Using Model Checking to Detect Simultaneous Masking in Medical Alarms

Bassam Hasanain, Andrew D. Boyd, and Matthew L. Bolton, Member, IEEE

Abstract—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: making them impossible to hear. Because masking may only occur in extremely specific and/or rare situations, experimental evaluation techniques are insufficient for detecting masking in all of the potential alarm configurations used in medicine. Thus, a real need exists for computational methods capable of determining if masking exists in medical alarm configurations before they are deployed. In this paper, we present such a method. Using a combination of formal modeling, psychoacoustic modeling, temporal logic specification, and model checking, our method is able to prove whether a modeled configuration of alarms can interact in ways that produce masking. This paper provides the motivation for this method, presents its details, describes its implementation, demonstrates its power with a case study, and outlines future work.

Index Terms—Formal methods, medical alarms, model checking, patient safety, psychoacoustics, psychophysics.

I. 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. 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 [2]. Significant numbers of false alarms can desensitize humans to them (a condition known as alarm fatigue); alarms can be poorly designed, reducing their effectiveness [2]; and concurrently sounding alarms can perceptually interact in ways that make them difficult to identify [3] or mask each other (make one or more of them imperceptible) [4]. Unfortunately, problems caused by the masking of concurrently sounding alarms can be very difficult to identify because they may occur under rare or unusual conditions or through the interaction of particular alarms within or between alarming systems. Thus, while auditory masking has been experimentally detected in clinical settings [5], [6], the vast majority of the work has focused on other aforementioned problem areas [7]. Thus, there is a very real and urgent need for methods capable of identifying if masking is present in medical alarm configuration before they are used in medical practice.

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 an analysis tool, widely used in the analysis of safety critical computer systems that is specifically designed to find problems in models of concurrent systems using a form of automated theorem proving [8]. Psychoacoustic models are capable of mathematically indicating if concurrently sounding alarms might interact in ways that could produce masking [4], [9], [10]. When used together in our method [1], these techniques can allow healthcare providers to determine if masking exists in a modeled configuration of alarms computationally. With such a detection capability, healthcare providers should be able to deploy systems that will avoid masking: enabling medical personnel to respond to alarms appropriately and potentially save patient lives.

In this paper, we describe this method and illustrate its utility. We first discuss the literature relevant to understanding our method. This includes material on masking in medical alarms, psychoacoustic models of masking, and model checking. We then describe our method: its conceptualization, design, and implementation. To illustrate its power, we use it to evaluate a realistic configuration of medical alarms. We conclude by discussing our results and future avenues of research.

II. RELATED WORK

A. Concurrently Sounding Medical Alarms

Auditory medical alarms have a number of problems [7] making them one of the most significant technological hazards to patient safety for over a decade [11], [12]. 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 resulting in at least 12 deaths [13]. Medical device manufacturers have reported 216 “alarm-related” deaths to the FDA between January 2005 and June 2010 [14]. An event alert issued by the Joint
B. Psychoacoustic Models of Simultaneous Masking

A number of models exist for predicting auditory masking [4], [9], [10], [30], [38]–[41]. However, psychoacoustic models are the most appropriate to this study 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. The most successful of these 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) [9], [42]–[46].

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 [47], indicating which of the 24 critical bands of hearing the sound falls in. For a given sound (sound) with a frequency \( f_{\text{sound}} \) in hertz, the frequency is converted into the Bark scale using the formula

\[
z_{\text{sound}} = 13 \cdot \arctan(0.00076 \cdot f_{\text{sound}}) + 3.5 \cdot \arctan\left(\left(f_{\text{sound}}/7500\right)^2\right)
\]  

where \( z_{\text{sound}} \) is the frequency of the sound in Barks [48].

The masking curve for a given sound (a masker) is generally formulated as a function of both the sound and its frequency’s distance from another, potentially masked, sound (a maskee) on the Bark scale. This difference, \( \delta z \), is represented as

\[
\delta z = z_{\text{maskee}} - z_{\text{masker}}
\]  

Then, the masker’s masking curve is represented as

\[
\text{curve}_{\text{masker}}(v_{\text{masker}}, \delta z) = \text{spread}_{\text{masker}}(v_{\text{masker}}, \delta z) + v_{\text{masker}} - \Delta
\]

where \( v_{\text{masker}} \) is the volume of the masker in dB, spread is a function that defines how the volume changes as \( \delta z \) moves away from zero, and \( \Delta \) represents the minimum difference between a masker’s and maskee’s volume under which masking can occur [9].

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 [9] 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 [9], where masked audio data are removed to reduce file size.

For example, the spreading function used as the basis for the MPEG2 audio codec [43] is formulated as

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

1. A critical band is the frequency bandwidth of the filters produced by the inner ear’s basilar membrane in the cochlea.
This spreading function is tuned to normal hearing. It also has only one independent variable ($\delta z$). However, other spreading functions can take volume ($v_{\text{masker}}$) as an argument [9].

There can be different formulations of $\Delta$ depending on the nature of the sound. For tonal maskers [49], like those used in most medical alarms, $\Delta$ (in dB) is formulated as

$$\Delta = 14.5 + z_{\text{masker}}. \quad (5)$$

For a given masking curve, we know that the masker (with volume $v_{\text{masker}}$ and Bark frequency $z_{\text{masker}}$) is masking the maskee (with volume $v_{\text{maskee}}$ and frequency $z_{\text{maskee}}$) if

$$\text{curve}_{\text{masker}}(v_{\text{masker}}, z_{\text{maskee}} - z_{\text{masker}}) \geq v_{\text{maskee}}. \quad (6)$$

C. Formal Verification With Model Checking

Formal verification is an analysis technique that falls within the broader category of formal methods. Formal methods are a set of well-defined mathematical languages and techniques for the specification modeling, and verification of systems [50]. Specification are formulated to describe desirable system properties in rigorous unambiguous notations. Systems are modeled using mathematically based languages that support well-established theoretical formalisms such as finite-state automata. The verification process mathematically proves whether or not the model satisfies the specification. Formal verification has been successfully in a number of applications, especially computer hardware design, where performance must be guaranteed.

Model checking is a highly automated approach used to verify that a formal model of a system satisfies a set of desired properties (a specification [51]). A formal model describes a system as a set of variables and transitions between variable states. Specification properties are usually represented in a temporal logic using formal system model variables and temporal operators to construct propositions asserting temporal relationships between system elements [52]. Verification is performed automatically by exhaustively searching a system’s state space to determine if these propositions hold. If they do, the model checker returns a confirmation. If there is a violation, an execution trace called a counterexample is produced. This counterexample depicts a model state (the value of the model’s variables) corresponding to a specific violation along with a list of the incremental model states leading to the 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 event systems. However, hybrid modeling and analysis techniques can allow formal verification to be used with models that contain continuous quantities [53]-[55]. 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 analysts use timed automata [53], [56], a modeling approach where every discrete transition in a formal model is assigned a real numbered time.

Researchers have used formal verification to evaluate issues related to human–automation interaction (see [57] 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. 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.

III. 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 demonstrates that this is possible. 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. This method is 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.

The following section describes the method we developed. This includes an overview of the framework and a detailed description of its components. Then, to demonstrate the utility of the method, we use it to evaluate a realistic medical alarm configuration. Finally, our results are discussed and avenues of future work are explored.

IV. METHODS

In the method we have developed (see Fig. 1), an analyst must: 1) examine the documentation associated with a configuration of medical alarms and model their behavior using our formal modeling architecture (see Fig. 2); 2) specify the absence of masking using specification property patterns we provide; and 3) 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 (see Fig. 2) to be based on timed automata. Timed automata [53] provide a means of modeling time as a real-valued continuous quantity in a formal model. This architecture has multiple submodels 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.

We have implemented this method using the tools available in the Symbolic Analysis Laboratory (SAL) [58], [59]. In particular, we have designed our method to work with SAL’s infinit bounded model checker [53], [59], a tool capable of evaluating formal models containing timed automata. SAL’s infinit 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 specific 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 how 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.

### A. Formal Modeling Architecture

An overview of the SAL implementation of the architecture can be seen in Fig. 3. This has eight distinct parts. First, 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 nonnegative 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, \( \text{delatConst} \), represents the volume used in the computation of \( \Delta \). The second, \( \text{bigMax} \), 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 IV-A3.

The clock submodel, which is responsible for maintaining and advancing time, is next. It is described in Section IV-A1.

A series of submodels representing the behavior of the different alarms in the alarm configuration follow. Each of these represents the behavior of various alarms. Section IV-A2 describes the generic formal modeling pattern used for modeling each alarm in a configuration (with \( N \) alarms) in the architecture.

The masking computation submodel evaluates the outputs of the alarm submodels and uses the define functions to compute whether masking is occurring at the given clock-indicated time. This is developed further in Section IV-A3.

Each of the submodels 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 specification are described in Section IV-B.

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

1) Clock: The clock submodel (see Fig. 4) is responsible for advancing time and communicating the current time to the other
elements of the system. It receives a maximum on the amount that time can advance to (maxNextTime) as input 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 that is always greater than the current global time and less than or equal to the maximum next time.

Fig. 3. Overview of the implementation of the formal modeling architecture for modeling medical alarm configuration (see Fig. 2). This implementation is written using the notation of SAL (see [58]). Note that in this listing (and all subsequent listings in Figs. 4–7), 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 (see Figs. 4–7) or indicates an incremental series of like components or operations (e.g., the synchronous compositions of the alarm submodels: alarm1 || ... || alarmN).

2) Alarms: 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 (see Fig. 5). Each alarm has a constant value representing the length of its sounding cycle in seconds (alarmCycleTime) with an analyst specific value [TCycle] in Fig. 5). Each alarm also has a variable 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 (it can restart at any later time). Thus, the amount of time the alarm has been sounding is computed as the difference between the global time and the alarm’s start time (alarmTimeInCycle = globalTime - 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 global time in the next state (see TRANSITION in Fig. 5). 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. 5), the alarm will assume different values for frequency and volume. It should be noted that, in the model shown in Fig. 5, the value of the alarm’s volume is determined by the alarm’s frequency. However, analysts 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. 5. Within this, the analyst needs to describe the

```plaintext
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
```
Fig. 5. Generic SAL code for representing alarm behavior. Note that bracketed words in red represent numerical values that should be explicitly specified by the analyst. \([TCycle]\) represents the alarm’s cycle time in seconds. \([TFreq1]–[TFreqN]\) represent relative times (from the start time) that the frequency and volume change in increasing order in seconds. \([Freq1]–[FreqN]\) and \([Vol1]–[VolN]\) represent different frequencies (in Hz) and volumes (in dB), respectively.

Fig. 6. Method implementation of (1)–(5) used by the maskingComputation submodel to compute whether masking is occurring between two alarms.
alarmCycleTime (the \([TCycle]\) value in Fig. 5) and the logic defining the alarmFrequency, alarmVolume, and alarmNextTime by specifying the appropriate values ([\(TFreq1\)], \([Freq1]\), \([Vol1]\), etc).

3) Masking Computation: The masking computation model (see Fig. 7) 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 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 (see Fig. 6) that implement the equations in (1)–(5) in the formal model. Because model checkers are limited in their ability to consider nonlinear 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 precomputed and accessed in the formal model using a large IF...THEN...ELSIF statement. For the conversion of frequency (in hertz) to the Bark scale, the lookup table was computed using (1) rounded to the nearest tenth of a Bark for the full range of Bark scale values (0–24). For the spreading function \((spread)\), (4) was used to compute the spread rounded up the nearest dB for the full range of possible values for \(\delta z\) \((\delta z\) in Fig. 6). Note that this computation rounds up so that it biases the masking curve slightly in favor of detection. Further, the spreading function from (4) was chosen because it uses only \(\delta z\) in its computation, making the corresponding lookup table 1-D and thus much simpler to implement formally. This is discussed in more detail in Section IV-A.

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 hertz); 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 (6).

The second thing the masking computation model does is calculate the next time variables \((NextTime_{\text{Alarm}})\) 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.

In creating the masking computation model, an analyst is responsible for creating the “alarmMasked” variable and its
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)));

Fig. 8. Specificatio 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.

definition for each alarm using the pattern shown in Fig. 7. A model must also create the maxNextTime definition again using the pattern in Fig. 7.

B. Specification

To model check whether or not masking is present in a model, specification 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 study, 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 (see Fig. 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 (see Fig. 8) for each alarm.

V. CASE STUDY

To illustrate the ability of our method to detect masking in a realistic configuratio of medical alarms, we have used it to evaluate a simple case study. In this target configuratio (see Table I), there were three alarms. In a given cycle, each alarm played a two-tone pattern with a pause in between. Each sound used a frequency commonly found in tonal alarms [5], [32]. Durations and volumes were also consistent with the IEC 60601-1-8 international standard [32].

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 I. The last contained the implementation of all three alarms. Each model was implemented by instantiating the formal model architecture discussed in Section IV-A. This entailed creating an alarm module for each alarm included in the configuratio using the pattern in Fig. 5. This meant explicitly including the bracketed values in the pattern based on the desired alarm behavior. For the three alarms in Table I (for which N = 3), these are shown in Table II.

In all of the models, specification were created using the patterns from Fig. 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 test 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 specificatio in each model was evaluated using SAL’s infinit bounded model checker [59] (with search depth 12) on a Linux workstation with a 3.3-GHz Intel Xeon processor and 64 GB of RAM.

All verification results can be seen in Table III. 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 specificatio (which can be seen at http://fhsl.eng.buffalo.edu/resources/) were visualized (see Fig. 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.

VI. DISCUSSION

This study has introduced a novel method for identifying masking in configuratio of medical alarms. This method uses
a formal modeling architecture, psychoacoustic models of masking, specificatio 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 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 verification 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.

As the number of alarms in medical environments increases and causes more and more alarm interactions, there will be even more chances for total alarm masking conditions. As such, the presented method could be used by hospital personnel to evaluate the safety of different medical alarm configurations by considering all of the possible alarm interactions. Thus, this study 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. However, despite its success, this method has some limitations, which will be addressed in future work.

### A. Additional Masking Considerations

Our current implementation of the method uses the spreading function from the MPEG2 audio codec for normal hearing (4). This particular spreading function was chosen because it is 1-D: only varied as a function of $\delta z$. This made it easier to include the spreading function calculations in the formal model. However, a number of different spreading functions have been developed, all with slightly different shapes and thus different tradeoffs between miss and false alarm rates for detection [9]. An alarm that is not totally masked but very close to a masking threshold will still be difficult for an operator to perceive. Thus, for our method, there is greater utility in creating a digitally based masking detector (one that errs toward false alarms) rather than a conservative one (one that errs toward misses). Future work should investigate which spreading function provides the best desired detection behavior and integrate it into our method.

Currently, the method is only set up to detect simultaneous masking for individuals with normal hearing. However, this may be an unrealistic expectation for everybody that may need to perceive different medical alarms. Thus, future work should investigate how different spreading functions could be used to account for individuals with different hearing proficiencies.

Another limitation of the current approach is that it does not account for additive masking. Additive masking describes a condition where two simultaneous sounds can produce masking greater than or equal to the sum of their respective masking curves [9], [60]. Specifically, additive masking is computed as

$$ I_N = \left( \sum_{n=1}^{N} I_n^{\alpha} \right)^{1/\alpha} \tag{7} $$

where $I_N$ represents the combined masking intensity of $N$ maskers at a given frequency, $I_n$ represents the masking intensity of masker $n$ (its masking curve value) at that frequency, and $\alpha$ is a constant with range $(0, \infty)$. With an $\alpha$ of 1, masking effects are purely additive. However, Lutfi [61] found that an $\alpha$ of 0.33 was best suited to tonal sounds, a condition that leads to the “over adding” of different sounds' masking effects. For medical alarms, additive masking can manifest in two different contexts: either multiple alarms can be sounding and interact to create an aggregate masking curve and/or the alarm may contain multiple prominent auditory harmonics which can contribute to a given alarm’s masking [32]. Temporal masking can also occur [4], where the temporal relationships between sounds can mask those not concurrently sounding. Given that these phenomena suggest many additional alarm interactions that can cause masking beyond those considered in the current implementation, future work should investigate how to include them in our method.

It is also uncommon for alarms to be operating in a completely quiet environment. Thus, alarms may interact with environmental noise in ways that could exacerbate masking conditions. Future work should investigate how other environmental sounds could be incorporated into our method.

Many alarms are periodic in that they will sound repeatedly until a problem is dealt with. However, while our current method does allow alarms to sound over multiple cycles, it only checks whether an alarm is partially or completely masked in any given single cycle. It is our contention that even seconds of delay in operator response could have profound impacts on human health in medical environments. Thus, even the imperceptibility of a single cycle is dangerous and thus indicative of a problem that needs to be addressed. However, it may not always be possible to completely eliminate masking. In these situations, analysts may wish to determine if alarms will be perceptible within a

<table>
<thead>
<tr>
<th>Model with</th>
<th>Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarms</td>
<td>Alarm</td>
</tr>
<tr>
<td>1 &amp; 2</td>
<td>Alarm 1</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td></td>
<td>Alarm 2</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td>1 &amp; 3</td>
<td>Alarm 1</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td></td>
<td>Alarm 3</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td>2 &amp; 3</td>
<td>Alarm 2</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td></td>
<td>Alarm 3</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td>1, 2, &amp; 3</td>
<td>Alarm 1</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td></td>
<td>Alarm 2</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
<tr>
<td></td>
<td>Alarm 3</td>
</tr>
<tr>
<td></td>
<td>Total Masking</td>
</tr>
</tbody>
</table>

Note. ✓ indicates a verification confirmation and × indicates a verification failure with a counterexample.
Future work should investigate how the method could be used such a solution would be in a dynamic healthcare environment.

However, it is not clear how feasible and thus support localization) that could improve the chances of alarm perceptibility. Even in this situation, the presented method should not prevent it. However, through iterative modeling and verification, an analyst could use the method to find alarm settings or subsets of alarms that would not produce masking. Slight modification to our method could allow us to search this parameter space to determine if they allow for the existence of masking in terms of detection time.

B. Experimental Validation

Our method is based on established psychological principles and is thus expected to give accurate predictions. However, to show the validity of the method, it would be good to validate its predictions against actual human subject experimental results in realistic operational environments. Future work should pursue this. Additionally, human subject experiments could help us choose and tune spreading functions to achieve the desired prediction results. This should also be explored in future work.

C. Additional Case Studies and Use in Design

While the presented case study illustrates the method’s utility, there are many medical alarm configuration [62]. Further, standards such as IEC 60601-1-8 [32] have a number of open parameters that can be used to represent the behavior of alarms. Slight modification to our method could allow us to search this parameter space to determine if they allow for the existence of masking. Future work will investigate this.

To date, the method has only been used to detect masking, not prevent it. However, through iterative modeling and verification, an analyst could use the method to find alarm settings or subsets of alarms that would not produce masking. However, it is also conceivable that an analyst will encounter situations where he or she must use an alarm configuration that produces masking. Even in this situation, the presented method should have utility as it could allow the analyst to identify interventions (such as alarm positioning to target different listener ears and thus support localization) that could improve the chances of alarm perceptibility. However, it is not clear how feasible such a solution would be in a dynamic healthcare environment. Future work should investigate how the method could be used to provide analysts with realistic decision support for alarm configuration design, selection, and positioning.

Finally, alarms are critical to safety in domains beyond medicine including automotives, aviation, and industrial settings. Future work should explore how our method could be used to detect masking in these environments.

D. Scalability

All model checking-based verification have scalability problems: where the size of the model grows exponentially as concurrent elements are included. This can quickly lead to models that are too big or take too long to verify due to computational and/or physical memory limitations [51]. The case study presented here took slightly more than 21 min to verify. Thus, for more complex case studies, it is likely that scalability problems will be constraining. Such problems will likely be exacerbated as more complex masking conditions (i.e., additive or temporal masking) are included. However, there are potential opportunities for improving the scalability of our approach. In particular, some of the computations the formal model performs could be done using preprocessing. For example, instead of dynamically calculating alarm masking curves for each alarm in a given configuration these could be precalculated so that the associated lookup tables would be optimized for the configuration. Future work will investigate how to incorporate this and other scalability improvements into our method.

Even if scalability proves to be a persistent problem with this approach, the method may still have utility. Specifically, rather than evaluating dynamically created alarm configurations for immediate deployment in a healthcare environment, the method could be used to preassemble a database of alarm configuration across medical devices certified to avoid masking. Such analyses would not need to be done dynamically in the field and could be done without temporal constraints and on more sophisticated hardware. Thus, if scalability proves to be a significant constraint on the utility of our method, we will pursue this certificatio approach.

E. Other Alarm Considerations

As mentioned in Section 1, there are many other problems facing medical alarms. As this is the first attempt to explore alarm problems formally, there may be many future opportunities for
extending our work to explore other alarm issues. For example, there is good evidence suggesting that human mental workload can contribute to alarm mistrust, fatigue, and inattentional deafness [20], [23]. Formal methods could help researchers discover when these conditions could occur. Such an analysis will need to integrate formal approaches for modeling alarm perception, workload [63], and task behavior [64]–[66] to be successful. This should be explored in future work. Additionally, excessive false alarms can cause people to intentionally ignore alarms thought to be spurious (the “cry wolf” phenomenon [67]). By reducing the number of false alarms, masking potential could inherently be reduced. Future work should explore how formal methods could be used to help reduce false alarms in addition to the masking analyses discussed here.

F. Tool Usability

Finally, the method as currently implemented requires that the analyst be familiar with and implement formal models of alarm behavior. While our formal modeling architecture can assist analysts in this task, it will be cumbersome for those with no formal modeling experience. Ideally, our method would allow analysts to easily explore different alarm configuration with little to no knowledge of formal techniques. Future work should investigate how to develop tools that will enable analysts to quickly construct and analyze alarm configuration without the need for manual formal modeling and specification.

ACKNOWLEDGMENT

The authors would like to thank A. House and K. Molinaro for their help proofreading this manuscript.

REFERENCES


Bassam Hasanain received the B.S. degree in electrical engineering from the University of Toledo, Toledo, OH, USA, in 2008, and the M.S. degree in engineering management from the University of Dayton, Dayton, OH, in 2009. He is currently working toward the Ph.D. degree with the Department of Industrial Engineering, University of Illinois at Chicago, Chicago, IL, USA.

His research interests include model checking, formal methods, psychoacoustics, and psychophysics, in particular model checking applied to medical alarms.

Andrew D. Boyd received the B.S. degree in chemistry and mathematics for Southern Methodist University, Dallas, TX, USA, and the M.D. degree from the University of Texas Southwestern Medical School, Dallas.

He is currently an Assistant Professor with the Department of Biomedical and Health Information Sciences, University of Illinois at Chicago, Chicago, IL, USA.

His research interests include data simplification to improve clinical outcomes in cardiology and oncology.

Matthew L. Bolton (S’05–M’10) received the B.S. degree in computer science in 2003, the M.S. degree in systems engineering in 2006, and the Ph.D. degree in systems engineering in 2010, all from the University of Virginia, Charlottesville, VA, USA.

He is currently an Assistant Professor with the Department of Industrial and Systems Engineering, University at Buffalo, State University of New York, Amherst, NY, USA. His research interests include the development of tools and techniques for using human performance modeling, task analysis, and formal methods to analyze, design, and evaluate complex safety-critical systems.