

An Alert-Generation Framework for Improving Resiliency in Human-Supervised, Multi-Agent Teams

Sarah Al-Hussaini¹, Jason M. Gregory², Shaurya Shriyam¹, Satyandra K. Gupta¹

¹Viterbi School of Engineering, University of Southern California, CA, USA
 {salhussa, shriyam, guptask}@usc.edu

²US CCDC Army Research Laboratory, Adelphi, MD, USA
 jason.m.gregory1.civ@mail.mil

Abstract

Human-supervision in multi-agent teams is a critical requirement to ensure that the decision-maker's risk preferences are utilized to assign tasks to robots. In stressful complex missions that pose risk to human health and life, such as humanitarian-assistance and disaster-relief missions, human mistakes or delays in tasking robots can adversely affect the mission. To assist human decision making in such missions, we present an alert-generation framework capable of detecting various modes of potential failure or performance degradation. We demonstrate that our framework, based on state machine simulation and formal methods, offers probabilistic modeling to estimate the likelihood of unfavorable events. We introduce smart simulation that offers a computationally-efficient way of detecting low-probability situations compared to standard Monte-Carlo simulations. Moreover, for certain class of problems, our inference-based method can provide guarantees on correctly detecting task failures.

Introduction

With the advancement of robotic systems and artificial intelligence, there is a growing interest in more-intelligent, multi-agent teams working collaboratively to accomplish missions. These teams show especially great promise in dull, dirty, and dangerous applications, such as military operations and humanitarian-assistance and disaster-relief (HA/DR) efforts (Gregory et al. 2016). Despite the widespread use and ever-increasing capabilities of robotic systems, researchers anticipate that human team members will continue to be necessary - and not be replaced by technology - because of various advantages, including diverse expertise, adaptive decision-making, and the potential for synergy (DeCostanza et al. 2018). More importantly, HA/DR missions involve tasks with literal life-or-death consequences and so human-in-the-loop operations are mandatory to ensure proper management of resources and critical decision-making authority. Because multi-agent systems will require extensive collaboration, it is imperative that systems be developed with both the strengths and weaknesses of humans in mind, just as researchers and engineers design for the robotic agent. The disaster response system

needs to integrate humans, robots, and the software agents for effective response to emergency situations (Ramchurn et al. 2016). Connecting human language (Chai et al. 2016), adapting to human intents (Levine and Williams 2014), and co-development of joint strategies (Ramakrishnan, Zhang, and Shah 2017) can assist successful human-robot teaming.

It is well-understood that the relevant missions in the context of HA/DR are extremely complex (Murphy 2014). Due to the size and unstructured characteristics of the operational environment, restrictive communication limitations, and the constant threat of system and task-level failure, there is a large amount of uncertainty in the availability and efficiency of agents. The dynamic conditions introduced by the environment lead to intermittent data flow and require that the agents, including the humans, constantly adapt using the currently-available information. Humans are specifically prone to making mistakes and must overcome cognitively- and emotionally-fatiguing situations in the stressful situations of HA/DR missions (Murphy 2004). This can lead to the issuing of erroneous, slow-paced, or ill-advised commands. In an effort to prevent dangerous scenarios and the catastrophic degradation of performance, the team must be *resilient* to these various challenges, and the accompanying contingencies (Shriyam and Gupta 2018), by minimizing the impact of human-made errors.

Alerts provide an efficient and effective way for enabling and improving resiliency because they offer a means to prevent human-introduced mistakes and expedite decision-making. Already, alert systems have been deployed in a number of technologies for every-day tasks, like blind spot detection or lane departure warning systems for drivers to prevent vehicular accidents. An intelligent alert system can also provide tremendous benefit to human-robot teams operating in time-sensitive, safety-critical scenarios. This is because an alert system can assist the team with the required decision-making that is inherently challenging, taxing, and severely-consequential. Already, researchers have investigated the human factor concerns associated with supervisory control of multi-robot systems (Wong and Seet 2017), (Sherwood 2018), (Chien et al. 2018) and there exists several different alert-generating architectures and interfaces such as an augmented reality-based solution (Makris et al.

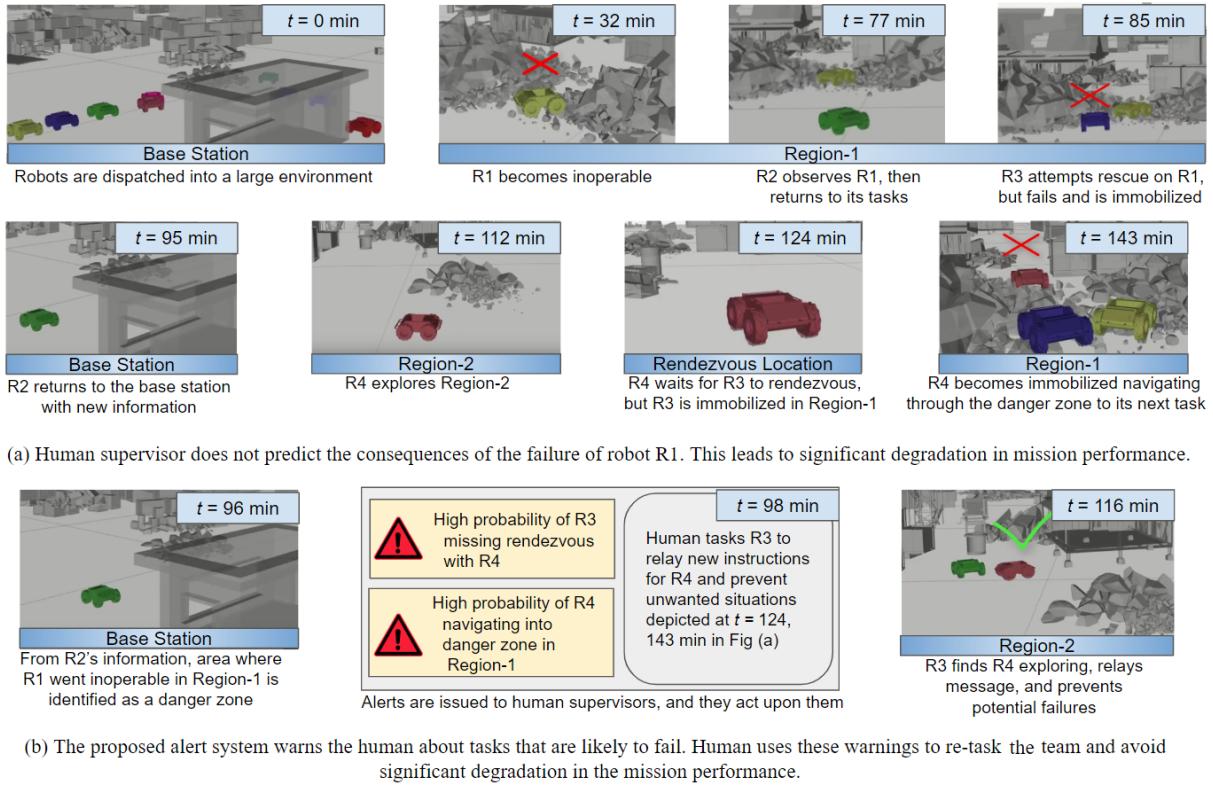


Figure 1: A motivating example scenario with four robots, R1 (yellow), R2 (green), R3 (blue), R4 (brown), visualized in Rviz.

2016) and the various alert systems used by NASA (Mosier et al. 2017). There have also been several systems designed specifically for human-robot teams in the HA/DR context, both with (Barnes et al. 2014) (Jentsch 2016) and without alert systems (Kruijff et al. 2014). These alert-generation frameworks are purely reactive, where a notification is provided to the human once an undesirable event has occurred. The most relevant work to date is the predictive conventional interface and predictive virtual reality interface designs (Roldán et al. 2017). These two interfaces predict the risk and relevance of a robot performing some task. In this work, we also seek a proactive approach to providing alerts; however, our focus is on the generation of alerts for potential dangers or unwanted situations in the mission, as well as, erroneous and inefficient task assignments issued by humans. The Human Factors Analysis Classification System framework, applied for human operators in different applications (Shappell et al. 2007), (Diller et al. 2014), identifies decision errors, perceptual errors, and violations as the triggers to unsafe acts, which have preconditions from environmental, personnel factors, and mental and physical condition of operators. Our alert system is aimed to address all these factors, and improve the performance of human supervisors. A motivating mission scenario is illustrated in Figure 1.

Building a useful alert system suitable for multi-agent systems requires careful consideration. First, a proper language-based scheme is necessary to facilitate intuitive interaction with the human teammates. Unlike the afore-

mentioned alert systems for every-day tasks, an alert system for multi-agent systems cannot simply rely on sensors and comprehensive observations. Instead, a fieldable alert system requires inference and probabilistic model estimation to account for the inherent uncertainty in mission operations. An alert system must also be flexible to the types of alerts offered, be tailored to the human preferences and mission needs, and generate meaningful alerts in a timely fashion so that agents can take the necessary, corrective actions. In this work, we propose a novel, alert-generation framework that overcomes these challenges to improve resiliency of multi-agent teaming. This work provides two noteworthy contributions. First, we define a formal language and a state machine-based simulation architecture to model the likelihood of salient system states during the execution of a human-commanded mission. Second, we present an inference engine that compares human-specified alert conditions with the probabilistic outcomes of the state machine simulations to produce worthwhile alerts. Using a probabilistic temporal logic framework, we enable the human to specify alert conditions based on their requirements and preferences in an effort to improve the value of the reported alerts. To demonstrate the usefulness of our proposed framework, we provide some example scenarios to be detected as results. They show the detection of unwanted situations based on new information which can be complex for humans to infer by themselves in time-critical missions. We also demonstrate the usefulness of smart simulation which can be useful

in detecting low probability events in computationally efficient manner compared to Monte-Carlo simulations.

Overview of Approach

Mission Description

We consider a disaster-stricken environment, e.g., a city after an earthquake, flood, or wildfire, as a representative environment for a generic HA/DR mission. The outdoor environment is assumed to be on the order of several square kilometers and comprised of complex, unstructured terrain. A human-supervised team of robots is deployed for exploring affected regions efficiently, collecting important information, and performing certain tasks, according to human preferences. As the team of robots navigate through the environment, there is some nonzero probability of operational failure, which could be a result of spatial factors, such as complex terrain, or stochastic events, such as hardware or software failures. There also are communication challenges because of the large scale operational environment. Limited communications cause substantial delays in humans receiving information from the robots in the field. We encode these typical challenges and characteristics of a generic HA/DR operation in our specific mission in order to present and test our proposed alert generation framework. Thus, any other multi-robot mission, related to HA/DR-relevant application might be reduced to a variation of our defined mission, and our framework can be tailored for any such operation.

In a large-scale environment, usually there are some regions of higher importance which should be given priority in the exploration process. We assume that the humans, typically the first responders, can use their expertise and protocols to identify some *areas-of-interest* (AoIs), chosen at the beginning of the mission, or dynamically throughout based on latest information. We also assume, there are a few, very sparsely located beacons, and robots need to be physically close to them in order to communicate with humans. Each robot carries out several tasks based on its observations and the situation-specific instructions, and then navigates back to a beacon after some time to reconnect with its supervisors.

In our mission, human supervisors at the base station provide instruction-set to each robot in the team, and dispatch them for exploration and data collection. In addition to navigation and exploration, humans can instruct a robot to rendezvous with another robot, or relay new instructions to another robot. Humans can also command a robot to provide assistance to a temporarily-disabled robot in an attempt to make it functional again. Every time humans receive new information from a returning robot, they need to assess the current situation, and make intelligent decisions based on the latest update. Humans might want to re-task the fielded robots, which may be currently out of communications range, in order to avoid unwanted situations and prevent undesirable contingencies. Thus, the human supervisors are under constant pressure to make decisions quickly in order to utilize the robots most effectively. Due to the large size of the environment, humans may only receive updates from each robot after some extended time. Therefore, the robots need to be sufficiently tasked for prolonged peri-

ods, and the stochasticity and danger involved in the mission may prompt the humans to give instruction-sets which are incredibly complex. This necessitates a systematic way for humans to command robots, and deployed systems should be equipped to describe a complete instruction-set with adequate complexity.

Language For Commanding Robots

We present a formal language in this section that can decompose the complex, human-provided instructions in a systematic way. First, we define a set of tasks that a robot can execute; these are: explore, navigate, rescue, rendezvous, relay, return, wait. The uncertainties in the mission, and different stochastic phenomena demand several other actions from the robots, in addition to exploration and navigation. In order to improve resiliency in a mission with a high probability of failure, robots need to provide assistance to temporarily-disabled teammates by attempting *rescues*, as defined in our previous works (Al-Hussaini, Gregory, and Gupta 2018a), (Al-Hussaini, Gregory, and Gupta 2018b). This rescue operation has stochastic outcomes, such as successful rescue, failed rescue, and, in the worst case, the rescuer robot also becoming inoperable during the attempt. In order to tackle the challenge of scarce communication in a HA/DR mission, we have included *rendezvous* and *relay* tasks. In a collaborative exploration-based mission, it is useful for multiple robots from different regions to meet at pre-scheduled times and locations to exchange information, referred to as *rendezvous*. The *relay* task requires a robot to go to a specific region, search for another robot, and relay a specific piece of information to that robot. Here, we have limited the scope of relay tasks to initially clear the other robot's old instructions and then issue a new command-set. Since one of the focuses of this work is to provide alerts for future contingencies, this *relay* task is particularly helpful. It can be used in sending an available robot to prevent an adverse situation happening to a robot already in the field. Finally, the return and wait tasks correspond to the robot navigating back to within communications range of the human supervisor, or remaining stationary in one location, respectively.

Let $\mathbf{R} = \{R_1, R_2, \dots, R_N\}$ be the set of N robots, deployed in the mission. At any time t , each robot R_i has state S_{R_i} , location L_{R_i} , event list \mathbf{E}_{R_i} , list of other robot's most recent status updates (location, state) \mathbf{I}_{R_i} , and its own state history list $StateHist_{R_i}$. A robot goes through a series of states in order to perform a particular task. For example, *TravelToRend*, *WaitToRend*, *Rendezvouzing* are the states corresponding to task *rendezvous*; all having the same argument-list, i.e., robot ID to meet, rendezvous location, and time window. The element $StateHist_{R_i}^{\tau}$ gives its state from a time that was τ instances before the current time. A complete list of events should include relevant environmental events as well as some task related events with a robot itself or other teammates. An example event type is *rescue attempt*, with arguments robots IDs, location, time, and outcome. Every unique event a robot R_i learns about is stored in \mathbf{E}_{R_i} . In order to create time-based and situation-dependent instructions, humans need to construct different conditional statements which make use of time, the robot's information

Table 1: Supporting Functions for Robot Instructions and Alert Conditions

Functions or Literals	Description of Return Variables or Values
$TravelDuration_i(\mathcal{X}_1, \mathcal{X}_2)$	Estimated travel robot R_i to navigate from region \mathcal{X}_1 to \mathcal{X}_2
$isRiskyRegion(\mathcal{X})$	$True$ when the region \mathcal{X} is risky, $False$ otherwise
$NearbyRobotID_i(st, d)$	ID of robot with state type st , and within distance d from robot R_i
$CountNearbyRobots_i(st, d)$	Number of robots with state type st , within distance d from R_i
$CountExploringRobots(\mathcal{X})$	Number of robots exploring region \mathcal{X}
$CountEvents(e, j_1, a_1, j_2, a_2, \dots, i_m, a_m)$	Number of events in \mathbf{E}_{R_i} whose $type$ is e , and $arg^{j_1}, arg^{j_2}, \dots, arg^{j_m}$ are respectively a_1, a_2, \dots, a_m
$ToRend_i = \begin{cases} 1, & \text{if } S_{R_i}.type = TravelToRend \\ 0, & \text{otherwise} \end{cases}$	$True$ if robot R_i is travelling to rendezvous location
$EndRend_i = \begin{cases} 1, & \text{if } (StateHist_{R_i}^1.type = WaitToRend, \\ & S_{R_i}.type \neq WaitToRend) \text{ or } S_{R_i}.type = Rend \\ 0, & \text{otherwise} \end{cases}$	$True$ if robot R_i moves to a new task after a rendezvous attempt
$IsNav_i = \begin{cases} 1, & \text{if } S_{R_i}.type \in \{Navigating, TravelToExpl, \\ & TravelToResc, TravelToRel, TravelToRend\} \\ 0, & \text{otherwise} \end{cases}$	$True$ if robot R_i is navigating or travelling to a task location
$NeverRescue(i, \mathbf{J}) = \begin{cases} 1, & \text{if } \forall s \in \mathbb{S}_i^*, \neg(s.type = rescue, s.arg^1 \in \mathbf{J}) \\ 0, & \text{otherwise} \end{cases}$	$True$ if robot R_i will never attempt rescue on a robot with ID $\in \mathbf{J}$
$NeverRelay(i, \mathbf{J}) = \begin{cases} 1, & \text{if } \forall s \in \mathbb{S}_i^*, \neg(s.type = relay, s.arg^1 \in \mathbf{J}) \\ 0, & \text{otherwise} \end{cases}$	$True$ if robot R_i will never attempt relay on a robot with ID $\in \mathbf{J}$
$MinTravelT(i, L) = \frac{EuclideanDistance(L_i, L)}{MaxNavigationSpeed_{R_i}}$	Minimum navigation time for R_i from its location L_i to location L

* Defined in Inference-based Approach Section

state at the instance, and model estimation of different parts of the system. For crafting these conditions, we present a compiled list of functions in Table 1. We use items from this list for the robot’s instruction-sets and alert conditions presented in Figure 2, and Tables 2, 3, respectively.

Humans issue complex instruction-sets to the robotics teammates. Each instruction-set is defined as a set of tasks, arguments, event- and temporal-based conditionals. Based on the different outcomes of the stochastic parameters related to the environment, events, and the states of other teammates, a robot may perform various sequences of tasks for the human-provided instructions. The first step is to convert the instructions to a pseudocode format in order to properly identify the *if* and *while* statements along with the conditions. All the conditional statements of interest in this work can be expressed mathematically with the functions and variables described in this section. Additionally, we identify proper arguments for each task in the instruction-set. These aid with generating a *task transition model* for each robot where each transition is dictated by a condition. We provide an illustrative example of language decomposition for a snippet of an instruction-set in Figure 2.

System Architecture

Our proposed framework is targeted to be used by humans supervising a team of robots in a challenging large-scale mission. The task transition model derived from complex instruction-set is used to model each robot’s behavior as a state machine. Whenever any robot returns to a human with

new information, they provide state information of other robot teammates encountered in the field. We assume there are some estimated models of different stochastic parameters within the entire system. Using these models, the latest state information, and the state machine models of the robots, forward simulations of the entire system are performed. These simulations are done at appropriately high-level, in order to perform a large number of simulations quickly and generate immediate alerts. Each simulation run is a collection of parallel, but inter-dependent, state machine simulations for all robots in the field. The results of simulations give probabilistic estimates on feasible outcomes in the mission. Simultaneously, humans define their preferred list of unwanted situations that they feel are important to detect. These contingency conditions are then expressed as mathematical propositions. Our framework provides an inference engine that utilizes the results from the simulations, and finds *Truth* values for the user-specified alert conditions. If any of these become *True*, the framework shows the corresponding alerts to the humans. This alert can be based on probabilistic estimates from the simulations, or it can provide guarantees on particular situations happening with 100% certainty. The proposed framework is represented in the block diagram in Figure 3.

Specification of Alert Conditions

We have identified some exemplary alert-triggering scenarios that humans may find useful and relevant to an HA/DR mission. We also outline mathematical expressions of the

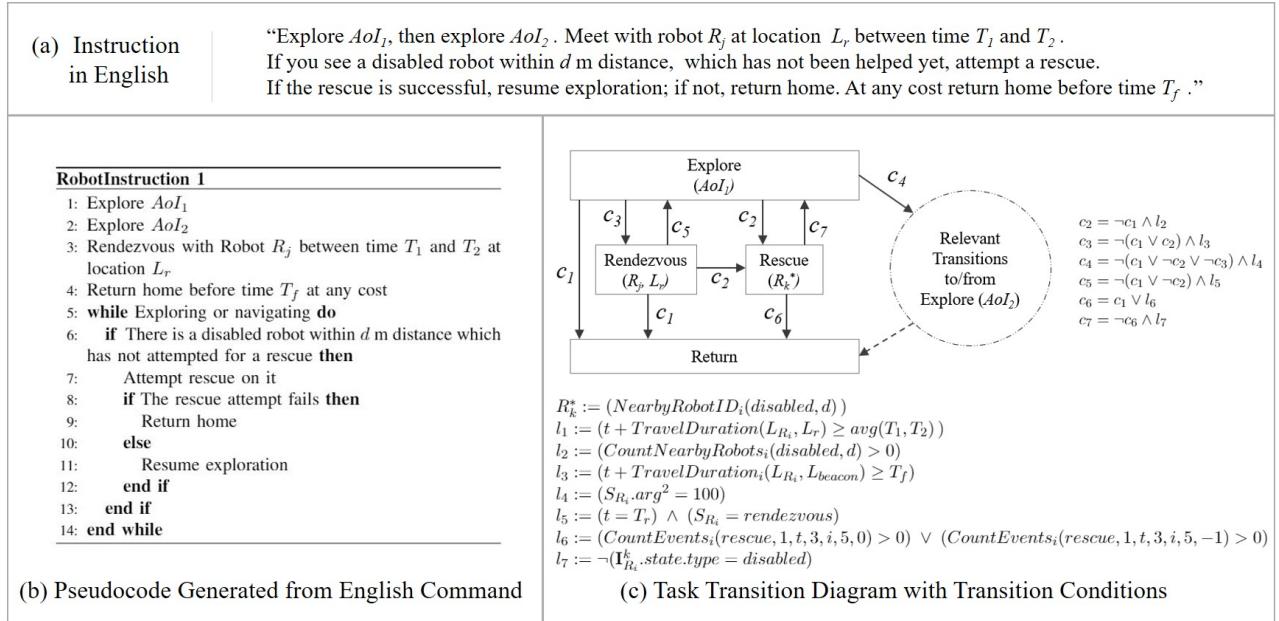


Figure 2: An example of the language decomposition offered by our proposed framework. English instructions provided by the human (a) are converted to pseudocode (b) that is used to generate a task transition model (c).

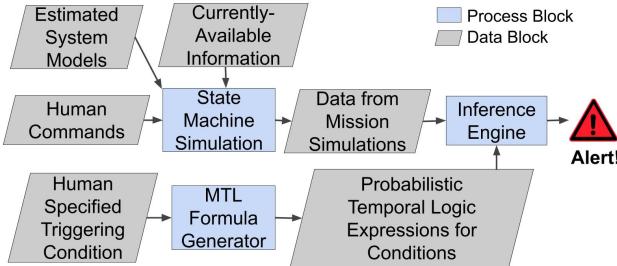


Figure 3: Block diagram of the proposed alert-generation framework. The grey slanted rectangles are data blocks and blue rectangles are processing components of the system.

alert conditions for detection of these situations. We formulate the conditions in a probabilistic temporal logic framework (Konur 2013), using different parameters and functions. Probabilistic temporal logic is a powerful language to mathematically express many kinds of complex conditions. Humans are free to choose different alert conditions from a potential list relevant to each mission, or craft their own conditions based on their preferences. The description of several alert situations are enumerated in this section, and the mathematical expressions relevant to these contingencies are provided. Note, this is not a comprehensive list for a mission, rather we provide some worthwhile examples depicting different types of fundamental expressions and conditions.

In our proposed framework we use Metric Temporal Logic (MTL) (Ouaknine and Worrell 2008) specifications to detect a particular unwanted situation in a single mission outcome. Mission progression ends at some

time T_{end} once all of the robots are in *returned* or *disabled* states; after that time we can assume the entire system remains constant. Let Ψ be a set of atomic propositions, crafted from the aforementioned items relevant to the mission threads, and the MTL formulae are built from Ψ using Boolean connectives (*and* \wedge , *or* \vee , *not* \neg), propositions \top (*True*) and \perp (*False*), and time-constrained or -unconstrained versions of temporal operators (*eventually* \diamond , *always* \square , *next* \bigcirc , *until* \mathcal{U}). A time-constrained temporal operator is Γ_I , where $\Gamma \in \{\diamond, \square, \mathcal{U}\}$, and time interval $I \subseteq (0, \infty)$, while the unconstrained version is $\Gamma = \Gamma_{(0, \infty)}$. If we generate a large number of probable strings of mission threads, representing the different ways the mission can progress, we can compute the fraction of strings that satisfies an MTL formula. This fraction can be taken as the estimated probability which is compared with a threshold value p_{th} in order to detect high or low probability conditions. $P_{\sim p_{th}} \Phi$ indicates that the probability of Φ being *True* is $\sim p_{th}$, where, $\sim \in \{<, \leq, >, \geq, =\}$, $0 \leq p_{th} \leq 1$, and Φ is an MTL formula. For clarity, we provide some examples of MTL formula similar to the ones used in Table 2, along with their implications in words.

- $(\diamond_{(t_1, t_2)} \phi)$ is *True iff* eventually at some time between t_1 and t_2 , ϕ is *True*
- $(\diamond(\phi_1 \rightarrow \bigcirc \phi_2))$ is *True iff* eventually at some time ϕ_1 is *True* and right after that (at the next time instance), ϕ_2 is *True*.
- $(\diamond(\phi_1 \rightarrow (\phi_2 \mathcal{U}_{\{t'\}} \top))$ is *True iff* eventually at some time ϕ is *True*, and right after that ϕ_2 remains *True* for next $t' - 1$ time instances.

The following items narrate some contingency situations that human supervisors may want to receive alerts for. The

Table 2: Example Alert Condition Specifications: *Probabilistic estimation for enumerated situations*

Alert #	Condition descriptions	Expressions of conditions
1	Non-zero probability of robot R_i having states S_1, S_2, S_1, S_2 respectively in four consecutive time steps (oscillation between states S_1, S_2)	$\mathbf{P}_{>0}[\Diamond((S_{R_i} = S_1) \rightarrow \bigcirc(S_{R_i} = S_2) \rightarrow \bigcirc(S_{R_i} = S_1) \rightarrow \bigcirc(S_{R_i} = S_2))]$ Where, $(\{S_1, S_2\} \subset \mathbb{S}) \wedge (S_1 \neq S_2)$
2	(i) High probability of robot R_i and R_j never being together at the scheduled location (X) within time (t_1, t_2) (ii) High probability of one of the robots R_i, R_j being at scheduled time & location while other one does not	(i) $\mathbf{P}_{\geq p_{th}}[\Diamond_{(t_1, t_2)} \neg(L_{R_i} = X \wedge L_{R_j} = X)]$ (ii) $\mathbf{P}'_{\geq p'_{th}}[B_j \vee B_i]$ Where, $B_i := (\Diamond_{(t_1, t_2)} L_{R_j} = X) \wedge (\Box_{(t_1, t_2)} \neg(L_{R_i} = X))$ $B_j := (\Diamond_{(t_1, t_2)} L_{R_i} = X) \wedge (\Diamond_{(t_1, t_2)} \Box \neg(L_{R_j} = X))$
3	(i) Low probability of robot R_i attempting rescue on R_j at location X (ii) Non-zero probability of robot R_i attempting rescue on R_j at location X more than n times	(i) $\mathbf{P}_{\leq p_{th}}[\Diamond(CountEvents_i(rescue, 2, X, 3, i, 4, j) > 0)]$ (ii) $\mathbf{P}_{>0}[\Diamond(CountEvents_i(rescue, 2, X, 3, i, 4, j) > n)]$
4	High probability of robot R_i navigating through a risky region	$\mathbf{P}_{\geq p_{th}}[\Diamond isRiskyRegion(L_{R_i})]$
5	High probability of having more than N_Y number of robots exploring region Y at once (more than a time duration t'), where N_Y is an upper bound on number of exploring robots for that region	$\mathbf{P}_{\geq p_{th}}[\Diamond(\varphi_Y \rightarrow (\varphi_Y U_{\{t'\}} \top))]$ Where, $\varphi_Y := (CountExploringRobots(Y) \geq N_Y)$
6	(i) High probability of R_i travelling to rendezvous for more than a time duration t' (ii) High probability of R_i navigating for more than t' time duration towards the next task, after a successful or failed rendezvous	(i) $\mathbf{P}_{\geq p_{th}}[\Diamond(ToRend_{R_i} \rightarrow (ToRend_{R_i} U_{\{t'\}} \top))]$ (ii) $\mathbf{P}_{\geq p_{th}}[\Diamond(EndRend_i \rightarrow (IsNav_i U_{\{t'+1\}} \top))]$

detection condition corresponding to each situation and its mathematical expression are provided in Table 2. How to choose appropriate probability threshold values, in accordance with human preferences on certain situations in a mission, is to be considered for our future work.

1. Wastage of time due to oscillatory states

Poorly-defined conditionals in an instruction-set can cause a robot to oscillate between two states which might not be intended by the human commander. This may waste a significant amount of time without any progress.

2. Improbable rendezvous

A pre-scheduled rendezvous between two robots might not happen if humans make mistakes in issuing the commands. It can also be missed if a robot becomes inoperable, or skips the rendezvous to execute a separate branch of tasks due to situations and specifications. In crafting the condition expression, we assume that both robots arriving at the designated location within the scheduled time period is sufficient for a successful rendezvous.

3. Improbable or redundant rescue attempts

A specific rescue task instructed to a robot may never actually occur due to certain, obstructing events. Another unwanted situation can be where a robot continuously attempts multiple rescues of the same robot, when it is not intended behavior. Incomplete or improper conditional statements tied to the rescue task can be a possible reason for this issue. In a broader sense, redundant rescue might also refer to multiple robots attempting the same rescue.

4. Navigation through a risky region

If humans receive information about a newly-assessed, risky region they might want to check whether any robot in the field is likely to navigate through that region based

on its instructions. Moreover, humans can erroneously assign a robot to a risky region. If humans are notified of this condition in a timely fashion, dangers can be prevented.

5. Redundant exploration

Every region, based on its characteristics, has an optimal number of robots for achieving fast, collaborative exploration. If there are too many robots exploring a relatively small region together, it may make the exploration process inefficient. We have provided an expression for such conditions in Table 2. Likewise, redundant exploration may occur when a robot is exploring a region that has already been explored by another robot.

6. Excessive travel time for rendezvous or rescue

Humans might want to avoid having rendezvous or rescue at a location where a robot needs to navigate for a long time from its previous task or to its next task. We have provided expressions for rendezvous in Table 2.

There can be many other alert conditions that humans may want to detect, such as, risky rescue, long travel-time for an unsuccessful task, etc. Any condition of a person's choice can be mathematically formulated, used for detection of adverse situations, and produce an alert for the humans.

Methods to Detect Alert Conditions

Given the specification of alert conditions, the alert system must identify when an alert condition is expected to hold *True*, and issue an alert to the humans. The alert condition detection can be simulation- or inference-based. Our simulation-based approach issues alerts based on probability estimates from simulations, but it lacks any guarantee. On the other hand, the inference-based approach may not be possible in every situation, but when applicable, it is able to provide a guarantee on certain situation.

Simulation-based Approach We use high-level, discrete-time simulations of the mission, where each simulation uses the system model and the latest information. It generates data on a single instantiation that the system could progress throughout the mission, out of infinitely many possibilities. We perform a large number of simulations, and observe what percentage of the simulations have a specific, unwanted situation of occurring. This is considered as the estimated probability of the alert condition to hold true. If this probability meets the thresholding specification set by humans, the interface shows an alert about the possible contingency.

If we consider detection of an extremely low-probability situation, repeated random sampling or Monte-Carlo simulations may become computationally infeasible due to the excessively large number of simulations required. Our smart simulation feature provides a computationally feasible way to detect even these low-probability circumstances. There are often some critical events that prompt the unwanted situation to be detected. Such critical events are identified from the state or task transition conditions, and are artificially triggered during the simulation runs. The probability estimate from these simulations give the conditional probability, and using the probability of the critical event, we estimate the actual probability of the unwanted situation.

Inference-based Approach The estimated probability of alert-triggering situations, calculated from simulations, might produce a poor representation of the real scenario, as the uncertainty and size of the system increases. Also, it is difficult to build an adequate model of the stochastic parameters in the mission. Therefore, we attempt to perform quick inferences using only the non-stochastic parts of the system, (i.e., instruction-sets of the robots, maximum navigation speed) instead of using simulations to detect certain contingencies with significantly-higher confidence. For a complicated mission, it might not be always possible; nevertheless, it can produce useful results in some cases.

Many of the alert conditions that we have presented refer to the probability of certain tasks being completed, i.e., certain states and/or locations that are reached by one or more robots within a time range. If an alert condition can be modeled in this way, and the simulation-based testing shows absolutely zero probability, this inference-based approach can be attempted. This method firstly converts the state transition model of each robot into a directed graph without the transition conditions. Using a graph search algorithm, like Depth First Search (DFS), to conduct reachability tests, and considering some simple Boolean literals, this approach can provide the humans with a much stronger assessment of an alert condition. In fact, in these cases it can guarantee absolute certainty.

It may appear as though confirming reachability for an individual agent may be sufficient in this inference. However, HA/DR missions are actually much more challenging and tasks, such as *rescue* and *relay*, can complicate the inference process. Let, the latest information available to humans on each robot $R_i \in \mathbf{R}$ is from a time in the past, t_i . Its state, previous state before arriving to the latest state, and location are $S_i, S_{prev,i}, L_i$, respectively. We can perform a graph

search for each robot R_i , and obtain the set of all possible states that R_i can reach from any state s . Note, an inoperative robot does not have any reachable state unless it is successfully rescued by another robot, and then it returns back to its previous state. Let, \mathbb{S}_i represent the set of all reachable states for robot R_i , from state S_i if it is functional, and from state $S_{prev,i}$ if it is not functional. Let M_{alive}, M_{dis}, M represent the sets of IDs of functional, non-functional, and all robots, respectively. Some instruction-set for a robot may require dynamic assignment of an argument of a certain task, for which the state transition model of the robot's corresponding states might have unknown argument values. For example, the instruction can be “if you see a robot within 20m, rescue it”. If this *rescue* state is s in the state-graph, $s.arg^1$ will be ‘unknown’, hence $s.arg^1 \notin M$. It is important to differentiate this unknown argument from known arguments in order to properly assess the probability of rescue or relay operation for a robot. Given all the specifications, we would like to prove whether R_i will never reach state S_f at location L_f by time T_f . The conditions to be checked for this purpose are given in Table 3, using some supporting functions from Table 1. The first three conditions of Table 3 test for reachability to the final state S_f , while the last two check whether it is physically possible for R_i to navigate to the destination L_f in time T_f .

Results

We have tested several alert conditions in our Python-based custom simulator for some mission scenarios that capture the inherent challenge for a human to quickly process a complex inference or dependency while observing only raw data. We then visualize the resulting outcomes using Rviz from the Robot Operating System (ROS). We provide a few examples in this section to illustrate the value of our alert system and to demonstrate the applicability of alert condition detection. An example case shows how the inoperability of one robot decreases a second robot’s probability of successful rendezvous by between 0 – 41%, depending upon the time and location of the disabled robot. Without an alert system, it might not be possible for the humans to infer such contingencies under pressure, and would not be able to make intelligent decisions that support resilient operations.

Using Simulation to Issue Alerts

The individual robots as state machines in the simulation are dependent on each other and the environment. While staying within a state, a robot keeps performing some low-level actions, which affect itself and its teammates. At the start of the simulation, each state machine, i.e., robot model, needs to be properly initiated according to latest update. We start simulation from the earliest update time instance among the robots, and forward simulate all of the robots whose statuses are not known for respective time instances.

Is there a non-zero probability of robot R_i oscillating between two tasks T_1 and T_2 ? Let robot R_i be currently in the field, with the instruction-set described in Figure 2, except that the commander, forgetfully or otherwise, excluded the conditional phrase, “which has not been helped yet”,

Table 3: Inference-Based Alert Conditions: *Prove that robot R_i can not reach state S_f and location L_f by time T_f*

Condition descriptions	Expressions of conditions
1) R_i is functioning, S_f is not reachable from its current state, and no functioning robot will attempt a rescue on any disabled robot, and not relay new instruction to R_i	$(i \in \mathbf{M}_{alive}) \wedge (S_f \notin \mathbb{S}_i) \wedge (\forall j \in \mathbf{M}_{alive}, (NeverRelay(j, \{i\}) \wedge NeverRescue(j, \mathbf{M}_{dis})))$
2) R_i is disabled and no functioning robot will attempt any rescue on any robot who are disabled	$(i \in \mathbf{M}_{dis}) \wedge (\forall j \in \mathbf{M}_{alive}, NeverRescue(j, \mathbf{M}_{dis}))$
3) R_i is disabled, S_f is not a reachable state even if R_i is rescued, and no robot (functioning, or disabled robot after being rescued) will attempt relaying new instruction to R_i	$(i \in \mathbf{M}_{dis}) \wedge (S_f \notin \mathbb{S}_i) \wedge (\forall j \in \mathbf{M} - \{i\}, NeverRelay(j, \{i\}))$
4) R_i is functioning, but its earliest possible arrival time to destination location L_f is later than time T_f	$(i \in \mathbf{M}_{alive}) \wedge (t_i + MinTravelT(i, L_f) > T_f)$
5) R_i is disabled; even if it is successfully rescued at the earliest possible time based on other robots' times and locations, it still can not reach L_f in time T_f	$(i \in \mathbf{M}_{dis}) \wedge (EarliestRescueTime(i) + MinTravelT(L_i, L_f) > T_f)$ where, if τ is an array with elements $\tau^j = t_j + MinTravelT(j, L_i)$, $\forall j \in \mathbf{M} - \{i\}$, $EarliestRescueTime(i) = max(min(\tau), t_i)$

about other inoperable robots to attempt rescues. In this case, if a rescue attempt fails (disabled robot remains disabled after attempted to be rescued), R_i will go back to previous task explore (T_1), and immediately come back to the same *rescue* task (T_2) again. Repeated failed rescue attempts will keep this cycle going on. Effectively, R_i will keep attempting consecutive rescue attempts until R_j is revived, and waste a lot of time. A more serious problem may occur if it gets disabled after making multiple failed rescue attempts, in case of a high-risk rescue. So, it might be worthwhile to detect any possibility of such situation so that the instruction can be corrected before robot R_i 's deployment.

As we described, other robots getting disabled nearby as well as failed rescue attempts are the critical events that can cause this oscillation. These critical conditions are identified from the transition conditions on task transition model of robot R_i , between the two corresponding tasks. We perform detection of this situation with Monte-Carlo (MC) simulations and smart simulations, and compare the performances. The probability of critical events are estimated from the mission model. We use this example of low-probability situation to illustrate the value of smart simulations.

20,000 MC simulations give probability of oscillation to be 0.00015. Therefore, it requires at least 14,000 simulations to detect this situation confidently. However, with only 500 simulations using our proposed framework – a reduction in the *required* number of simulations by a factor of at least 100 – we compute the probability of oscillation to be 0.00013. Thus, smart simulation can provide an efficient way to approximate the estimated probability, and issue alerts.

Using inferences to issue alerts

We use arithmetic operations and graph search algorithms on state transition models to make quick inferences, and issue an alert instantly, when applicable.

Is there a non-zero probability of Robot R_i rescuing Robot R_j ? Let robot R_i be in the field with after given the instruction “Explore AoI-1 and AoI-2 sequentially, and rescue any disabled robot within the exploring AoIs. If event- A

happens, move to AoI-3, and explore AoI-3 and AoI-4 sequentially.” Humans receive information on robot R_j being inoperable in AoI-2, and R_i is last known to be exploring AoI-3. We need to analyze whether R_j will be attempted to be rescued by R_i . From a reachability test, the system may conclude in a moment that rescuing R_j in AoI-2 is not a reachable state for R_i , given its latest known state. Any other robots, given their latest known states, do not have any *relay* task among their reachable states, therefore no on-field robot's instruction-set will change. Therefore, it is guaranteed that R_i will *not* rescue R_j .

Conclusions and Future Work

We presented an alert generation framework for human-supervised multi-robot teams deployed in challenging applications. We identified several useful warning situations, and have expressed their conditions using Metric Temporal Logic (MTL) specification. Our state machine simulation with an abstract model of the mission is computationally-efficient, which is essential in HA/DR applications for rapid alert generation. More specifically, we probabilistically assessed complex conditions with forward simulations of the system and leveraged smart simulations to ensure a computationally-efficient way of detecting situations that have low probability. We also proposed an inference-based approach to detect some alert conditions with absolute certainty.

In the future, we would like to study the facets on the user end, like natural language processing for extracting the commands to a mathematical framework. We would also like to perform a human study with a well-designed interface, and work on identifying reasonable probability threshold values for the alert conditions by modelling a human user's preference from efficient choice experiments (Louviere et al. 2008) on mission outcomes.

Acknowledgment: This work was supported by the U.S. Army Research Laboratory. Any opinions or conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsor.

References

- [Al-Hussaini, Gregory, and Gupta 2018a] Al-Hussaini, S.; Gregory, J. M.; and Gupta, S. K. 2018a. Generation of context-dependent policies for robot rescue decision-making in multi-robot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- [Al-Hussaini, Gregory, and Gupta 2018b] Al-Hussaini, S.; Gregory, J. M.; and Gupta, S. K. 2018b. A policy synthesis-based framework for robot rescue decision-making in multi-robot exploration of disaster sites. In *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*.
- [Barnes et al. 2014] Barnes, M. J.; Chen, J. Y.; Jentsch, F.; Oron-Gilad, T.; Redden, E.; Elliott, L.; and Evans III, A. W. 2014. Designing for humans in autonomous systems: Military applications. Technical report, ARMY RESEARCH LAB ABERDEEN.
- [Chai et al. 2016] Chai, J. Y.; Fang, R.; Liu, C.; and She, L. 2016. Collaborative language grounding toward situated human-robot dialogue. *AI Magazine* 37(4):32–45.
- [Chien et al. 2018] Chien, S.-Y.; Lin, Y.-L.; Lee, P.-J.; Han, S.; Lewis, M.; and Sycara, K. 2018. Attention allocation for human multi-robot control: Cognitive analysis based on behavior data and hidden states. *International Journal of Human-Computer Studies* 117:30–44.
- [DeCostanza et al. 2018] DeCostanza, A. H.; Marathe, A. R.; Bohannon, A.; Evans, A. W.; Palazzolo, E. T.; Metcalfe, J. S.; and McDowell, K. 2018. Enhancing human-agent teaming with individualized, adaptive technologies: A discussion of critical scientific questions. Technical report, US Army Research Laboratory Aberdeen Proving Ground United States.
- [Diller et al. 2014] Diller, T.; Helmrich, G.; Dunning, S.; Cox, S.; Buchanan, A.; and Shappell, S. 2014. The human factors analysis classification system (hfacs) applied to health care. *American Journal of Medical Quality* 29(3):181–190. PMID: 23814026.
- [Gregory et al. 2016] Gregory, J.; Fink, J.; Stump, E.; Twigg, J.; Rogers, J.; Baran, D.; Fung, N.; and Young, S. 2016. Application of Multi-Robot Systems to Disaster-Relief Scenarios with Limited Communication. *Field and Service Robotics: Results of the 10th International Conference* 639–653.
- [Jentsch 2016] Jentsch, F. 2016. *Human-robot interactions in future military operations*. CRC Press.
- [Konur 2013] Konur, S. 2013. A survey on temporal logics for specifying and verifying real-time systems. *Frontiers of Computer Science* 7(3):370–403.
- [Kruijff et al. 2014] Kruijff, G.-J. M.; Janíček, M.; Keshavadas, S.; Larochelle, B.; Zender, H.; Smets, N. J.; Mioch, T.; Neerincx, M. A.; Diggelen, J.; Colas, F.; et al. 2014. Experience in system design for human-robot teaming in urban search and rescue. In *Field and Service Robotics*, 111–125. Springer.
- [Levine and Williams 2014] Levine, S. J., and Williams, B. C. 2014. Concurrent plan recognition and execution for human-robot teams. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.
- [Louviere et al. 2008] Louviere, J. J.; Street, D.; Burgess, L.; Wasi, N.; Islam, T.; and Marley, A. A. 2008. Modeling the choices of individual decision-makers by combining efficient choice experiment designs with extra preference information. *Journal of choice modelling* 1(1):128–164.
- [Makris et al. 2016] Makris, S.; Karagiannis, P.; Koukas, S.; and Matthaakis, A.-S. 2016. Augmented reality system for operator support in human–robot collaborative assembly. *CIRP Annals* 65(1):61–64.
- [Mosier et al. 2017] Mosier, K. L.; Fischer, U.; Burian, B. K.; and Kochan, J. A. 2017. Autonomous, context-sensitive, task management systems and decision support tools i: Human-autonomy teaming fundamentals and state of the art.
- [Murphy 2004] Murphy, R. R. 2004. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 34(2):138–153.
- [Murphy 2014] Murphy, R. R. 2014. *Disaster robotics*. MIT press.
- [Ouaknine and Worrell 2008] Ouaknine, J., and Worrell, J. 2008. Some recent results in metric temporal logic. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 1–13. Springer.
- [Ramakrishnan, Zhang, and Shah 2017] Ramakrishnan, R.; Zhang, C.; and Shah, J. 2017. Perturbation training for human-robot teams. *Journal of Artificial Intelligence Research* 59:495–541.
- [Ramchurn et al. 2016] Ramchurn, S. D.; Huynh, T. D.; Wu, F.; Ikuno, Y.; Flann, J.; Moreau, L.; Fischer, J. E.; Jiang, W.; Rodden, T.; Simpson, E.; et al. 2016. A disaster response system based on human-agent collectives. *Journal of Artificial Intelligence Research* 57:661–708.
- [Roldán et al. 2017] Roldán, J.; Peña-Tapia, E.; Martín-Barrio, A.; Olivares-Méndez, M.; Del Cerro, J.; and Barrientos, A. 2017. Multi-robot interfaces and operator situational awareness: Study of the impact of immersion and prediction. *Sensors* 17(8):1720.
- [Shappell et al. 2007] Shappell, S.; Detwiler, C.; Holcomb, K.; Hackworth, C.; Boquet, A.; and Wiegmann, D. A. 2007. Human error and commercial aviation accidents: An analysis using the human factors analysis and classification system. *Human Factors* 49(2).
- [Sherwood 2018] Sherwood, S. M. 2018. The effect of task load, automation reliability, and environment complexity on uav supervisory control performance.
- [Shriyam and Gupta 2018] Shriyam, S., and Gupta, S. K. 2018. Incorporating potential contingency tasks in multi-robot mission planning. In *IEEE International Conference on Robotics and Automation (ICRA)*.
- [Wong and Seet 2017] Wong, C. Y., and Seet, G. 2017. Workload, awareness and automation in multiple-robot supervision. *International Journal of Advanced Robotic Systems* 14(3):1729881417710463.