of the scope of this paper.

Definition 3. *A* ***treatment*** *is defined as a tuple $<$ Agent, Action, PS, SS, ES, L $>$, where*
Agent is the entity that performs the treatment, Agent $\in$ *{MedicalDevices, MedicalStaff},*
Action is the set of executable instructions,
PS is a set of preconditions that must be satisfied before preforming the Action, PS $\subset$ *PCS,*
SS is a set of potential side effects, SS $\subset$ *PCS*
ES is a set of expected responses after performing the Action, ES $\subset$ *PCS*
L is the life cycle, which specifies the time interval between the treatment being performed and the treatment has no further effect on the patient[9]*.*

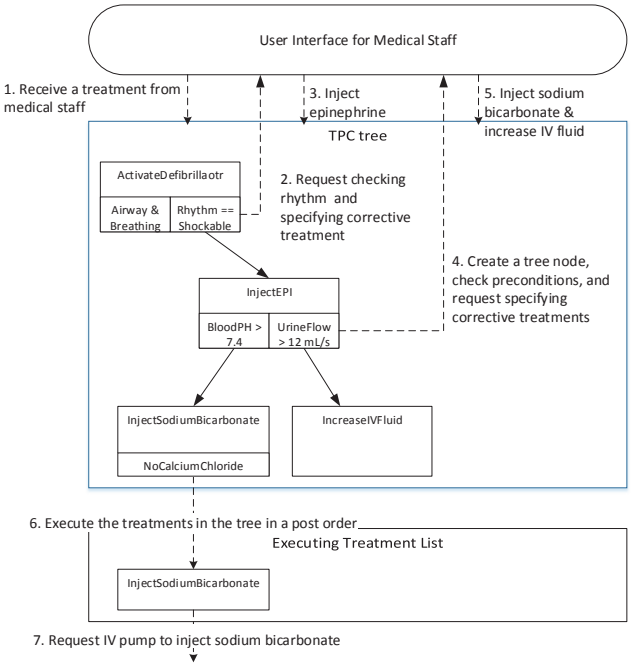
For example, the treatment of injecting epinephrine is defined as <IVPump, InjectEpinephrine, {<System, BloodPH, $\geqslant$, 7.4 $>$, $<$ System, UrineFlow, $\geqslant$, 12 mL/s $>$}, {$<$ System, BloodPressure, $\geqslant$, 210 $>$}[10], <MedicalStaff, EKGRhythm, =, ShockableRhythm $>$, 4 min $>$

III. Treatment Validation Protocol

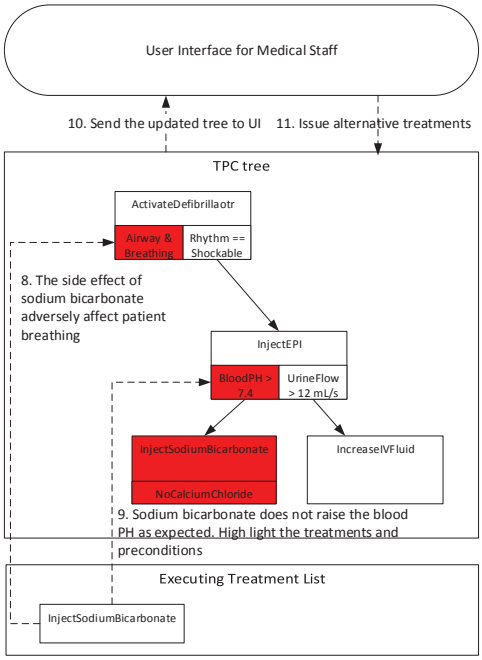
A *Treatment Precondition and Correction (TPC)* tree is proposed in Section III-A to capture the standard medical practice through structuring treatments and preconditions. Then, we further develop a validation protocol to dynamically adapt the TPC tree to the patient conditions and side effects in collaboration with the medical staff.

A. Treatment Precondition and Correction Tree Structure

We propose a *Treatment Precondition and Correction (TPC)* tree for structuring the preconditions and treatments. Based on the TPC tree, the medical staff can keep track of the preconditions and treatments with concise and comprehensive physiological information. A tree node represents a treatment, and the number of children equals the number of the preconditions of the treatment. An edge represents the relation of a precondition and the corresponding corrective treatment. The tree is built in a top-down manner, and the root node is the treatment that the medical staff intend to perform in



(a) TPC tree construction and execution



(b) Side effect monitoring and expected responses checking

Fig. 3. The validation protocol

[8]Some physiological conditions can be automatically checked by the system, such as threshold and trend. Some complicated conditions must be checked by medical staff, such as EKG rhythm.

[9]Different treatments have different life cycle. For example, the life cycle of a drug is decided based on the drug metabolism.

[10]Epinephrine may adversely cause elevation in blood pressure.

the beginning. If any precondition of the treatment is not satisfied, a child node for the corrective treatment is added. Since the corrective treatment may have its own preconditions and further corrective treatments, the height of the tree increases accordingly. The leaf nodes are the treatments that either have no preconditions or the preconditions are satisfied. In addition, due to the dynamics of patient conditions and potential side effects, the TPC tree is not static and requires to be dynamically updated. The algorithms of building and updating the TPC tree will be detailed in the next section.

Let us use the following scenario to illustrate how validation of preconditions and corrective treatments can be achieved with the proposed TPC tree. Suppose the medical staff send a request to activate a defibrillator. The system builds the TPC tree rooted at *ActivateDefibrillator*. Since activating defibrillator has two precondition: airway and breathing are under control, and EKG shows a shockable rhythm. Assume that the first precondition is satisfied, but our system cannot automatically analyze the EKG rhythm and requires medical staff's diagnosis. The system requests the medical staff to check EKG rhythm and specifies a corrective treatment if EKG shows a non-shockable rhythm. Suppose the medical staff determine that EKG rhythm is non-shockable and specify giving epinephrine, as illustrated in step 1–3 of Figure 3(a). Giving epinephrine has two preconditions: patient's blood pH larger than 7.4 and urine flow rate larger than 12 mL/s. In this scenario, these two preconditions can be checked by the system and neither of them are satisfied. The system, then, requests the medical staff to specify the corresponding corrective treatments. One corrective treatment is injecting sodium bicarbonate for correcting blood pH value, and the other is increasing IV fluid volume for improving urine flow rate, as illustrated in step 4 and 5 of Figure 3(a). Since injecting sodium bicarbonate requires that calcium chloride is not currently being injected[11], the system must check the IV pump status. Suppose IV pump is not currently injecting calcium chloride and there is no precondition for increasing IV fluid volume. The construction of TPC tree is complete, because there is no further corrective treatment to be added.

The system sends the TPC tree to the medical staff for approval. If the medical staff disapproves the treatments, the system discards the TPC tree and waits for the medical staff to issue a new treatment. Otherwise, the system executes the treatments in a post order, since the treatment of the leaf node must be performed first to correct the precondition. In this case, the system requests the IV pump to inject sodium bicarbonate, as illustrated in step 6 and 7 of Figure 3(a).

The system monitors the potential side effects and checks the expected responses of sodium bicarbonate. The following cases might occur:

Case 1: Injecting sodium bicarbonate does not invalidate any precondition and successfully raises patient's blood pH value higher than 7.4. The system removes *InjectSodiumBicarbonate* node from the TPC tree and executes *IncreaseIVFluid*.

Case 2: Sodium bicarbonate adversely affects the patient breathing. The system "highlights" the preconditions in the TPC tree and requests the medical staff to specify a corrective treatment, as illustrated in step 8 of Figure 3(b). Suppose the medical staff specifies providing assisted ventilation. The system adds a treatment node to the TPC tree*AssistedVentilation*.

Case 3: Sodium bicarbonate fails to raise patient's blood pH value. The system "highlights" the treatment node and the corresponding preconditions, as illustrated in step 9 of Figure 3(b). The system sends the TPC tree to the medical staff and requests them to specify an alternative corrective treatment. Once the medical staff specify a new corrective treatment, the system must update the TPC tree and check the preconditions of the new treatment, as described before.

B. Description of the Protocol

Algorithm III.1 Pseudo code for constructing TPC tree

```
1  checkPrecondAndBuildTree(TPCTreeNode root){
2    PrecondSet precondSet←root.treatment.PS;
3    TreatmentSet correctiveTreatmentSet;
4    Precondition precond;
5
6    while(precondSet ≠ ∅){
7      for each precond ∈ precondSet{
8        if(precond.Checker is medical devices){
9          if(precond is satisfied)
10           precond.status←SATISFIED;
11         else
12           precond.status←UNSATISFIED;
13       } else if(precond.Checker is medical staff)
14         precond.status←UNKNOWN;
15
16     }
17     ...
18     correctiveTreatmentSet←receiveFromUI();
19
20     // check the completeness of corrective
           treatments
21     for each precond ∈ precondSet{
22       if(precond.status == UNKNONW || (precond.
             status == UNSATISFIED && no corrective
             treatments))
23         sendExceptionToUI(INCOMPLETE);
24         ...
25     }
26
27     for each treatment ∈ correctiveTreatmentSet{
28       // create child nodes for the corrective
             treatment and insert the childNode to
             the tree
29       ...
30       // update the precondition set
31       precondSet←∪treatment.PS
32     }
33
34   }
35 }
```

In this section, we describe the validation protocol in detail and show part of the pseudocode in Algorithm III.1 and Algorithm III.2. Algorithm III.1 shows the construction of the TPC tree. Algorithm III.2 shows the post-order execution and

[11]Sodium bicarbonate and calcium chloride cannot be simultaneously injected through the same IV pump, because they would form calcium carbonate and make the two drugs ineffective.

Algorithm III.2 Pseudo code for post order execution and side effect monitoring

```
1  postOrderExecution(TPCTreeNode node) {
2    ...
3    TPCTreeNode childNode;
4    for each childNode ∈ node.childNodeList{
5      postOrderExecution(childNode);
6    }
7
8    if(all preconditions are satisfied){
9      performingTreatment(node);
10     setUpTimerForExpectedResponse(node);
11     ...
12   } else{
13     sendExceptionToUI(
           IneffectiveCorrectiveTreatment);
14     ...
15   }
16 }
17
18 // Periodically monitors the side effects of the
       treatments in the executing list
19 runTimeMonitoring(TPCTreeNode root) {
20   TPCTreeNodeSet affectedNodeSet←∅;
21   // Monitoring side effect
22   for each treatment in the executingList{
23     if(sideEffects affect other treatments ||
           sideEffects invalidate the preconditions)
           {
24       affectedNodeSet←affectedNodeSet∪
             getAffectedNodeSet(treatment);
25       setTreeNodeStatus(affectedNodeSet);
26   }
27
28   checkPrecondAndBuildTree(root);
29   sendToUI(root);
30
31   if(medical staff approves the tree){
32     postOrderExeuction(root);
33   } else{
34     // abort the treatment
35     ...
36   }
37 }
```

dynamic side effect monitoring. Our protocol consists of the following phases.

1. TPC tree construction phase: The system receives a treatment from the medical staff and starts to build a TPC tree in a breath-first manner. The system checks the preconditions of the received treatment. If any precondition is not satisfied or must be checked by the medical staff, the system sends the tree to the medical staff and requests them to check the preconditions and specify the corrective treatments, as shown in the line 7–19 of Algorithm III.1. After getting the input from the medical staff, the system checks if each unsatisfied precondition has a corresponding corrective treatment. If the corrective treatments are incomplete, an exception is sent to the medical staff, as shown in the line 21–24 of Algorithm III.1. The system then adds the corrective treatments as child nodes to the TPC tree. Since the corrective treatments may introduce a new set of preconditions, the system checks the preconditions and expands the tree. If there are no further preconditions to check or all the preconditions are satisfied, TPC tree is sent to the medical staff for approval. If the medical staff approves the TPC tree, the system enters the execution and monitoring phase.

2. Execution and monitoring phase: The system executes the treatments in the TPC tree in a post order. In order to keep track of all the ongoing treatments, the system maintains an executing treatment list. Since patient conditions dynamically change, the system checks the preconditions of the treatment again before performing it[12]. If the preconditions are satisfied, the system inserts the treatment into the executing list and requests the medical devices to perform the treatment. The system needs to check the expected response after a time interval, specified by the medical staff, as shown in the line 8–15 of Algorithm III.2. The details of checking expected responses will be explained in the next phase. In addition, the system periodically monitors or requests the medical to check the potential side effects of the treatments in the executing list. The side effects may lead to the following situations:
2-a The side effects of a treatment interfere the other ongoing treatments. Specifically, the side effects cause the patient's physiological measurements changing in an opposite direction to the expected responses of other treatments.
2-b The side effects invalidate the previously satisfied preconditions in the TPC tree.
In both cases, the system will highlight the interfered treatments and the corresponding preconditions in the TPC tree and send an exception to the medical staff, as shown in the line 22–25 of Algorithm III.2. The medical staff can adjust the existing treatments, such as increasing or decreasing the drug dosage, or specifying alternative treatments. The system then updates the tree, as described in the previous phase.

After the system informs the side effects to the medical staff and updates the TPC tree with their approval, the system restarts the post order execution.

3. Checking expected responses phase: As explained in the previous phases, the system must check patient's conditions against the expected responses of the treatment when the timer fires. If the patient conditions are as expected, the system removes the corresponding treatment node from the TPC tree and executes the next treatments based on the post order of the TPC tree. If the patient conditions do not improve as expected, the system highlights the unsuccessful preconditions and the corresponding corrective treatments on the TPC tree for the medical staff. The medical staff can specify an alternative corrective treatment, and the system updates the TPC tree accordingly and restarts the post order execution.

By following the above procedures, the system preforms the treatments and corrects the preconditions in a bottom-up manner. Even if the side effects adversely affect other treatments or invalidate the preconditions, the system is capable of updating the TPC tree and let medical staff change the treatments.

[12]If the precondition that the treatment intends to correct is satisfied before the treatment is performed, the treatment is removed from the tree.

IV. Correctness of the Protocol and Discussion

A. Correctness of the Protocol

Theorem 1. *Under the proposed protocol, a treatment is performed only if all preconditions of the treatment are satisfied.*

Proof: Proof by contradiction. We assume that a TPC tree node n_α is executed, but one of the preconditions of n_α is not satisfied. Since the protocol adopts post-order execution, child nodes of n_α must be executed before n_α can be executed. A child node is removed from the tree if and only if its expected responses are satisfied. Consequently, the preconditions of n_α must all be satisfied before the child nodes are removed from the tree. In addition, according to the line 8–9 of Algorithm III.2, the precondition is checked again before the treatment is performed. We arrive at contradiction. ■

We then prove that our protocol can correct the unsatisfied preconditions and reach the root node, which is the treatment that the medical staff intend to perform in the beginning.

Definition 4. *An TPC tree is **well-formed** if each unsatisfied preconditions have a tree node for the corrective treatment.*

Theorem 2. *The root node of a well-formed TPC tree is reachable if the corrective treatments are effective and the preconditions are not invalidated by the side effects.*

Proof: Proof by induction. Let N be the number of preconditions in a TPC tree.
Base case: When N = 0, the statement is trivially true.
Induction step: Assume the statement is true for $1 \leqslant N \leqslant k$, Consider N = k + 1. There are two possible cases.
Case 1: The (k+1)th precondition is satisfied, the protocol starts post order execution as if there are only k preconditions in the tree.
Case 2: The (k+1)th precondition is not satisfied. Since the tree is well-formed, by definition, the (k+1)th precondition has a corresponding corrective treatment node. Moreover, the corrective treatment can successfully correct the (k+1)th precondition because the corrective treatments are effective and the preconditions are not invalidated by the side effects. After the (k+1)th precondition is satisfied, the protocol traverses the tree as if there are only k preconditions.
In either case, the induction case holds. Therefore, by induction, the statement is true. ■

Lemma 1. *The checkPrecondAndBuildTree function, as shown in Algorithm III.1, either builds a well-formed TPC tree or raises an exception.*

Proof: Proof by contradiction. Assume Algorithm III.1 build a non-well-formed tree, and no exception is raised. Suppose a precondition p_α, is not satisfied and there is no corresponding corrective treatment node in the TPC tree. The status of the precondition p_α will be set to *UNSATISFIED*, as shown in line 8–12. After the medical staff specify the corrective treatments, the protocol checks if all the unsatisfied preconditions have corresponding corrective treatments. If not, an exception is sent to the medical staff, as shown in line 21–25. We reach a contradiction. ■

Theorem 3. *Suppose side effects of a treatment invalidate any precondition and make the TPC tree become non-well-formed. The protocol updates the tree to be well-formed if the medical staff correctly specifies the corrective treatments.*

Proof: The protocol periodically monitors the potential side effects of the treatments in the executing list and checks if any precondition is invalidated. As shown in the line 22–27 of Algorithm III.2, if any previously satisfied precondition is invalidated due to the side effects, the protocol "highlights" the affected tree nodes and preconditions. The protocol, then, calls the *checkPrecondAndBuildTree()* to update the tree. According to Lemma 1, since the medical staff correctly specifies the corrective treatments, the updated tree is well-formed. ■

The above theorems prove that our protocol validates treatments in respect of validating preconditions, monitoring side effects and checking expected responses and adapts the TPC tree to the dynamics of patient conditions and non-determinism of the treatments.

B. Discussion

In this section, we discuss the rationale behind the proposed protocols as well as the limitations and potential solutions.

First, the proposed protocol executes the treatments nodes of a TPC tree in a post order instead of all the leaf nodes concurrently. The reason for limiting the concurrent treatments is medical staff generally perform treatments in sequence. For example, **A**irway, **B**reathing, and **C**irculation is a commonly accepted sequence for assessment and treatment of patients[13]. In addition, concurrent treatments make the medical staff hard to reason the effectiveness of treatments and side effects of treatments. However, the system allows the medical staff to specify compatible treatments, which are the treatments that can be performed concurrently. The system performs the compatible treatments concurrently as long as their preconditions are satisfied.

Second, the proposed protocol requires medical staff to specify corrective treatments if certain preconditions are not satisfied. To improve usability, when the medical staff click the unsatisfied precondition on the user interface, a drop down list of commonly used medications or treatments that can correct this precondition is displayed. Nevertheless, the system does not recommend which treatment is the best to this patient's specific conditions.

V. Verification

In this section, we first describe a resuscitation scenario as our case study. We, then, model the proposed protocol in UPPAAL [3], which is a model checking tool, to verify the safety and correctness properties.

[13]Other variations, such as CAB and ABCDE, are proposed for different medical scenario [13]

A. Resuscitation Case Study

Cardiac arrest is the abrupt loss of heart function and can lead to death within minutes. American Heart Association (AHA) provided resuscitation guidelines for the urgent treatment of cardiac arrest [5]. The general procedures of resuscitation consist of the following steps.

1. Cardiopulmonary resuscitation (CPR): the medical staff perform CPR for at least two minutes. In the mean time, other medical staff try to access intravenous vein (IV therapy) and inject drugs.

2. Check heart rhythm: the medical staff check the heart rhythm. If the rhythm is non-shockable, the medical staff should keep performing CPR and giving drugs, such as vasopressin and epinephrine. If the rhythm is shockable, go to the defibrillation step.

3. Defibrillation: if the medical staff determine the heart rhythm is a shockable one, they should activate a defibrillator to deliver electrical energy to the heart to regulate the heart rhythm. If the patient's heart rhythm is still abnormal, the medical staff should perform CPR again (back to step 1).

Furthermore, the side effects of the treatments may cause adverse interactions. Therefore, the medical staff should closely monitor the patient's conditions and preform alternative treatments if the side effects occur.

B. UPPAAL Model and Verification

We model the proposed protocol in UPPAAL. The system consists of the following models: user interface, validation protocol, side effect monitor, EKG monitor, defibrillator, IV Pump, blood pH monitor, and urine flow rate sensor. The models communicate using UPPAAL synchronization channels and shared variables.The user interface models follows the three-step resuscitation procedure and contains a list of pre-defined preconditions and treatments. The medical devices send the patient's physiological measurements, which are modeled as non-deterministic transitions, to the validation protocol. In addition, the medical devices also receive the treatment requests from the protocol and change the states accordingly.

Due to the space limit, we present a simplified UPPAAL model of the treatment validation protocol in Figure 4. The model reflects Algorithm III.1, builds a TPC tree based on the input from the user interface and checks the expected responses. When the model receives a treatment request from the user interface, the model creates a *RootNode* and checks the preconditions of the treatment using boolean function *checkPreconditions*. If the preconditions are satisfied and the medical staff confirm the treatment, the model goes to the *StartToExecute* state. Otherwise, the model goes to the *Unsatisfied* state and iteratively creates tree nodes, which is implemented in *assignChildNode* function, based on the corrective treatments received from the user interface. If any unsatisfied precondition does not have a corrective treatment, the model sends *IncompleteTreatmentsAlert* to the user interface. Otherwise, the model starts post order execution if the user interface approves the tree. If the preconditions are satisfied, the model adds the treatment to the executing list and sends action commands to the corresponding devices. If the preconditions are not satisfied, the model loops back to the *Unsatisfied* state. In addition, the model checks the expected response of the treatment after the timer fires.

We show part of the verified safety and correctness properties in Table I. We verified two sets of properties in UPPAAL: medical safety properties and protocol correctness properties. The medical safety properties capture the safety requirements of the resuscitation scenario, which are given by the medical staff. On the other hand, the protocol properties guarantee the correctness of the proposed protocol. For instance, property *P8* can be checked by the formula:
SideEffectMonitor.sideEffectOccur==true --> SideEffecMonitor.UpdateTree && isWellFormed(RootNode), where *isWellFormed* is a boolean function to check if the TPC is well-formed, as defined in Section IV. The above formula verifies that if variable *sideEffectOccur* is true, the *UpdateTree* state is eventually reached and *isWellFormed* returns true. Other safety and correctness properties can be verified with similar formula.

VI. Related Work

Medical Device Plug-and-Play (MDPnP) provides flexibility and interoperability for medical systems [7], and this work is part of the ongoing effort. There are several dimensions to design safe and flexible medical systems, including supervisory control, system models, verification and validation. Some closed-loop control frameworks are proposed to mitigate safety hazards [2]. System modeling and formal verification are important issues for developing and analyzing medical systems [11]. King *et al.* [11] proposed a formal specification language to express and reason safety properties of on-demand medical systems. Pajic *et al.* combined simulation-based analysis and model checking to guarantee safety of closed-loop medical systems [15]. In this work, the proposed protocol validates the treatments according to the pathophysiological models specified by the medical staff.

Expert systems aim to capture domain-specific expert knowledge in order to allow systems to induce rules, make inferences and arrive at a specific conclusion [8]. Several inference and decision making methodologies have been proposed on medical expert systems, such as rule-based reasoning [17], fuzzy logic [9] and probabilistic network [1]. However, many clinical problems are complicated and involving many dynamic decision-making, so straightforward attempts to chain together larger sets of rules encounter major difficulties. Unlike those decision support systems, in this work, we focus on validating the treatments in accordance with best practice medical guidelines.

VII. Conclusion and Future Work

In this paper, we develop a treatment validation protocol to enforce the correct execution sequence regarding precondition validation, side effects monitoring, and expected responses checking. The proposed TPC tree structures the preconditions and corrective treatments, which provides a logical path for

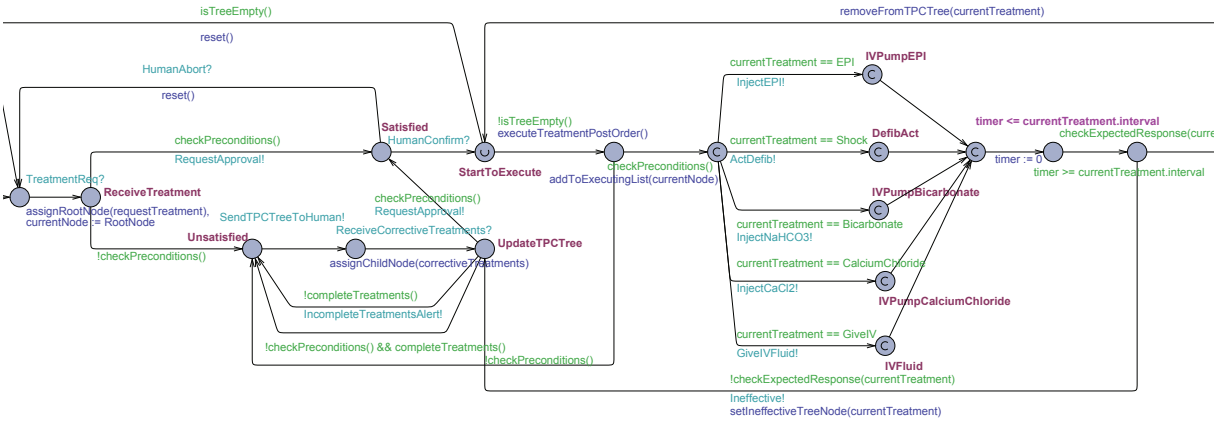


Fig. 4. The validation protocol

TABLE I
VERIFIED PROPERTIES OF THE RESUSCITATION SCENARIO

	Verified Properties
Medical safety properties	P1: Defibrillator is activated only if the EKG rhythm is a shockable one and airway and breathing is normal.
	P2: Epinephrine is injected only if the blood pH value is larger than 7.4 and urine flow rate is higher than 12 mL/s.
	P3: If the side effect of sodium bicarbonate adversely affects the breathing, the tree is updated with a new treatment node for assisted ventilation.
	P4: If epinephrine does induce a shockable rhythm, the tree is updated with an alternative treatment node, *vasopressin*.
Protocol properties	P5: There is no deadlock in the system.
	P6: A treatment is performed only if all its preconditions are satisfied.
	P7: If side effect does not occur, the root node of the TPC tree is added to the executing list
	P8: If side effects invalidate a precondition, the TPC tree is updated and well-formed.

medical staff to keep track of the medical procedure. In collaboration with medical staff, the proposed protocol adapts the TPC tree to the dynamics of patient conditions and non-deterministic behavior of treatments. We use resuscitation as a case study to verify the safety and correctness properties of the developed system.

In this paper, we only verify the safety and correctness properties of the proposed validation protocol. As future work, we would like to collect medical error case studies from Food and Drug Administration (FDA) and evaluate the reduction of the medical errors with the proposed protocol.

Acknowledgement: This work is supported in part by NSF CNS 13-29886, NSF CNS 13-30077, by ONR N00014-12-1-0046 and by Massachusetts General Hospital

REFERENCES

[1] S. Andreassen, F. V. Jensen, and K. G. Olesen. Medical expert systems based on causal probabilistic networks. *International journal of biomedical computing*, 28(1):1–30, 1991.

[2] D. Arney, M. Pajic, J. Goldman, I. Lee, R. Mangharam, and O. Sokolsky. Toward patient safety in closed-loop medical device systems. In *Proceedings of the 1st ACM/IEEE ICCPS*. ACM, 2010.

[3] G. Behrmann, A. David, and K. Larsen. A tutorial on uppaal. *Lecture Notes in Computer Science*, pages 200–236, 2004.

[4] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 359–364. Springer, 2002.

[5] J. M. Field, M. F. Hazinski, M. R. Sayre, L. Chameides, S. M. Schexnayder, R. Hemphill, R. A. Samson, J. Kattwinkel, R. A. Berg, F. Bhanji, et al. Part 1: executive summary 2010 american heart association guidelines for cardiopulmonary resuscitation and emergency cardiovascular care. *Circulation*, 122(18 suppl 3):S640–S656, 2010.

[6] J. Goldman, S. Whitehead, S. Weininger, and M. Rockville. Eliciting clinical requirements for the medical device plug-and-play (MD PnP) interoperability program. *Anesthesia & Analgesia*, 102:S1–54, 2006.

[7] J. Goldmann. Medical devices and medical systems—essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE)—part 1: General requirements and conceptual model. *draft ASTM TC F*, 29, 2009.

[8] D. L. Hudson. Medical expert systems. *Wiley Encyclopedia of Biomedical Engineering*, 2006.

[9] D. L. Hudson and M. E. Cohen. Fuzzy logic in medical expert systems. *Engineering in Medicine and Biology Magazine, IEEE*, 13(5):693–698, 1994.

[10] W. Kang, P. Wu, L. Sha, R. B. Berlin, and J. M. Goldman. Towards safe and effective integration of networked medical devices using organ-based semi-autonomous hierarchical control. Technical report, University of Illinois at Urbana Champaign, 2012.

[11] A. L. King, L. Feng, O. Sokolsky, and I. Lee. A modal specification approach for on-demand medical systems. In *Proceedings of the Third International Symposium on Foundations of Health Information Engineering and Systems*, 2013.

[12] L. T. Kohn, J. M. Corrigan, M. S. Donaldson, et al. *To err is human: building a safer health system*, volume 627. National Academies Press, 2000.

[13] D. R. Kool and J. G. Blickman. Advanced trauma life support®. abcde from a radiological point of view. *Emergency radiology*, 14(3):135–141, 2007.

[14] A. Latif, N. Rawat, A. Pustavoitau, P. J. Pronovost, and J. C. Pham. National study on the distribution, causes, and consequences of voluntarily reported medication errors between the icu and non-icu settings*. *Critical care medicine*, 41(2):389–398, 2013.

[15] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. Goldman, and I. Lee. Model-driven safety analysis of closed-loop medical systems. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 2012.

[16] M. Perrin and A. Fletcher. Laparoscopic abdominal surgery. *Continuing Education in Anaesthesia, Critical Care & Pain*, 4(4):107–110, 2004.

[17] S. Tsumoto and H. Tanaka. Automated discovery of medical expert system rules from clinical databases based on rough sets. In *KDD*, pages 63–69, 1996.