A Formal Approach for Detecting

Masking in Medical Alarms

by

Bassam Hasanain

B.S., University of Toledo, 2008

M.S., University of Dayton, 2009

Thesis

submitted as partial fulfillment of the requirements

for the degree of Doctor of Philosophy in Industrial Engineering and Operation Research

in the Graduate College of the

University of Illinois at Chicago, 2016

Chicago, Illinois

Defense Committee:

David He, Chair

Matthew L. Bolton, Advisor

Andrew D. Boyd, Biomedical and Health Information Sciences

Michael A. Brown

Prashant Banerjee

This dissertation is dedicated to my parents, my wife, and the rest of my family for all their support during those long years. Thank you for always believing in me and telling me that I can do it. I am truly blessed to have such a family.

ACKNOWLEDGEMENT

This work would not have been possible to accomplish without the continues and valuable support I had from my Dissertation Advisor Dr. Matthew L. Bolton. I would like to thank Dr. Boloton for believing in me as a researcher, his generous help over the years helped me kept going. Dr. Bolton is one of the few mentors who really cares about his student during their studies and even after they graduate, and continuously try to help in anyway he can. Working with him was an absolute pleasure.

I would also like to acknowledge my committee members Dr. David He, Dr. Andrew D. Boyd, Dr. Michael A. Brown, Dr. Prashant Banerjee, for taking the time to serve on my committee. Also for the many valuable discussions that helped me improved my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

dB	Decibels
IEC	International Electrotechnical Commission
MPEG	Moving Picture Experts Group
SAL	Symbolic Analysis Laboratory

SUMMARY

The ability of people to hear and respond to auditory medical alarms is critical to the health and safety of patients. Unfortunately, concurrently sounding alarms can perceptually interact in ways that mask one or more of them: make them impossible to hear [37, 38]. Masking may only occur in extremely specific and/or rare situations. Thus, experimental evaluation techniques are insufficient for detecting masking in all of the potential alarm configurations used in medicine. Therefore, there is a real need for computational methods capable of determining if masking exists in medical alarm configurations before they are deployed [37, 38]. In this work, we present such a method. Using a combination of formal modeling, psychoacoustic modeling, temporal logic specification, and model checking, our method is able to investigate and prove whether a configuration of concurrently sounding medical alarms can interact in ways that produce masking [37, 38]. This dissertation motivates and presents this method. Specifically, we describe how the method was developed over two steps. In the first, we developed the method to account for the masking of pairs of alarms in a configuration of multiple alarms. In the second, we included the ability to account for the additive effect of multiple potential masking sounds. For both implementations, we present case study applications that illustrate the predictive power of the method. The results associated with each stage of the method are discussed. Ultimately, general conclusions about the method are discussed and directions for future research are explored.

CHAPTER 1: INTRODUCTION

Medical alarms (which are usually auditory) are used by automation to notify human observers that monitored patient health measures have passed a threshold, indicating a potentially unsafe condition that requires immediate attention. Clearly, the ability of humans to perceive, understand, and respond to alarms is critical to patient safety.

Unfortunately, there are many limitations of modern medical alarm systems [28]. Significant numbers of false alarms can desensitize humans to alarms (a condition known as alarm fatigue); alarms can be poorly designed, reducing their effectiveness [28]; and concurrently sounding alarms can perceptually interact in ways that make them difficult to identify [47] or mask each other (make one or more of them imperceptible) [34]. It is important to note that all of the above problems are also relevant to alarms in other domains. However, as we will see in subsequent chapters, these problems are particularly acute in medicine due to the sheer number of alarms in use. Unfortunately, problems caused by the masking of concurrently sounding alarms can be very difficult to identify because they may occur under rare or unusually conditions or through the interaction of particular alarms within or between alarming systems. Thus, while auditory masking has been experimentally detected in clinical settings [51, 65], the vast majority of the work has focused on other problem areas. Thus, there is a very real and urgent need for methods capable of identifying if masking is present in medical alarm configurations before they are used in medical practice.

In this work, we describe a method we developed that is capable of doing such analyses. Our method makes use of two computational analysis techniques: model checking and psychoacoustic modeling. Model checking is predominantly used in the analysis of computer software and hardware, it is an analysis tool widely used in the analysis of safety critical computer systems. Psychoacoustic models are capable of mathematically indicating if concurrently sounding alarms might interact in ways that could produce masking [13, 16, 34]. When used together in our method, these techniques can allow health care providers to computationally determine if masking exists in a configuration of alarms. With such a detection capability, health care providers should be able to deploy systems guaranteed to avoid masking, ensuring that medical personnel will always hear a sounding alarm, enabling them to properly respond, and potentially saving patients' lives.

Our method was developed over two phases. The first developed the initial version of the method that was capable of detecting simultaneous masking between pairs of alarms in a configuration of multiple alarms. This development was the first work to ever account for psychoacoustic concepts in form verification analyses. It also demonstrated that masking detection was possible with model checking. The second development phase extended the method to be able to account for the additive effect of masking. This effort improved the masking detection capabilities of the method while simultaneously improving its usability and its ability to scale.

This dissertation describes our method and illustrates its utility. The next chapter is a review of the the literature relevant to understanding our method and uses it to motivate our research. The third chapter describes the first version of our method along with an application that demonstrates its predictive power. Chapter 4 discusses the second version of the method and explains how it was realized. Chapter 4 also contains several case study applications that demonstrate the improvements to the method as well as its ability to detect masking in a realistic configuration of alarms. We ultimately discuss our results, our research contributions, and implications for future work.

CHAPTER 2: REVIEW OF THE RELEVANT LITERATURE

This chapter reviews the literature relevant to understanding our method. This includes a discussion of masking in medical alarms, psychoacoustic models of masking, and model checking. We ultimately use this discussion to motivate our research.

2.1 Concurrently Sounding Medical Alarms

Auditory medical alarms have a number of problems [29] making them one of the most significant technological hazards to patient safety for over a decade [25, 62]. The Pennsylvania Patient Safety Authority reports that there have been 194 documented problems with operators' responses to telemetry monitoring alerts from June 2004 to December 2008 [1] and at least 35 deaths [26]. Medical device manufacturers have reported 216 "alarm-related" deaths to the FDA between January 2005 and June 2010 [26]. An event alert issued by the Joint Commission in April 2013 stated that reports voluntarily submitted to the Joint Commission's Sentinel Event database contained 98 events linked to alarms from Jan 2009 to Jun 2012: 80 of these events resulted in patient death, 13 produced "permanent loss of function", and 5 extended the stay of patients in the hospital [64].

There are a number of different perceptual problems that can arise with medical alarms (see [29] for an overview). For the purpose of this dissertation, we are primarily concerned with perceptibility of concurrently sounding alarms. Specifically, alarms that sound in close temporal proximity may produce auditory masking [34, 52], a condition where multiple sounds interact in a way that prevents the human perceptual system from hearing one of or more of them.

Different sounds that can be used for auditory alarms [60]. However, most alarms are either represented abstractly as sounds with a distinctive tone [28], or as a melody of such sounds [44]. Unfortunately, these types of sounds are particularly susceptible to masking in the presence of other alarms. Although many medical alarm experts have acknowledged auditory masking between concurrent medical alarms as a threat to patient safety [27, 30, 31, 46, 50, 51, 57, 65], it has been given very little research attention. In an analysis of 49 different alarms used in the intensive care unit and the operating rooms of a Canadian teaching hospital, Momtahan et al. [51] found several instances where alarms masked other alarms using a combination of physical auditory measurement, psychophysical modeling, and human subject psychophysical experiments. In a separate analysis, [65], who used psychoacoustic models to evaluate audio data recorded for medical alarms and other common hospital alarm noises (including phone ringing and beeper notifications), also found masking of medical alarms.

While there are a number of ways that auditory masking can occur [13, 34, 52], for tonal alarms, the most important is simultaneous masking. Simultaneous masking describes particular relationships between frequencies and volumes (determined by the human perceptual system) that can result in sounds being undetectable.

As the number of medical alarms increases and more and more alarms from different systems interact, the presence of these masking conditions will significantly increase [27]. Further, it is impractical to expect hospitals to use the experimental techniques of [51] and [65] to detect masking conditions in all of the possible alarm configurations that could occur in the hospital. Luckily, psychoacoustic models exist that are capable of detecting if simultaneous masking will occur between concurrent sounds.

2.2 The Psychoacoustic of Simultaneous Masking

A number of different models exist for predicting auditory masking [13, 16, 34, 42, 43, 52, 55, 56]. However, psychoacoustic models of masking are the most appropriate to this work because they quantitatively relate a sound's physical characteristics (its frequency/tone and volume) to the masking effect the sound has on human perception using mathematical formulas. Luckily, there are psychoacoustic models capable of expressing all of the different ways masking can manifest between concurrent tonal alarm sounds. The most successful of these models use heuristics based on the expected excitation patterns of the human ear's basilar membrane (the physical structure largely responsible for allowing humans to distinguish between different sounds)[3, 4, 13–15, 61].

These psychoacoustic models represent a sound's masking threshold for different frequencies of concurrently occurring sounds (its masking curve) as a function of the sound's volume in decibels (dB) and frequency in Barks. The Bark scale is psychoacoustic in that it represents a sound's frequency from 1 to 24 [24], indicating which of the 24 critical bands of hearing the sound falls in.[1] For a given sound (*sound*) with a frequency (f_{sound}) in Hz, the frequency is converted into the bark scale using the formula

$$z_{sound} = 13 \cdot \arctan(0.00076 \cdot f_{sound}) + 3.5 \cdot \arctan\left((f_{sound}/7500)^2\right), \tag{2.1}$$

where z_{sound} is the frequency of the sound in Barks.

The masking curve for a given sound (a *masker*) is generally formulated as a function of both the sound and the frequencies distance of another, potentially masked, sound (a *maskee*) on the bark scale. This difference, dz is represented as

$$\delta z = z_{maskee} - z_{masker}. \tag{2.2}$$

Then, the masker's masking curve is represented as

$$\text{curve}_{\text{masker}}(v_{masker}, \delta z) = \text{spread}_{\text{masker}}(v_{masker}, \delta z) + v_{masker} - \Delta \tag{2.3}$$

where v_{masker} is the volume of the masker in dB, spread is a function that defines how the volume

[1] A critical band is the frequency bandwidth of the filters produced by the inner ear's basilar membrane in the cochlea.

changes as δz moves away from zero, and Δ represents the minimum difference between a masker's and maskee's volume under which masking can occur.

There are a number of different psychoacoustic spreading functions that have been developed. Each makes tradeoffs between misses and false alarms in the detection of masking [13] and have been tuned to different applications. For example, many of these spreading functions were developed to compute the masking functions that are used in lossy audio compression formats like MPEG 2 and MP3 [13], where masked audio data is removed to reduce file size.

For example, the spreading function used as the basis for the MPEG2 audio codec [61] is formulated as

$$\text{spread}_{\text{masker}}(\delta z) = \quad 15.81 + 7.5 \cdot (\delta z + 0.474) - 17.5 \cdot \sqrt{1 + (\delta z + 0.474)^2}. \tag{2.4}$$

This spreading function is tuned to normal hearing. It also has only one independent variable (δz). However, other spreading functions can also take volume (v_{masker}) as an argument.

There can be different formulations of Δ depending on the nature of the sound. For tonal maskers [45], like those used in most medical alarms, Δ (in dB) can be formulated as

$$\Delta = 14.5 + z_{masker}. \tag{2.5}$$

If we are only considering the masking effect between a pair of alarms in a configuration of multiple alarms, we know that the masker (with volume v_{masker} and Bark frequency z_{masker}) is masking the maskee (with volume v_{maskee} and frequency z_{maskee}) if

$$\text{curve}_{\text{masker}}(v_{masker}, z_{maskee} - z_{masker}) \geq v_{maskee}. \tag{2.6}$$

2.2.1 Additive masking

The psychoacoustics described above assume that masking is occurring between two alarms. However, if there are multiple potential masking sounds, masking is additive. Additive masking describes a condition where two simultaneous sounds can produce masking greater than or equal to the sum of their respective masking curves [13].

This additive masking [13, 43] is modeled by combining the masking curve values of each potential masker on the power scale. We use the following equation to represent a volume (in dB) on the power scale

$$\text{power}(v) = 10^{v/10}. \tag{2.7}$$

Then, for a given maskee and N potential maskers, the threshold (in dB) the maskee must exceed to be heard is calculated as [13, 43]

$$\begin{aligned}\text{power}(\text{mthresh}_{maskee}) =& \text{power}(\text{abs}_{maskee}) \\ &+ \left(\sum_{n=1}^{N} \text{power}(\text{curve}_{masker_n}(z_{maskee}))^{\alpha}\right)^{1/\alpha}\end{aligned} \tag{2.8}$$

where α is a constant with range $(0, \infty)$ [36] and abs_{maskee} is the absolute threshold of hearing (in dB) at the maskee's frequency (f_{maskee} in Hz) calculated by [63]

$$\text{abs}_{maskee} = 3.64 \cdot \left(\frac{f_{maskee}}{1000}\right)^{-0.8} - 6.5 \cdot e^{-0.6\left(\frac{f_{maskee}}{1000} - 3.3\right)^2} + 10^{-3} \cdot \left(\frac{f_{maskee}}{1000}\right)^4. \tag{2.9}$$

These psychoacoustics have shown themselves to be valid and useful for predicting masking for normal human hearing for decades [13]. Most relevantly, they have been used to identify masking between recorded medical sounds [65].

2.3 Formal Verification with Model Checking

Formal verification falls within the broader category of formal methods that uses efficient, exhaustive search techniques to prove whether a mathematical model of a systems behavior (usually represented as a state machine) adheres to desirable, logically-asserted properties. Verification mathematically proves whether the model adheres to the properties in the specification. Formal verification has been used successfully in computer hardware and software.

Model checking automatically performs formal verification [17]. The formal model describes a system as a finite state machine (variables and discrete transitions between valuations of those variables). Specifications are asserted using logical operators, system variables, and temporal logic operators [32]. The model is verified against the specification properties by exhaustively searching through the system's state space while evaluating whether the properties are true. If they are, the model checker returns a confirmation. If there is a property is not true, a counterexample is returned. The counterexample represents a trace through the system model (a sequence of model states)that led up to the property violation. Because of its approach, model checking is particularly good at finding problems in systems with concurrency, where independent system elements can interact in ways unanticipated by designers.

The majority of formal verification analyses are concerned with discrete even systems. However, hybrid modeling and analysis techniques can allow formal verification to be used with models that contain continuous quantities [23, 40, 58]. In such models, a discrete state (such as a particular configuration of sounding alarms) can be associated with continuous quantities (this could include precise times, frequencies, and volumes) that can also be used in the assertion of specifications. For example, to model time formally, formal modelers use timed automata [2, 23], a modeling approach where every discrete transition in a formal model is assigned a real number time.

Researchers have used formal verification to evaluate issues related to human-automation interaction (see [12] for a review). These techniques focus on abstract models from the human factors literature that can be represented with discrete mathematical models and used in analyses of a scope such that specific human factors problems can be discovered [12]. Collectively, these studies have shown that formal verification can be very useful for finding problems related to human factors in automated systems. However, none of them have explored how human perception and problems associated with it can be included in these formal analyses.

2.4 Objective

Because of its ability to detect problems in complex, concurrent systems, formal verification should be capable of detecting if masking can manifest in a particular configuration of medical alarms. The work presented here strived to demonstrate this. As an initial step we developed a method that allows an analyst to specify a configuration of alarms and use formal verification to detect if there are any situations where each alarm is masked. The method would only account for simultaneous masking between any given two sounds in a configuration of alarms. We then extend this method to account for the effects of additive masking, thus extending the method to account for the masking effect of multiple, concurrently sounding alarms. Both version of the method are built around a formal modeling architecture that allows for the sounding behavior of medical alarms to be represented formally. Our framework includes psychoacoustic functions capable of indicating when masking can occur and temporal logic specification property patterns for asserting the absence of masking conditions. Thus, formal verification with model checking can be used to detect if masking exists in models constructed around the framework.

CHAPTER 3: THE INITIAL METHOD

In the initial version of our method, we were predominately concerned with demonstrating the feasibility of our method by having it be able to detect masking between pairs of alarms in a configuration of multiple alarms. In this version of the method (Fig. 3.1), an analyst must: (a) examine the documentation associated with a configuration of medical alarms and model their behavior using our formal modeling architecture (Fig. 3.2); (b) specify the absence of masking using specification property patterns we provide; and (c) use model checking to formally verify that the specification properties hold for the model. If no masking exists, the model checker will return a confirmation in its verification report. Otherwise, a counterexample will be produced which will illustrate how masking can occur. This can be used by the analyst to determine how the discovered masking condition might be avoided.

Timing of concurrently sounding alarms can have a profound impact on whether alarms are masked or not, thus we need to evaluate all of the different ways alarms can temporally overlap. Therefore, we have designed our formal modeling architecture (Fig. 3.2) to be based on timed automata. Timed automata [23] provide a means of modeling time as a real-valued continuous quantity in a formal model. This architecture has multiple sub-models that are synchronously composed together: a clock (the timed automaton) that keeps and advances time; models of the behavior of the alarms in a given configuration; and a model that computes whether masking is occurring for each alarm and determines the maximum advance of the clock.

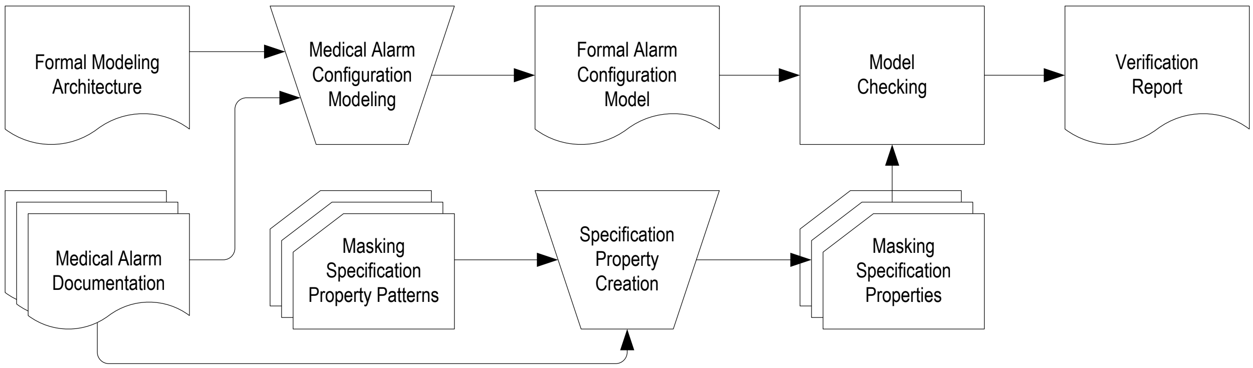

Figure 3.1: The original method for using formal verification to detect auditory masking in medical alarm configurations.

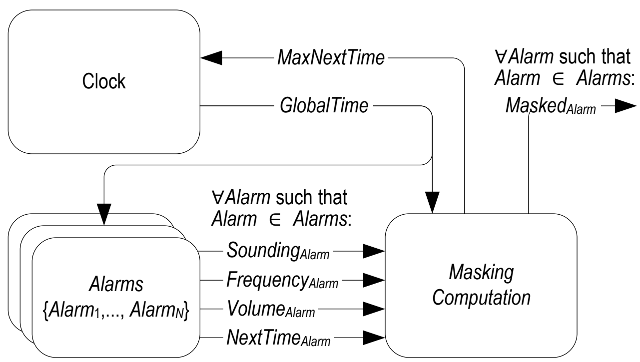

Figure 3.2: The original architecture for formally modeling a configuration of auditory medical alarms. Boxes represent sub-models of the larger system model and arrows represent variables with input-output relationships between the sub-models. Arrows with no target indicate outputs.

3.1 Method Implementation

We have implemented this method using the tools available in the Symbolic Analysis Laboratory (SAL) [20, 54]. In particular, we have designed our method to work with SAL's infinite bounded model checker [23, 54], a tool capable of evaluating formal models containing timed automata. SAL's infinite bounded model checker uses satisfiability modulo theories to check properties in formal models that contain continuous variables. It is bounded in the sense that it takes a number of steps (the bound) as input. The model checker then proves whether or not the checked specification properties hold for up to the specified number of steps through the model.

Our implementation of the method is designed so that it will require a limited amount of analyst-created code. What is required follows systematic patterns. The remainder of this section describes how our implementation of the method was realized. First, we describe the details of the formal modeling architecture. This is followed by a description of the specification property patterns analysts can use to assert the absence of masking. Finally, we explain how the model checker can be used to evaluate a medical alarm configuration. Throughout, we highlight where analyst effort is required.

An overview of the SAL implementation of the architecture can be seen in Fig. 3.3. This has eight distinct parts. Firstly, it contains a collection of type definitions. These represent variable types that are used by other elements in the modeling architecture for representing alerting concepts and include non-negative, real-valued time, volume, and frequency.

Next, the model contains two constants that are used to represent standard values used in other parts of the architecture. The first, `delatConst`, represents that constant volume used in the computation of Δ (Eq. (2.5)). The second, `bigMax`, represents an arbitrarily large maximum on the amount time can increase in a given step through the model.

The constant definitions are followed by function definitions. These represent mathematical expressions that are used by other model constructs to compute quantities used in the detection of masking. These are discussed in Section 3.1.3.

The clock sub-model, which is responsible for maintaining and advancing time, is next. It is described in Section 3.1.1.

A series of sub-models representing the behavior of the different alarms in the alarm configuration follow. Each of these represents the behavior of a given alarm at the global current time indicated by the clock. Section 3.1.2 describes the generic formal modeling pattern used for modeling each alarm in a configuration (with `N` alarms) in the architecture.

The masking computation sub-model evaluates the outputs of the alarm sub-models and uses the defined functions to compute whether masking is occurring at the given clock-indicated time. This is developed further in Section 3.1.3.

Each of the sub-models is ultimately synchronously composed into a full system model.

Finally, specification properties are used to assert the absence of masking in a model constructed using the architecture. The generic patterns used for composing such specifications are described in Section 3.1.4.

Of the architectural components, only the alarm sub-models, the masking computation sub-model, the system model composition, and the specification properties require any analyst effort. All of the other components are standard.

3.1.1 Clock Sub-model

The clock sub-model (Fig. 3.4) is responsible for advancing time and communicating the current time to the other elements of system. It receives a maximum on the amount that time can advance to (`nextTime`) and outputs the current and/or global time (`globalTime`). The global time is initially set to 0. Then, for every subsequent step through the model, the global time is advanced to an arbitrary new time such that the new time is always greater than the current global time and less than or equal the maximum next time.

3.1.2 Alarms Sub-model

The behavior of each alarm (which is assumed to be a pattern of tones) is described in a separate model, where each alarm model follows a similar implementation pattern (Fig. 3.5). Each alarm has a constant value representing the length of its sounding cycle in second (`alarmCycleTime` with constant value `TCycle` in Fig. 3.5). Each alarm also has a start time (`alarmStartTime`, which is initially 0) that is used to indicate if an alarm is sounding (`alarmSounding = alarmStartTime > 0`) and, if it is, when the alarm started doing so.

The alarm model is responsible for setting the start time and computing the amount of time the alarm has been sounding. Our model assumes that an alarm will sound for a single cycle and then stop. Thus, the amount of time the alarm has been sounding is computed as the difference between the global time and the alarms start time (`alarmTimeInCycle = globalTime -`

```
formalArchitecture : CONTEXT =
BEGIN

  %Type definitions
  TIME        : TYPE    = {X : REAL | X >= 0}; % in s
  VOLUME      : TYPE    = {X : REAL | X >= 0}; % in dB
  FREQUENCY   : TYPE    = {X : REAL | X >= 0}; % in Hz

  %Constant definitions
  deltaConst : VOLUME = 14.5;
  bigMax      : TIME    = 60;

  %Function definitions
  ...

  %Clock sub-model
  clock : MODULE =
  BEGIN
    ...
  END;

  %Alarm sub-models
  alarm1 : MODULE =
  BEGIN
    ...
  END;

  ...

  alarmN : MODULE =
  BEGIN
    ...
  END;

  %Masking computation sub-model
  maskingComputation : MODULE =
  BEGIN
    ...
  END;

  %Composition of the full system model
  system : MODULE = clock || alarm1 || ... || alarmN || maskingComputation;

  %Specification properties
  ...

END
```

Figure 3.3: An overview of the SAL implementation of the formal modeling architecture for modeling medical alarm configurations (Fig. 3.2). This implementation is written using the notation of SAL (see [20]). Note that in this listing (and all subsequent listings in Figs. 3.4–3.6), code highlighting is used to improve readability: SAL language reserved words (including built-in basic types) are blue; declared types are dark blue; constants are green; functions are orange (these appear in subsequent listing); and everything else is black. Ellipses "..." are used to indicate the omission of content that is either detailed in subsequent listings (Figs. 3.4–3.6) or indicates an incremental series of like components or operations (e.g. the synchronous compositions of the alarm sub-models: `alarm1 || ... || alarmN`).

```
clock : MODULE =
BEGIN

  INPUT  maxNextTime : TIME
  OUTPUT globalTime  : TIME

  INITIALIZATION
    globalTime = 0;

  TRANSITION
    globalTime' IN {X: TIME | (X > globalTime) AND (X <= maxNextTime)};

END;
```

Figure 3.4: SAL code for representing the clock in the formal model of the original method.

`alarmStartTime`). At any given global time, an alarm that is not sounding can begin sounding in the next state by setting the start time to the globaltime in the next state (see `TRANSITION` in Fig. 3.5). If the alarm is sounding and has not been sounding for longer than its cycle time in the next state's global time, the alarm keeps its current start time in the next state. If the alarm has been sounding for its full cycle time at the next global time, the alarm ceases to sound (sets the start time to zero) in the next state.

If the alarm is sounding, then the alarm model must update its frequency (`alarmFrequency`), volume (`alarmVolume`), and next time (`alarmNextTime`) output variables based on the alarm's time in cycle. Specifically, for set times less than or equal to the alarm's cycle time (i.e. `TFreq1` – `TFreqN` from Fig. 3.5), the alarm will assume different values for frequency and volume. It should be noted that, in the model shown in Fig. 3.5, the value of the alarm's volume is determined by the alarm's frequency. However, modelers can have the alarm volume change independently of the alarm's frequency if desired. The purpose of the next time output is to communicate the next global time that the alarm will experience a change in its frequency and/or volume. Thus, the next time variable should update to reflect this based on the current time in cycle.

All alarm models follow the implementation pattern shown in Fig. 3.5. Within this, the analyst needs to describe the `alarmCycleTime` (the `[TCycle]` value in Fig. 3.5) and the logic defining the `alarmFrequency`, `alarmVolume`, and `alarmNextTime` by specifying the appropriate values (`[TFreq1]`, `[Freq1]`, `[Vol1]`, etc.).

```
alarm : MODULE =
  BEGIN

  INPUT   globalTime        : TIME

  LOCAL   alarmStartTime    : TIME
  LOCAL   alarmCycleTime    : TIME
  LOCAL   alarmTimeInCycle  : TIME

  OUTPUT  alarmSounding     : BOOLEAN
  OUTPUT  alarmFrequency    : FREQUENCY
  OUTPUT  alarmVolume       : VOLUME
  OUTPUT  alarmNextTime     : TIME

  INITIALIZATION
    alarmStartTime   = 0;

  DEFINITION
    alarmCycleTime   =  TCycle;
    alarmTimeInCycle =  globalTime - alarmStartTime;
    alarmSounding    =  alarmStartTime > 0;
    alarmFrequency   =  IF    alarmSounding AND alarmTimeInCycle < TFreq1 THEN Freq1
                        ...
                        ELSIF alarmSounding AND alarmTimeInCycle < TFreqN THEN FreqN
                        ELSE                                                   0
                        ENDIF;
    alarmVolume      =  IF    alarmSounding AND alarmFrequency = Freq1 THEN Vol1
                        ...
                        ELSIF alarmSounding AND alarmFrequency = FreqN THEN VolN
                        ELSE                                                0
                        ENDIF;
    alarmNextTime    =  IF    NOT alarmSounding         THEN bigMax
                        ELSIF alarmTimeInCycle < TFreq1 THEN alarmStartTime + TFreq1
                        ...
                        ELSIF alarmTimeInCycle < TFreqN THEN alarmStartTime + TFreqN
                        ELSE  alarmStartTime + alarmFullCycleTime
                        ENDIF;

  TRANSITION
  alarmStartTime' IN {X: TIME | ((alarmStartTime = 0) AND ((X = globalTime') OR (X = 0)))
                        OR ((alarmStartTime > 0)
                            AND (globalTime' < (alarmStartTime + alarmCycleTime))
                            AND (X = alarmStartTime))
                        OR ((alarmStartTime > 0)
                            AND (globalTime' >= (alarmStartTime + alarmCycleTime))
                            AND (X = 0))
                      };
END;
```

Figure 3.5: Generic SAL code for representing alarm behavior. Note that words in red represent numerical values that should be explicitly specified by the modeler. `TCycle` represents the alarm's cycle time. `TFreq1` – `TFreqN` represent relative times from the start time that the frequency and volume change in increasing order. `Freq1` – `FreqN` and `Vol1` – `VolN` represent different frequencies (in Hz) and volumes (in dB) respectively.

3.1.3 Masking Computation Sub-model

The masking computation model (Fig. 3.6) has two roles. First, at every time assumed by the clock, it looks at the frequency and volume of each alarm and computes whether it is being masked by

```
maskingComputation : MODULE =
BEGIN

  INPUT  globalTime       : TIME
  INPUT  alarm1Volume     : VOLUME
  INPUT  alarm1Frequency  : FREQUENCY
  INPUT  alarm1Sounding   : BOOLEAN
  INPUT  alarm1NextTime   : TIME
  ...
  INPUT  alarmNVolume     : VOLUME
  INPUT  alarmNFrequency  : FREQUENCY
  INPUT  alarmNSounding   : BOOLEAN
  INPUT  alarmNNextTime   : TIME

  OUTPUT alarm1Masked     : BOOLEAN
  ...
  OUTPUT alarmNMasked     : BOOLEAN

  OUTPUT maxNextTime      : TIME

  DEFINITION
    alarm1Masked     =  masking(alarm2Sounding,   alarm2Frequency,   alarm2Volume,
                                alarm1Sounding,   alarm1Frequency,   alarm1Volume)
                        OR ...
                        OR masking(alarmNSounding,   alarmNFrequency,   alarmNVolume,
                                   alarm1Sounding,   alarm1Frequency,   alarm1Volume);
    ...
    alarmNMasked     =  masking(alarm1Sounding,   alarm1Frequency,   alarm1Volume,
                                alarmNSounding,   alarmNFrequency,   alarmNVolume)
                        OR ...
                        OR masking(alarmN-1Sounding, alarmN-1Frequency, alarmN-1Volume,
                                   alarmNSounding,   alarmNFrequency,   alarmNVolume);
    maxNextTime      IN {X: TIME | (X = alarm1NextTime OR ... OR X = alarmNNextTime)
                         AND (X <= alarm1NextTime AND ... AND X <= alarmNNextTime)};

END;
```

Figure 3.6: Generic SAL code for the masking computation sub-model in the original method.

the other sounding alarms. These computations are synthesized into a single Boolean variable for each alarm that indicates if that alarm is being masked (`alarm1Masked` – `alarmNMasked`).

They are performed using a set of functions (Fig. 3.7) that implement the equations in Eqs. (2.1) to (2.5) in the formal model. Because model checkers are limited in their ability to consider non-linear variable arithmetic in their formal input models, two of these functions were implemented using lookup tables: values of the functions over a range of acceptable values are pre-computed and accessed in the formal model using a large `IF...THEN...ELSIF` statement. For the conversion of frequency (in Hz) to the Bark scale, the lookup table was computed using Eq. (2.1) rounded to the nearest tenth of a Bark for the full range of Bark scale values (0 to 24). For the spreading function (`spread`), Eq. (2.4) was used to compute the spread rounded up the nearest dB for the full range of possible values for δz (`dz` in Fig. 3.7). Note that this computation rounds up so that it biases the masking curve slightly in favor of detection.

Further, the spreading function from Eq. (2.4) was chosen because it uses only δz in its computation, making the corresponding lookup table one-dimensional and thus much simpler to implement formally. This is discussed in more detail in Section 2.2.1. The masking function (`masking`) uses both the `bark` and `spread` functions to compute whether or not a given alarm (the masker) masks another alarm (the maskee). `sMasker`, `fMasker`, `vMasker`, `sMaskee`, `fMaskee`, `vMaskee` represent the following concepts for masker and maskee alarms respectively: whether or not the alarm is sounding; its frequency (in Hz); and volume (in dB). If neither the masker nor the maskee are sounding, no masking is possible. If this is not true and the volume of the maskee is 0 (for example, when the alarm is sounding but in a pause between alarm tones), then the maskee is being masked. Otherwise, the function computes whether the masker masks the maskee using Eq. (2.6).

```
bark(f: FREQUENCY): REAL
  = IF f <= 5     THEN 0    ELSIF  f <= 15    THEN 0.1  ... ELSIF
       f <= 15145 THEN 23.9 ELSE 24   ENDIF;

spread(dz: REAL): REAL
  = IF dz <= -24    THEN -572 ELSIF dz <= -23.9 THEN -570 ... ELSIF
       dz <= 23.9   THEN -228 ELSE -229 ENDIF;

masking(sMasker: BOOLEAN, fMasker: FREQUENCY, vMasker: VOLUME,
        sMaskee: BOOLEAN, fMaskee: FREQUENCY, vMaskee: VOLUME): BOOLEAN
  = IF    NOT(sMasker AND sMaskee) THEN FALSE
    ELSIF vMaskee = 0              THEN TRUE
    ELSE  ((spread(bark(fMaskee) - bark(fMasker)) + vMasker -
           (deltaConst + bark(fMasker))) >= vMaskee)
    ENDIF;
```

Figure 3.7: Method implementation of Eqs. (2.1) to (2.5) used by the `maskingComputation` submodel to compute whether masking is occurring between two alarms.

The second thing the masking computation model does is calculate the next time variables ($NextTime_{Alarm}$) from all of the alarms and communicates it to the clock as `maxNextTime`. It does this by selecting a time value from a set of times that are equal to at least one of the alarm's next times and less than or equal to all of the alarm next times.

```
alarmPartialMasking : THEOREM system
  |- G(NOT(alarmVolume > 0 AND alarmMasked));
alarmTotalMasking   : THEOREM system
  |- G(NOT((NOT alarmSounding)
       AND X(alarmSounding AND alarmMasked
             AND U(alarmSounding AND alarmMasked,
                   NOT alarmSounding))));
```

Figure 3.8: Specification property patterns for a given alarm. `alarmPartialMasking` asserts the absence of masking for a given alarm. `alarmPartialMasking` asserts that a given alarm will never be completely masked.

In creating the masking computation model, an analyst is responsible for creating the "`alarmMasked`" variable and its definition for each alarm using the pattern shown in Fig. 3.6. A model must also create the `maxNextTime` definition, again using the pattern in Fig. 3.6.

3.1.4 Specification

To model check whether or not masking is present in a model, specifications must assert its absence. Our method uses property patterns to do this, where an analyst must instantiate the specification pattern for each alarm in a configuration. In this method, we are interested in detecting whether, for a given alarm, there is any situation where the alarm is masked and if the alarm can be totally masked (completely imperceptible). Thus, specification property patterns are created for each alarm asserting the absence of both phenomena (Fig. 3.8). For a given alarm (`alarm`), `alarmPartialMasking` uses linear temporal logic to assert that: for all paths through the model (**G**), there should never be a situation where the alarm is making noise (`alarmVolume > 0`) and the alarm is masked. `alarmTotalMasking` asserts that: for all (**G**) paths through the model, we never want it to be true that the alarm goes from not sounding to sounding and masked in the next (**X**) state such that, from then on, the alarm is sounding and masked until (**U**) it is no longer sounding.

When creating the specifications, an analyst needs to replicate the specification property patterns (Fig. 3.8) for each alarm.

3.1.5 Model Checking

To determine whether masking exists in a modeled configuration, the analyst must evaluate the model with a model checker. For our implementation, this means using SAL's infinite bounded model checker [54]. Once installed on a compatible system (see [21]), this is run on the command line as follows:

```
sal-inf-bmc [Model] [Specification] -d [Depth]
```

Here, `sal-inf-bmc` is the command for invoking the model checker. The analyst must specify the name of the model (`[Model]`), the name of the specification to be checked (`[Specification]`, and the search depth (`[Depth]`; the maximum number of transitions to consider in an analysis). In our analyses, depths should be set high enough to account for all of the possible transition states an analyst wants to consider.

To ensure that all of the different temporal alarm overlap possibilities in the configuration are considered, this bound should be the sum of the total number of different alarm transitions that can occur. For example, in a configuration with two alarms that each sound twice with a pause in between, each alarm has four transitions (off to sounding, sounding to pause, pause to sounding, sounding to off). Thus, there would be a total of 8 transitions for the two alarms and an analysis of the model should have a search depth of at least 8. However, it is important to note that smaller search depth can be used. Because increasing the search depth will likely increase verification time, an analyst may wish to save time by using smaller depth. Any counterexamples returned from shorter searchers will still constitute valid results. However, a failure to find a counterexample at a depth below the suggested one may not genuinely indicate the absence of masking.

3.2 Application

To illustrate the ability of our method to detect masking in a realistic configuration of medical alarms, we have used it to evaluate a simple application. In this target configuration (see Table 3.1) there were three alarms. In a given cycle, each alarm played a two-tone pattern with a pause in

Table 3.1: Case Study 1 Alarm Configuration

Alarm	Tone 1			Pause	Tone 2		
	Freq. (Hz)	Vol. (dB)	Time (s)	Time (s)	Freq. (Hz)	Vol. (dB)	Time (s)
Alarm 1	261	80	0.25	0.100	370	80	0.25
Alarm 2	277	60	0.15	0.050	277	60	0.15
Alarm 3	524	85	0.20	0.075	294	85	0.20

between. Each sound used a frequency commonly found in tonal alarms [44, 51]. Durations, and volumes were also consistent with the IEC 60601-1-8 international standard [44].

These alarms were used to construct four separate formal models using the above implementation of the method. Three of these models contained the implementation of each pair of alarms from Table 3.1. The last contained the implementation of all three alarms. In all of the models, specifications were created using the patterns from Fig. 3.8 to assert that each of the included alarms should never be partially or totally masked (see `http://fhsl.eng.buffalo.edu/resources/` for full listings of all models). By evaluating each of these models separately, we are able to evaluate the ability of the method to detect masking within the possible interactions of any two alarms as well as all of the alarms together.

Every specification in each model was evaluated using SAL's infinite bounded model checker [54] (with search depth 12) on a Linux workstation with a 3.3 GHz Intel Xeon processor.

The model checker produced counterexample needed to manually interpreted by an analysts. We found that a variation of a vertical bar graph could be used to effectively show how alarms in a counterexample sounded in relation to each other and indicate when masking was occurring, all verification results can be seen in Table 3.2. This shows that no masking was detected when only Alarm 1 and Alarm 3 were in the model. However, in the model where Alarm 1 and Alarm 2 were present and the model where Alarm 2 and Alarm 3 were present, partial masking was detected but not total masking. In the model with all three of the alarms, both partial and total masking were detected. The counterexamples returned by the model checker for each specification were visualized (Fig. 3.9) to determine exactly how the detected masking manifested. This revealed that the first tone of Alarm 1 and the second tone of Alarm 3 were both capable of masking the tones of Alarm 2.

3.3 Discussion

This work has introduced a novel method for identifying masking in configurations of medical alarms. This method uses a formal modeling architecture, psychoacoustic models of masking, specification property patterns, and formal verification with model checking to prove whether or not each alarm in a modeled configuration will be perceptible with normal hearing. We have

Table 3.2: Case Study 1 Verification Results

Model with			Verification Output	
Alarms	Alarm	Specification	Time (s)	Outcome
1 & 2	Alarm 1	Partial Masking	87.26	✓
		Total Masking	63.76	✓
	Alarm 2	Partial Masking	99.31	×
		Total Masking	56.11	✓
1 & 3	Alarm 1	Partial Masking	67.05	✓
		Total Masking	32.88	✓
	Alarm 3	Partial Masking	66.37	✓
		Total Masking	127.57	✓
2 & 3	Alarm 2	Partial Masking	180.56	×
		Total Masking	95.82	✓
	Alarm 3	Partial Masking	85.49	✓
		Total Masking	69.52	✓
1, 2, & 3	Alarm 1	Partial Masking	392.76	✓
		Total Masking	320.62	✓
	Alarm 2	Partial Masking	1281.26	×
		Total Masking	492.76	×
	Alarm 3	Partial Masking	297.92	✓
		Total Masking	1205.43	✓

Note. ✓ indicates a verification confirmation and × indicates a verification failure with a counterexample.

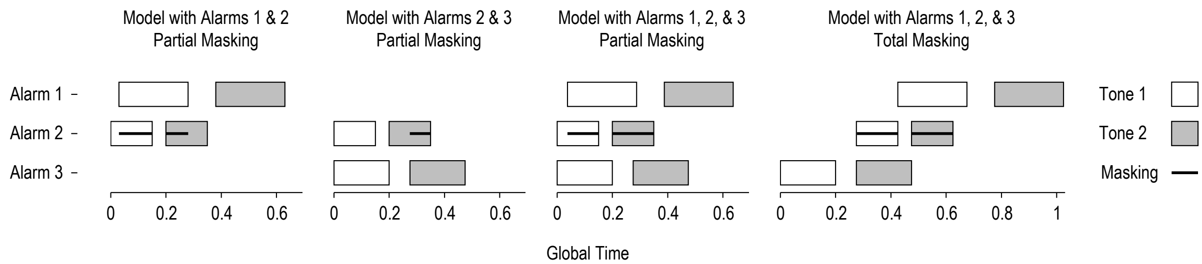

Figure 3.9: Illustration of the counterexamples returned when the model checker failed to prove that Alarm 2 would not be partially or completely masked. For the partial masking results, Alarm 2's tones were masked by Alarm 1's first tone and Alarm 2's second tone was masked by Alarm 3's second tone. For the total masking result, the second tone of Alarm 3 masks the first tone of Alarm 2 and Alarm 2's second tone was masked by Alarm 1's first tone.

implemented a version of this method in SAL using timed automata. To demonstrate the method's power, we presented a realistic medical alarm configuration and showed how our method could be used to find masking conditions. The power of the method is particularly well illustrated by the multiple verifications that were performed. While partial masking was detected in models that only contained two alarms, total masking was only observed when the interactions between all three alarms were considered simultaneously.

While this original version of the method presented in this chapter makes these significant contributions, it does have several limitations. First, it only considered masking between pairs or alarm sounds in a configuration of multiple alarms and thus does not account for the additive effect of masking [13, 43]. Second, this initial version also used a psychoacoustic spreading function (Eq. (2.4)) selected to account for the computational limitations of our method. Specifically, it is one dimensional in that it only varied as a function of δz. This made it easier to include the spreading function calculations in the formal model as a lookup table. There are psychoacoustics better suited to modeling the effect of tonal alarms. Third, the method scaled badly, resulting in analyses that took prohibitively long to give useful results [38]. Fourth, the method required manual formal modeling and specification by analysts and provided no user support for interpreting analysis results. All of these limitations were addressed in the subsequent version of the method, discussed next.

CHAPTER 4: THE NEW METHOD

The work presented in this chapter shows how we extended our method to account for the limitations of our initial version of the method (see Chapter 3). To this end, we enable our method to account for the additive of masking. We also update the psychoacoustics used to compute masking curves to better reflect the tonal nature of the masking sounds of alarms. We further create a computer program that enables the lookup tables to be optimized to improve scalability while using new psychoacoustics. Finally, this program was given features to simplify the model creation process and allow analysts to automatically visualize counterexamples to identify when and how masking can occur. Below we describe how these features were realized. To demonstrate the scalability improvements of the method and show that the new version has comparable detection capabilities to the original, we re-evaluate the application presented in Chapter 3 and compare the results. We also demonstrate the ability of the method to detect additive masking in another simple application. Finally, we show the ability of the method to detect additive masking in a realistic application by applying it to the alarms used in a telemetry monitoring system.

The updated version of our method is shown in Fig. 4.1. To use it, an analyst first examines alarm documentation and describes the behavior of the alarms using a MS Excel spreadsheet, where each alarm is described as a sequences of tones (and pauses between tones) each with a defined frequency (Hz), volume (dB), and duration (s). Figure 4.2 shows an example of how a single alarm would be described in our spreadsheet. When done modeling alarms, the analyst uses the computer program to automatically convert the described alarm configuration's behavior into a formal model.

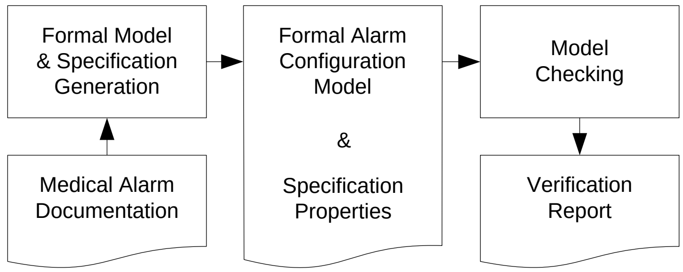

Figure 4.1: A sequence diagram of our new masking detection method.

Name	Freq (Hz)	Vol (dB)	Time (s)	...
AnAlarm	523	72	0.1	
	0	0	0.1	
	698	72	0.1	
	0	0	0.1	
	784	72	0.1	

Figure 4.2: An example of an alarm as it would be described in a spreadsheet in our implementation of the method. The presented alarm has three tones separated by two pauses, where the order of tones and pauses is specified from top to bottom. The ... is used to indicate that additional alarms would be described similarly to the right of the presented alarm.

4.1 Updated Formal Modeling Architecture

The formal model used in the new method has a slightly updated architecture (Fig. 4.3). The formal system model is made of a set of synchronously composed sub-models, each with a particular purpose. The clock sub-model uses a timed automaton [2, 23] to advance model time (*globalTime*) and communicate it to the other sub-models. Each alarm is represented as a sub-model that can start or stop sounding at appropriate times and adjust its state based on its current state and how long it has been sounding. Alarm state represents each of the distinct tones or pauses that occur over a complete sounding. For example, the alarm shown in Fig. 4.2 would have six states: one for when it is not sounding and one for each of the listed tones and pauses. A single masking computation sub-model uses the current state of each alarm, its associated "power alpha" (discussed subsequently), and the psychoacoustics of simultaneous masking to determine if any alarm is masked by the other sounding alarms. This sub-model also find the minimum of the alarms' recommended next times (the *alarmNextTime* variables) to recommend a maximum amount (*maxNextTime*) to advance the clock.

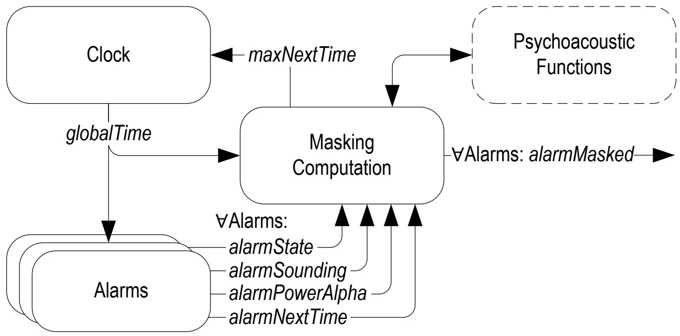

Figure 4.3: A sequence diagram of our masking detection method.

4.2 The New Method's Psychoacoustics

Model checkers cannot handle the nonlinear arithmetic of model variables [21]. Thus our method uses a pre-computed lookup tables (functions) to represent nonlinear psychoacoustic computations. However, the size of your lookup tables can reduce the efficiency of your model. Thus, in our new method, we encapsulate all of the necessary non-linear mathematical operations into a single lookup table. This was optimized to ensure the minimum number of necessary entries for any given model. This was done to reduce verification time.

The "power alpha" value discussed above and in Fig. 4.3 plays an important role in this optimization and allows for the detection of masking using a model checker. Specifically, by transforming the maskee's volume and the masking effect of maskers into "power alpha" values using lookup tables, masking can be detected using only linear arithmetic operations. Figure 4.4 explains the rational for the "power alpha" transformation and shows how it occurs.

Our method uses the relationship from Eq. (4.6) (Fig. 4.4) as the basis for its optimization. Specifically, the computer program pre-computes each alarm's associated "power alpha" value (using Eq. (4.4)) for each of the alarm's states. In the formal model, the alarm's state and "power

We know from Eq. (2.8) that a set of maskers will mask a maskee if

$$\text{power}\left(v_{maskee}\right) \leq \text{power}\left(\text{absolutethreshold}_{maskee}\right) + \left(\sum_{n=1}^{N} \text{power}\left(\text{curve}_{masker_n}\left(z_{maskee}\right)\right)^{\alpha}\right)^{1/\alpha}. \tag{4.1}$$

Using basic algebraic operations, we know that

$$\text{power}\left(v_{maskee}\right) - \text{power}\left(\text{absolutethreshold}_{maskee}\right) \leq \left(\sum_{n=1}^{N} \text{power}\left(\text{curve}_{masker_n}\left(z_{maskee}\right)\right)^{\alpha}\right)^{1/\alpha} \tag{4.2}$$

and thus that

$$\left(\text{power}\left(v_{maskee}\right) - \text{power}\left(\text{absolutethreshold}_{maskee}\right)\right)^{\alpha} \leq \sum_{n=1}^{N} \text{power}\left(\text{curve}_{masker_n}\left(z_{maskee}\right)\right)^{\alpha}. \tag{4.3}$$

If we let

$$\text{poweralpha}_{maskee} = \left(\text{power}\left(v_{maskee}\right) - \text{power}\left(\text{absolutethreshold}_{maskee}\right)\right)^{\alpha} \tag{4.4}$$

and

$$\text{maskingpoweralpha}_{masker}(maskee) = \text{power}\left(\text{curve}_{masker}\left(z_{maskee}\right)\right)^{\alpha}, \tag{4.5}$$

then we know that the maskee will be masked by the set of N maskers if

$$\text{poweralpha}_{maskee} \leq \sum_{n=1}^{N} \text{maskerpoweralpha}_{masker_n}(maskee). \tag{4.6}$$

Figure 4.4: Explanation of “power alpha” and how it is used to determine if masking is occurring.

alpha” value are communicated to the masking computation sub-model (see Fig. 4.3). The masking computation sub-model then uses the lookup table (pre-computed by the computer program to implement Eq. (4.5) and optimized to minimize the number of entries) to obtain the “power alpha” value associated with each potential maskee-masker pair of alarms, based on each alarm’s respective state. For each sounding alarm, the masking computation sub-model adds up each of the “power alpha” values with the given alarm as the maskee and compares it with the “power alpha” value from that alarm’s associated alarm sub-model to determine if masking is occurring (see Eq. (4.6)). Thus, the use of the “power alpha” values allows our method to implement additive masking detection formally. We use $\alpha = 0.33$, which Lutfi [49] found best captured the “over adding” of the masking effects of tones. However, the computer program allows for different analyst specified α values.

The “power alpha” computation in Eq. (4.5) relies on masking curve values. Because these values are pre-computed in our new version of the method, we were able to be more selective in what spreading function (see Eq. (2.3)) we use. Specifically, we can now use the more computationally complicated spreading function of

$$\text{spread}_{masker}(\delta z) = \begin{cases} -17 \cdot \delta z + 0.15 \cdot v_{masker} \\ \qquad \cdot (\delta z - 1) \cdot \theta(\delta z - 1) & \text{for } \delta z \geq 0 \\ -(6 + 0.4 \cdot v_{masker}) \cdot |\delta z| \\ -(11 + 0.4 \cdot v_{masker} \cdot (|\delta z| - 1)) & \text{otherwise} \\ \quad \cdot \theta(|\delta z| - 1) \end{cases} \tag{4.7}$$

where $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ otherwise. This particular spreading function was chosen because it is the most appropriate for modeling the masking effects of tones on other tones [15]. We also updated the way that Δ (from Eq. (2.3)) was computed. In the new version

$$\Delta = 6.025 + 0.275 \cdot z_{masker} \text{ dB}. \tag{4.8}$$

This new formulation was used for several reasons. First, it has been shown to be appropriate for tones [3]. It was also used in the MPEG audio codec [13], thus it has a well-established validity. Further, it will always be smaller than the Δ used in the original method (Eq. (2.5)). This means that it will increase the chances that our method will detect masking. Given that missing the detection of masking has significantly worse consequences than a false alarm, this was a preferable value of Δ for our purposes.

4.3 Formal Model and Specification Generation

When the computer program generates the formal model and specifications, it creates a formal model in the input language of the symbolic analysis laboratory's (SAL's) infinite bounded model checker [21]. An overview of the file can be seen in Fig. 4.5. Note that this is slightly different from the overview previously shown in Fig. 3.3.

The model has eight parts. First, there are type definitions. These represent variable types that

```
alarmConfiguration : CONTEXT =
BEGIN

  %Type definitions
  TIME       : TYPE = {X : REAL | X >= 0}; % in s
  POWERALPHA : TYPE = {X : REAL | X >= 1};
  ALARMSTATE : TYPE
    = {Alarm1_0, Alarm1_1, ..., Alarm1_N1,
       Alarm2_0, Alarm2_1, ..., Alarm2_N2,
       ...
       AlarmM_0, AlarmM_1, ..., AlarmM_NM};

  %Constants
  bigMax     : TIME = 60;

  %Function definitions
  ...

  %Clock sub-model
  ...

  %Alarm sub-models (alarm1 - alarmM)
  ...

  %Masking computation sub-model
  ...

  %Composition of the full system model
  system : MODULE = clock || alarm1 || ... || alarmM || maskingComputation;

  %Specification properties
  ...

END
```

Figure 4.5: An overview of the implementation of the formal modeling architecture (Fig. 4.3) as generated by our computer program. This implementation is written using the notation of SAL (see [20]). Note that in this listing (and all subsequent listings), code highlighting is used to improve readability. SAL language reserved words (including built-in basic types) are blue; declared types are dark blue; constants are green; functions are orange (these appear in subsequent listing); comments start with a % and are gray; and everything else is black. Ellipses "..." are used to indicate the omission of content that is either detailed in subsequent listings or indicates an incremental series of like components or operations (e.g. the synchronous compositions of the alarm sub-models: `alarm1 || ... || alarmM`).

are used by other elements in the modeling architecture for representing real-valued time (which cannot be negative), "power alpha" values, and alarm state. Note that alarm state assumes there are `M` alarms, where each alarm will have `N` states and `N` can be different for each alarm. This is followed by constant definitions. The model contains only one constant, `bigMax`, which represents an arbitrarily large maximum on how much time can advance in any given modeled step. The constant definitions are followed by function definitions. This represents the lookup table used for computing the "power alpha" values of masking curves. See Section 4.3.3 for a deeper discussion of how this is computed.

The clock sub-model, which is responsible for maintaining and advancing time, is next. It is described in Section 4.3.1. A series of sub-models representing the behavior of each alarm in the configuration come next. Each of these represents the behavior of a given alarm. Section 4.3.2 describes how each alarm is modeled. The masking computation sub-model follows and is responsible for determining if masking is occurring at any given clock-indicated time. This is described further in Section 4.3.3. All of the sub-models are ultimately synchronously composed into the complete system model.

Finally, specification properties are used to assert the absence of masking in a model. These properties are discussed in Section 4.3.4.

4.3.1 The Clock Sub-model

The clock sub-model is unchanged from the previous version of the method [38]. Please see Section 3.1.1 for a description of the clock sub-model.

4.3.2 The Alarm Sub-Models

As with the previous version of the method, the behavior of each alarm is described in separate sub-models, where each alarm model follows the same implementation pattern. The updated code pattern for representing the behavior of an alarm is show in Fig. 4.6. Each alarm has a constant value representing the length of its sounding cycle in seconds (`alarmXCycleTime`) which is set to a value `[TCycle]` derived by the description of the alarm behavior used in the generation process. Each alarm also has a variable representing its start time (`alarmXStartTime`). This is initially 0.

The alarm model is responsible for setting the start time and computing the amount of time the alarm has been sounding. Our model assumes that an alarm will sound for a single cycle and then stop (it can restart at any later time).

Thus, at any given `globalTime`, an alarm that is not sounding can begin sounding in the next state by setting the start time to the `globalTime` in the next state (see the code under `TRANSITION` in Fig. 4.6.). A start time greater than zero indicates that the alarm is sounding (`alarmXSounding = alarmXStartTime > 0`), information used by the masking computation sub-model. If sounding, the alarm computes how long it has been doing so as the difference between the global time and the alarm's start time (`alarmXTimeInCycle = globalTime - alarmXStartTime`). If the alarm is sounding and has not been sounding for longer than its cycle time in the next state's global time, the alarm keeps its current start time in the next state. If the alarm has been sounding for its full cycle time at the next global time, the alarm ceases to sound (sets the start time to zero) in the next state. Note that this behavior is the same as described in our original version of the method [38].

In the original method [38], an alarm would update its volume, frequency, and next time in response to the amount of time the alarm had been sounding. In the new method, the alarm model updates its state in response the alarms sounding time. Based on this state, the alarm sub-model computes its other output values: the "power alpha" (`alarmXPowerAlpha`) associated with that state (a value pre-computed by the generating computer program) and the alarm next time (`alarmXNextTime`; the time that the next state change will occur).

4.3.3 The Masking Computation Sub-model

The masking computation sub-model was updated from the previous method to account for the newly updated psychoacoustics. At every time assumed by the clock (`globalTime`), the masking computation model (Fig. 4.8) does two things. First, it uses the state and information of all of the alarms to determine if each alarm is being masked, where a Boolean variable associated with each alarm (`alarm1Masked` – `alarmNMasked`) is computed to be true or false if masking is or is not occurring respectively. The values of these variables are determined using the pre-computed "power alpha" lookup table (a function) generated by the computer program (see Fig. 4.7). Specif-

```
alarmX : MODULE =
  BEGIN

  INPUT  globalTime         : TIME

  LOCAL  alarmXStartTime    : TIME
  LOCAL  alarmXCycleTime    : TIME
  LOCAL  alarmXTimeInCycle  : TIME

  OUTPUT alarmXState        : ALARMSTATE
  OUTPUT alarmXSounding     : BOOLEAN
  OUTPUT alarmXPowerAlpha   : POWERALPHA
  OUTPUT alarmXNextTime     : TIME

  INITIALIZATION
    alarmXStartTime    =  0;

  DEFINITION
    alarmXCycleTime    =  alarmXTCycle;
    alarmXTimeInCycle  =  globalTime - alarmXStartTime;
    alarmXSounding     =  alarmStartTime > 0;
    alarmXState        =  IF    alarmXStartTime > 0 AND (alarmXTimeInCycle < alarmXTime1)
                             THEN alarmX_1
                          ELSIF alarm1StartTime > 0 AND (alarmXTimeInCycle < alarmXTime2)
                             THEN alarmX_2
                          ...
                          ELSE  alarmX_0 ENDIF;
    alarmXPowerAlpha   =  IF    alarmXState = alarmX_1  THEN alarmXPAlpha1
                          ELSIF alarmXState = alarmX_2  THEN alarmXPAlpha2
                          ...
                          ELSE 1 ENDIF;
    alarmXNextTime     =  IF    alarmXState = alarmX_1  THEN alarmStartTime + alarmXTime1
                          ELSIF alarmXState = alarmX_2  THEN alarmStartTime + alarmXTime2
                          ...
                          ELSE BigMax ENDIF;

  TRANSITION
    alarmStartTime' IN {X: TIME | ((alarmStartTime = 0)
                                    AND ((X = globalTime') OR (X = 0)))
                          OR ((alarmXStartTime > 0)
                              AND (globalTime' < (alarmXStartTime + alarmXCycleTime))
                              AND (X = alarmXStartTime))
                          OR ((alarmXStartTime > 0)
                              AND (globalTime' >= (alarmXStartTime + alarmXCycleTime))
                              AND (X = 0))};
END;
```

Figure 4.6: Generic SAL code for representing alarm behavior. `alarmX` is the name of the generic alarm that would be replaced with an actual alarm name in the generated model code. Note that red names represent alarm-dependent values that are inserted into the model by the computer program. `alarmXTCycle` represents the alarm's cycle time in seconds. `alarmXTime1` ... represent the times at which each corresponding alarm state ends. For example, the alarm is in state `alarmX_1` if `alarmXTimeInCycle < alarmXTime1`. `alarmXPAlpha1` ... represent the "power alpha" values associated with each state.

ically, for each possible maskee, the `thresholdPowerAlpha` function values are computed for each other alarm being treated as a masker and summed together. This sum is then compared to

```
thresholdPowerAlpha(maskerState : ALARMSTATE, maskeeState : ALARMSTATE): POWERALPHA
  = IF     maskerState = Alarm1_1  AND maskeeState = Alarm2_1 THEN powerAlpha1_1_2_1
    ELSIF maskerState = Alarm1_1  AND maskeeState = Alarm2_2 THEN powerAlpha1_1_2_2
    ...
    ELSE 1 ENDIF;
```

Figure 4.7: Lookup table implementation of Eq. (4.5) for use in the determination of whether masking (including addictive masking) is occurring. Note that this uses the masking curve formulation from Eq. (2.3) with the spreading function from Eq. (4.7) and the Δ from Eq. (4.8). The number of entries in this table is computed by the computer program based on the number of possible pairs between potential maskee and masker alarms. For example, `powerAlpha1_1_2_1` represents the pre-computed "power alpha" value associated with masker Alarm1 in state `Alarm1_1` and maskee Alarm2 in state `Alarm2_1`.

```
maskingComputation : MODULE =
BEGIN

  INPUT  globalTime        : TIME
  OUTPUT maxNextTime       : TIME

  INPUT  alarm1State       : ALARMSTATE
  INPUT  alarm1PowerAlpha  : POWERALPHA
  INPUT  alarm1NextTime    : TIME
  INPUT  alarm1Sounding    : BOOLEAN
  OUTPUT alarm1Masked      : BOOLEAN
  ...

  DEFINITION
    alarm1Masked = alarm1Sounding AND (thresholdPowerAlpha(alarm2State, alarm1State)
                    ...
                    + thresholdPowerAlpha(alarmMState, alarm1State)) >= alarm1PowerAlpha;
    ...

    maxNextTime      IN  {x: TIME | (x = alarm1NextTime OR ... OR x = alarmMNextTime)
                              AND (x <= alarm1NextTime AND ... AND x <= alarmMNextTime)};

END;
```

Figure 4.8: Generic SAL code for the masking computation sub-model in the new method.

the given maskee's "power alpha" value. If the sum is greater than or equal to this value, then the Boolean variable is true. Otherwise it is false.

The second responsibility of the masking computation sub-model is to calculate the next time variable (`maxNextTime`) to represent the maximum amount of time the clock can advance to in the next step. This is done by selecting the minimum of all of the alarm next times.

```
alarmXPartialMasking : THEOREM system
  |- G(NOT(alarmXPowerAlpha > 1 AND alarmXMasked));
alarmXTotalMasking   : THEOREM system
  |- G(NOT((NOT alarmXSounding)
       AND X(alarmXSounding AND alarmXMasked
             AND U(alarmXSounding AND alarmXMasked,
                   NOT alarmXSounding))));
```

Figure 4.9: Specification property patterns for a given alarm (`alarmX`). `alarmXPartialMasking` asserts the absence of any masking for a given alarm. `alarmXTotalMasking` asserts that a given alarm will never be masked over its entire sounding cycle. Note that `alarmXPartialMasking` is different from the comparable property in Fig. 3.8. Specifically, the `alarmXPowerAlpha` value is used to determine if the alarm is actually making noise instead of the alarm's volume parameter.

4.3.4 Specification Properties

In the original method, specification properties had to be manually created using patterns. In the new version, our computer program generates the properties automatically. Further, the properties have been slightly modified to account for the changes in the psychoacoustics that were used. For each alarm, two properties are generated (Fig. 4.9).

The first (`alarmXPartialMasking`) is used to detect any masking that can occur. This uses linear temporal logic, it assert that, for all paths through the model (**G**), there should never be a situation where the alarm is making noise (`alarmXPowerAlpha > 1`) and the alarm is masked. The second specification (`alarmTotalMasking`) is meant to check that an alarm is never masked for its entire sounding cycle. This states that, for all (**G**) paths through the model, we never want it to be true that the alarm goes from not sounding to sounding and masked in the next (**X**) state such that, from then on, the alarm is sounding and masked until (**U**) it is no longer sounding.

4.3.5 Running the Model Checker

Running the model checker is done the same as in the previous version of the method. Please see Section 3.1.5.

4.4 Counterexample Visualization

In the original method, a model checker produced counterexample needed to manually interpreted by analysts. We found that a variation of a vertical bar graph could be used to effectively show how alarms in a counterexample sounded in relation to each other and indicate when masking was occurring [38]. Thus, in our new version of the method, we have added the ability to automatically create these graphs from a counterexample input.

Created graphs list the names each alarm in a configuration along the vertical axis. Time is shown on the horizontal axis. Bars are plotted in the chart to show when the tones of each alarm are sounding. Smaller black lines are overlaid on the bars to show when a particular tone (or part of a tone) is masked. Examples of these plots can be seen later in Figs. 4.10 and 4.11.

4.5 Application Case Studies

Below we present three different case studies that illustrate the power of the presented method. First, we evaluate the case study originally presented by Hasanain et al. [38] to show both that the new formulation of the method can detect the masking of the original, but also that it does so more quickly. Second, we apply the method to a simple case study that demonstrates the ability of the method to detect additive masking. Third, we apply the method to a realistic application based on the GE CARESCAPE™Monitor B850 [35], a telemetry monitoring system.

In all of the reported case studies, as with the previous method's analyses, formal verifications was performed using SAL's infinite bounded model checker on a Linux workstation with a 3.3 GHz Intel Xeon processor and 64 GB of RAM.

4.5.1 Case Study 1: The Original Application

In the case study originally presented in [38] (and the previous chapter), there were three alarms (Table 3.1). All three of these had a cycle featuring two tones separated by a pause. Each tone frequencies, tone and pause lengths, and volumes were all consistent with those commonly found in medical alarms [44, 51].

Table 4.1: Case Study 1 Verification Results

Model Alarms	Alarm	Masking Spec.	Original Output Time (s)	Original Output Result	New Outputs Time (s)	New Outputs Result	Decrease
1 & 2	1	Partial	87.26	✓	0.15	✓	99.83%
		Total	63.76	✓	0.11	✓	99.83%
	2	Partial	99.31	×	0.47	×	99.53%
		Total	56.11	✓	0.24	✓	99.57%
1 & 3	1	Partial	67.05	✓	0.11	✓	99.84%
		Total	32.88	✓	0.12	✓	99.64%
	3	Partial	66.37	✓	0.16	✓	99.76%
		Total	127.57	✓	0.1	✓	99.92%
2 & 3	2	Partial	180.56	×	1.24	×	99.31%
		Total	95.82	✓	0.17	✓	99.82%
	3	Partial	85.49	✓	0.15	✓	99.82%
		Total	69.52	✓	0.09	✓	99.87%
1, 2, & 3	1	Partial	392.76	✓	6.65	✓	98.31%
		Total	320.62	✓	1.58	✓	99.51%
	2	Partial	1281.26	×	89.97	×	92.98%
		Total	492.76	×	148.29	×	69.91%
	3	Partial	297.92	✓	3.74	✓	98.74%
		Total	1205.43	✓	1.46	✓	99.88%

Note. ✓ indicates a verification confirmation and × indicates a verification failure with a counterexample.

When evaluated with the original method, analyses were conducted on four different models. One model for each possible pair of alarms from Table 3.1 and one with all three. Each of these models were also modeled and evaluated with our new method to ensure the same analysis results were achieved and to compare verification times. A comparison of the analysis results with both methods can be seen in Table 4.1. These show that each specification property produced the same outcome when verified. Further, an examination of the counterexamples with our visualizer revealed that they both discovered the same masking conditions. Alarm 2 can be partially masked by Alarm 1 when Alarm 2's second tone overlaps with Alarm 1's first tone. Alarm 2 is partially masked by Alarm 3 when Alarm 2's first tone overlaps with Alarm 3's second tone. With specific timing such that Alarm 2's first tone completely overlaps with Alarm 3's second tone and Alarm 2's second tone completely overlaps with Alarm 1's first tone, Alarm 2 is completely masked. An illustration of this condition found by both versions of the method can be seen in Fig. 3.9.

The results in Table 4.1 also demonstrate the scalability improvements of our new method. Specifically, the new method was able to perform all of the analyses substantially faster than the original method, with reductions in verification times from 69.96% to 99.92 %.

This case study illustrates the performance improvements that were achieved with the new version of the system. It also shows that comparable detection capabilities exist in this new version. However, this case study does not illustrate the additive masking detection capabilities of the method because, as the result in Fig. 3.9 show, masking does not require multiple maskers to overlap to produce masking. Additive masking detection is demonstrated in the next case study.

4.5.2 Case Study 2: Additive Masking Detection

The second case study evaluated the alarms shown in Table 4.2. These particular alarms were chosen because, with the exception of minor timing differences, these alarms are consistent with reserved alarm sounds from the international medical alarm standard [44].

Because we were particularly interested in seeing if our method could detect additive masking, we used these alarms to construct four different models. One for each possible pair of alarms and one with all three alarms. By using our method to evaluate all four of these models we were able to determine if our method could find additive masking. Specifically, if we found masking that occurred due to the overlapping of two or more masking alarms with a maskee, where masking did not occur when each potential masker alone overlapped the maskee, then our method could find additive masking conditions.

We checked the specification properties for each alarm (for both partial and total masking) in each model created as part of our method. For models containing two alarms, verification search depths were set to 12. The model with all three alarms used a search depth of 18 for all verifications. Verification results are shown in Table 4.3.

These results reveal that the only masking that occurs happens when all three alarms are in the model. The results also show that only partial masking occurs for Alarms B and C. When

Table 4.2: Case Study 2 Alarm Configuration

Name	Freq. (Hz)	Vol. (dB)	Time (s)
Alarm A	261	84	0.1
	0	0	0.1
	329	84	0.1
	0	0	0.1
	392	84	0.1
Alarm B	261	84	0.1
	0	0	0.1
	329	84	0.1
	0	0	0.1
	293	84	0.1
Alarm C	523	84	0.1
	0	0	0.1
	293	84	0.1
	0	0	0.1
	392	84	0.1

Table 4.3: Case Study 2 Verification Results

Model		Masking	Verification Output	
Alarms	Alarm	Spec.	Time (s)	Outcome
A & B	Alarm A	Partial	8.37	✓
		Total	2.98	✓
	Alarm B	Partial	4.31	✓
		Total	1.37	✓
A & C	Alarm A	Partial	5.25	✓
		Total	1.70	✓
	Alarm C	Partial	3.81	✓
		Total	1.70	✓
B & C	Alarm B	Partial	6.37	✓
		Total	1.61	✓
	Alarm C	Partial	4.79	✓
		Total	2.20	✓
A , B & C	Alarm A	Partial	248.2	✓
		Total	24.8	✓
	Alarm B	Partial	1583.8	×
		Total	13.3	✓
	Alarm C	Partial	1410.1	×
		Total	31.6	✓

Note. ✓ indicates a verification confirmation and × indicates a verification failure with a counterexample.

the counterexamples associated with these verification failures were visualized (Fig. 4.10) they showed that masking only occurred as a result of additive masking. Specifically, partial masking of Alarm B's third tone occurs when it sounds concurrently with Alarm A's first tone and Alarm C's second tone. Alarm C's second tone can be partially masked if it sounds concurrently with Alarm A's first tone and Alarm B's third tone. Since no masking occurs in the models that only contain two alarms, this indicates that the observed masking is additive.

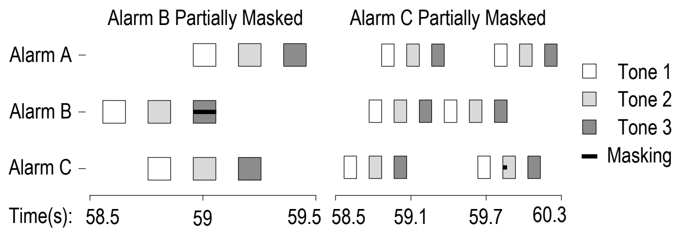

Figure 4.10: Method-created counterexample visualization of how Alarms B and C can be partially masked by additive masking caused by the other two alarms in the configuration.

It is important to note that the partial masking shown in the two plots of Fig. 4.10 both occur in the same condition: the concurrent sounding of Alarm A's first tone, Alarm B's third tone, and Alarm C's second tone. This would mean that, when actually heard by a human, that either of the potentially masked tones (Alarm B's third or Alarm C's second) would be unhearable. It is a potential limitation of the method that it cannot identify which of these would actually be masked. Despite this limitation, the presented analyses do demonstrate the ability of the method to detect additive masking. However, even though the included alarms are realistic, this case study is still artificial. The ability of the method to detect masking in a more realistic context is explored in the third case study.

4.5.3 Case Study 3: The GE CARESCAPE$^{\text{TM}}$Telemetry Monitor

To evaluate a realistic application, we used our updated method to analyze the alarms in the GE CARESCAPE$^{\text{TM}}$Monitor B850 [35], an telemetry monitoring system compatible with the international medical alarm standard [44]. The GE monitor had the alarms shown in Table 4.4. There were four high-priority alarms that each played the same ten-tone alarm melodies (with the same timings), one medium-priority alarm with a three-tone melody, and one low-priority alarm with

Table 4.4: Alarms from Case Study 3, the GE CARESCAPE Telemetry Monitoring System

Name	Freq. (Hz)	Vol. (dB)	Time (s)	Name	Freq. (Hz)	Vol. (dB)	Time (s)	Name	Freq. (Hz)	Vol. (dB)	Time (s)
CPU-C1	523	72	0.1	D15K	523	81	0.1	D19KT	523	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	698	72	0.1		698	81	0.1		698	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	784	72	0.1		784	81	0.1		784	82	0.1
	0	0	0.3		0	0	0.3		0	0	0.3
	880	72	0.1		880	81	0.1		880	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	988	72	0.1		988	81	0.1		988	82	0.1
	0	0	1		0	0	1		0	0	1
	523	72	0.1		523	81	0.1		523	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	698	72	0.1		698	81	0.1		698	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	784	72	0.1		784	81	0.1		784	82	0.1
	0	0	0.3		0	0	0.3		0	0	0.3
	880	72	0.1		880	81	0.1		880	82	0.1
	0	0	0.1		0	0	0.1		0	0	0.1
	988	72	0.1		988	81	0.1		988	82	0.1
	0	0	5		0	0	5		0	0	5
SystemHigh	523	84	0.1	SystemMedium	523	83	0.2	SystemLow	523	79	0.2
	0	0	0.1		0	0	0.2				
	698	84	0.1		784	83	0.2				
	0	0	0.1		0	0	0.2				
	784	84	0.1		988	83	0.2				
	0	0	0.3		0	0	19				
	880	84	0.1								
	0	0	0.1								
	988	84	0.1								
	0	0	1								
	523	84	0.1								
	0	0	0.1								
	698	84	0.1								
	0	0	0.1								
	784	84	0.1								
	0	0	0.3								
	880	84	0.1								
	0	0	0.1								
	988	84	0.1								
	0	0	5								

Note. CPU-C1, D15K, D19KT, and SystemHigh are high-priority alarms. SystemMedium is a medium-priority alarm. SystemLow is a low priority alarm.

one tone. Our analyses assumed that any of these alarms could sound simultaneously.

We modeled these alarms in our new method and evaluated each alarm to see if they were ever partially or totally masked. Because of the complexity of the model, we anticipated scalability problems. Thus we attempted to minimize the search depth used in the verifications. Specifically, all properties were verified iteratively starting with the minimum depth capable of detecting masking. If no masking was found, the search depth was increased by one for each verification until masking was discovered or the verification took a prohibitively long amount of time. For partial masking properties, this meant search depths started at 2 and increased from there. For total masking properties, search depths started at the total number of states in the associated alarm and were iteratively increased up to 21. Note that search depths greater than 21 were not considered because of the amount of time required for the analyses. Further, because a depth of 21 would encapsulate what was likely to be the worst possible masking condition for the three high-priority alarms (when they all sounded at the same time as each other due to them all having the same tones), this was seen as sufficient.

Table 4.5: Case Study 3 Verification Results

Alarm	Masking Spec.	Search Depth	Time (s)	Outcome
CPU-C1	Partial	2	145.70	×
	Total	21	60,967.05	×
D15K	Partial	2	135.21	×
	Total	21	14,5870.50	✓
D19KT	Partial	2	135.21	×
	Total	21	14,8252.81	✓
SystemHigh	Partial	2	139.02	×
	Total	21	395,441.48	✓
SystemMedium	Partial	2	104.24	×
	Total	21	203,702.73	✓
SystemLow	Partial	2	81.24	×
	Total	4	216.66	×

Note. Because a full search depth of $21 \times 4 + 7 + 2 = 93$ was not used in these analyses, it is possible (but unlikely) that a verified property might be untrue at a higher search depths. Thus, verified properties are only guaranteed to be true at the presented search depth.

A summary of our results can be seen in Table 4.5. These show that masking is possible between the alarms of the GE CARESCAPE. Specifically, partial masking was observed for all of the alarms. Two of the alarms can be totally masked: CPU-C1 and SystemLow (Fig. 4.11(b) and (h) respectively). This is concerning because it means that these alarms may not be heard or responded to by an observer. Because SystemLow is a low-priority alarm, one could argue that it being masked by higher priority alarms is not of that much concern. However, CPU-C1 is a high-priority alarm. The length of this alarm's cycle is 8.2 seconds. This means that if the alarm is masked, someone will not respond to it for at least that long. In a safety critical medical environment, this is an extremely long time. As such, the ability of CPU-C1 to be completely masked represents a significant patient safety problem with this device.

4.6 Discussion

The work presented in this chapter has introduced a novel extension of our original method that accounts for its shortcomings. By changing the way that masking detection is performed in our computational model, we are now able to properly account for the additive effect of multiple maskers as well as use a more appropriate spreading function. This means that our masking prediction is more accurate than in the previous version and can thus detect masking conditions that it could not previously. In implementing our new approach to masking detection, we have also improved the scalability of our method. This improves the usefulness and applicability of the method. By providing a computer program that can generate formal models from alarm configuration descriptions and visualize counterexamples, we have improved the usability of the method.

In addition to these contributions, we have provided three different case studies that demonstrate different capabilities of the method. The first case study illustrates the significant scalability improvements that were achieved without any loss of detection capabilities. The second one shows that the method is indeed capable of detecting additive masking in a simple alarm configuration. Finally, the last case study showed how the method could be used to find masking conditions in a real application. As such, the method clearly has utility and, if used by medical device engineers and/or hospitals to evaluate and design alarm configurations, could significantly increase the chance that medical alarms are perceivable. This could have a profound impact on patient safety.

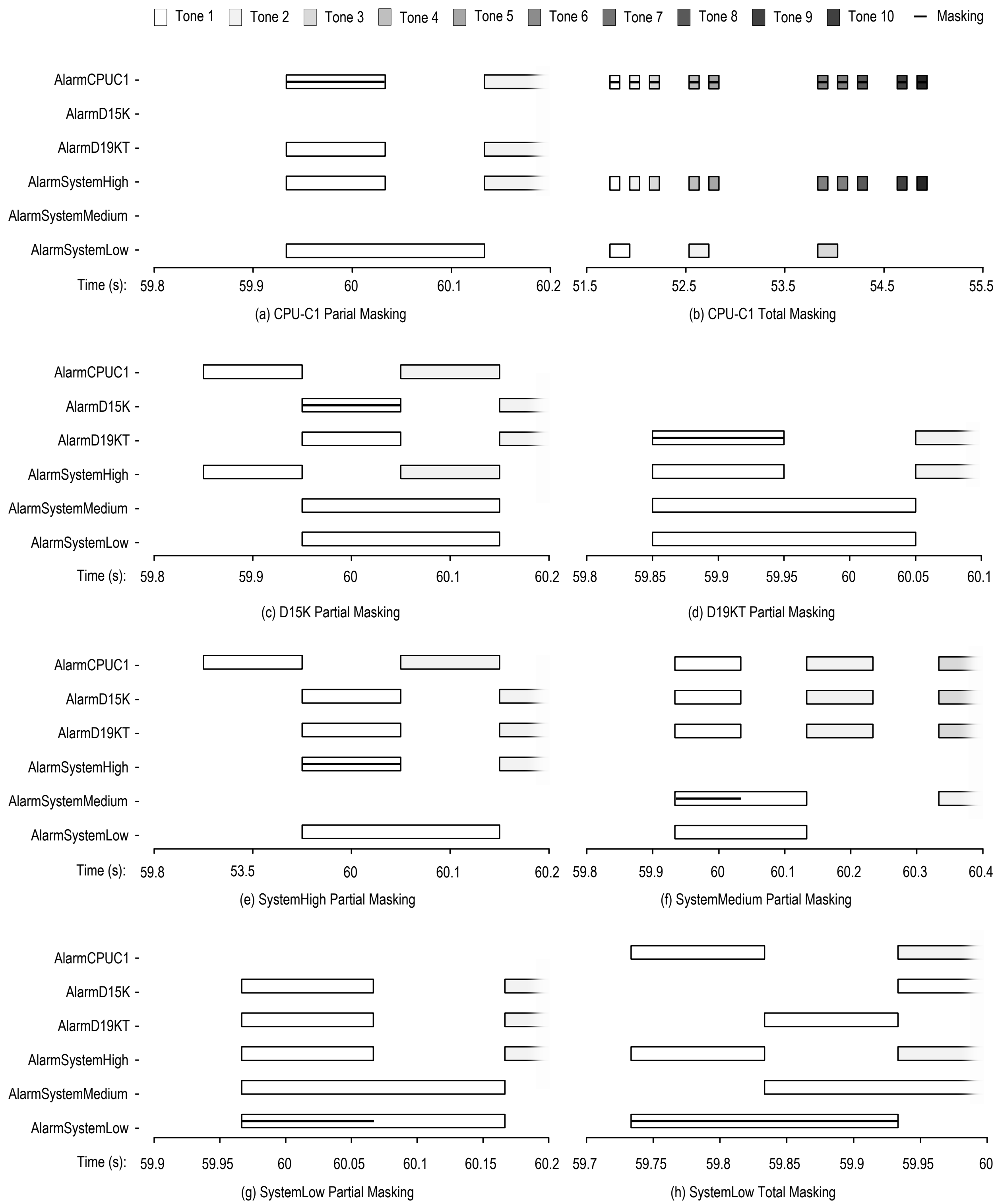

Figure 4.11: Method-created counterexample visualizations of how the Alarms in the GE CARESCAPE can mask each other. Note that only masking found in the respective analyses are shown in the graphs. The x-axis of the plots was reduced to only show the period of masking and not the full sounding cycles of the alarms.

CHAPTER 5: CONCLUSIONS AND GENERAL DISCUSSION

This dissertation has introduced an unprecedented way of synergistically using psychoacoustic modeling and model checking to detect medical alarm masking. In doing this, it provides analysts with an innovative solution to a problem that was not previously possible. As the number of alarms in medical environments increases and causes more and more alarm interactions, there will be even more chances for alarm masking conditions. As such, the presented method could be used by hospital personnel or medical system designers to evaluate the safety of different medical alarm configurations by considering all of the possible alarm interactions. Thus, this work has the potential to significantly improve patient safety. Further, our method is a contribution because it represents the first successful attempt to model psychophysical concepts in a formal model.

Beyond the method itself, this work has made a number of significant research contributions: (a) a unique way of accounting for non-linear psychoacoustics in formal models in a way that is scalable; (b) methods for specifying the absence of both partial and total masking; (c) a software program that facilitates the easy creation of formal model representing an alarm configuration; and (d) a new way of visualizing counterexamples to help analysts interpret analysis results.

To date the work has produced two published papers. One conference paper and one journal paper:

- B. Hasanain, A. Boyd, and M. L. Bolton. An approach to model checking the perceptual interactions of medical alarms. In *Proceedings of the 2014 International Annual Meeting of the Human Factors and Ergonomics Society*, pages 822–826, Santa Monica, 2014. HFES

- B. Hasanain, A. Boyd, and M. Bolton. Using model checking to detect simultaneous masking in medical alarms. *IEEE Transactions on Human-Machine Systems*, 2015. ISSN 2168-2291. doi: 10.1109/THMS.2014.2379661

An additional journal paper is under review:

- B. Hasanain, A. Boyd, J. Edworthy, and M. Bolton. A formal approach to discovering simultaneous additive masking between auditory medical alarms. *Applied Ergonomics*, ND. Under Review

Because the presented work is so unique, it also opens up many possibilities for future extensions and research. These are discussed below.

5.1 Scalability

Although our new version of the method significantly improved the method's scalability, the results shown for the third case study still indicate that the method does not scale well. The approach for iteratively increasing the search depth should allow analysts to mitigate some scalability concerns. However, scalability will definitely limit what systems the method can be applied to and when it will be appropriate. For example, the time required to run an analysis on a complex application would likely take a prohibitively long time for use in a dynamic environment like a hospital. There may be additional ways to improve the scalability of the method. Compositional verification [18] is a process where large models can be verified using smaller independently verified, model components. As such, it may be possible to use compositional verification to check the interactions of specific tones across multiple analyses and use these analyses to draw conclusions about more complex configurations that use the analyzed tones. Many alarms, like those in the GE CARESCAPE, can have repeated patterns both within and between alarms. Thus, it may be possible to exploit symmetry in the models to further improve the scalability of the method [33]. Future efforts should investigate how these and other approaches could be used to improve our method's scalability.

Even if scalability persists as a limitation, this does not preclude the usefulness of the method. Specifically, the method could still be used for simple alarm configurations. Alternatively, system designers should have enough time to evaluate designed systems with other common alarm sounds to ensure they avoid masking. Further, analyses could target alarm standards so that, although analyses may take a long time, their results could influence standard development and thus impact the safety of medical alarms across the industry without the method needing to be used by designers or hospitals. This is further explored in the next section. Thus, while future efforts may improve the scalability of the method, the method should still have utility.

5.2 The International Medical Alarm Standard

The analyses presented under case studies 2 and 3 have interesting implications for future analyses. Specifically, the GE CARESCAPE alarms (case study 3) are in conformance with the IEC 60601-1-8 international medical alarm standard [44]. The alarms in case study 2 are nearly identical to reserved alarms sounds from IEC 60601-1-8 (there are only minor and likely inconsequential timing difference). Because masking was detected in both of these case studies, this would indicate that there are masking problems for alarms designed to adhere to the standard. This is potentially very dangerous. The method presented here offers the capabilities to systematically evaluate the alarm requirements and reserved sounds found in IEC 60601-1-8 and potentially explore solutions to discovered problems. This should be a priority of future work.

5.3 More Complex Alarm Behavior and Sounds

There are other types of alarms beyond the tonal sounds currently supported by the method. In fact, features of IEC 60601-1-8 are not currently supported by the method. Specifically IEC 60601-1-8 alarms can have sub-frequencies, additional simultaneous frequencies that make each tonal sound more complicated [44]. Given that the method currently supports additive masking, accounting for these sub-harmonics should be an easy extension of the method. This should be the subject of future work. Additionally, there are a number of different spreading functions for representing the masking properties of different types (tonal vs noisy) sounds [13]. Thus future extensions of the method should enable the ability to account for different types of sounds. Finally, the current formulation of our method works with discrete transitions in alarm state. However, alarm sounds can have many dynamic elements related to the frequencies and volumes of different components of a sound. Accommodating these dynamics would require a significant change in the way the method models alarms. Future work should investigate if there are formal modeling abstraction techniques that can be used to model more dynamic types of sounds in our method.

5.4 Additional Masking Detection

Our method is capable of detecting simultaneous masking. However, there is also a phenomena called temporal masking [34]. However, background and transient noises in health care environments can mask alarms or exacerbate other masking conditions [8, 48]. There is also a phenomenon known as temporal masking [34]. Temporal masking describes a situation where non-concurrent sounds, but ones that sounds in close temporal proximity, are masked. Such a phenomenon could increase the instances of masking in a given configuration. Psychoacoustics exist for accounting for these factors, however they are not readily adaptable to formal modeling. Thus, future work could investigate how to include these in our method using either extended formal modeling techniques or through clever exploitation of other analysis approaches with formal verification.

5.5 Deeper Analysis Support

As illustrated in the result for case study 2, our method can detect masking conditions where it is unclear which of two or more alarms would actually be masked. Future work should investigate how to disambiguate such analysis results. Additionally, it is the nature of the model checking analyses that they only ever find one instance of a problem (a specification violation). This means that there could be more masking conditions in any given configuration than those initial found by the method. Ideally, our method would be able to find all of the conditions in a configuration that produce masking. Such capabilities should be investigated in the future. Finally, while our method can find masking problems, it is not clear how the method should be used to find solutions to those problems. Future work should focus on extending the method to support the exploration of design solutions that will avoid masking.

5.6 Experimental Validation

The psychoacoustics used in our method have been well validated over the years and have served as the basis for the MPEG family of lossy audio codecs [13]. Thus, we expect our method to give accurate masking predictions. However, experiments with actual human subjects in realistic listening environments would allow us to validate our method's findings. This should be the subject of future work.

5.7 Other Application Domains

The work presented here and in previous analyses [37, 38] have focused on medical alarms. However, the perceivability of alarms can play a critical role in the safety of aviation [5], automobile [6], and industrial systems [59]. Thus, future work should investigate how our method could be used in these and other safety critical domains.

5.8 Other Alarm Problems

There are many problems facing medical alarms beyond simultaneous masking [29]. Because our work constitutes the first attempt to address alarm problems formally, there may be many future opportunities for extending our work to explore other alarm issues. In particular, there is good evidence suggesting that human mental workload can contribute to alarm mistrust, fatigue, and inattentional deafness [7, 19, 22, 29, 66]. Formal methods could help researchers discover when these conditions could occur. Such an analysis will need to integrate formal approaches for modeling alarm perception [37, 38], workload [41, 53], and task behavior [9–11] to be successful. This should be explored in future work.

BIBLIOGRAPHY

[1] P. P. S. Advisory. Connecting remote cardiac monitoring issues with care areas. *PA Patient Safety Authority*, 6(3):79–83, 2009.

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

[3] E. Ambikairajah, A. Davis, and W. Wong. Auditory masking and mpeg-1 audio compression. *Electronics & communication engineering journal*, 9(4):165–175, 1997.

[4] F. Baumgarte, C. Ferekidis, and H. Fuchs. A nonlinear psychoacoustic model applied to ISO/MPEG layer 3 coder. In *Proceedings of the Audio Engineering Society Convention*, New York, 1995. Audio Engineering Society.

[5] J. P. Bliss. Investigation of alarm-related accidents and incidents in aviation. *The International Journal of Aviation Psychology*, 13(3):249–268, 2003.

[6] J. P. Bliss and S. A. Acton. Alarm mistrust in automobiles: How collision alarm reliability affects driving. *Applied Ergonomics*, 34(6):499–509, 2003.

[7] J. P. Bliss and M. C. Dunn. Behavioural implications of alarm mistrust as a function of task workload. *Ergonomics*, 43(9):1283–1300, 2000.

[8] F. E. Block. Evaluation of users' abilities to recognize musical alarm tones. *Journal of Clinical Monitoring and Computing*, 8(4):285–290, 1992.

[9] M. L. Bolton and E. J. Bass. Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 43(6):1314–1327, 2013.

[10] M. L. Bolton, R. I. Siminiceanu, and E. J. Bass. A systematic approach to model checking human-automation interaction using task-analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 41(5):961–976, 2011.

[11] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu. Using phenotypical erroneous human behavior generation to evaluate human-automation interaction using model checking. *International Journal of Human-Computer Studies*, 70:888–906, 2012.

[12] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu. Using formal verification to evaluate human-automation interaction in safety critical systems, a review. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 43:488–503, 2013.

[13] M. Bosi and R. E. Goldberg. *Introduction to digital audio coding and standards*, volume 721 of *The Springer International Series in Engineering and Computer Science*. Springer, New York, 2003.

[14] K. Brandenburg and M. Bosi. Overview of MPEG audio: Current and future standards for low bit-rate audio coding. *Journal of the Audio Engineering Society*, 45(1/2):4–21, 1997.

[15] K. Brandenburg and G. Stoll. ISO/MPEG-1 audio: A generic standard for coding of high-quality digital audio. *Journal of the Audio Engineering Society*, 42(10):780–792, 1994.

[16] R. P. Carlyon, S. McAdams, R. P. Carlyon, and S. McAdams. The psychophysics of concurrent sound segregation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 336(1278):347–355, 1992.

[17] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, 1999.

[18] J. Cobleigh, D. Giannakopoulou, and C. Păsăreanu. Learning assumptions for compositional verification. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 331–346, 2003.

[19] M. Cvach. Monitor alarm fatigue: An integrative review. *Biomedical Instrumentation & Technology*, 46(4):268–277, 2012.

[20] L. De Moura, S. Owre, and N. Shankar. The SAL language manual. Technical Report CSL-01-01, Computer Science Laboratory, SRI International, Menlo Park, 2003.

[21] L. De Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In *Proceedings of the 16th International Conference on Computer Aided Verification*, pages 496–500. Springer, 2004.

[22] F. Dehais, M. Causse, F. Vachon, N. Régis, E. Menant, and S. Tremblay. Failure to detect critical auditory alerts in the cockpit evidence for inattentional deafness. *Human Factors*, 56 (4):631–644, 2014.

[23] B. Dutertre and M. Sorea. Timed systems in SAL. Technical Report NASA/CR-2002-211858, SRI Internation, 2004.

[24] E. Zwicker and R. Feldtkeller. *Das Ohr als Nachrichtenempfnger*. Stuttgart, Hirzel Verlag, Germany, 1967.

[25] ECRI Institute. Top 10 health technology hazards for 2013. *Health Devices*, 41, 2012.

[26] ECRI Institute. Alarm safety resource site: Guidance and tools to help healthcare facilities improve alarm safety, 2013. Retrieved May 27, 2013, from `https://www.ecri.org/Forms/Pages/Alarm_Safety_Resource.aspx`.

[27] J. Edworthy. The design and implementation of non-verbal auditory warnings. *Applied ergonomics*, 25(4):202–210, 1994.

[28] J. Edworthy. Designing effective alarm sounds. *Biomedical Instrumentation & Technology*, 45(4):290–294, 2011.

[29] J. Edworthy. Medical audible alarms: A review. *Journal of the American Medical Informatics Association*, 2012.

[30] J. Edworthy and E. Hellier. Fewer but better auditory alarms will improve patient safety. *Quality and Safety in Health Care*, 14(3):212–215, 2005.

[31] J. Edworthy and E. Hellier. Alarms and human behaviour: Implications for medical alarms. *British Journal of Anaesthesia*, 97(1):12–17, 2006.

[32] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, A. R. Meyer, M. Nivat, M. Paterson, and D. Perrin, editors, *Handbook of Theoretical Computer Science*, chapter 16, pages 995–1072. MIT Press, Cambridge, 1990.

[33] E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal methods in system design*, 9(1):105–131, 1996.

[34] H. Fastl and E. Zwicker. *Psychoacoustics: Facts and models*, volume 22. Springer, 2006.

[35] GE Healthcare. CARESCAPE™monitor B850 technical specifications supplement. Technical Report 2040386-084D, 2010.

[36] D. M. Green. Additivity of masking. *The Journal of the Acoustical Society of America*, 41: 1517–1525, 1967.

[37] B. Hasanain, A. Boyd, and M. L. Bolton. An approach to model checking the perceptual interactions of medical alarms. In *Proceedings of the 2014 International Annual Meeting of the Human Factors and Ergonomics Society*, pages 822–826, Santa Monica, 2014. HFES.

[38] B. Hasanain, A. Boyd, and M. Bolton. Using model checking to detect simultaneous masking in medical alarms. *IEEE Transactions on Human-Machine Systems*, 2015. ISSN 2168-2291. doi: 10.1109/THMS.2014.2379661.

[39] B. Hasanain, A. Boyd, J. Edworthy, and M. Bolton. A formal approach to discovering simultaneous additive masking between auditory medical alarms. *Applied Ergonomics*, ND. Under Review.

[40] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, Washington, 1996. IEEE Computer Society. ISBN 0-8186-7463-6.

[41] A. Houser, L. M. Ma, K. Feigh, and M. L. Bolton. A formal approach to modeling and analyzing human taskload in simulated air traffic scenarios. In *Proceedings of the IEEE International Conference on Complex Systems Engineering*, Piscataway, 2015. IEEE. (6 pages). In Press.

[42] L. G. Huettel and L. M. Collins. Using computational auditory models to predict simultaneous masking data: Model comparison. *IEEE Transactions on Biomedical Engineering*, 46 (12):1432–1440, 1999.

[43] L. E. Humes and W. Jesteadt. Models of the additivity of masking. *The Journal of the Acoustical Society of America*, 85(3):1285–1294, 1989.

[44] IEC 60601-1-8. *Medical electrical equipment - Part 1-8*. International Electrotechnical Commission, Geneva, 2003-08-14.

[45] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, 1993.

[46] A. Konkani, B. Oakley, and T. J. Bauld. Reducing hospital noise: A review of medical device alarm management. *Biomedical Instrumentation & Technology*, 46(6):478–487, 2012.

[47] P. Lacherez, E. Seah, and P. Sanderson. Overlapping melodic alarms are almost indiscriminable. *Human Factors*, 49(4):637–645, 2007.

[48] C. Laroche, H. T. Quoc, R. Hétu, and S. McDuff. 'detectsound': A computerized model for predicting the detectability of warning signals in noisy workplaces. *Applied Acoustics*, 32(3): 193–214, 1991.

[49] R. A. Lutfi. Additivity of simultaneous masking. *The Journal of the Acoustical Society of America*, 73:262, 1983.

[50] C. Meredith and J. Edworthy. Are there too many alarms in the intensive care unit? an overview of the problems. *Journal of advanced nursing*, 21(1):15–20, 1995.

[51] K. Momtahan, R. Hetu, and B. Tansley. Audibility and identification of auditory alarms in the operating room and intensive care unit. *Ergonomics*, 36(10):1159–1176, 1993.

[52] B. C. J. Moore. Masking in the human auditory system. In *Proceedings of the Audio Engineering Society Conference: Collected Papers on Digital Audio Bit-Rate Reduction*, pages 9–19, New York, 1996. Audio Engineering Society.

[53] J. Moore, R. Ivie, T. Gledhill, E. Mercer, and M. Goodrich. Modeling human workload in unmanned aerial systems. In *2014 AAAI Spring Symposium Series*, Palo Alto, 2014. AAAI.

[54] L. Moura, S. Owre, H. Rue, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In *Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 496–500. Springer, Berlin, 2004.

[55] A. J. Oxenham and B. C. J. Moore. Modeling the additivity of nonsimultaneous masking. *Hearing research*, 80(1):105–118, 1994.

[56] A. J. Oxenham and B. C. J. Moore. Additivity of masking in normally hearing and hearing-impaired subjects. *The Journal of the Acoustical Society of America*, 98:1921, 1995.

[57] R. D. Patterson, T. F. Mayfield, R. D. Patterson, and T. F. Mayfield. Auditory warning sounds in the work environment [and discussion]. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 327(1241):485–492, 1990.

[58] A. Podelski and S. Wagner. Model checking of hybrid systems: From reachability towards stability. In *Hybrid Systems: Computation and Control*, pages 507–521. Springer, 2006.

[59] D. H. Rothenberg. *Alarm management for process control: a best-practice guide for design, implementation, and use of industrial alarm systems*. Momentum Press, New York, 2009.

[60] RSSB. Alarms and alerts guidance and evaluation tool, 2009. URL `http://www.rssb.co.uk/sitecollectiondocuments/pdf/research-toolkits/T326/main.html`. Accessed 23 May 2012.

[61] M. R. Schroeder, B. S. Atal, and J. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *The Journal of the Acoustical Society of America*, 66: 1647–1652, 1979.

[62] W. W. Stead, H. S. Lin, et al. *Computational technology for effective health care: Immediate steps and strategic directions*. National Academies Press, Atlanta, 2009.

[63] E. Terhardt. Calculating virtual pitch. *Hearing Research*, 1(2):155–182, 1979.

[64] The Joint Commission. Medical device alarm safety in hospitals. *Sentinel Even Alert*, 50, April 8, 2013.

[65] O. Toor, T. Ryan, and M. Richard. Auditory masking potential of common operating room sounds: A psychoacoustic analsysis. In *Anesthesiology 2008*, volume 109, page A1207, Park Ridge, 2008. American Society of Anesthesiologists.

[66] R. B. Way, S. A. Beer, and S. J. Wilson. Whats that noise? Bedside monitoring in the emergency department. *International Emergency Nursing*, 22(4):197–201, 2014.

APPENDICES

IEEE Requesting permission to reuse content from an IEEE publication

Title: Using Model Checking to Detect Simultaneous Masking in Medical Alarms

Author: Hasanain, B.; Boyd, A.D.; Bolton, M.L.

Publication: Human-Machine Systems, IEEE Transactions on

Publisher: IEEE

Copyright © 1969, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

SAGE

Title: An Approach to Model Checking the Perceptual Interactions of Medical Alarms:

Author: Bassam Hasanain, Andrew D. Boyd, Matthew L. Bolton

Publication: Proceedings of the Human Factors and Ergonomics Society Annual Meeting

Publisher: SAGE Publications

Date: 09/01/2014

Gratis Reuse

- Without further permission, as the Author of the journal article you may:
  - post the accepted version (version 2) on your personal website, department's website or your institution's repository. You may NOT post the published version (version 3) on a website or in a repository without permission from SAGE.
  - post the accepted version (version 2) of the article in any repository other than those listed above 12 months after official publication of the article.
  - use the published version (version 3) for your own teaching needs or to supply on an individual basis to research colleagues, provided that such supply is not for commercial purposes.
  - use the accepted or published version (version 2 or 3) in a book written or edited by you. To republish the article in a book NOT written or edited by you, permissions must be cleared on the previous page under the option 'Republish in a Book/Journal' by the publisher, editor or author who is compiling the new work.
- When posting or re-using the article electronically, please link to the original article and cite the DOI.
- All other re-use of the published article should be referred to SAGE. Contact information can be found on the bottom of our 'Journal Permissions' page.

BACK | CLOSE WINDOW

VITA

NAME: Bassam Hasanain

EDUCATION:

University of Illinois at Chicago	**Chicago, IL**
PhD in Industrial Engineering & Operation Research,	Dec 2015
Current GPA	3.33 out of 4
University of Dayton	**Dayton, OH**
Master of Science in Engineering Management,	Dec 2009
Overall GPA	3.50 out of 4
University of Toledo	**Toledo, OH**
Bachelor of Science in Electrical Engineering Technology,	May 2008
Overall GPA	3.54 out of 4

DISSERTATION: A Formal Approach for Detecting Masking in Medical Alarms

HONORS: Dean's List (Fall '06, Spring '07, Spring '08)

PROFESSIONAL MEMBERSHIP: Optimist Club "tutor and mentor young people".
UsToUs "empower students achieving a positive social impact in the society".

PUBLICATIONS: Hasanain, B., Boyd, A.D., & Bolton, M. L. (2015). Using Model Checking to Detect Simultaneous Masking in Medical Alarms. IEEE Transactions on Human-Machine Systems, DOI: 10.1109/THMS.2014.2379661

Hasanain, B., Boyd, A.D., & Bolton, M. L. (2014). An approach to model checking the perceptual interactions of medical alarms. Submitted to the 2014 HFES Annual Meeting. Chicago: Human Factors and Ergonomics Society.

Languages: Fluent in both English & Arabic