Model and Integrate Medical Resource Availability into Verifiably Correct Executable Medical Guidelines - Technical Report

Chunhui Guo, Zhicheng Fu, Zhenyu Zhang, Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616, USA
Email: {cguo13, zfu11, zzhang111}@hawk.iit.edu, ren@iit.edu

Lui Sha
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
Email: lrs@illinois.edu

***Abstract*—Improving effectiveness and safety of patient care is an ultimate objective for medical cyber-physical systems. A recent study shows that the patients' death rate can be reduced by computerizing medical guidelines [19]. Most existing medical guideline models are validated and/or verified based on the assumption that all necessary medical resources needed for a patient care are always available. However, the reality is that some medical resources, such as special medical equipment or medical specialists, can be temporarily unavailable for an individual patient. In such cases, safety properties validated and/or verified in existing medical guideline models without considering medical resource availability may not hold any more.**

The paper argues that considering medical resource availability is essential in building verifiably correct executable medical guidelines. We present an approach to explicitly and separately model medical resource availability and automatically integrate resource availability models into an existing statechart-based computerized medical guideline model. This approach requires minimal change in existing medical guideline models to take into consideration of medical resource availability in validating and verifying medical guideline models. A simplified stroke scenario is used as a case study to investigate the effectiveness and validity of our approach.

I. Introduction and Related Work

Medical guidelines play an important role in today's medical care. Over past two decades, significant amount of efforts have also been made in obtaining various computer-interpretable models and developing tools for the management of medical guidelines, such as Asbru [4], GLIF [21], GLARE [23], EON [24], and PROforma [8]. Along with the well development and use of formal techniques on system design [17], [16], [26], our previous work [10] also designed a platform to model medical guidelines with statecharts and automatically transform statecharts [11] to timed automata [1] for formal verification. Furthermore, runtime verification is proposed and well adapted to working directly on the medical guidance systems [15], [14], [9] to improve the system performance. All these work is based on medical guidelines presented in medical handbooks.

However, medical guidelines often focus on medical procedures and with implicit assumption that all required medical resources for treatments are always available. By medical resources, we mean medical professionals, supplies, and equipments[1]. Most existing computer-based medical guideline models inherit the implicit assumption and are validated and/or verified based on that all required medical resources are constantly available. Unfortunately, the reality is that some medical resources, such as special medical equipments or medical specialists, can be temporarily unavailable for patients. In such cases, some processes of medical guideline models may be blocked and safety properties validated and/or verified may fail and put patients in danger. We use a simplified stroke scenario to illustrate the cases as follows. For illustration purpose, we ignore some medical details from computer science perspective in the simplified stroke scenario.

Stroke Scenario: *An ischemic stroke occurs when a clot or a mass blocks a blood vessel, cutting off blood flow to a part of the brain and results in a corresponding loss of neurologic function [2]. The goal of acute treatment is to keep the amount of brain injury as small as possible. The only FDA approved drug to treat ischemic stroke is tissue plasminogen activator (tPA), a clot busting drug [2]. The intravenous (IV) tissue plasminogen activator (tPA) injection is a standard treatment for ischemic stroke patients and it is most effective during the initial 3-hour window from the onset of stroke symptoms [3]. The treatment window can be extended from 3 to 4.5 hours for certain patients, but the risks are increased [13]. Some patients can be treated by dripping tPA directly on the clot through a intra-arterial (IA) micro-catheter within 6 hours from the onset of stroke symptoms [22]. However, the IA tPA treatment requires specialists to control tPA dose, special equipments to put the micro-catheter into blood vessels, and technicians to operate the special equipment.*

In addition, in order to use the tPA treatment, we must ensure that (1) CT scan does not show hemorrhage, and (2) the patient's blood pressure is under control. To derive the conclusion that the patient does not have hemorrhage, we would need medical resources including a CT machine, a CT technician, and a radiologist. If a patient's blood pressure is not within the range for tPA administration, a specialist is required to control blood pressure.

In the simplified stroke scenario, there are three medical properties needed to be guaranteed in the patient care:

- **P1**: the tPA is injected only if a CT scan shows no hemorrhage and systolic and diastolic blood pressure are smaller than or equal to 185 mm Hg and 110 mm Hg;
- **P2**: the IV tPA administration is completed within 3 hours from onset of symptoms;
- **P2**: the IA tPA administration is completed within 6 hours from onset of symptoms.

Assume a stroke patient's onset time is 0 and a physician orders CT scan for the patient at time 20 (minutes). If the CT machine is always available, the tPA administration can be completed within the 3-hour window. However, if the CT machine is unavailable until 200 minutes. In such case, the tPA administration can not be completed within 3 hours due to temporarily unavailable CT machines. Hence, modeling

[1]Patients are not considered as medical resources. They can be treated as preconditions of treatments and can be validated with the protocol presented in [25].

medical resources in existing medical guideline models and validating and verifying safety properties with consideration of medical resource availability are essential for improving patient care safety.

One approach to address the medical resource availability issue in existing medical guideline models is to directly add medical resource availability as guards to corresponding transitions or as state constraints. We call this method as direct modification approach. The timed and resource-oriented statecharts [18] takes the direct modification approach by specifying required resource information in states. Christov *et al.* [6], [7] uses Little-JIL to model the processes in medical guidelines and represents resource as preconditions of process steps. The mentioned work has shown that adding medical resource availability as transition guards, state constraints, or process preconditions is a practicable approach to address medical resource temporal unavailable issue in medical guideline models. But these approaches also face the following challenges. First often times, a medical guideline represents a generalized treatment procedure for a disease, it is not defined for a specific hospital. As medical resource availabilities at different medical facilities can be significantly different, to use such direct modification approaches, we would have to build different medical guideline models for different medical facilities. Second, even within a same medical facility, medical resource availability can change over time, therefore the corresponding medical guideline models need to be changed as well. Third, for a failed safety property, identifying the errors that cause the failure becomes more challenging as errors both in medical resource availabilities and medical guideline model itself could cause the safety property to fail. Forth, medical guideline models with medical resource built in would increase the difficulty for medical professionals to understand and clinically validate the models, and would unnecessarily require medical staffs to know the medical resource availability at medical facilities.

In this paper, we present an approach to model and integrate medical resource availability into executable medical guideline models. Our approach separates resource models from medical guideline models to minimize the change impact of both guidelines and resources, as well as leaving the syntax and semantics of medical guideline models unchanged. In particular, we first define the procedures that how physicians to annotate required resources for actions in medical guideline models. To explicitly take medical resource availability into medical guideline system design, we represent an approach to explicitly and separately model medical resource availability. The medical resource availability models are then integrated into medical guideline models so that the integrated medical guideline models can be validated and safety properties in the presence of temporarily unavailable resources can be formally verified. A simplified stroke scenario is used as a case study to explain the proposed approach. The main contributions of the paper are:

- Take medical resource availability into consideration in validating and verifying executable medical guideline models.
- Present an approach to explicitly and separately model medical resource availability with statecharts.
- Develop an approach to automatically integrate resource availability models with verifiably correct executable medical guideline models.

The rest of the paper is organized as following: we introduce a framework for building verifiably correct executable medical guideline models in Section II. Section III describes the approaches for explicitly and separately modeling medical recourses and their variabilities. Section IV defines the procedure for integrating medical resource availability models into medical guideline models. A simplified stroke case study is given in Section V to illustrate the effectiveness of the presented approach. We draw conclusions and point out future work in Section VI.

II. Verifiably Correct Executable Medical Best Practice Guidelines

Our previous work [10] designed a platform to build verifiably correct executable medical guidelines. The high level abstract of the platform is depicted in Fig. 1. In particular, we use statecharts [11] to model medical guidelines and interact with medical professionals to validate the correctness of the medical guideline models. The statecharts built with Yakindu tool [12] are then automatically transformed to timed automata [1] by the developed Y2U[2] tool, so that the safety properties required by the model, UPPAAL timed automata [5] in particular, can be formally verified. We use the simplified stroke scenario presented in Section I as an example to illustrate our previous approaches on how to build verifiably correct executable medical guidelines.

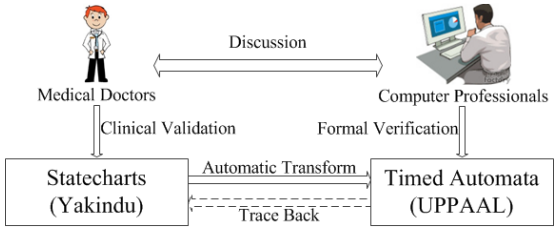

Fig. 1. A Platform for Building Verifiably Correct Executable Medical Guidelines

We use Yakindu statecharts to model the stroke treatment guideline [13]. For illustration and easy understanding purpose, we show a simplified stroke statechart model in Fig. 2, which only focuses on the CT scan and IV tPA administration procedures and omits details of other medical procedures. The full version of stroke statechart model is available in the case study (Section V). Hypertension is present in up to 84% of patients presenting with acute stroke [20]. In the simplified statechart shown in Fig. 2, we assume that upon patient arrival, treatments to control blood pressure have been immediately performed. A patients blood pressure is either quickly brought within the range or not possible.

In the statechart shown in Fig. 2, two medical actions `CTscan` and `givetPA` are modeled by Yakindu statechart *events*. In Yakindu statecharts, *events* can be raised by both states and transitions. For instance, the *entry action* of state "CT" (`entry/ raise CTscan`) raises *event* `CTscan` when state "CT" is entered. The *event* `givetPA` is raised by the transition from state "tPAcheck" to state "tPA" if tPA is administrated (the value of boolean variable `tPAad` is `true`). In the simplified stroke statechart model (Fig. 2), the two two timing related variables `curT` and `onsetT` represent the current system time and the onset time of stroke symptoms, respectively. We assume that the time unit in the simplified stroke statechart model is minute. Hence, the remaining time of the 3-hour tPA treatment window can be calculated by formula $180 - (\texttt{curT} - \texttt{onsetT})$.

The simplified stroke statechart model in Fig. 2 is transformed to UPPAAL time automata as shown in Fig. 3 with our Y2U[2] tool [10]. The properties **P1** and **P2** are verified in UPPAAL by formula (1) and formula (2), respectively.

[2]The Y2U tool is available at www.cs.iit.edu/~code/software/Y2U/index.html.

$$A[\,]\ \texttt{Stroke.tPA}\ imply\ \texttt{systolicBP} <= 185\ \&\&\ \texttt{diastolicBP} <= 110\ \&\&\ !\ \texttt{hemorrhage} \quad (1)$$

$$A[\,]\ \texttt{Stroke.tPAcheck}\ imply\ \texttt{tpaT} - \texttt{onsetT} <= 180 \quad (2)$$

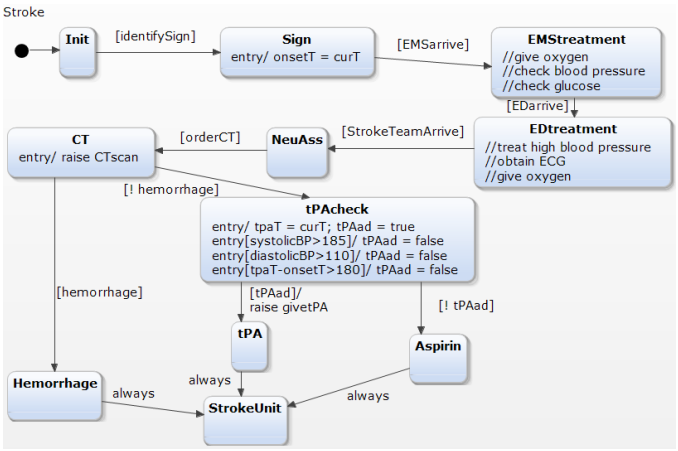

Fig. 2. Simplified Stroke Yakindu Statechart Model

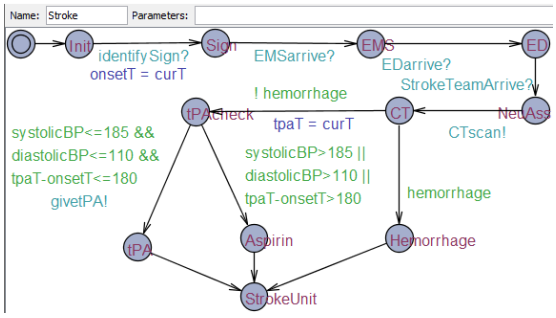

Fig. 3. Simplified Stroke UPPAAL Model

As medical guideline handbooks often assume that all required medical resources are available. With the assumption, both clinical validation results of stroke Yakindu model in Fig. 2 and formal verification results of stroke UPPAAL model in Fig. 3 show that both properties **P1** and **P2** are satisfied. However, the assumption on medical resource availability may not always hold in reality. For example, the `CTscan` medical action in state "CT" of the stroke statechart model in Fig. 2 requires CT machines and CT technicians. If both CT machines and CT technicians are available after 200 minutes from onset of the symptoms, the stroke statechart is then blocked at state "CT" for 200 minutes. In this scenario, the safety property **P2** fails.

The example reveals a fact that safety properties validated and verified in medical guideline models based on the assumption that medical resources are always available can fail because of temporarily unavailable resources. Hence, taking into consideration of medical resource availability in developing verifiable medical guideline models is essential in validating and verifying the safety properties of the guideline models. We model medical resource availability with statecharts and integrate medical resource availability models with medical guideline statecharts to validate and verify safety properties in the following two sections.

III. Model Medical Resource Availability with Statecharts

In this section, we model medical resource availability with statecharts in two steps: (1) automatically annotate required medical resources in executable medical guidelines and (2) explicitly model medical resource availability with statecharts based on resource annotations and given availability information.

A. Annotate Medical Resources in Executable Medical Guideline Models

To model medical resource availability, we need to identify which resources are required by which medical actions and represent the required resources in executable medical guidelines.

We use the simplified stroke statechart model shown in Fig. 2 as an example to illustrate medical resources required by medical guidelines. In the state "CT", a medical action `CTscan` which is modeled as an *event* in Yakindu statecharts is raised by the *entry action* of the "CT" state. According to medical professionals, the `CTscan` medical action requires CT machines and CT technicians. Similarly, a medical action `givetPA` is raised by the *action* of the *transition* from state "tPAcheck" to state "tPA". The `givetPA` medical actioin requires tPA fluid. The examples show that (1) medical actions are modeled as statechart *events* and can be raised in both *states* and *transitions* and (2) medical resources required by medical actions are implicit and not represented in medical guideline statecharts. They need to be provided by medical professionals.

As medical professionals participate in model building and clinical validation processes of medical guideline statecharts, one intuitive method to represent required resources in medical guideline statecharts is that medical professionals review each *state* and *transition* of medical guideline statecharts and manually annotate required medical resources in each *state* and *transition*. The intuitive method works but has a disadvantages that medical professionals need to check all *states* and *transitions* in guideline statecharts when validating the correctness of annotated resource information.

To avoid the disadvantage, we propose an approach to annotate medical resources in executable medical guidelines with two steps: (1) represent medical actions required resources given by medical professionals by a map structure and (2) automatically annotate required medical resources in *states* and *transitions* according to the resource map and raised medical actions in corresponding *states* and *transitions*. Compared to the above intuitive resource annotation method, the proposed approach has an advantage that medical professionals only need to check the resource map when validating the correctness of medical resource information.

In the resource map structure $(\texttt{key}, \overrightarrow{\texttt{value}})$, the *key* is medical actions that are represented by corresponding *event* names in the medical guideline statecharts. The *value* of the resource map is required medical resources of the corresponding *key* (medical action). As a medical action may require multiple resources, we use an array of all required medical resources to represent the *value* in the resource map structure. In the resource array, we replace spaces in resource names with underscores (_). In current work, we only consider the multiple resources required by the same medical action are pairwise independent and leave dependent resources as our future work. As multiple resources are independent, the sequence of multiple resources in a resource array is not important. We give the formal definition of the resource map structure in Definition 1 and show the resource map of the simplified stroke scenario in Example 1.

Definition 1: Given an executable medical guideline model $\mathcal{G}$, a set of medical actions $A = \{a_1, a_2, \ldots, a_n\}$ in the medical guideline $\mathcal{G}$,

and a set of medical resources $R_i = \{r_1^i, r_2^i, \ldots, r_m^i\}$ required by the medical action a_i, the medical resource map $\mathcal{M}$ is defined as

$$\begin{aligned}\mathcal{M} =&\{(a_1, [r_1^1, r_2^1, \ldots, r_{m_1}^1]),\\ &(a_2, [r_1^2, r_2^2, \ldots, r_{m_2}^2]),\\ &\ldots\ldots\ldots\\ &(a_n, [r_1^n, r_2^n, \ldots, r_{m_n}^n])\}.\end{aligned} \tag{3}$$

Example 1: The simplified stroke statechart model shown in Fig. 2 has two medical actions `CTscan` and `givetPA`. Suppose the `CTscan` medical action requires CT machines and CT technicians and the `givetPA` medical actioin requires tPA. According to Definition 1, the resource map of the simplified stroke scenario is

$$\{(\texttt{CTscan}, [\texttt{CT\_machine}, \texttt{CT\_technician}]), (\texttt{givetPA}, [\texttt{tPA}])\}. \tag{4}$$

The required medical resource information represented in the map $\mathcal{M}$ is independent of executable medical guideline models. To model medical resource availability, we also need to annotate the required resources in executable medical guideline models. With the purpose of not affecting execution behaviors and validation/verification results of medical verifiably correct executable medical guideline models, we annotate medical resources by Yakindu statechart *comments*. The annotation is defined as follows.

Definition 2: Given a state S (or a transition T) in a executable medical guideline model $\mathcal{G}$, a set of medical actions $A_S = \{a_1, a_2, \ldots, a_k\}$ modeled in state S (or transition T), and a medical resource map $\mathcal{M}$ of $\mathcal{G}$, the annotation of state S (or transition T) is represented as

$$//@\texttt{RES} : r_1^1, \ldots, r_{m_1}^1, \ldots, r_1^2, \ldots, r_{m_2}^2, \ldots, r_1^k, \ldots, r_{m_k}^k. \tag{5}$$

Based on the medical resource map and the medical resource annotation definitions, we annotate required medical resources in executable medical guideline statecharts with following two steps: first search each *state* S (and transition T) in the given medical guideline statechart $\mathcal{G}$; second, if the actions of state S (or transition T) contain medical actions in the given medical resource map $\mathcal{M}$, add annotation, i.e., formula (5), to state S (or transition T). Algorithm 1 gives the details of the annotation procedure, where the operation $R + R'$ in Line 6 returns the concatenation of R and R'. The time complexity of Algorithm 2 is $O(L * M * N)$, where L is the element number of the medical resource map $\mathcal{M}$, M is the number of medical resources required by the medical guideline model $\mathcal{G}$, and N is the sum of states' number and transitions' number in $\mathcal{G}$.

Example 2: Given the simplified stroke statechart model shown in Fig. 2 and a resource map of formula (4). The state "CT" has a medical action `CTscan`. We use `CTscan` as the key to search the resource map given by formula (4) and find resource array [`CT_machine`, `CT_technician`]. According to Definition 2, we add the annotation "//@`RES` : `CT_machine`, `CT_technician`" to state "CT". Similarly, we add the annotation "//@`RES` : `tPA`" to the transition from state "tPAcheck" to state "tPA". The annotated stroke statechart model by Algorithm 1 is depicted in Fig. 4, where the annotated states and transitions are marked by red rectangle.

Algorithm 1 ANNOTATION

Input: An executable medical guideline model $\mathcal{G}$ and a medical resource map $\mathcal{M}$ (formula (3)).
Output: The annotated medical guideline model $\mathcal{G}'$.

```
1:  for each state S or transition T in G do
2:      Define a resource array R = [ ]
3:      for each raised action a in S or T do
4:          Find R' with key a in M
5:          if R' is not NULL then
6:              R = R + R'
7:          end if
8:      end for
9:      if R is not empty then
10:         Add an annotation in the format of formula (5) to state S
            or transition T
11:     end if
12: end for
13: return G
```

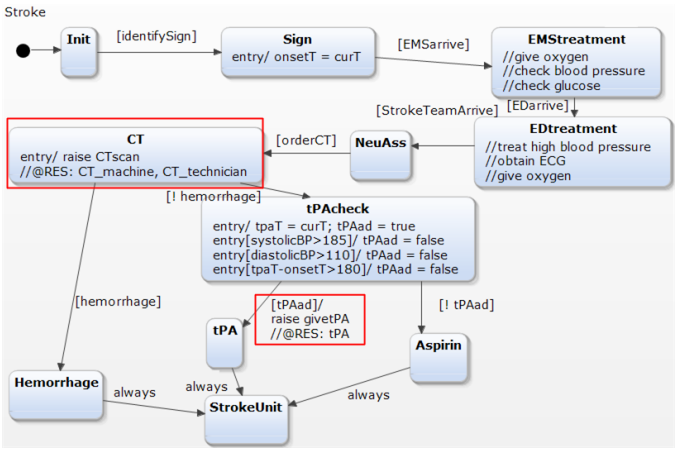

Fig. 4. Annotated Stroke Yakindu Model

B. Model Medical Resource Availability with Statecharts

Given a resource map $\mathcal{M}$ and resource availability information, we develop statecharts to model medical resource availability in three steps: (1) design a `Timer` statechart to record current system time; (2) declare a boolean variable for each resource to denote its availability at current time; and (3) build a statechart for each resource to represent its given availability information.

For the `Timer` statechart, we use an integer variable `curT` to denote current system time and let a `Timer` statechart to increase current time `curT`. The `Timer` statechart only contains one state which has a self-loop transition to increase current time `curT` by 1 every one time unit. Fig. 5 shows an example `Timer` statechart with time unit minute, which increases `curT` by 1 `every 60s`.

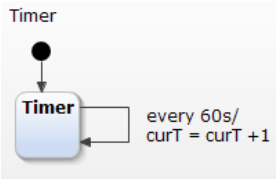

Fig. 5. Timer Statechart

To represent resource availability related variables, we declare an interface named `RES`. For each unique resource r in a given resource map $\mathcal{M}$, we declare a boolean variable V_r in the interface `RES` to denote the resource r's availability at current system time. The variable V_r has the same name as the corresponding resource r and default value `false` that means the resource r is not available ini-

tially. For example, the resource map of the simplified stroke scenario given in formula (4) contains three medical resources `CT_machine`, `CT_technician`, and `tPA`. The declared resource availability variables of the simplified stroke scenario is shown in Fig. 6.

```
interface RES:
var CT_machine : boolean = false
var CT_technician : boolean = false
var tPA : boolean = false
```

Fig. 6. Resource Availability Variables

For each unique resource r in the given resource map $\mathcal{M}$, we build a statechart to represent its given availability information. Each resource statechart contains only one state S that has a self-loop transition T with guard `true`. The transition T ensures that the resource r's availability is checked at each statechart execution cycle and maintains the latest value. The entry actions of the state S check the resource r's availability at current time `curT` based on given resource availability information. If the resource r is available, the entry action assigns `true` value to the corresponding resource boolean variable V_r; otherwise, the resource variable V_r is assigned as `false`. We use Example 3 to show resource statecharts for the simplified stroke scenario.

Example 3: For the simplified stroke scenario, given the resource map as formula (4) and resource availability information as follows: (1) both `CT_machine` and `CT_technician` are available after 200 minutes and (2) the `tPA` is always available. For resource `CT_machine`, we build the "CT_machine" statechart with only one state named "CT_machine" which has a self-loop transition with guard `true`. According to given resource availability information that the `CT_machine` is available after 200 minutes, we add two entry actions to the state "CT_machine":

1) `entry`$[$`curT` $>$ `200`$]$/`RES.CT_machine` $=$ `true` assigns variable `CT_machine` as `true` if current time `curT` is larger than 200 minutes and denotes that the resource `CT_machine` is available after 200 minutes;
2) `entry`$[$`curT` $<=$ `200`$]$/`RES.CT_machine` $=$ `false` assigns variable `CT_machine` as `false` if current time `curT` is smaller than or equal to 200 minutes and denotes that the resource `CT_machine` is not available until 200 minutes.

Similarly, we build two statecharts for resource `CT_technician` and `tPA`, respectively. The resource statecharts are shown in Fig. 7.

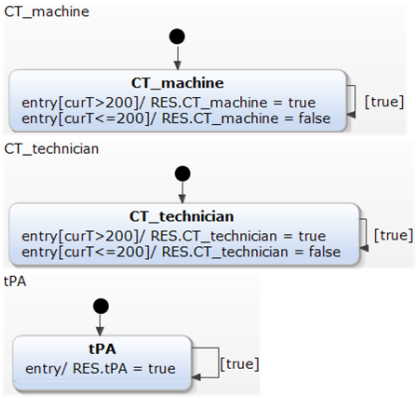

Fig. 7. Resource Statecharts

IV. Integrate Medical Resource Availability Models with Medical Guideline Statecharts

To clinically validate and formally verify the safety of medical guideline models with consideration of medical resource availability, we need to integrate medical resource availability models with medical guideline statecharts.

According to the medical resource availability modeling approach presented in Section III-B, for each resource r, a boolean variable V_r is declared. We use the declared resource availability variable V_r to bridge the communication between medical resource availability models and medical guideline statecharts and modify medical guideline statecharts with following integration rules.

- **Integration Rule 1**: For each transition T with guard G, if it is annotated by "//@`RES` : $r_1, r_2, \ldots, r_n$", the guard G is modified by $G = G$ && V_{r_1} && V_{r_2} && $\ldots$ && V_{r_n};
- **Integration Rule 2**: For each state S, if it is annotated with "//@`RES` : $r_1, r_2, \ldots, r_n$", apply **Integration Rule 1** to all incoming transitions of the state S with the annotation.

Algorithm 2 gives the integration procedure. The time complexity of Algorithm 2 is $O(M * N^2)$, where M is the number of medical resources required by the medical guideline model $\mathcal{G}$ and N is the sum of states' number and transitions' number in $\mathcal{G}$. Example 4 illustrates how we apply the integration rules to integrate the CT machine, CT technician, and tPA fluid availability models with the simplified stroke statechart.

Algorithm 2 Integration

Input: An annotated medical guideline model $\mathcal{G}$.
Output: The integrated medical guideline model $\mathcal{G}'$.

```
for each state S in G do
    if S is annotated with "//@RES : r_1, r_2, ..., r_n" then
        for each incoming transition T with guard G of state S do
            G = G && V_{r_1} && V_{r_2} && ... && V_{r_n}
        end for
    end if
end for
for each transition T with guard G in G do
    if T is annotated with "//@RES : r_1, r_2, ..., r_n" then
        G = G && V_{r_1} && V_{r_2} && ... && V_{r_n}
    end if
end for
return G
```

Example 4: We integrate the resource availability models in Fig. 7 with the annotated stroke statechart model in Fig. 4. The transition T_1 from state "tPAcheck" to state "tPA" is annotated with "//@`RES` : `tPA`" and has guard G_1 = `tPAad`. Based on **Integration Rule 1**, the transition T_1' guard is set as G_1 = `tPAad` && `RES.tPA`. The state "CT" is annotated by "//@`RES` : `CT_machine`, `CT_technician`" and only has one incoming transition T_2 with guard G_2 = `orderCT` from state "NeuAss". According to **Integration Rule 2**, we apply **Integration Rule 1** to the transition T_2 and set the guard as G_2 = `orderCT` && `RES.CT_machine` && `RES.CT_technician`. Fig. 8 shows the integrated stroke statechart, where the modified transitions are marked by red rectangle.

To clinically validate and formally verify the safety of the stroke statechart with the consideration of resource availability, we run simulations of the integrated stroke model (Fig. 8) through Yakindu, transform it to integrated stroke UPPAAL model (Fig. 9), and verify the two safety properties (**P1** and **P2**) in UPPAAL. The

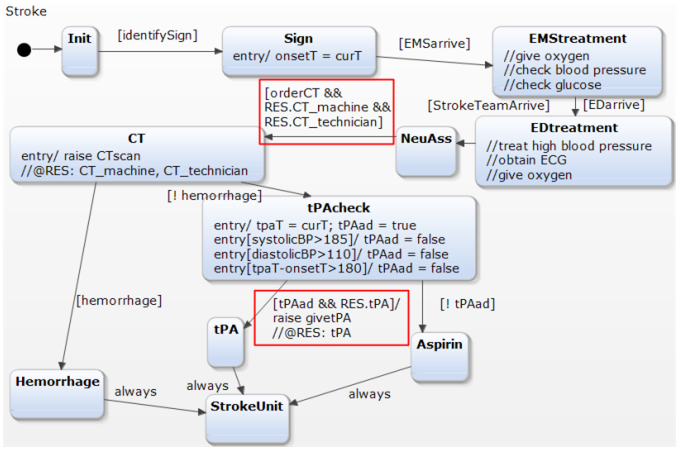

Fig. 8. Integrated Stroke Yakindu Model

resource availability is given in Example 3, i.e., both `CT_machine` and `CT_technician` are available after 200 minutes, the `tPA` is always available. Both simulation and verification results show that the property **P1** holds while **P2** fails.

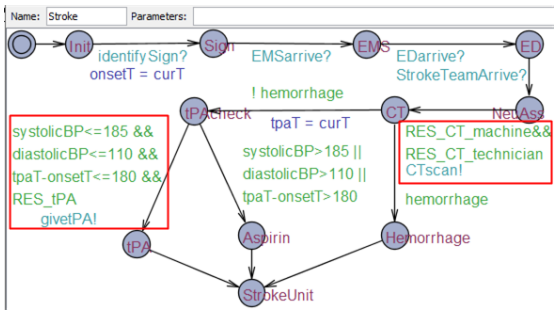

Fig. 9. Integrated Stroke UPPAAL Model

V. Simplified Stroke Case Study

The stroke statechart model given in Fig. 2 has only focused on the CT scan and IV tPA administration procedures, but omitted the details of other medical procedures. To validate the effectiveness of the proposed approach, we extend the simplified stroke model by considering following scenarios with different patient conditions: (1) if a patient's blood pressure is not within the range required by tPA administration, a blood pressure control procedure needs to be performed; (2) if tPA administration is approved within 3 hours from onset of stroke symptoms, an IV tPA procedure is performed; (3) if tPA administration is approved in the 3-6 hour window from the onset time, an IA tPA procedure is performed; and (4) if tPA is not approved, aspirin is given to patients.

We use the proposed approach to annotate resources, model resource availabilities, and integrate resource models with the extended stroke statechart model. The integrated stroke statechart is shown in Fig. 10.

To clinically validate and formally verify the safety of the stroke statechart with the consideration of resource availabilities, we run simulations of the integrated stroke statechart model (Fig. 10) through Yakindu, transform the integrated stroke model to an UPPAAL model with the Y2U tool [10] (Fig. 11), and verify the safety properties in UPPAAL.

In addition to the properties **P1** and **P2** given in formula (1) and formula (2), we also need to verify property **P3** that the IA tPA

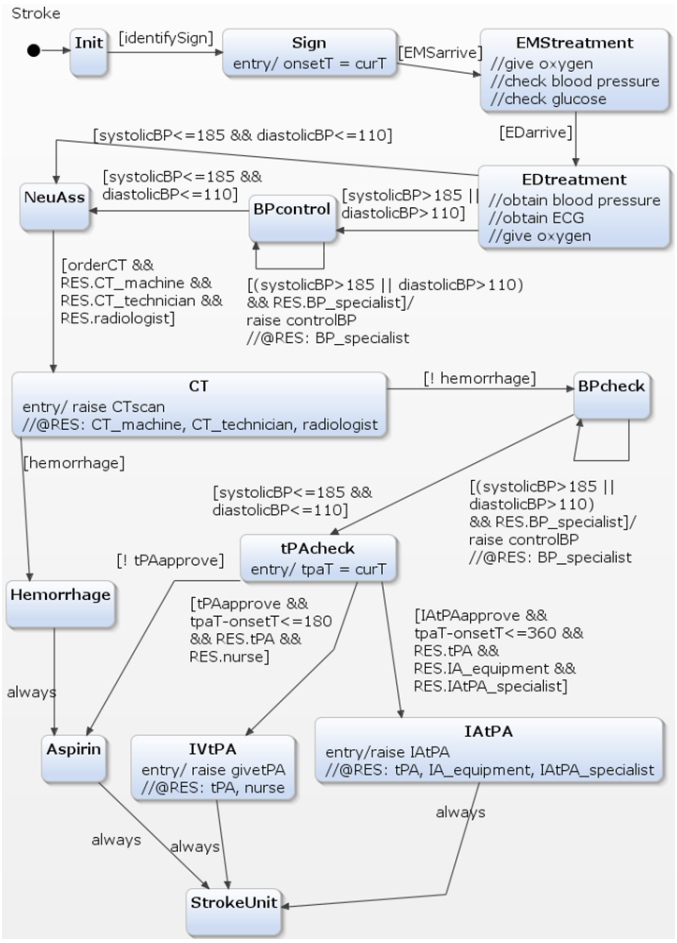

Fig. 10. Integrated Stroke Statechart

administration must be completed within 6 hours from onset of stroke symptoms, i.e.,

$$A[\,]\ \texttt{Stroke.IAtPA}\ \mathit{imply}\ \texttt{tpaT} - \texttt{onsetT} <= 360. \qquad (6)$$

Assume a patient's onset time of stroke symptom is 0, the resource schedule is given in Fig. 12, where resources are not available during shaded time slots. Both simulation and verification results show that the safety property **P1** and **P3** hold, but **P2** fails.

The case study demonstrates that the proposed approach is effective in capturing safety property fails caused by temporarily unavailable resources in both clinical validation and formal verification process.

VI. Conclusion

Medical guidelines often assume that all required medical resources are available. Unfortunately, the reality at medical facilities is that some medical resources can be temporarily unavailable. Hence, taking into consideration of medical resource availability in developing verifiable medical guideline models is essential in validating and verifying safety properties. The paper presents an approach to separately model medical resource availability with statecharts and automatically integrate medical resource availability statecharts with verifiably correct executable medical guideline models. The proposed approach allows to minimize the change impact on medical guideline models caused by resource availability variations. Applying separation of concern methodology in our approach further allows different

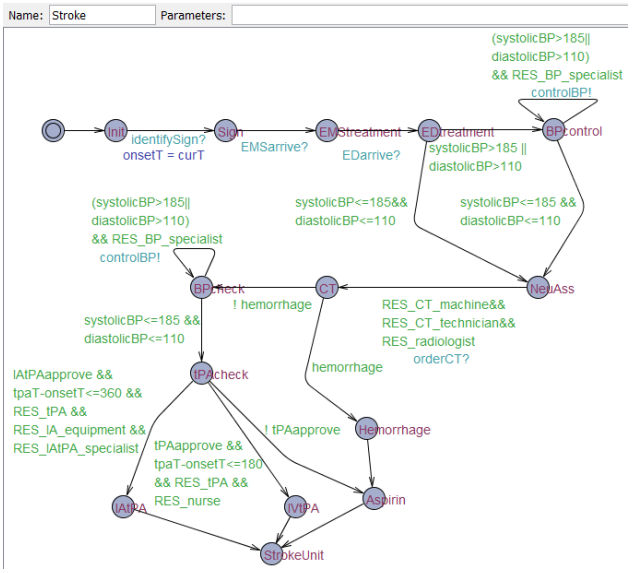

Fig. 11. Stroke UPPAAL Model

	Time Slots (30 minutes)									
	30	60	90	120	150	180	210	240	270	300
CT Machine										
CT Technician										
Radiologist										
tPA										
Nurse										
BP Specialist										
IA Equipment										
IA tPA Specialist										

Fig. 12. Stroke Resource Schedule

professionals to focus on only their own domains, e.g., medical professionals and resource administrators focus on medical guidelines and medical resource availability information, respectively. The separation also improves model understandability for both medical professionals and resource administrators. In addition, this approach can be easily implemented in our existing platform [10] with which the medical guideline models with the consideration of resource availability can be clinically validated by medical professionals and formally verified with existing tools. We also use a simplified stroke scenario as a case study to investigate the effectiveness and validity of our approach. In this paper, we consider the multiple resources required by the same medical action are pairwise independent. Our future work is to extend the presented approach to support dependent resources.

ACKNOWLEDGEMENT

We would like to thank Professor Yu Jiang from Tsinghua University for his valuable suggestions in this work. The work is supported in part by NSF CNS 1545008 and NSF CNS 1545002.

REFERENCES

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[2] A. S. Association. Ischemic stroke. https://www.strokeassociation.org/idc/groups/stroke-public/@wcm/@hcm/documents/downloadable/ucm_309725.pdf.

[3] A. S. Association. Stroke treatment. http://www.strokeassociation.org/STROKEORG/AboutStroke/Treatment/Stroke-Treatment_UCM_492017_SubHomePage.jsp.

[4] M. Balser, C. Duelli, and W. Reif. Formal semantics of asbru - an overview. *Proc. of the 6th Biennial World Conference on Integrated Design and Process Technology*, 5(5):1–8, 2002.

[5] G. Behrmann, A. David, and K. Larsen. A tutorial on uppaal. In *Formal Methods for the Design of Real-Time Systems*, pages 200–236. Springer, 2004.

[6] S. Christov, B. Chen, G. S. Avrunin, et al. Formally defining medical processes. *Methods of Information in Medicine*, 47(5):392, 2008.

[7] S. Christov, B. Chen, G. S. Avrunin, et al. *Rigorously Defining and Analyzing Medical Processes: An Experience Report*, pages 118–131. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[8] J. Fox, N. Johns, and A. Rahmanzadeh. Disseminating medical knowledge: the proforma approach. *Artificial Intelligence in Medicine*, 14(1-2):157 – 182, 1998.

[9] C. Guo, Z. Fu, S. Ren, Y. Jiang, and L. Sha. Towards verifiable safe and correct medical best practice guideline systems. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, July 2017.

[10] C. Guo, S. Ren, Y. Jiang, P.-L. Wu, L. Sha, and R. Berlin. Transforming medical best practice guidelines to executable and verifiable statechart models. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10, April 2016.

[11] D. Harel. Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274, 1987.

[12] Itemis. Yakindu statechart tools (sct). https://www.itemis.com/en/yakindu/statechart-tools/, 2016.

[13] E. C. Jauch, B. Cucchiara, O. Adeoye, et al. Part 11: adult stroke: 2010 american heart association guidelines for cardiopulmonary resuscitation and emergency cardiovascular care. *Circulation*, 122(18 suppl 3):S818–S828, 2010.

[14] Y. Jiang, H. Liu, H. Kong, R. Wang, M. Hosseini, J. Sun, and L. Sha. Use runtime verification to improve the quality of medical care practice. In *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*, pages 112–121. IEEE, 2016.

[15] Y. Jiang, H. Song, R. Wang, M. Gu, J. Sun, and L. Sha. Data-centered runtime verification of wireless medical cyber-physical system. *IEEE Transactions on Industrial Informatics*, 13(4):1900–1909, 2017.

[16] Y. Jiang, Y. Yang, H. Liu, H. Kong, M. Gu, J. Sun, and L. Sha. From stateflow simulation to verified implementation: A verification approach and a real-time train controller design. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*, pages 1–11. IEEE, 2016.

[17] Y. Jiang, H. Zhang, Z. Li, Y. Deng, X. Song, M. Gu, and J. Sun. Design and optimization of multiclocked embedded systems using formal techniques. *IEEE transactions on industrial electronics*, 62(2):1270–1278, 2015.

[18] J. Kim, I. Kang, J. Y. Choi, and I. Lee. Timed and resource-oriented statecharts for embedded software. *IEEE Transactions on Industrial Informatics*, 6(4):568–578, Nov 2010.

[19] B. A. McKinley, L. J. Moore, J. F. Sucher, et al. Computer protocol facilitates evidence-based care of sepsis in the surgical intensive care unit. *Journal of Trauma and Acute Care Surgery*, 70(5):1153–1167, 2011.

[20] M. McManus and D. S. Liebeskind. Blood pressure in acute ischemic stroke. *Journal of Clinical Neurology (Seoul, Korea)*, 12(2):137–146, 2016.

[21] V. L. Patel, V. G. Allen, J. F. Arocha, and E. H. Shortliffe. Representing clinical guidelines in glif. *Journal of the American Medical Informatics Association*, 5(5):467–483, 1998.

[22] E. A. Prince, S. H. Ahn, and G. M. Soares. Intra-arterial stroke management. In *Seminars in interventional radiology*, volume 30, pages 282–287. Thieme Medical Publishers, 2013.

[23] P. Terenziani, S. Montani, A. Bottrighi, et al. The glare approach to clinical guidelines: main features. *Stud. Health Technol. Inform.*, pages 162–166, 2004.

[24] S. W. Tu and M. A. Musen. Modeling data and knowledge in the eon guideline architecture. *Medinfo*, pages 280–284, 2001.

[25] P. Wu, D. Raguraman, L. Sha, R. Berlin, and J. Goldman. A treatment validation protocol for cyber-physical-human medical systems. In *Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on*, pages 183–190, Aug 2014.

[26] Y. Yang, Y. Jiang, M. Gu, and J. Sun. Verifying simulink stateflow model: timed automata approach. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 852–857. ACM, 2016.