SAFE MEDICAL DEVICE INTERACTION RULES

BY

YUN YOUNG LEE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Research Associate Professor Ralph Johnson

ABSTRACT

The advent of computerized medical devices has resulted in better accuracy and increased safety for patients and clinicians, yet medical accidents are still not uncommon. With the computerized medical devices and their technical complexity, any given clinical environment is a composition of complex computer systems. This introduces new form of risks, not only with the usage of a single device but with the interactions of multiple devices, which increase exponentially with respect to the number of devices. Even the most advanced medical devices fail to serve their purpose if their users cannot effectively use them, or if their complexity deters the users.

This thesis presents an English-like configuration language for defining device interactions rules, which makes it easier for non-programmers like clinicians and surgeons to configure a complex medical system. The user-defined rules are then automatically and formally proven for its safety, by means of a state machine and the reachability of safe states within it. The automatic checker ensures that there is no error-by-omissions and conflicts between the rules.

This thesis is dedicated to my mother, the strongest person I know who sacrificed everything to give her children a chance at a better future. It takes some serious guts to raise three children alone in a foreign country. I love you, mum!

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Ralph Johnson for his guidance, encouragement, and support in this project, as well as for his gracious understanding in my frequent out-of-town trips. He gave me an exciting project to work on and still allowed me to pursue my sometimes vague and random interests, and I am immensely grateful for that.

I would like to thank the MDPnP project team – Prof. Lui Sha, Prof. ChengXiang Zhai and Prof. Jose Meseguer for their guidance, and Charlie Kim, Maryam Rahmaniheris and others for being a great fun to work with.

Finally I would like to thank my previous Master's advisor, Prof. Darko Marinov, for his guidance and support throughout my early years in the graduate school; my dearest group of friends who support, spoil and entertain me, for making this small town and grueling graduate school feel like home; and my family for their love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

EPIC	Explicitly Parallel Instruction Computing
AST	Abstract Synyax Tree
ASTM	American Society for Testing and Materials
CPB	Cardiopulmonary Bypass
EKG	Electrocardiography
ICE	Integrated Clinical Environment
IV	Intravenous
MDPnP	Medical Device Plug and Play
NASS	Network-Aware Supervisory System
OR	Oerating Room
PACU	Post-anesthesia care unit
PCA	Patient-Controlled Analgesia

CHAPTER 1

INTRODUCTION

In ancient Rome, physicians used finely-toothed jawed forceps to crush the uvula before it is amputated in order to prevent hemorrhaging [1]. This, aside from being excruciatingly painful, was also very unsafe. A modern counterpart to the uvula crushing forceps is the airway laser, which works by focusing enormous heat on a small area of tissue and allowing instantaneous sealing of small vessels and lymphatics with minimal damage to surrounding tissues [2].

Medical devices have evolved greatly over time, and recent advancements in computer technologies have only accelerated this trend. Airway lasers allow quick upper airway surgeries, radiation therapy machines kill cancerous cells without requiring surgical incisions, and ventilators help patient's respiration with a flick of a switch. This makes it especially ironic that with such highly computerized medical devices that provide better accuracy and safety, medical accidents - some of which are fatal - are still not uncommon. There are reported incidents of fire during airway laser surgery [3], where the laser ignites surgical tools such as sponges or alcohol due to high concentration of oxygen or nitrous oxide often used as anesthesia; the Therac 25 incident is a well-known medical accident where patients were given massive overdose of radiation, approximately 100 times the intended dose, and resulted in deaths of three patients [4]; a patient died because a clinician did not remember to restart the ventilator after taking an X-Ray during surgery [5].

The problem is that even though these computerized devices give better accuracy and safety, they also introduce technical complexity in operation that may not be easily understood or controlled by their users, that is, the clinicians. There are many reasons, including the quality of software and hardware of the devices, but perhaps more importantly, many of the devices are designed as stand-alone devices that are often proprietary, and clinicians have little or no control over how devices operate in conjunction with one

another. Also, the number of possible interactions increases exponentially with respect to the number of devices. Clinicians in ancient Rome with forceps probably did not have to worry about the possibility of fire during an airway surgery.

Perhaps such complications that arise from the device interactions are unavoidable; as the medical devices evolve, their risks evolve as well. However, despite how advanced these medical devices become, clinicians must have complete control over them. Every medical decision needs to come from the clinicians and not the devices or the engineers who built the devices, in order to ensure that the responsibility of medical practices is on the clinicians. By providing clinicians the means to control device interactions, not only the devices interoperate based on the professional intelligence of clinicians, but the devices can also alert the clinicians when an abnormality is detected, allowing them to promptly act upon it which may sometimes be the only way to ensure patients' well-being.

There are a number of ways to give clinicians the control of the devices. They may be given training on operating the devices, or they could follow a strict set of instructions. These however can be restrictive of clinicians' activities. Clinicians should dictate the device operations and not be led by them. There is a need for more intuitive approach for clinicians to decide how the devices should interoperate, that preferably do not require extensive technical expertise. Furthermore, these approaches should also provide the safety and accuracy that computerized devices are designed for.

This thesis presents ongoing research in medical device interoperability. Specifically, it introduces a very natural and English-like language for configuring safety rules for device interactions, along with an automated tool for checking the consistency of the user-specified rules. This work emphasizes the important role of clinicians in a medical environment by putting them in control of the devices. Chapter 2 provides the background and motivation of this work, and chapter 3 lists the use cases derived from on-site interviews with medical personnels, which formed the basis for derivation of safety rules. Chapter 4 describes the language syntax, while chapter 5 describes the semantics of the language in terms of a finite state machine. Chapter 6 details how the safety is checked, and chapter 6 concludes with brief outline of future work.

CHAPTER 2

BACKGROUND

2.1 Medical Device Plug and Play

Ensuring safe interoperability of medical devices is not a new challenge. The advent of computerized devices introduced new set of risks and complexities, and now any given clinical environment is a composition of complex computer systems. Even though there exists protocols such as ISO/IEEE 11073 Medical/Health Device Communication Standards [6] which enables communication for point-of-care medical and personal health devices with external computer systems, these involve device specializations which means there are different protocols for different devices. The generality that can help widespread adoption of such protocols still lacks, and there is a growing need for better communication standards that can address the following challenges [7]:

- Correct interpretation of medical data and effective monitoring of patients' conditions are context-dependent. Operational contexts may be different at different stages of a complex medical procedure. How can we obtain and distribute timely medical context information consistently and correctly? How can we use the context information and real time monitoring data to provide better decision support?

- Medical devices have different safety levels and different levels of reliability. How do we verify that a dynamically configured system is safe at each stage of a complex medical procedure and is safe during the transition from one stage to next? This is the medical workflow safety challenge.

- There are many commonly used medical device configuration patterns. How can we formally describe them, verify their safety and use them

as building blocks in a medical procedure driven workflow architecture?

To address these challenges, a consortium of professionals from medical, industrial, academic, and government groups have been collaborating through the Medical Device Plug and Play (MDPnP) research program initiated by the Center for Integration of Medicine & Innovative Technology (CIMIT) to identify open standards and technology for safe integration of medical devices. A number of researchers at the University of Illinois at Urbana-Champaign have been actively participating in the program, combining expertise from multiple computer science disciplines. The research group has identified the following goals [7]:

1. Develop a representative set of use cases to guide MDPnP Research in general and user interface in particular.

2. Develop verifiably safe architecture patterns as building blocks for safe MDPnP.

3. Develop context-sensitive MD PnP and decision support.

4. Develop medical grade wireless networking technologies for safe MDPnP.

The work presented in this thesis addresses the first goal, the use cases, and introduces one approach to the third goal, providing safer and easier decision support by allowing clinicians to configure safe device interaction rules.

2.2 Motivation

With highly advanced and specialized medical devices, any given clinical environment is a composition of complex computer systems. The exponential increase in number of possible device interaction with respect to number of devices only adds to an already-complicated medical procedures, and incorrect interactions can have serious, possibly fatal, effects on patients. In addition, the nature of medical procedures is inherently fluid and non-deterministic because it depends on patients' conditions, clinical history, and

many other variables. As such, there is no set of device interaction rules completely specifying any medical procedure, and thus surgeons and clinicians need to define such rules based on their expertise and experiences. Then, how can surgeons safely configure complex medical systems without extensive technical understanding that is often required, while ensuring the safety of patients? The need for an accessible way for surgeons - or to generalize, non-technical users - to configure a complex medical system prompted for an easy-to-use configuration language that hides the complicated technical aspects. The natural English-like configuration language presented in this thesis satisfies that purpose.

As a result of an ongoing research on MDPnP, the research group has developed a generic framework called Network-Aware Supervisory System (NASS). NASS integrates medical devices into a clinical interoperability system, which also provides a development environment in which medical device supervisory logic can be developed based on the assumptions of an ideally robust network[8]. More specifically, NASS generates a set of synchronized plans, based on a supervisory logic, for each device to carry out in case of a network, system or device failure. The supervisory logic defines device interactions, according to which the NASS coordinates devices by automating device (de)activations. The safe device interaction rules are thus the core of NASS. Given an unsafe set of rules, NASS will not help prevent medical accidents.

Given the configuration language, suregeons can define the safe device interaction rules that essentially scriptize the supervisory logic required by the NASS framework. The rules are parsed into an abstract syntax tree (AST), which in turn are mapped to the NASS framework's representation of safety rules.

CHAPTER 3

USE CASES

Appendix B of part 1 in American Society for Testing and Materials's general requirements and conceptual architecture standards (ASTM standards) [5] details a number of scenarios providing clinical context for the development of an integrated medical device systems. This chapter contains these use cases in a form that is more usable from the software engineering perspective. Some of those use cases that are relevant to device-device interaction are included here for completeness. The first use case on airway laser surgery is derived from [3] by the current author, and the rest are based on scenarios in the ASTM standards and were written by Maurice Rabb.

3.1 Airway Laser Surgery with Ventilator

Goal

Surgeons use a ventilator and a laser devices to perform airway or any other relevant surgery, without causing fire that may harm/kill a patient or themselves.

Summary

During a laser surgery, a ventilator may be in use in conjunction with a laser. If both the laser and the ventilator are in use at the same time, there is a risk of fire where the laser may ignite combustible material, such as surgical gloves or sponges. This may also cause the patient to inhale the burning vapors, or spread fire with exhalation. The surgeons are also at risk of burns.

Rationale

"A 35 year-old male presented for laser ablation of recurrent laryngeal pa-

pillomata. ... History and physical exam were unremarkable other than a 20 pack-year smoking history, a bushy mustache, and a weight of 129 kg. ... Anesthesia was induced with thiopental and fentanyl. Vecuronium was given to provide relaxation. Mask ventilation with O_2, N_2O, and isoflurane was without problems. ... the surgeon inserted an adult Dedo laryngoscope. Jet ventilation was instituted with oxygen via a 13 gauge cannula inserted in the left light-carrier channel of the Dedo laryngoscope. A thumb-controlled valve and 50 psi oxygen from the piped-in system powered the jet. The patient's face and the perioral area were covered with soaking wet towels to the extent that only the barrel of the Dedo laryngoscope was visible. Anesthesia was maintained with thiopental and fentanyl during jet ventilation. There were no problems, although arterial oxygen saturation (SaO_2) dipped briefly into the high 80% range during episodes of apnea requested by the surgeons to eliminate movement of the vocal cords. Spot checks of end-tidal CO_2 during jetting were in the 30-40 mm Hg range. Near the end of the surgical procedure, the surgeon yelled "Fire" and bright blue and orange flames accompanied by a muffled roar were observed coming out of and around the laryngoscope. Jet ventilation was stopped, the towels removed, and the patient's blazing mustache extinguished with the wet towels. The surgeon, was in a great deal of pain, and noted that two of the fingers on his right hand had had the latex glove burned completely away. ... The patient suffered a second degree burn to his right upper lip... The surgeon suffered second degree burns of the right index and middle fingers."

Equipment

- Airway Laser
- Anesthesia or oxygen ventilator

Actors

- Patient
- Surgeon
- Anesthesiologist
- Additional operating room (OR) staff

Pre-conditions

- The patient is connected to the anesthesia machine.
- The patient is under anesthesia.
- The anesthesia machine is connected to the integrated hospital information system.
- The laser is connected to the integrated hospital information system.

Post-conditions

- Success conditions
  - Laser is used without causing fire.
  - The ventilators normal operation is: a) not interrupted at all, during synchronized use; or b) not
  - interrupted beyond a set threshold.
  - The patient continues to be properly respirated.
- Failure conditions
  - Laser used in conjunction with ventilator causes fire.
  - The ventilator stops or is interrupted too long.
  - The patient goes without respiration too long.

Triggers

- The surgeon requests the use of laser.

Scenarios

- A: Standard operation
  - The surgeon requests laser use.
  - The anesthesiologist stops ventilator before laser is started.

  - The surgeon waits a while to make sure that the gas/oxygen from the ventilator is dissipated. The length of time for waiting may be predefined.
  - The surgeon continues with the procedure using the laser.
  - The surgeon signals the end of laser use.
  - The anesthesiologist waits for few seconds. The waiting period may be predefined.
  - The anesthesiologist resumes the ventilator.

- B: Ventilator fails to stop
  - The laser is prohibited until the ventilator stops.
  - When the ventilator stops, surgeon waits for the designated period of time before starting the laser.

- C: Patient has low blood-oxygen (SpO_2) level
  - The laser is prohibited and ventilator is enforced.

3.2 Morphine Administration

Goal

Patient uses a patient-controlled analgesia (PCA) system to self-administer morphine sulfate for pain management, without injuring or killing him/herself.

Summary

While on the PCA infusion pump, the patient is monitored with a respiration rate monitor and a pulse oximeter. If physiological parameters move outside the pre-determined range, the infusion can be stopped and alarms sent to notify the clinical staff to examine the patient and restart the infusion if appropriate. The use of two independent physiological measurements of respiratory function (oxygen saturation and respiratory rate) enable a smart monitor to optimize sensitivity to detect respiratory compromise while reducing false alarms.

Rationale

"A 49-year-old woman underwent an uneventful total abdominal hysterectomy and bilateral salpingo-oophorectomy. Postoperatively, the patient complained of severe pain and received intravenous morphine sulfate in small increments. She began receiving a continuous infusion of morphine via a PCA pump. A few hours after leaving the PACU and arriving on the floor, she was found pale with shallow breathing, a faint pulse, and pinpoint pupils. The nursing staff called a 'code', and the patient was resuscitated and transferred to the intensive care unit on a respirator. Based on family wishes, life support was withdrawn and the patient died. Review of the case by providers implicated a PCA overdose."

Equipment

- PCA pump containing morphine sulfate
- Large volume infusion pump acting as a carrier for saline
- Pulse oximeter
- Non-invasive blood pressure device when is this used
- Respiration rate monitor what type (belt or airway sensor)
- Patient-nurse call system
- The integrated system

Actors

- Patient
- Physician
- Nurse
- Clinical assistant

Pre-conditions

- The PCA pump system has been set up by the nurse or clinical assistant.

- The nurse has performed an IV line assessment for the patient.
- The pulse oximeter is connected to the patient.
- The respiration rate monitor is connected to the patient.

Post-conditions

- Success conditions
  - The PCA pump administers medication at the proper dosage and the proper frequency.
  - The patient is able to self-administer pain relief as needed.
- Alarm conditions
  - The PCA pump fails
  - The patient's SpO_2 or respiration rate are detected to be outside the defined acceptable range
  - The patient or family members pushes delivery button too often
- Failure conditions
  - The patient overdoses.
  - The patient doesn't receive adequate pain relief.
  - The nurse call system receives too many spurious alarms.

Triggers

- The nurse activates the PCA pumping system.
- The patient pushes the delivery button.
- A family member pushes the delivery button.

Scenarios

- Operational Scenarios
  A: Normal patient

- The nurse activates the PCA pump system.
- The patient's SpO_2 and respiration rate are monitored continuously.
- The system uses an algorithm based on the set-up information and the current SpO_2 and respiration rate to determine if the system is operating in a safe range.
- The patient's vital-signs remain within acceptable range throughout the operation of the infusion pump

B: Patient at risk

- Same setup as A
- The algorithm detects decreases in the patient's SpO_2 and/or respiration rate below the calculated or pre-set threshold, a message is sent via the nurse call system with the appropriate level of alarm.
- If the alarm is at a high enough level, a command is sent to stop the PCA pump.

- Hazard Scenarios

  A: PCA Pump detects internal error

  - The PCA pump detects that it is in an unsafe state (eg. overflow, sporadic flow).
  - The PCA pump stops pumping.
  - The PCA pump sends failure information to the integrated system
  - The PCA pump or the integrated system generates an alarm

  B: Integrated Clinical Environment (ICE) detects PCA pump disconnect

  Conditions: An intentional or unintentional disconnection of the PCA pump from the rest of the sytem, or a failure of PCA pump.

  - The ICE has not received any acknowledgements of connectivity from the PCA pump
  - The ICE gives an alarm

C: PCA detects ICE disconnect
Conditions: An intentional or unintentional disconnection of the PCA pump from the rest of the sytem, or a failure of the ICE.

- The PCA pump has not received any input for more than a specific amount of time, either from the SpO_2 or respiration monitors, from a nurse, or from the ICE.
- The PCA pump has been in operation without any input.
- The PCA pump stops pumping.
- The individual pump gives an alarm

D: Patient wants to self-administer pain relief

- Also applies to family members
- The patient pushes the delivery button.
- If the patient's request is within the frequency allowance, an immediate dose of the drug is administered. Otherwise, the request is ignored.

E: Patient goes for a walk

- The PCA pump becomes disconnected from the other devices. hazard scenarios B and C apply
- If the patient is not in good condition or is high risk, a message is sent via the nurse call system with the appropriate level of alarm.
- If the PCA pump remains disconnected beyond a specific amount of time, a message is sent via the nurse call system with the appropriate level of alarm.
- If the PCA pump remains disconnected beyond a specific amount of time, the PCA stops pumping.

F: Patient or family member pushes the delivery button too often
Conditions: A stationary patient

- The patient or a family member pushes the delivery button too frequently.

  - A message is sent via the nurse call system with the appropriate level of alarm. should be non-critical alarm or warning

- Failure Scenarios

  A: Patient is incorrectly considered at too low a risk

  Same as operational scenarios except:

  - Improperly, the patient's SpO_2 and respiration rate are not below the calculated or pre-set threshold.
  - No alarm, or an inappropriately low-level alarm message is sent via the nurse call system.
  - The PCA pump continues to run.
  - The patient overdoses.

  B: Patient is incorrectly considered at too high a risk

  Same as operational scenarios except:

  - Improperly, the patient's SpO_2 and respiration rate are below the calculated or pre-set threshold.
  - The algorithm detects decreases in the patient's SpO_2 and/or respiration rate below the calculated or pre-set threshold.
  - Spurious alarm messages are sent via the nurse call system.
  - The alarm level cause the PCA pump to stop pumping.

  C: Algorithm improperly calculates the safety range too optimistically

  - Same effect as failure scenario A.

  D: Algorithm improperly calculates the safety range too conservatively

  - Same effect as failure scenario B.

  E: SpO_2 and Respiration Rate detected incorrectly

  - Condition: Failure of sensors or detachment of sensors
  - May lead to either failure scenario A or B.

3.3 Taking X-Ray while on Ventilator

Goal

Taking clear X-Ray images of a patient during who is connected to an anesthesia machine ventilator, at a specific phase of respiration.

Summary

A patient undergoing a surgical procedure is connected to a ventilator. The radiology technician uses the the X-Ray machine to take an image of the patient. The X-Ray machine is synchronized with the operation of the ventilator.

Rationale

"A 32-year-old woman had a laparoscopic cholecystectomy (gall bladder removal) performed under general anesthesia. At the surgeon's request, a plain film X-Ray was shot during a cholangiogram. The anesthesiologist stopped the ventilator for the film. The X-Ray technician was unable to remove the film because of its position beneath the table. The anesthesiologist attempted to help her, but found it difficult because the gears on the table had jammed. Finally, the X-Ray was removed, and the surgical procedure recommenced. At some point, the anesthesiologist glanced at the electrocardiography (EKG) and noticed severe bradycardia. He realized he had never restarted the ventilator. This patient ultimately expired."

Equipment

- Portable X-Ray machine
- Anesthesia machine ventilator
- Operating table
- The integrated system

Actors

- Patient
- Surgeon

- Radiology technician
- Anesthesiologist
- Additional OR staff

Pre-conditions

- The patient is connected to the anesthesia machine.
- The patient is under anesthesia.
- The anesthesia machine is connected to the integrated hospital information system.
- The system is aware of the patient's maximum safe pause time for respiration.

Post-conditions

- Success conditions
  - Clear X-Rays are taken of the patient in the desired phase of respiration.
  - The ventilators normal operation is: a) not interrupted at all, during synchronized use; or b) not interrupted beyond a set threshold.
  - The patient continues to be properly respirated.
- Failure conditions
  - Unclear X-Ray images, or no X-Ray images are taken.
  - X-Ray images are taken during wrong phase of respiration.
  - The ventilator stops or is interrupted too long.
  - The patient goes without respiration too long.

Triggers

- The surgeon requests the radiology technician take an X-Ray.

Scenarios

- A: Pure synchronized operation
  Conditions: The image exposure length is short enough to be synchronized with the ventilator.
  - The surgeon requests the radiology technician take an X-Ray.
  - The technician connects the portable X-Ray machine to the integrated hospital information system.
  - The machine communicates is operating specs including its exposure time and X-Ray tube latency with the system.
  - The technician moves it into positions.
  - The technician inputs when the image is to be captured (either inspiration or expiration).
  - The integrated system determines if there is sufficient time to obtain the X-Ray during the desired phase of the respiratory cycle.
  - All of the OR staff (except the technician) are instructed to leave the room.
  - The technician activates the X-Ray machine, and an image is automatically taken at the next occurance of the proper phase of respiration, and for the required exposure length.
  - The OR staff returns to the room.
  - The system notifies the anesthesiologist that respiration was continuous.
  - The surgical procedure continues.
- B: Interrupted operation
  Conditions: The image exposure length too long to be synchronized with the ventilator.
  Same as scenario A except once the technician activates the X-Ray machine:
  - The system pauses the anesthesia machine ventilator at either end-inspiration or end- expiration.
  - An X-Ray is taken for the required exposure length.

- The system resumes the anesthesia machine ventilator at the pre-image respiration rate.
- The X-Ray machine reports that it successfully took an image.
- The OR staff returns to the room.
- The system notifies the anesthesiologist that respiration was interrupted and has resumed.
- The surgical procedure continues.

- C: Interrupted operation with failure to photograph
  Conditions: The image exposure length too long to be synchronized with the ventilator.
  Same as scenario B except:
  - The system pauses the anesthesia machine ventilator at either end-inspiration or end- expiration.
  - For whatever reason an X-Ray fails to be taken.
  - The system resumes the anesthesia machine ventilator within a maximum safe pause time, at the pre-image respiration rate.
  - The system sends an alarm about the failure to produce and image.
  - The system notifies the anesthesiologist that respiration was interrupted and has resumed.
  - The surgical team decides how to proceed.

3.4 Transitions between Ventilator and Cardiopulmonary Bypass

Goal

Ensure the safe transitions between a ventilator and a cardiopulmonary bypass (CPB) machine.

Summary

A patient undergoing a surgical procedure is connected to a CPB and removed from a ventilator. At the end of the procedure the ventilator is resumed, and use of the CPB is discontinued. The integrated system manages the safe transitions from the ventilator to the CPB, and back. The system provides a smart alarm to warn the OR team if the CPB terminates and lung ventilation has not resumed.

Rationale

"Cardiac (heart) surgery, pulmonary (lung), and major vascular surgery may require the use of CPB. During CPB, the CPB machine takes over both the pumping function of the heart and the ventilation function of the lung. Therefore, during CPB, the anesthesia machine ventilator is not needed, and is turned off to prevent unnecessary ventilation-induced lung movement that may interfere with surgery. During this period, physiological respiratory and circulatory monitors may be turned off or their alarms inactivated to prevent nuisance alarms. At the conclusion of the CPB period, the heart resumes pumping blood, and the CPB machine pump is stopped. Lung ventilation must be resumed prior to discontinuation of CPB or non-oxygenated blood will circulate. The anaesthesia/surgical team must remember to resume ventilation and manually re-start the anaesthesia ventilator. Patient injuries and deaths occur when the team forgets to resume ventilation. This is a longstanding problem that continues to occur. Immediately following CPB, the heart and other major organs may be especially susceptible to injury from poorly oxygenated blood."

Equipment

- Anaesthesia workstation ventilator
- Cardiopulmonary bypass (CPB) machine
- Physiological monitors
- The integrated system

Actors

- Patient
- Surgeon
- Anesthesiologist
- A perfusion team

Pre-conditions

- The physiological monitors are connected to the integrated system.
- The anaesthesia workstation ventilator is connected to the integrated system.
- The physiological monitors are connected to the patient.
- The anaesthesia workstation ventilator is connected to the patient.
- The patient is under anesthesia.
- The system is aware of the maximum safe time for the patient to be without respiration support.

Post-conditions

- Success conditions
  - The patient's blood flow and oxygenation level stay within a predetermined safe range.
  - The OR staff are not distracted by nuisance alarms, or conversely numbed to valid alarms.
- Failure conditions
  - The patient dies, or is injured due to poorly oxygenated blood, organs or tissues.
  - The anaesthesiologist forgets to resume ventilation.

Triggers

- The surgeon determines that a CPB machine is necessary.

Scenarios

- A: Standard operation
  - The surgeon determines that a CPB machine is necessary.
  - The perfusion team sets up the CPB and connects it to the integrated system.
  - The CPB is connected to the patient.
  - After the system detects that CPB has begun, it waits to detect that the ventilator has been discontinued.
  - The system queries the anaesthesiologist about which alarms should be enabled.
  - Via a user interface, the anaesthesiologist enters whether or not alarms for the physiological respiratory and circulatory monitors should be surpressed, and whether a "smart ventilation" alarm should be provided.
  - At the end of the medical procedure, the surgeon stimulates the patient's heart to resume pumping blood.
  - The anaesthesiologist resumes manual ventilation.
  - The anaesthesiologist resumes the anaesthesia ventilator.
  - The anaesthesiologist instructs the perfusion team to turn off the CPB.
  - The system reenables the alarms for the physiological monitors.
  - The system disables the smart alarm 5 minutes after ventilation is detected.
- B: Detection failure
  Conditions: The system fails to detect that CPB has begun, or that the ventilator has stopped.
  - The system sets off an equipment detection failure alarm.

- C: Low CPB flow
  Conditions: The CPB flow decreases to less than 0.5 liters per minute for over 2 minutes.
  - The system sets off an smart ventilation alarm.
- D: CPB termination
  Conditions: The CPB prematurely terminates.
  - Same as scenario C.
- E: Ventilator has not resumed
  Conditions: The CPB has stopped and the ventilation has not been detected, or has occured for less than 5 minutes.
  - Same as scenario C.

CHAPTER 4

MEDICAL DEVICE INTERACTION CONFIGURATION LANGUAGE

Even the most advanced and accurate medical devices do not serve their purpose if the users cannot effectively use them. Worse, the complexity of configuration and usage may make users unwilling to use the devices. Clinicians must take full responsibility of any medical decisions and never allow devices to execute any processes without their knowledge.

These requirements prompted a need for a more usable and intuitive configuration language for defining safe device interactions. The configuration language presented in this work is a natural, English-like language that fulfills such requirements by making system configuration more accessible to clinicians, or to generalize, the non-programmers. In this section, the syntax and semantics of the safety rule configuration language is described in detail.

4.1 Syntax

As some of the use cases in the previous chapter show, some medical devices require exclusive usage while some may be used in unrestricted combinations. Computerized devices can aid clinicians by automating clinician-defined device interactions. For example, a device controller may *allow* activating the airway laser only after the ventilator has been *prohibited* for an adequate period of time, in order to avoid risk of fire. Similarly, if a patient's blood oxygen saturation level (saturation of peripheral oxygen, SpO2, measured by a pulse oximeter) is below a certain threshold, ventilator can be automatically *enforced*. The device interaction configuration language, with an English-like syntax yet much smaller in scope, provides an easier way of specifying such interactions. Figure 4.1 describes its grammar:

⟨configuration⟩ ::= ⟨rule⟩+
⟨rule⟩ ::= ⟨actionSatement⟩ ⟨ifRule⟩ ⟨conditionStatement⟩
⟨actionStatement⟩ ::= ⟨actions⟩ ⟨device⟩
⟨actions⟩ ::= ⟨action⟩ **or** ⟨action⟩ | ⟨action⟩
⟨action⟩ ::= **prohibit** | **allow** | **enforce**
⟨ifRule⟩ ::= **if** | **only if**
⟨device⟩ ::= ⟨ID⟩ . ⟨ID⟩
⟨conditionStatement⟩ ::= ⟨deviceCondition⟩ | ⟨patientCondition⟩
⟨deviceCondition⟩ ::= ⟨device⟩ is ⟨actionPast⟩ ⟨timeLimit⟩
⟨actionPast⟩ ::= **prohibited** | **allowed** | **enforced**
⟨timeLimit⟩ ::= ε | **for** ⟨NUMBER⟩ ⟨timeUnit⟩
⟨timeUnit⟩ ::= **seconds** | **minutes** | **cycles** | ...
⟨patientCondition⟩ ::= ⟨device⟩ **is** ⟨binaryCondition⟩ ⟨NUMBER⟩ ⟨valueUnit⟩
⟨binaryCondition⟩ ::= **less than** | **<** | **less than or equal to** | **<=** | **equal to** | **=** | **greater than** | **>** | **greater than or equal to** | **>=** | **not equal to** | **<>**
⟨valueUnit⟩ ::= **ml** | **%** | ...
⟨NUMBER⟩ ::= ⟨DIGIT⟩+
⟨DIGIT⟩ ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
⟨ID⟩ ::= (⟨CHAR⟩ | ⟨DIGIT⟩)+
⟨CHAR⟩::=(**a-z** | **A-Z** | **_** | **-**)

Figure 4.1: Configuration language grammar.

The following is a detailed description of each syntactic components.

- **rule** Each rule describes activation condition for a single device depending on a status of another device. A device may depend on itself; for example, a PCA may be prohibited for a certain period of time since it was allowed to prevent patient overdose.

- **actionStatement** A rule can specify single or multiple actions for a device to take depending on a condition. More specifically, a user can "prohibit or allow", or "allow or enforce" a device. It is also possible to "prohibit or enforce", but it is essentially the same as "allow".

- **ifRule** There are two kinds of ifRule, "if " and "only if". Given "A if B", B implies A. In contrast, with "A only if B", A implies B. For example, if a rule is defined"enforce ventilator.oxygenpump *if* ventailtor.oximeter is less than 90%", this implies that when the oximeter's reading is less than 90%, the oxygen pump is always enforced. However the reverse implication does not hold. Similarly, given a rule "allow airwaylaser.laser *only if* ventilator.oxygenpump is prohibited for 5 seconds", it implies that when the laser is allowed the oxygen pump has been prohibited for at least 5 seconds, while the reverse implication is

not always true.

- **device** A device may consist of multiple actuator and sensor functionalities. In order to distinguish such sub-device functionalities, a device in the language is identified as "device.function", for example, ventilator.pump and ventilator.oximeter.

- **conditionStatement** A rule may specify one of two types of condition statements; deviceCondition describes a condition on the activation status of other actuator, and patientCondition describes a condition on the data reported by a sensor.

- **deviceCondition** A deviceCondition describes the activation status of an actuator that another actuator depends on. It may specify the length of time that a device has been in the said status, where the time can be measured in seconds, minutes, cycles or any other relevant units.

- **patientCondition** A patientCondition essentially describes binary conditions on sensors, or more specifically, the data reported by them. The data may be measured in percent, milliliters, or any other relevant units.

4.2 Language Design and Examples

Each rule describes restrictions placed on the behaviour of one device by the status of another device. A device may be an actuator or a sensor. An actuator performs an action based on an input signal, whereas a sensor only detects and reports external signals which can be read by an observer or other devices. For example, a ventilator is an actuator that facilitates a patient's breathing by supplying fresh air, while a pulse oximeter is a sensor that measures the oxygen saturation of a patient's blood. Furthermore, a single device can have multiple sensor and actuator functionalities. It is common for a medical ventilator to also include a pulse oximeter. A rule may specify that an actuator's activation status is dependent on another actuator or a sensor, but a sensor does not depend on any other devices.

The rule allows for three actions for automated control of devices; *prohibit*, *allow* and *enforce*. These actions refer to activation of each device, where *prohibiting* a device prevents clinicians from accidentally turning it on when they should not, and *enforcing* a device prevents accidental deactivation of the device by clinicians. *Allowing* a device gives the control of the device to the clinicians. Also, it is assumed that sensors are always enforced while an actuator may change its activation state throughout a procedure.

A rule specifies an actuator to be prohibited, allowed or enforced depending on either 1) the activation status of other devices or 2) a patient's condition. In the former case, the device's activation depends on other actuators but not sensors (as sensors are always enforced), and the length of time each of them in certain status. This type of conditions will be referred to as a "device condition" for brevity. In the latter, the device changes its activation status conditionally on the evaluation of readings reported by sensors. This type of conditions will be referred to as a "patient condition".

The following are safety rules derived from each of the use cases in chapter 3. With the natural English-like syntax, it is trivial to encode a use case in the safety rules even though there is no stepwise method of deriving a set of safety rules from a use case.

- **AIRWAY LASER SURGERY**
  1. allow airwaylaser.laser only if ventilator.oxygenpump is prohibited for 5 seconds
  2. allow or enforce ventilator.oxygenpump only if airwaylaser.laser is prohibited for 2 seconds
  3. enforce ventilator.oxygenpump if ventilator.oximeter is less than 90%

- **MORPHINE ADMINISTRATION**
  1. allow pca.pump if pca.pump is prohibited for 10 minutes
  2. prohibit pca.pump if ventilator.oximeter is lower than 90%

- **XRAY WHILE VENTILATOR IS IN USE**
  1. allow xray.xray if ventilator is prohibited for 1 second

2. allow ventilator.oxygenpump if xray.xray is prohibited
3. enforce ventilator.oxygenpump if ventilator.oximeter is less than 90%

- **CPB**
  1. prohibit ventilator if CPB is enforced
  2. enforce ventilator if CPB is prohibited
  3. enforce ventilator if spo2 is lower than 90%

The English-like configuration language not only provide a syntax that is very similar to English, but also offers the familar semantics, except for a more rigorous logical evaluation of the ifRule component (refer to previous section). Consider the first rule for the airway laser surgery use case. The rule specifies that no airwaylaser.laser should be allowed if a ventilator.oxygenpump is not prohibited or prohibited for less than 5 seconds. In other words, if an airwaylaser.laser is allowed, it should always infer that the ventilator.oxygenpump is prohibited for minimum of 5 seconds. Due to the definition of the ifRule component "only if", however, the rule does not specify that if the ventilator.oxygenpump is prohibited for 5 seconds, the airwaylaser.laser must be allowed. Similarly, the third rule for the airway laser surgery use case defines that if the ventilator.oximeter's reading is less than 90%, the ventilator.oxygenpump must always be enforced. This however does not mean that the ventilator.oximeter's reading is less than 90% whenever the ventilator.oxygenpump is enforced. The both-way inference would require "if and only if" for its ifRule component, which is not included in the language as it is infeasible to require multiple medical devices to change their activation status perfectly synchronously, neither by clinicians or computers. The semantics of the safe device interactions rules can also be defined more rigorously using a mathematical model - chapter 5 provides a more formal semantics of the rules using a finite state machine.

CHAPTER 5

CONFIGURATION LANGUAGE SEMANTICS WITH FINITE STATE MACHINE

The language presented in this work is a *configuration language* and thus traditional kinds of semantics for programming languages do not adequately describe the language. Each rule describes safe actions of a device given the current status of another (or the same) device. That is, there are no statements or expressions that change the current state of the system. Given this characteristic of the language, a finite state machine is a natural representation of the semantics of the language, where each rule describes a safe transition among all possible transitions, and a state is a set of all devices with their respective statuses.

5.1 Finite State Machine

A finite state machine (FSM) is a mathematical abstraction of a system. It is a behavior model with a finite number of states and transitions between them, which allows deterministic specification of the system. More specifically, a finite number of states implicitly stores the information about what has happend before, and transitions represent the system's response to current input, which depends on the current state of the macine. A FSM is defined as a quintuple (Σ, S, s0, δ, F) where:

- Σ is a finite non-empty set of input symbols.
- S is a finite, non-empty set of states.
- s0 is an initial state where s0 $\in$ S.
- δ is the state-transition function: δ: S $\times$ Σ $\rightarrow$ S.
- F is the set of final states, a (possibly empty) subset of S.

In this work, a FSM is used to represent a medical environment consisting of multiple devices, where the devices react to one another, the surgeon's commands, the time elapsed, or to a patient's conditions. Therefore, a **state** is a vector of device states, where a device is either an actuator or a sensor. An actuator state is <status, time>, where status is the actuator's current activation status – prohibit, allow or enforce – and time marks the length of time for which the actuator has been in current activation status. A sensor state is <reading> which is the value that the sensor reports, since sensors are always enforced. A **transition** describes events which are time ticks, changing of a actuator status, and changes in sensor readings. Thus the FSM represntation of a medical environment is defined as:

- Σ: The time ticks, t, the changes in actuator status, a(device, newStatus), and changes in sensor readings, s(newSensorValue).

- S: Every state shares the same vector of devices (however not the device states). That is, each of states s, s' $\in$ S is a vector D of devices. For each device d $\in$ D, if d is an actuator, it has two attributes <status, time> where the status $\in$ {'prohibit', 'allow', 'enforce'} and time $\in$ [0 ... $\max_d$], where the $\max_d$ is defined by users. If d is a sensor, it has one attribute <reading> which is a value partition defined by users .

  Assuming there are functions time(state, device), status(state, device) and reading(state, device) that return the timer value of the actuator, actuator status, and readings of a sensor in a state, respectively, then s and s' are equal if for every d $\in$ D time(s, d) = time(s', d) and status(s, d) = status(s', d) and reading(s, d) = reading(s', d). The state generation is detailed in section 5.2.

- s0: A state with all actuators prohibited for their respective maximum duration, and every sensor reporting safe values which are deduced from the user-defined rules.

- δ: The time tick transition increments the timer value of each device by one, or loops it back to the maximum value if it is at its maximum. That is, s $\times$ t $\rightarrow$ s' where for every device d $\in$ D:

$$time(s', d) = \begin{cases} time(s, d) + 1 & \text{if time(s, d) } \max_d \\ time(s, d) & \text{if time(s, d) } = \max_d, \end{cases}$$

which results in a new state.

On the other hand, the actuator status transition changes the status of one actuator, d, and resets its timer value to 0. That is, s $\times$ a(d, u) $\rightarrow$ s' where status(s', d) = u and u $\in$ {'prohibit', 'allow', 'enforce'} - status(s, d), and time(s', d) = 0. Lastly, sensor value transitions changes the reading attribute of a sensor. Section 5.3 details the transitions and their generation.

- F: The set of final states is the collection of *safe* states that are deduced from the user-defined rules, described in section 6.2.

5.2 State Generation

The states in the FSM are generated based only on the user-defined rules. The $\max_d$ value of a device d is calculated as follows. With the safety rules, there is a maximum length of time per status that a device needs to keep track of. For example, given devices d, d_A, $d_B \in$ D and the rules

- allow d_A if d is prohibited for 2 seconds
- enforce d_B if d is prohibited for 4 seconds,

the device d only needs to keep track of its timer up to 4 seconds when it is prohibited. Any further time ticks on d will simply loop back to 4. This upper limit on the timer value bounds the state space to be finite, thus eliminating non-determinism. If no time limit is defined for a device in a status, its maximum will be 0.

The similar approach applies to sensors and their values. Instead of creating a state for every possible value that a sensor may read, which would significantly increase the state space, it is sufficient to have only two states: one where the reading is the literal translation of a rule, and the other that negates the condition. For example, given the following rule:

- allow d_A only if d is greater than 90%

then it will generate two states, a state where the sensor d's value is greater than 90%, and the other that is less than or equal to 90%. Furthermore, if two devices depend on different threshold values of a sensor, for example,

- allow d_A if d is less than 40%
- allow d_B if d is greater than 80%,

then the resulting states for the sensor d are the partitions of its values: $<$ 40%, $\geq$ 40% and $\leq$ 80%, and $>$ 80%.

With the three rules for the airway laser surgery use case, the tool generates aa FSM with 80 states where a state consists of a ventilator.oxygenpump, a ventilator.oximeter and an airwaylaser.laser. The ventilator.oxygenpump has 8 different states: 6 states for when it is prohibited with the timer value ranging from 0 to 5, 1 state each when it is allowed or enforced since there is no time constraint for these statuses. Similarly, the airwaylaser.laser has 5 different states in total; 3 for when it is prohibited, and 1 each for when it is allowed or enforced. Lastly, the ventilator.oximeter sensor can be either less than 90% or greater than or equal to 90%, resulting in 2 different states. The combination of all these produces $8 \times 5 \times 2 = 80$ states.

With the complete state machine, there must be one start state for it to be deterministic. In order to properly model a medical procedure, the tool designates as the start state the state where every actuator prohibited for their respective maximum length of time and all sensors reading safe values . If no rule is defined for a device, it is assumed that the device has no siginificant impact on the overall system and thus is not included in the resulting FSM. For the airway laser surgery, the start state has the ventilatr.oxygenpump prohibited for 5 seconds, the airwaylaser.laser prohibited for 2 seconds, and the ventilator.oximeter reading is greater than or equal to 90%.

5.3 State Transition Generation

There are three types of transitions, the time tick, the changes in activation status of a device, and the changes in sensor values.

1. Time Tick
   To properly model the time lapses in a medical environment, the time tick event transitions a state to another where all the devices have their

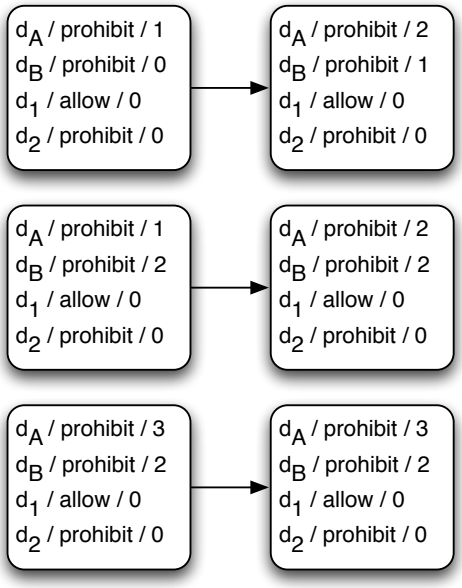

Figure 5.1: Valid time tick transitions

timer incremented by one, or at their respective maximum timer value. For example, given the rules

- allow d_1 if d_A is prohibited for 3 seconds
- prohibit d_2 if d_B is allowed for 2 seconds,

all the transitions shown in figure 5.1 are valid time tick transitions.

2. Activation status change of an actuator
   Another type of transition is the device activation transition. For simplicity, it is assumed that only one device may be prohibited, allowed or enforced at a time. For example, given a state with two actuators, changing the status of one actuator causes the transition to another state where the actuator has a new status with timer value 0 and the other with its timer value incremented by one or at its maximum value.

3. Sensor value changes
   The sensor value changes can happen at any state, as sensor values represent patient conditions which is indeed random and non-deterministic. For example, consider a state with one sensor (d_S) and an actuator (d_A), and that the sensor values are partitioned into three groups as described in the previous section ($< 40\%$, $\geq 40\%$ and $\leq 80\%$,

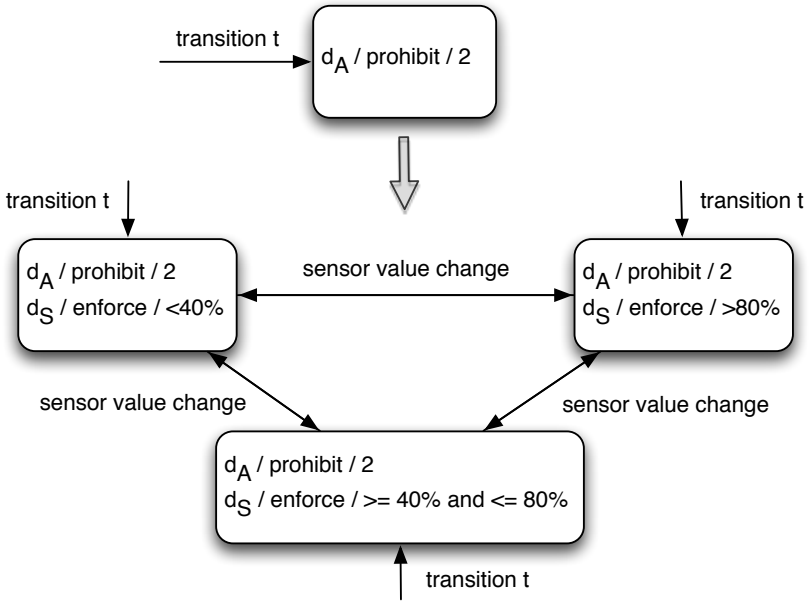

Figure 5.2: Valid sensor value changes and replicated transitions.

and $> 80\%$). Then for every state, there are three copies of each state with different value partition of the sensor and any transition between those states are allowed. Also, any transition to a state with one value partition of a sensor is also replicated for other states with different value partitions of the sensor (figure 5.2).

Any missing status of a device in safety rules is excluded from the FSM. For example, the safety rules for airway laser surgery defines no rule for enforcing the airwaylaser.laser and therefore there are no transitions to states where the airwaylaser.laser is enforced in the FSM. This is to prevent error by omission by not allowing "default" transitions that a user may not be aware of.

CHAPTER 6

CONSTRUCTING A SAFE SYSTEM

Providing clinicians with a more intuitive way to define safety rules for a medical procedure is useful, but only to an extent if the benefit stops at the ease of configuration. Proving that the safety rules defined by uesrs are sound and indeed safe is vital in any safety-critical system. Therefore an automatic checker providing mathematical proof of safety was also developed alongside the language. This chapter describes how the safety of the rules are formally proven with a FSM.

6.1 Open-Loop Safety

In a clinical environment where multiple devices are interconnected, the system needs to be fault-tolerant in case of network, system or device failures, that is, be open-loop safe. With medical devices being dependent on one another's activation status or readings, any failure that breaks the communication channel can have serious, even fatal, impact on patients. Therefore each device needs to be able to assume a *safe action* when there is a failure. A safe action ensures that the safe device interaction rules are satisfied. For example, NASS handles communication failures by providing synchronized plans for each device. These plans allow for gradual changes of action in each device to bring the entire system to safe states, to ensures that no device changes its activation status abrubtly which may violate some of the safety rules.

Given a set of user-defined safe device interaction rules, how can it be proven that these rules do not conflict with one another such that there is no safe state? That is, how can the set of rules be proven that it is open-loop safe? In order to provide a formal proof of safety, an automatic checker gen-

erates a complete finite state machine based on the user-defined rules, and transitions to unsafe states, which are deduced from the rules, are removed. With the resulting finite state machine, the tool checks if the safe states, also derived from the rules, are reachable from each and every state.

6.2 Application of Rules and Proving Safety

Given a complete FSM, rules are applied by removing transitions to unsafe states. Before the user-defined rules are applied, the complete state machine assumes that all the states and device status transitions are safe. For example, all the states in Figure 6.1, a subset of the complete FSM for the airway laser surgery (the starting state is not marked), are deemed safe, even though those highlighted in red are in fact unsafe according to the rules defined.

Essentially, the initial FSM allows for the device status transitions for each and every device in a state, as well as the time tick transitions. That is, the initial FSM models a medical system with no safety rules defined. This is a paradox because the initial FSM is generated based on the safety rules defined, but the first scan of the rules only extracts the data needed for the generation of a complete FSM, such as the devices involved and their actions and time limits. The conditions specified in rules are parsed and analysed only after the initial FSM is generated.

Each rule specifies two conditions, (A) a device's new status and (B) when it should happen, and a conditional phrase, "if" or "only if". Using the logical definition of the conditional phrase:

- "A if B" is evaluated to the boolean logic "A or not B".
- "A only if B" is evaluated to the boolean logic "not A or B".

The rules are therefore applied by removing transitions from a state to a new state that does not satisfy the boolean logic. For example, recall the rules defined for the airway laser surgery use case:

1. allow airwaylaser.laser *only if* ventilator.oxygenpump is prohibited for 5 seconds

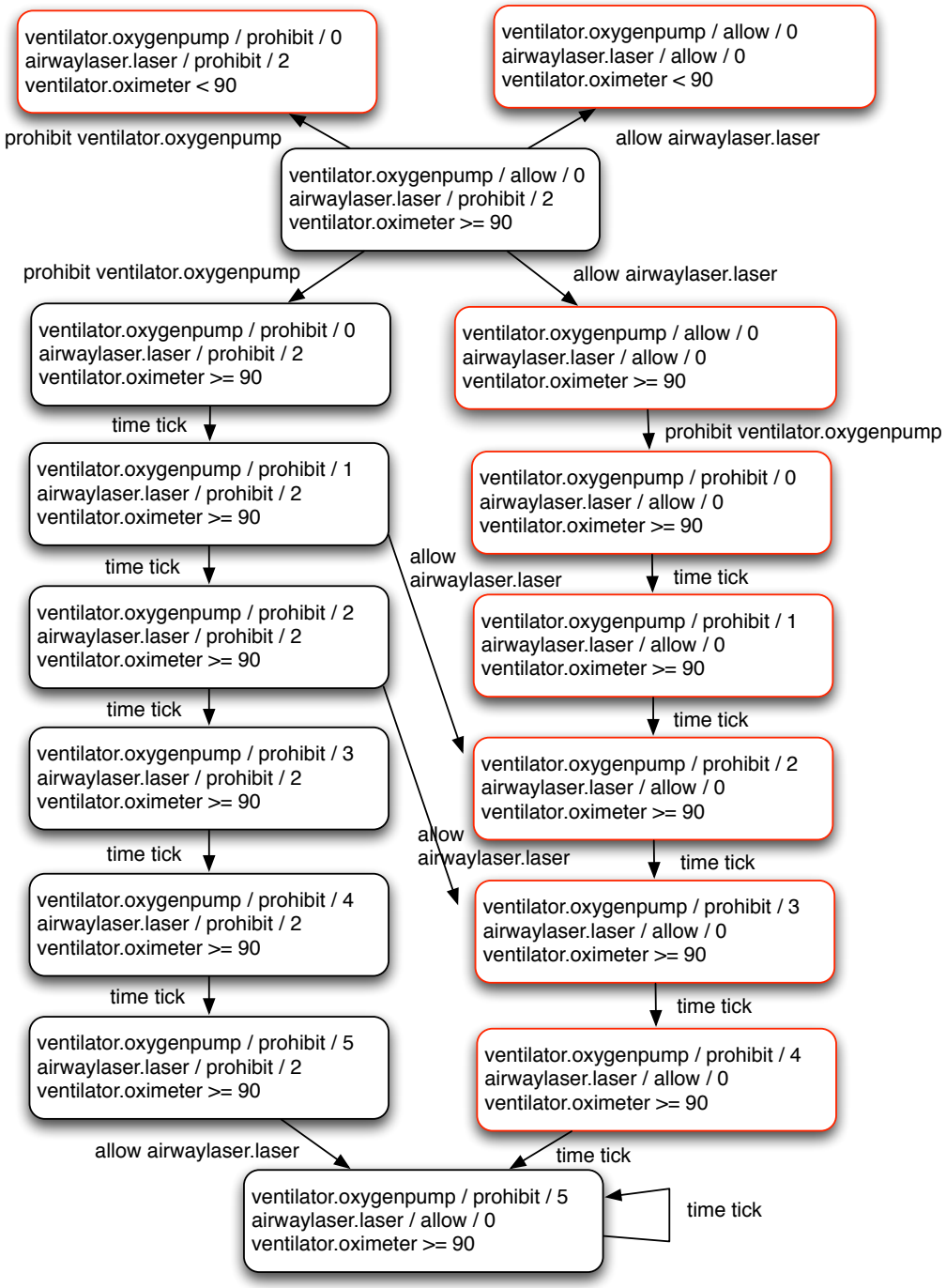

Figure 6.1: Subset of a complete FSM for the airway laser surgery.

2. allow or enforce ventilator.oxygenpump *only if* airwaylaser.laser is prohibited for 2 seconds

3. enforce ventilator.oxygenpump *if* ventilator.oximeter is less than 90%

According to the logical definitions above, these rules translate to:

1. airwaylaser.laser is not in allow status *or* ventilator.oxygenpump is prohibited for 5 seconds

2. ventilator.oxygenpump is in prohibit status *or* airwaylaser.laser is prohibited for 2 seconds

3. ventilator.oxygenpump is in enforce status *or* ventilator.oximeter is greater than or equal to 90%

In Figure 6.1, those states in black satisfy all of the conditions above, and thus determined to be safe. The states highlighted in red are those that do not satisfy one or more of the conditions and, therefore they are deemed unsafe. Iterating through every state, any transitions to unsafe states are removed, as shown in Figure 6.2. By removing transitions to unsafe states by construction, it is possible to prove that the FSM is *not unsafe*.

With the unsafe states excluded from the resulting FSM, it is important to prove that the FSM *is safe*, by checking that safe states are reachable from each and every state. The safe states are deduced from the written rules as follows:

1. For rules defining device conditions, "A if/only if B" is converted to "A and B".

2. Rules defining patient conditions specify what action a device need to take when a patient is in an undesirable condition. The checker takes a pessimistic approach by assuming the patient's condition is worsened, and thus "A if/only if B" is converted to "A".

3. Patient conditions take higher priority than device conditions.

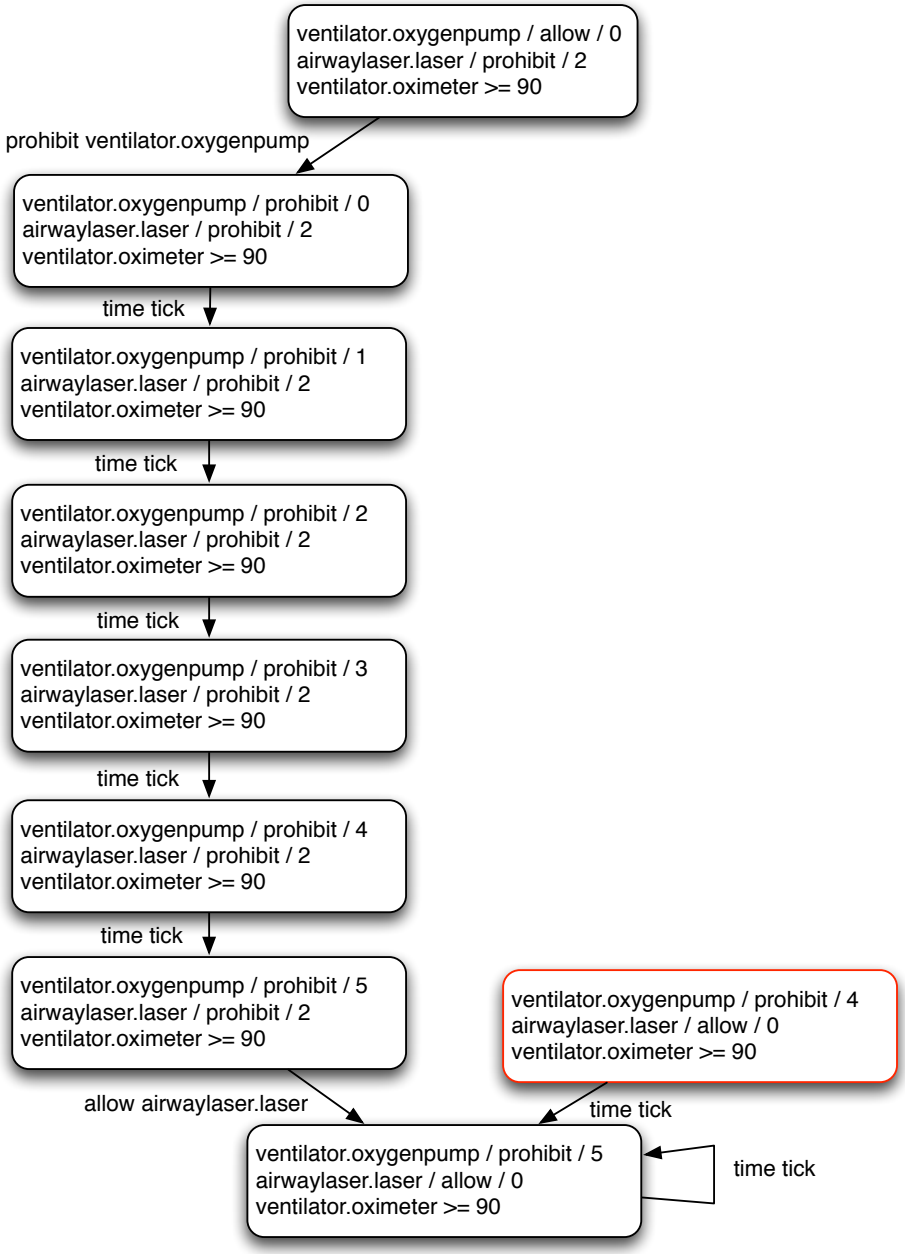

Figure 6.2: Subset of a complete FSM for the airway laser surgery, with transitions to unsafe states removed.

For example, given the airway laser surgery rules, the checker *assumes* that the ventilator.oximeter is less thant 90% and so the safe state has the ventilator.oxygenpump enforced. This in turn implies that the airwaylaser.laser must be prohibited for at least 5 seconds. Figure 6.3 shows the final FSM for the airway laser surgery use case. The start state is the left-most state on the top, and the two states on the bottom to the left-hand-side are the safe states.

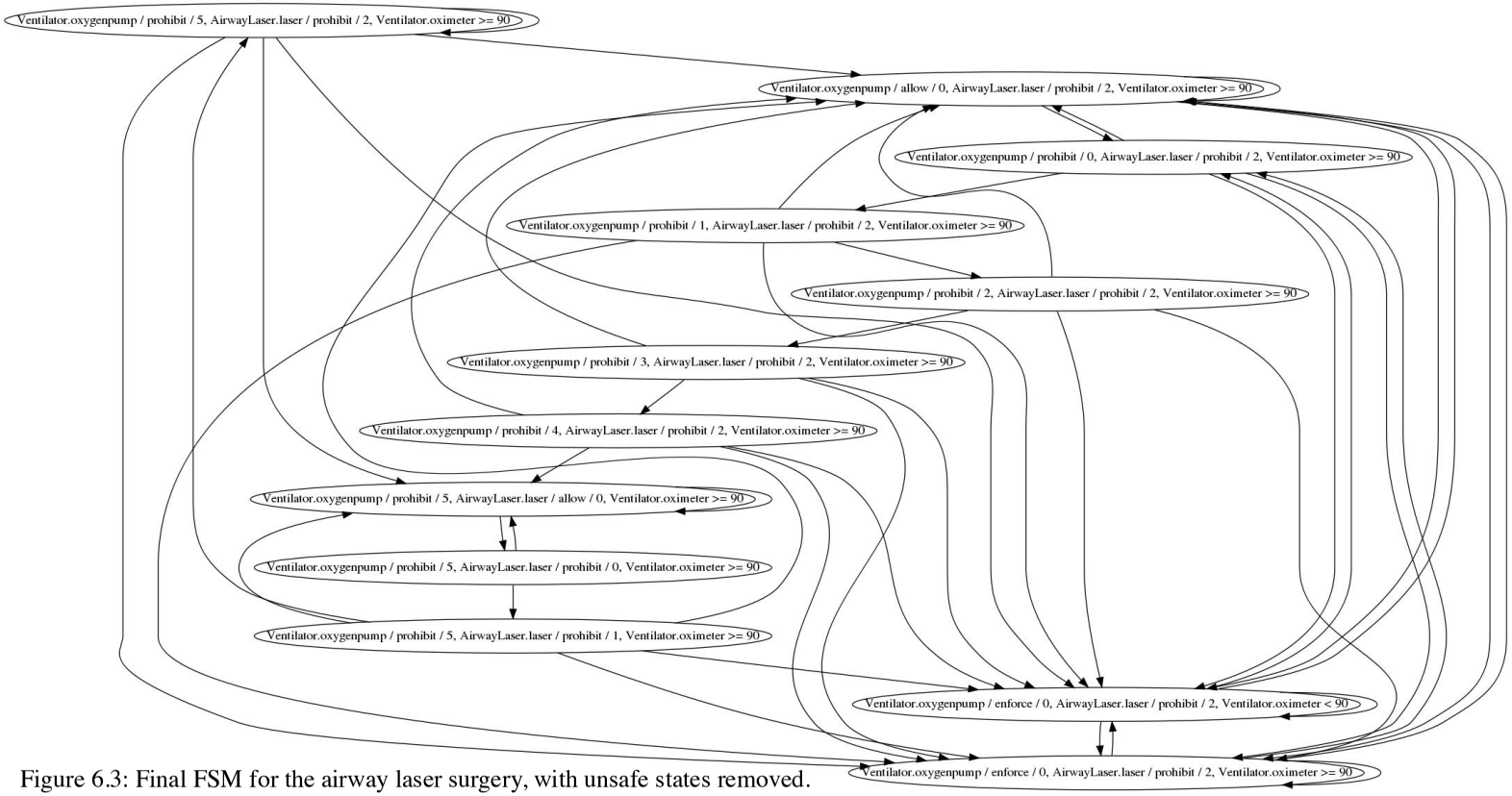

Figure 6.3: Final FSM for the airway laser surgery, with unsafe states removed.

CHAPTER 7

FUTURE WORK & CONCLUSIONS

7.1 Extended Configurations

While providing an easier way for a clinician to safely configure device interactions gives clinicians some control over a medical system, it is only a small part of the complexity. For example, some medical devices are usually used with one another; cautious clinicians may only activate a PCA when a pulse oximeter is activated, or start a procedure only when all the devices are connected. The configuration language can be extended to allow specification of device connections as well as interactions:

- start infusionpump.PCA only if ventilator.oximeter is allowed.
- start procedure only if ventilator.oxygenpump and ventilator.oximeter and airwaylaser.lsaer is started.

Given such connection rules, the FSM can incorporate them by including transitions with device connection actions and states with only the devices that are connected. A state with an empty device list will be the initial state.

One of the most critical challenges for clinicians in a medical environment, whether it be an operating room or an intensive-care unit, is to navigate through a numerous - sometimes sperfluous - alarms and warnings to ensure patients' safety. Each device has its own alarm conditions and multiple devices compounded can exacerbate the problem. The *alarm fatigue* experienced by nurses and clinicians is a serious problem sometimes with fatal consequences [9]. In order to reduce the superfluous alarms, the configuration language may be extended to allow clinicians to configure when an alarm should sound, consolidating many alarms from all the devices into one. For example, alarm rules may be defined as:

- start alarm if ventilator.oxygenpump is enforced and ventilator.oximeter is less than 80 %.
- stop alarm if ventilator.oximeter is greater than 90 %.

The FSM may extend such configuration by including an alarm flag in every state.

Lastly, even though the configuration language resembles English, it is still restrictive in terms of what tokens the lexer can accept. Specifically, it is infeasible to include every possible unit for time and value (e.g. minutes, cycles, ml, beats-per-minute). It may be beneficial to provide clinicians with a graphical use interface (GUI) to help rule specifications. The GUI may allow for easier extension of the grammar by providing users with options to add tokens prior to lexing and parsing the rules.

7.2 Formal Proof

In this work, a FSM is used to define the operational semantics of the configuration language, which serves the purpose of checking the safety of user-defined rules in terms of reachability of safe states. However, the state space of the resulting FSM tend to be very large, with the number of states increasing exponentially with respect to the number of devices. With such a large FSM, testing the correctness of its generation and manipulation can become difficult, failure of which renders its use for safety checking a moot purpose.

As the language is extended to include more configuration options, more sophisticated model and logic checkers may be used to formally prove the safety. For example, Real-Time Maude, a temporal logic checking tool supports the formal specification and analysis of real-time systems [10]. Such a tool may model the medical system where the time component plays a critical role. Anoter Maude-based tool, K [11], may also be used to define the operational semantics. K allows specification of parallelism which may more accurately describe the changes in device activation status.

Such formal checkers may also be used to simulate a medical system defined with the safety rules. For example, the Real-Time Maude allows users to execute a definition using rewrite logic, simulating the behaviour in a given time period. This may be useful for sanity check of the user-defined rules with certain time values. The FSM approach used in this work may also be extended with simulation capability. By searching the FSM randomly or exhaustively, users will be able to get a direct feedback on specific actions which can help in rule specifications.

7.3 Conclusions

With highly advanced and specialized medical devices, any given clinical environment is a composition of complex computer systems, with exponential increase in possible device interactions with respect to the number of devices. Even though computerized medical devices may provide better accuracy and safety, their complexity in configuration and use may deter the clinicians from using them. It is vital that clinicians use the devices *correctly*. By providing clinicians with a natural English-like configuration language, they can effectively and safely configure the complex device interactions, and the system operates based on their expertise and not on programmatical heuristics which can enanger patients' safety.

The automatic checking tool that formally proves the safety of user-defined rules also helps ensuring patients' well-being. Using a FSM, the tool checks if safe states are reachable from each and every state, thus proving that the system can be restored to a safe state in case of network, device or system failures.

REFERENCES

[1] "Surgical instruments from ancient rome." [Online]. Available: http://www.hsl.virginia.edu/historical/artifacts/antiqua/instruments.cfm

[2] A. Dick, "Laser safety in the operating room." [Online]. Available: http://clinicaldepartments.musc.edu/anesthesia/intranet/education/resident research/files/Andy Dick1.pdf

[3] F. S. E. Wegrzynowicz, N. Jensen, "Airway fire during jet ventilation for laser excision of vocal cord papillomata." [Online]. Available: http://www.anesthesiologyboards.com/pdfs/laserfire.pdf

[4] C. T. N. Leveson, "An investigation of the therac-25 accidents."

[5] A. S. for Testing and Materials, "Medical devices and medical systems – essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ice) part 1: General requirements and conceptual architecture." [Online]. Available: https://agora.cs.illinois.edu/download/attachments/23429568/ASTM +F29.21+draft+ICE+Part+I+N21.pdf

[6] ISO/IEEE, "Iso/ieee health informatics - point-of-care medical device communication - part 10201: Domain information model." [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1389297tag=1

[7] D. o. C. S. University of Illinois at Urbana-Champaign, "Medical device plug and play - safe composition of complex medical systems." [Online]. Available: https://agora.cs.illinois.edu/display/mdpnp/Home

[8] C. Kim, M. Sun, S. Mohan, H. Yun, L. Sha, and T. F. Abdelzaher, "A framework for the safe interoperability of medical devices in the presence of network failures," in *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ser. ICCPS '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1795194.1795215 pp. 149–158.
.

[9] L. Kowalczyk, "'alarm fatigue' linked to patient's death." 2010. [Online]. Available: http://www.boston.com/news/local/massachusetts/articles/2010/04/03/alarm_fatigue_linked_to_heart_patients_death_at_mass_general/

[10] P. Ölveczky, "Real-time maude 2.3 manual." 2007. [Online]. Available: http://www.ifi.uio.no/RealTimeMaude/intro.pdf

[11] T. Serbanuta and G. Rosu, "K-maude: A rewriting based tool for semantics of programming languages," in *Rewriting Logic and Its Applications*, ser. Lecture Notes in Computer Science, P. Ölveczky, Ed. Springer Berlin / Heidelberg, 2010, vol. 6381, pp. 104–122, 10.1007/978-3-642-16310-4_8. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16310-4_8