A Protocol of Coordinating Treatments for Cyber-Physical-Human Medical Systems

October 21, 2013

Abstract

In cyber-physical-human medical environments, coordinating supervisory medical systems and medical staff to perform treatments in a correct sequence is essential for patient safety. The coordination of treatments should be automated with medical staff's approval to assist the prevention of medical errors. However, the changing of patient conditions and the non-deterministic nature of potential side effects of treatments pose significant challenges. In this paper, we propose a coordination protocol to enforce the correct execution sequence of performing treatment, regarding preconditions validation, side effects monitoring, and expected responses checking. Moreover, the proposed protocol dynamically adapts to the patient conditions and side effects of treatments in collaboration with medical staff. A simplified cardiac arrest is used as a case study to verify the safety and correctness properties of the proposed protocol.

1 Introduction

Cyber-Physical medical systems tightly integrate physical medical devices, supervisory computers, and cooperative medical staff. Medical Device Plug and Play (MDPnP) [8, 9] is a generic framework for medical devices interoperability. Since the medical infrastructure is designed to improve patient safety, coordination of distributed medical devices, supervisory systems, and medical staff becomes an important issue. In an intensive care unit (ICU), patients are seriously ill, and there is often short of time and short of information. Statistics indicates that preventable medical error rate is highest in intensive ICU as compared to other hospital units [5, 18]. Most of those preventable medical errors result from concurrent and uncoordinated treatment actions [15]. In this paper, coordination is defined as enforcing the correct sequences of performing a treatment, including validating preconditions of the treatments, continuously monitoring potential side effects, and checking patient's responses. If a precondition is not satisfied, a corrective treatment should be performed in order to improve patient's condition and satisfy the precondition. Nevertheless, the corrective treatment may have a different set of preconditions and result in cascading of preconditions and treatments.

Example 1: In a cardiac arrest scenario, medical staff intend to activate a defibrillator to deliver a special type 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[1] and 2)the EKG monitor shows a shockable rhythm[2]. Suppose the patient's airway is open and breathing is under control. However, the EKG monitor shows a non-shockable rhythm[3]. 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.[4], and urine flow rate should be greater than 1 mL/min[5]. 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.

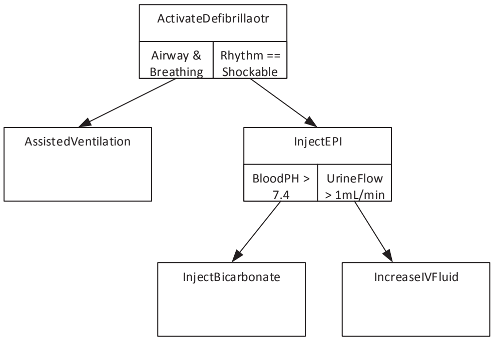


Figure 1: Treatments and preconditions tree

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 3.1. 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.

Example 2: One potential side effect of sodium bicarbonate is suppressed respiratory drive, 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

[1] Some medical staff prefer to perform treatments in the sequence of **A**irway, **B**reathing, and **C**irculation by assessing airway and breathing before treating cardiac arrest. For more details see http://www.patient.co.uk/doctor/adult-cardiopulmonary-arrest

[2] The shockable rhythms are ventricular fibrillation and pulseless ventricular tachycardia [7].

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

[4] Severe acidosis, which is an increased acidity in the blood and other body tissue, will significantly reduce the effectiveness of epinephrine [22]

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

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 [23] and airway laser surgery [12]. The purpose of this work is to develop mechanisms and protocols to dynamically adapt to the non-deterministic patient conditions and side effects. The proposed protocol structures preconditions and corrective treatments and provides feedback to the medical staff. In collaboration with medical staff, our protocol enforces the correct execution sequence of treatments. From a system verification perspective, we use a model checking tool, UPPAAL [3], to formally verify the correctness of the proposed protocol. The contributions of this paper can be summarized as follows:

- We propose a coordination protocol to structure preconditions and treatment 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 formally verify the safety and correctness properties of the proposed protocol.

The rest of paper is organized as follows. In Section 2, we describe the technical challenges and the targeted system model. The details of the proposed coordination protocol is described in Section 3. In Section 4, we prove the correctness of the proposed protocol and discuss the limitations and potential solutions. We show the model checking results in Section 5. Related work is discussed in Section 6.

2 System Overview

In this section, we first describe the technical challenges of developing a treatment coordination protocol. We then present the system models and formal definitions of treatments and physiological conditions used throughout the paper.

2.1 Treatment Coordination Requirements and Technical Challenges

From the Example 1 and Example 2 described in the previous section, we understand that the coordination of the treatments is essential for patient safety. In this work, we develop a coordination protocol to address the following three essential aspects.

- **Precondition**: A treatment can be performed only if the corresponding preconditions are satisfied. Unlike traditional cyber system preconditions, the medical system cannot lock or freeze 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 cascade of preconditions and treatments. Therefore, the system must structure preconditions and treatment to help medical staff keep track of preconditions and performed corrective 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 effect and alert medical staff to adjust the treatments. In addition, the system should dynamically change the structure of preconditions and treatments to reflect the adjustments from the 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.

In summary, followings are the major design challenges.

- The effectiveness of the treatments are non-deterministic, and the preconditions and corrective treatments may cause a cascade effect, as explained in the previous section. If medical staff lose track of any precondition or treatment, patient safety may be compromised.
- Potential side effects are dynamic and may interfere with other treatments or invalidate the preconditions. Therefore, medical staff must be informed to dynamically adjust the treatments.

2.2 System Models and Assumptions

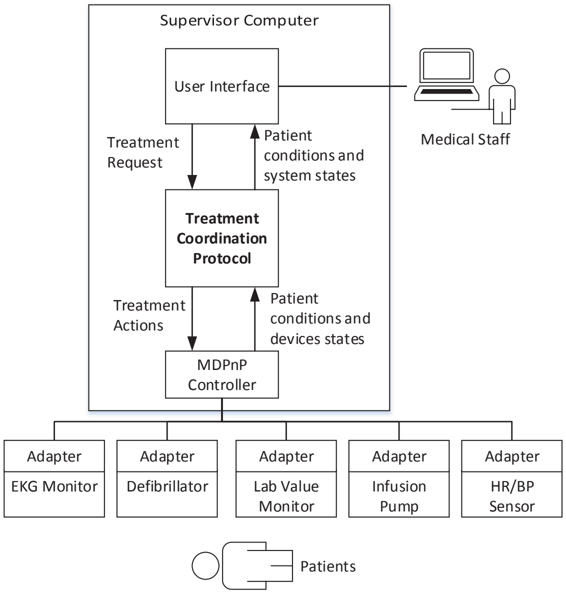


Figure 2: The System Architecture

In this section, we describe the system models and assumptions that are used throughout the paper.

Figure 2 shows the system architecture. We envision that Medical Device Plug and Play (MDPnP) will provide a centralized supervisory framework for integrating networked medical devices, such as EKG monitor and defibrillator. An adapter is attached to each medical device to provide network communication capability. In this paper, we take advantage of such an integrated framework and propose a *Treatment coordination protocol* on top of MDPnP. Note that the user interface, coordination protocol, and MDPnP boxes in the figure represent processes or threads, which can be physically placed in the same machine or distributed in different machines. In this work, we assume the three components are placed in a supervisor computer. Medical staff issue a treatment request through user interface. The proposed *Treatment coordination protocol* manages the correct execution sequence by validating the preconditions, monitoring potential side effects, and checking expected responses. If the execution sequence is correct and 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 from MDPnP controller, the proposed protocol dynamically checks the side effects and requests the medical staff to adjust the treatments. The details of the protocol will be presented in Section 3.

Now, let us formally define physiological conditions and treatments that this work focus on.

Let *RV* be the reference value of a monitored physiological measurement, which can be threshold, trend, or pattern.

Definition 1. ***PhysiologicalCondition (PC)*** *is defined as a tuple $<$ Checker, PM, Operator, RV $>$, where*
Checker is the entity capable of checking the condition[6]*, Checker $\in$ {System, MedicalStaff},*
PM (physiological measurement) $\in$ {EKGRhyhtm, HeartRate, BloodPressure...},
Operator $\in \{>, <, =, \leqslant, \geqslant\}$,

For example, the physiological condition of shockable rhythm is defined as <MedicalStaff, EKGRhythm, =, ShockableRhythm >.

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 $\subset$

[6] 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.

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[7].

For example, the treatment of injecting epinephrine is defined as <IVPump, InjectEpinephrine, $\{$<System, BloodPH, $\geqslant$, 7.4 $>$, $<$ System, UrineFlow, $\geqslant$, 1 mL/min $>\}$, $\{<$ System, BloodPressure, $\geqslant$, 210 $>\}$[8], <MedicalStaff, EKGRhythm, =, ShockableRhythm $>$, 4 min $>$

The developed system makes the following assumptions:

- Medical devices, controllers and the communication never fail. Fault tolerance is an important challenge that will be addressed in a separated paper.
- Medical devices periodically send the devices status, including the physiological measurement and device settings, to the treatment coordination protocol through MDPnP controller.
- The specifications of treatments coordination, which consist of preconditions, potential side effect, and expected responses, are prescribed by medical staff.
- Medical staff specifies the corrective treatments for the preconditions in order to improve patient conditions.

3 Treatment Coordination Protocol

A *Treatment Precondition and Correction (TPC)* tree is proposed in Section 3.1 to capture the standard medical practice through structuring treatments and preconditions. Then, we further develop a coordination protocol to dynamically adapt the TPC tree to the patient conditions and side effects in the collaboration with the medical staff.

3.1 Treatment Precondition and Correction Tree Structure

We propose a *Treatment Precondition and Correction (TPC)* tree for structuring the preconditions and treatments. 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 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

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

[8] Epinephrine may adversely cause elevation in blood pressure in up to 55%. http://www.drugs.com/sfx/epinephrine-side-effects.html

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 dynamically updating the TPC tree will be detailed in the next section. Like fault-tree [19], the proposed TPC tree aims to capture the cause and effect relations and analyze whether the root node can be reached or not. Unlike fault-tree, TPC tree intends to reach the root node by adding corrective treatments and must adapt to the dynamics of the medical environment.

Moreover, TPC tree also serves as an interface between the medical staff and the system. Since the TPC tree maintains a logical relations between preconditions and corresponding corrective treatments, medical staff can keep track of the progress of the medical procedures. In addition, the system provides feedback through the tree to the medical staff if the side effects of a treatment start to interfere other treatments or invalidate the previously satisfied preconditions. On the other hand, the medical staff specify the corrective treatments for the unsatisfied preconditions based on the tree structure.

Let us use the following scenario to illustrate how coordination 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 require 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 1 mL/min. 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[9], 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 request IV pump to inject sodium bicarbonate, as illustrated in step 6 and 7 of Figure 3(a).

[9] Sodium bicarbonate and calcium chloride cannot be used together, because they would form calcium carbonate and make the two drugs ineffective.

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 request the medical staff to specify a corrective treatment. Suppose the medical staff specifies providing assisted ventilation. The system adds a treatment node to the TPC tree*AssistedVentilation*, as illustrated in step 8 of Figure 3(b).
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 the medical staff to specify an alternative corrective treatment. Note that 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.

3.2 Description of the Protocol

In this section, we describe the coordination protocol in detail and show part of the pseudocode in Algorithm 3.1 and Algorithm 3.2. Algorithm 3.1 shows the construction of the TPC tree. Algorithm 3.2 shows the post-order execution and dynamic side effect monitoring. The proposed 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 examines the preconditions of the received treatment, which is the root of the tree. If any precondition is not satisfied or must be checked by medical staff, the system sends the tree to the medical staff and requests them to check the preconditions and specify corrective treatments, as shown in the line 7–23 of Algorithm 3.1. 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 26–30 of Algorithm 3.1. 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 is not further preconditions to check or all the preconditions are satisfied, the treatments in the TPC tree are ready to be performed. The system sends the TPC tree 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 is dynamically changes, the system checks the preconditions again before performing the

treatment. If the preconditions are satisfied, the system inserts the treatment into the executing list and request the corresponding medical devices to perform the treatment. The system needs to check the expected response after the a time interval, specified by the medical staff,[10], so the system setups a timer and checks the expected responses when the timer fires, as shown in the line 11–12 of Algorithm 3.2. The details of checking expected responses will be explained in the next phase. In addition, the system periodically monitors the potential side effects of the treatments in the the executing list. If the potential side effects must be checked by the medical staff, the system periodically requests the medical staff to check them. 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 measurement 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 27–29 of Algorithm 3.2. Since the tree provides a logical path of the reasons for preforming the treatments, medical staff can adjust the existing treatments, such as increase or decrease the drug dosage, or specify alternative treatments. The system then updates the tree according to the new treatments, as described in the previous phase.

After the system informs the side effects to the medical and updates the TPC tree with their approval, the system restarts the post order execution, as shown in the line 33–37 of Algorithm 3.2[11].

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, which means the corrective treatments successfully improve the patient conditions to satisfy the precondition, 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 and sends it to the medical staff. The medical staff can specify an alternative corrective treatment, and the system updates the TPC tree accordingly and restart 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. When the root treatment node is

[10] The timer interval depends on the treatment, patient conditions, and medical staff's judgment.

[11] Side effects may adversely affect patient's vital signs such as oxygen saturation level and heart rate, as well. The handling of those patient adverse event has been addressed in our previous work [29].

reached, the corresponding preconditions are satisfied and the treatment is safe to be performed.

4 Correctness of the Protocol and Discussion

In this section, we first prove the correctness of the protocol. We then discuss the limitations and potential solutions.

4.1 Correctness of the Protocol

Theorem 1. *Under the proposed protocol, a treatment is performed only if all the 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 10–11 of Algorithm 3.2, the precondition is checked again before the treatment is performed. Therefore, 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.
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 3.1, of the proposed protocol builds a well-formed TPC tree or an exception is raised.*

Proof. Proof by contradiction. Assume Algorithm 3.1 build a non-well-formed tree, and no exception is generated. 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 27–29. We reach a contradiction. Therefore, the protocol guarantees either the tree is well-formed or an exception is raised. □

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 26–30 of Algorithm 3.2, 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 would be well-formed. □

The above theorems prove that our coordination protocol enforces a correct sequence of performing treatments, including validating preconditions, monitoring side effects and checking expected responses. In addition, the protocol is capable of adapting the TPC tree to the dynamics of patient conditions and non-determinism of the treatments with medical staff's approval.

4.2 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. A treatment will be performed only if the expected response of the previous treatment is satisfied. 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 in many acute medical and trauma situations[12]. In addition, concurrent treatments make the medical staff hard to reasoning the effectiveness of treatments and side effects of treatments. However, the system allows the medical staff to specify compatible treatments, such as the drugs do not have adverse interference. The system perform the compatible treatments concurrently as long as their preconditions are satisfied. We are closely working with medical staff to collect the use cases compatible treatments to improve the flexibility without compromising safety of the system.

[12] Other variations, such as CAB and ABCDE, are proposed for different medical scenario [16].

Second, our protocol does not consider potential failure of medical devices and communication channels. We assume that each medical device has been approved Food and Drug Administration (FDA) and works as expected. In practice, some approved devices still had defects and were recalled. This is a serious challenge to be addressed in the future. Therefore, the failure of communication channels is one of the major issues in supervisory medical systems. When the communication channels fail, the proposed coordination protocol loses the capability of dynamically monitoring the patient conditions and checking preconditions or side effects. We address this safety issue in our separate papers [12, 11]. The fundamental concepts is that the supervisor computer, at run-time, generates contingency plans to direct each medical device to a fail-safe state in case of communication fails. The proposed coordination protocol can cooperate with these fail-safe mechanisms, but the detail design is out of the scope of this paper.

Third, 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. However, we would not recommend which treatment is the best to this patient's specific conditions. Treatment recommendations are outside the scope of this paper, because many drugs have potentially severe side effects, and complex trade off between medical decisions is involved.

5 Verification

In this section, we first describe a simplified 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.

5.1 Resuscitation Case Study

Cardiac arrest is the sudden loss of heart function, breathing and consciousness and can lead to death within minutes. The treatment is immediate defibrillation if a ”shockable” rhythm is present. Cardiopulmonary resuscitation (CPR) and medications, such as epinephrine and vasopressin, are used to provide circulatory support and to induce a ”shockable” rhythm. American Heart Association (AHA) provided resuscitation guidelines for the urgent treatment of cardiac arrest [7].

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 give drugs.

2. Check heart rhythm: the medical staff check the heart rhythm from the EKG monitor. The rhythm can be shockable rhythm, which is ventricular fibrillation or ventricular tachycardia, or non-shockable rhythm, which is asystole or pulseless electrical activity (PEA). 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, the medical staff should activate a defibrillator to deliver electrical energy to the heart in order to regulate the heart rhythm. If the patient's heart rhythm is still abnormal, the medical staff should perform CPR again (loop back to step 1).

Furthermore, the side effects of the treatments may cause adverse interactions. For example, sodium bicarbonate may adversely affect patient breathing, and epinephrine may adversely raise patient's blood pressure. Therefore, the medical staff must closely monitor the patient's status and preform alternative treatments if the side effects occur.

5.2 UPPAAL Model and Verification

We model the resuscitation scenario in UPPAAL; the system consists of following models: user interface, coordination protocol, side effect monitor, EKG monitor, defibrillator, IV Pump, blood PH monitor, and urine flow rate sensor. The models communicate using synchronization channels and shared variables.The user interface model follows the three-step resuscitation procedure and contains a list of pre-defined treatments, such as activating defibrillator and injecting epinephrine, as described in the previous section. The medical devices sends the patient's physiological measurements, which are modeled as non-deterministic transitions, to the coordination 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 show two simplified UPPAAL models of the treatment coordination protocol in Figure 4 to illustrate the modeled behavior. The model shown in Figure 4(a), which reflects Algorithm 3.1, builds the TPC tree based on the user interface input 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. Before performing a treatment, the model re-checks the preconditions of the treatment. If the preconditions are satisfied, the model adds the treatment to the executing list and sends action command 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. If the expected responses are satisfied, the corresponding tree node is removed from the TPC tree, and the model performs the next treatment in the tree. Otherwise, the model sets the corresponding tree node to be *ineffective* and sends the tree to the medical staff.

Figure 4(b) shows the design of side effects monitoring reflecting the Algorithm 3.2. The model checks each treatment node in the execution list against the preconditions in the TPC tree and other ongoing treatments in the execution list. If any precondition is violated or treatment is interfered, the model sets status of the corresponding tree node using *highlightTreeNode* function. The model then goes to the *UpdateTree* state and updates the tree nodes with corrective treatments received from the user interface, as described before. When the model finishes updating the TPC tree, it sends a *SideEffectAlert* because the protocol needs to restart post order execution.

We show part of the verified safety and correctness properties in Table 1. 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. For example, safety property *P2* can be checked by the UPPAAL temporal logic formula: *A[] CoordinationProtocol.IVPumpEPI imply*
LabBloodPH.Value > *7.4 && UrineSensor.Vaule* > *1* . The above formula verifies that for all reachable states, the *CoordinationProtocol* in the *IVPumpEPI* state implies the *Value* of *LabBloodPH* is larger than 7.4 and the *Value* of *UrineSensor* is larger than 1. On the other hand, the protocol properties guarantee the correctness of the proposed protocol. For instance, property *P10* can be checked by the UPPAAL 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 4. 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. According to the model checking results, we demonstrate that the proposed treatment coordination protocol is correctly designed to guarantee safety and correctness properties.

6 Related Work

Medical Device Plug-and-Play (MDPnP) provides flexibility and interoperability for medical systems [9], 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, 17, 13]. For example, Kramer *et al.* used a closed-loop approach to control IV infusion rate for treatment of hypovolemia [17]; Arney *et al.* proposed a closed-loop methodology for patient-controlled analgesia (PCA). In our previous work [29], we proposed a closed-loop architectural pattern to coordinate distributed medical devices and guarantee consistency control. In this paper, we adopt the closed-loop approach and focus on the treatment coordination issue.

Formal verification and validation are important issues to guarantee safety and correctness of the system. Formal method is widely used to specify and verify medical systems [1, 21, 10]. Pajic *et al.* combined simulation-based analysis and model checking to guarantee safety properties of closed-loop medical systems [21]. Jiang *et al.* developed a model for dual-chamber pacemaker and verified the safety properties using model checking techniques [10]. In this work, we use resuscitation as a case study and model the proposed protocol in UPPAAL [3] to formally verify the correctness and safety properties.

Table 1: 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 1 mL/min.
	P3: Sodium bicarbonate is injected only if calcium chloride is not currently being injected.
	P4: Calcium chloride is injected only if sodium bicarbonate is not currently being injected.
	P5: If the side effect of sodium bicarbonate adversely affects the breathing, the tree is updated with a new treatment node for assisted ventilation.
	P6: If epinephrine does not make patient's heart rhythm become shockable, the tree is updated with an alternative treatment node for drug *vasopressin*.
Protocol properties	P7: There is no deadlock in the system
	P8: A treatment is performed only if all its preconditions are satisfied.
	P9: If side effect does not occur, the root node of the TPC tree is added to the executing list
	P10: If side effects invalidate a precondition, the TPC tree is updated and well-formed.

System modeling and formal specification is an important issue for developing and analyzing medical systems [26, 14, 24]. King *et al.* [14] proposed a formal specification language to express and reason safety properties of on-demand medical systems. Rahmaniheris *et al.* [24] described an abstract representation of dynamic clinical environment. A modular architecture is also introduced to support safe system reconfiguration. In this work, we assume the physiological models specified by the medical staff are correct, and the proposed protocol ensures coordination of the treatments according to the model specifications.

Concurrency control has be extensively investigated to address coordination issue in distributed systems [4]. Some locking mechanisms, such as two-phase locking and tree-locking, are developed to guarantee a correct execution sequence [25, 27]. However, in cyber-physical-human environments, the system cannot lock or freeze the states of physical components, such as patient conditions. In addition, if the preconditions are invalidated due to side effects, the system cannot perform backward recovery like traditional database systems, [20]. In this work, we address these issues by proposing a TPC tree to structure preconditions and corrective treatment. The developed coordination protocol updates the TPC tree in the collaboration with medical staff to prevent medical errors.

7 Conclusion and Future Work

In this paper, we develop a treatment coordination 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 medical staff to keep track of the medical procedure. In collaboration with medical staff, the proposed protocol adapts the the TPC tree to the dynamics of patient conditions and non-deterministic behavior of treatments. Therefore, the medical errors due to uncoordinated treatments can be reduced. 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 coordination 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. In addition, human computer interaction (HCI) and situation awareness [28] is an important aspect for developing supervisory medical systems with human in the loop. We believe the proposed TPC tree can help medical staff keep track of the medical procedures and decide the treatments that best fit patient's conditions, thereby improving the situation awareness among the medical staff. We plan to implement a resuscitation assistant system and evaluate medical staff's mental workload using a trace-driven simulation.

References

[1] R. Alur, D. Arney, E. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. Van Albert, and J. Zhou. Formal specifications and analysis of the computer-assisted resuscitation algorithm (cara) infusion pump control system. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(4):308–319, 2004.

[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] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in database systems*, volume 370. Addison-wesley New York, 1987.

[5] D. J. Cullen, B. J. Sweitzer, D. W. Bates, E. Burdick, A. Edmondson, and L. L. Leape. Preventable adverse drug events in hospitalized patients: a comparative study of intensive care and general care units. *Critical care medicine*, 25(8):1289–1297, 1997.

[6] N. P. Day, N. H. Phu, N. T. H. Mai, D. B. Bethell, T. T. H. Chau, P. P. Loc, L. Van Chuong, D. X. Sinh, T. Solomon, G. Haywood, et al. Effects of dopamine and epinephrine infusions on renal hemodynamics in severe malaria and severe sepsis. *Critical care medicine*, 28(5):1353–1362, 2000.

[7] 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.

[8] 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.

[9] 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.

[10] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and verification of a dual chamber implantable pacemaker. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 188–203. Springer, 2012.

[11] 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.

[12] C. Kim, M. Sun, S. Mohan, H. Yun, L. Sha, and T. Abdelzaher. A framework for the safe interoperability of medical devices in the presence of network failures. In *Proceedings of the 1st ACM/IEEE ICCPS*, pages 149–158. ACM, 2010.

[13] A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter. Prototyping closed loop physiologic control with the medical device coordination framework. In *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*, pages 1–11. ACM, 2010.

[14] 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.

[15] 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.

[16] 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.

[17] G. C. Kramer, M. P. Kinsky, D. S. Prough, J. Salinas, J. L. Sondeen, M. L. Hazel-Scerbo, and C. E. Mitchell. Closed-loop control of fluid therapy for treatment of hypovolemia. *The Journal of Trauma and Acute Care Surgery*, 64(4):S333–S341, 2008.

[18] 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.

[19] W.-S. Lee, D. Grosh, F. A. Tillman, and C. H. Lie. Fault tree analysis, methods, and applications ? a review. *Reliability, IEEE Transactions on*, 34(3):194–203, 1985.

[20] Z. Luo, A. Sheth, K. Kochut, and J. Miller. Exception handling in workflow systems. *Applied Intelligence*, 13(2):125–147, 2000.

[21] 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.

[22] A. Papastylianou and S. Mentzelopoulos. Current pharmacological advances in the treatment of cardiac arrest. *Emergency medicine international*, 2012, 2011.

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

[24] M. Rahmaniheris, W. Kang, L.-J. Lee, L. Sha, R. B. Berlin, and J. M. Goldman. Modeling and architecture design of an mdpnp acute care monitoring system. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pages 514–515, 2013.

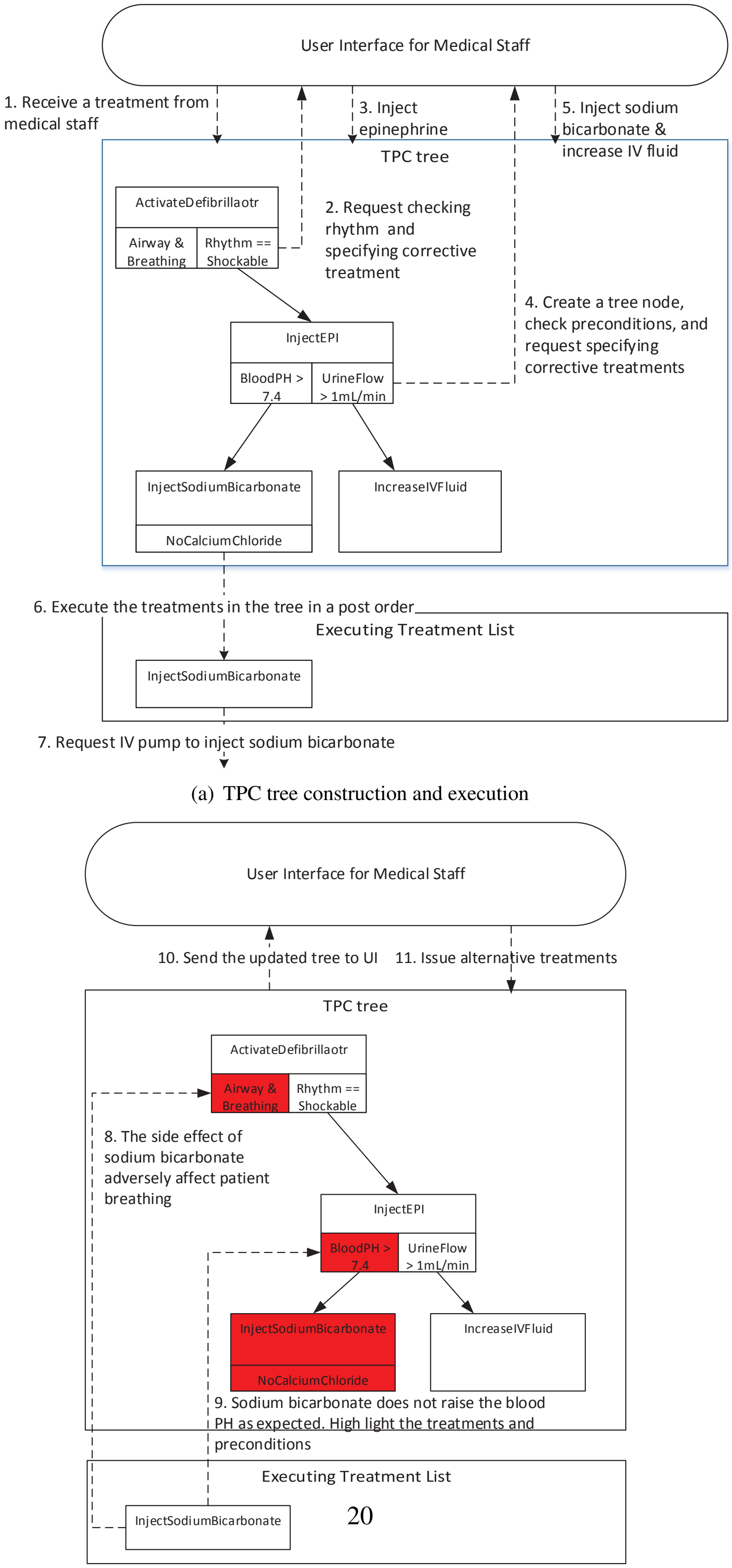
[25] K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Transactions on Computer Systems*, 7(1):61–77, 1989.

[26] V. C. Rideout. *Mathematical and computer modeling of physiological systems*. Prentice Hall Englewood Cliffs, NJ:, 1991.

[27] F. B. Schneider. Synchronization in distributed programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(2):125–148, 1982.

[28] M. Wright, J. Taekman, and M. Endsley. Objective measures of situation awareness in a simulated medical environment. *Quality and Safety in Health Care*, 13(suppl 1):i65–i71, 2004.

[29] P.-L. Wu, W. Kang, A. Al-Nayeem, L. Sha, R. B. Berlin Jr, and J. M. Goldman. A low complexity coordination architecture for networked supervisory medical systems. In *Proceedings of the 4th ACM/IEEE ICCPS*, pages 89–98. ACM, 2013.


User Interface for Medical Staff
1. Receive a treatment from medical staff
3. Inject epinephrine
5. Inject sodium bicarbonate & increase IV fluid
TPC tree
ActivateDefibrillaotr
Airway & Breathing
Rhythm == Shockable
2. Request checking rhythm and specifying corrective treatment
InjectEPI
BloodPH > 7.4
UrineFlow > 1mL/min
4. Create a tree node, check preconditions, and request specifying corrective treatments
InjectSodiumBicarbonate
NoCalciumChloride
IncreaseIVFluid
6. Execute the treatments in the tree in a post order
Executing Treatment List
InjectSodiumBicarbonate
7. Request IV pump to inject sodium bicarbonate
User Interface for Medical Staff
10. Send the updated tree to UI
11. Issue alternative treatments
TPC tree
ActivateDefibrillaotr
Airway & Breathing
Rhythm == Shockable
8. The side effect of sodium bicarbonate adversely affect patient breathing
InjectEPI
BloodPH > 7.4
UrineFlow > 1mL/min
InjectSodiumBicarbonate
NoCalciumChloride
IncreaseIVFluid
9. Sodium bicarbonate does not raise the blood PH as expected. High light the treatments and preconditions
Executing Treatment List
InjectSodiumBicarbonate

(a) TPC tree construction and execution

(b) Side effect monitoring and expected responses checking

Figure 3: The coordination protocol

Algorithm 3.1 Pseudo code for constructing TPC tree

```
checkPrecondAndBuildTree(TPCTreeNode root){
  PrecondtionSet precondSet←root.treatment.PS
  TreatmentSet correctiveTreatmentSet;
  Precondition precond;

  while(precondSet ≠ ∅){
    for each precond ∈ precondSet{
      if(precond.Checker is medical devices){
        if(precond is satisfied)
          precond.status←SATISFIED;
        else
          precond.status←UNSATISFIED;
      } else if(precond.Checker is medical staff)
        precond.status←UNKNOWN;

    }

    if(all preconditions are SATISFIED)
      break;

    // request medical staff to check non−machine−checkable preconditions and specify the corrective treatments
    sendTreeToUI(root);
    correctiveTreatmentSet←receiveFromUI();
    ...

    // check if all the unsatisfied preconditions have corrective treatments
    for each precond ∈ precondSet{
      if(precond.status == UNKNONW || (precond.status == UNSATISFIED && no corrective treatments))
        sendExceptionToUI(INCOMPLETE);
        ...
    }
    Treatment treatment;
    // create child nodes for the corrective treatment
    for each treatment ∈ correctiveTreatmentSet{
      TPCTreeNode childNode ← new TPCTreeNode(treatment);
      if(childNode is already in the tree)
        sendExceptionToUI(DUPLICATE);
      // insert the childNode to the tree
      ...
      // update the precondition set
      precondSet←∪treatment.PS
    }

  }
}
```

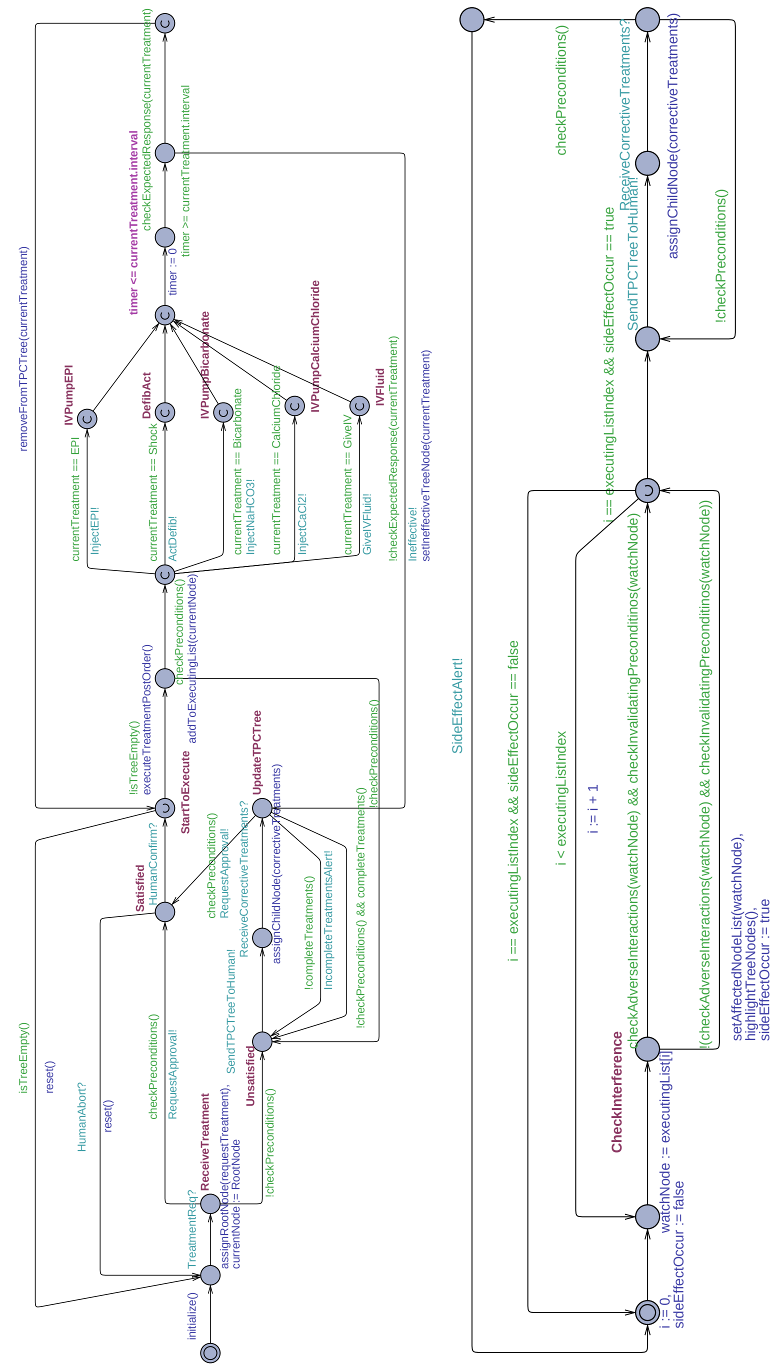
Algorithm 3.2 Pseudo code for post order execution and side effect monitoring

```
postOrderExecution(TPCTreeNode node)
{
  if(node == null)
    return;
  TPCTreeNode childNode;
  for each childNode $\in$ node.childNodeList{
    postOrderExecution(childNode);
  }
  // check precondition again because patient conditions dynamically change
  if(all preconditions are satisfied){
    performingTreatment(node);
    setUpTimerForExpectedResponse(node);
    ...
  } else{
    sendExceptionToUI(IneffectiveCorrectiveTreatment);
    ...
  }
}

// Periodically monitors the side effects of the treatments in the executing list
runTimeMonitoring(TPCTreeNode root)
{
  Treatment treatment;
  TPCTreeNodeSet affectedNodeSet$\leftarrow\emptyset$;
  // Monitoring side effect
  for each treatment in the executingList{
    // Interfere with other treatments
    if(treatment.sideEffects adversely affect other treatments || treatment.sideEffects invalidate the preconditions in the OTBC tree){
      affectedNodeSet$\leftarrow$affectedNodeSet$\cup$getAffectedNodeSet(treatment);
  }

  // update tree and request for new treatments
  checkPrecondAndBuildTree(root);
  sendToUI(root);

  if(medical staff approves the tree){
    postOrderExeuction(root);
  } else{
    // abort the treatment
    ...
  }
}
```


initialize()
TreatmentReq?
assignRootNode(requestTreatment),
currentNode := RootNode
ReceiveTreatment
checkPreconditions()
RequestApproval!
SendTPCTreeToHuman!
!checkPreconditions()
Unsatisfied
isTreeEmpty()
reset()
HumanAbort?
reset()
Satisfied
HumanConfirm?
StartToExecute
!isTreeEmpty()
executeTreatmentPostOrder()
checkPreconditions()
addToExecutingList(currentNode)
UpdateTPCTree
assignChildNode(correctiveTreatments)
ReceiveCorrectiveTreatments?
checkPreconditions()
RequestApproval!
!completeTreatments()
IncompleteTreatmentsAlert!
!checkPreconditions() && completeTreatments()
!checkPreconditions()
checkPreconditions()
currentTreatment == EPI
InjectEPI!
IVPumpEPI
currentTreatment == Shock
ActDefib!
DefibAct
currentTreatment == Bicarbonate
InjectNaHCO3!
IVPumpBicarbonate
currentTreatment == CalciumChloride
InjectCaCl2!
IVPumpCalciumChloride
currentTreatment == GiveIV
GiveIVFluid!
IVFluid
!checkExpectedResponse(currentTreatment)
Ineffective!
setIneffectiveTreeNode(currentTreatment)
timer := 0
timer <= currentTreatment.interval
timer >= currentTreatment.interval
checkExpectedResponse(currentTreatment)
removeFromTPCTree(currentTreatment)
i := 0,
sideEffectOccur := false
watchNode := executingList[i]
CheckInterference
checkAdverseInteractions(watchNode) && checkInvalidatingPreconditinos(watchNode)
!(checkAdverseInteractions(watchNode) && checkInvalidatingPreconditinos(watchNode))
setAffectedNodeList(watchNode),
highlightTreeNodes(),
sideEffectOccur := true
i < executingListIndex
i := i + 1
i == executingListIndex && sideEffectOccur == false
i == executingListIndex && sideEffectOccur == true
SendTPCTreeToHuman!
ReceiveCorrectiveTreatments?
assignChildNode(correctiveTreatments)
!checkPreconditions()
checkPreconditions()
SideEffectAlert!

(a) TPC tree construction and expected responses checking

(b) Run-time side effect monitoring

Figure 4: The coordination protocol