Safety Verification of Autonomous Systems: A Multi-Fidelity Reinforcement Learning Approach

Jared J. Beard and Ali Baheri

Abstract—As autonomous and semi-autonomous agents become more integrated with society, validation of their safety is increasingly important. The scenarios under which they are used, however, can be quite complicated; as such, formal verification may be impossible. To this end, simulation-based safety verification is being used more frequently to understand failure scenarios for the most complex problems. Recent approaches, such as adaptive stress testing (AST), use reinforcement learning, making them prone to excessive exploitation of known failures, limiting coverage of the space of failures. To overcome this, the work below defines a class of Markov decision processes, the knowledge MDP, which captures information about the learned model to reason over. More specifically, by leveraging, the “know what it knows” (KWIK) framework, the learner estimates its knowledge (model estimates and confidence, as well as assumptions) about the underlying system. This formulation is vetted through MF-KWIK-AST which extends bidirectional learning in multiple fidelities (MF) of simulators to the safety verification problem. The knowledge MDP formulation is applied to detect convergence of the model, penalizing this behavior to encourage further exploration. Results are evaluated in a grid world, training an adversary to intercept a system under test. Monte Carlo trials compare the relative sample efficiency of MF-KWIK-AST to learning with a single-fidelity simulator, as well as demonstrate the utility of incorporating knowledge about learned models into the decision making process.

I. INTRODUCTION

Safety validation evaluates a system to ensure performance criteria are met or to characterize the nature of their failures [1]. Such techniques are of utmost importance in scenarios where human lives are placed at great risk or failures could significantly damage expensive equipment. Some examples include aircraft autopilots, driverless cars, and space-flight systems. Formal verification methods, which have been widely used, build a rigorous mathematical or computational model of the system under test to reliably verify the system or exhaustively examine whether safety specifications have been fulfilled [2]. The key challenge to employing formal methods for safety verification is the curse of dimensionality.

This paper is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof. The U.S. Government does not endorse products or manufacturers. Trade or manufacturers’ names appear herein solely because they are considered essential to the objective of this paper. The findings and conclusions are those of the authors and do not necessarily represent the views of the funding agency. This document does not constitute FAA policy.

Another approach to safety validation is black-box and grey-box safety verification, wherein, the system is not known to a meaningful degree or provides limited information to reason over, respectively [1]. These techniques range from planning algorithms [3], [4] and optimization tools [5], [6] to reinforcement learning [7]. From the reinforcement learning perspective, adaptive stress testing (AST) aims to identify the most likely failure scenarios by framing the problem as a Markov decision process (MDP) and searching for an adversarial policy that maps environmental states to disturbances causing system failure. AST has been applied to practical domains, ranging from aircraft collision avoidance [7] to autonomous vehicles [8]. Despite overall success, sampling-based approaches, such as reinforcement learning, suffer from two key drawbacks. They fail to find particularly sparse failures [9]. Additionally, they gather data from high-fidelity simulators, which may exceed computational budgets [10].

To reduce the need for samples from high-fidelity simulators, recent reinforcement learning approaches use multiple fidelities of simulators and integrate information from each [11]. The level of fidelity indicates the extent to which a simulator simplifies the system model and its assumptions. Low-fidelity simulators make strong assumptions and greatly simplify the underlying system, enabling relatively fast execution. Conversely, high-fidelity simulators make fewer assumptions and exhibit more realistic behaviors and dynam-
ics. Their execution may be significantly slower than low-fidelity simulators depending on how closely the simulators approximate reality. To date, there has been one multi-fidelity (MF) safety validation work [12]. The work of Koren, et al serves as the sole MF alternative to our proposed approach and passes information from low to high-fidelity simulators (unidirectionally). Unidirectional techniques limit information passing, hence the efficiency with which information can be exploited. Bidirectional approaches (passing information both directions) such as the work of this paper, by contrast, have been shown to have lower sample complexity. The upper bound on high-fidelity samples of bidirectional approaches being equivalent to the unidirectional case [11]. Cutler et al’s bidirectional approach, while not used for safety validation, offers insight to the problem. Their approach uses the “knows what it knows” (KWIK) framework to provide metrics over which to heuristically reason about the relative worth of a sample in a given fidelity.

The objective of this work is to develop a framework that minimizes the simulation cost while maximizing the number of failure scenarios discovered. In contrast to previous safety validation techniques, we propose a learner that uses knowledge about itself to reformulate the safety validation problem and present a bidirectional algorithm MF-KWIK-AST to solve this problem (Fig. 1). Results are tested in a grid world setting with puddles that inhibit motion. An MK-KWIK-AST adversary is trained to intercept a system under test attempting to reach a goal coordinate.

A. Contributions

The contributions of this work are as follows:

- We propose a novel knowledge Markov decision process (KMDP) framework, which factors knowledge of the learned model into the decision making process and explicitly distinguishes the system model from the knowledge of this model;

- we demonstrate MF-KWIK-AST an algorithm using the KWIK framework to train learners from multiple fidelities of simulators in a bidirectional manner. MF-KWIK-AST also uses a knowledge based update to prevent early convergence of the learned model and encourage further exploration.

B. Outline

The paper is structured as follows: Section II briefly outlines AST and KWIK, as well as introduces the KMDP. Section III details the MF-KWIK-AST algorithm. Section IV covers the experimental setup, results, and some limitations of the implementation. Lastly, Section V presents concluding remarks and future directions of work.

II. BACKGROUND

A. Notation

This work relies on MDP so it is useful to introduce notation which will form the foundation of this work. MDP are defined by the tuple \( \langle S, A, T, R, \gamma \rangle \). Here, \( S \) is the set of all states \( s \) and \( A \) the set of all actions \( a \). The stochastic transition model is \( T(s' \mid s, a) \), where \( s' \) represents a state outcome when taking \( a \) from \( s \). \( R(s, a, s') \) is the reward for some transition. Lastly, \( \gamma \) is a temporal discount factor. By solving an MDP, an optimal control policy \( \pi^*(s) \forall s \in S \) can be achieved. Here, the optimal policy is defined as follows:

\[
\pi^*(s) = \arg \max_a Q^*(s, a),
\]

where \( Q^*(s, a) \) the optimal state-action value function is

\[
Q^*(s, a) = \max_{s'} T(s' \mid s, a)[R(s, a, s') + \gamma Q^*(s', a')].
\]

B. Adaptive Stress Testing

This work builds on the concept of AST. The key objective of which is to find the most likely failure scenarios for a (decision-making/control) system under test \( \mathcal{M} \). Here, failure scenarios are defined as the sequence of transitions that lead to some failure condition (e.g., midair collision). The system under test is placed in a black-box (usually high-fidelity) simulator \( \Sigma \), so the AST learner (with planner \( P \)) does not have direct access to the internal workings of the system. AST then samples scenarios iteratively to learn and uses the planner to apply disturbances (e.g., control agents in the environment, supply noise, or adjust parameters of the environment). More specifically, AST uses the simulations to minimize the distance to a failure metric, while maximizing the log-likelihood of trajectories to guide the most likely failures. The work presented here seeks to reduce the need for expensive simulations by using MF simulations with bidirectional information passing to improve the sample efficiency of such techniques.

C. Knows What It Knows Learner

Use of MF simulators in reinforcement learning has come about from the desire to reduce number of samples in and expense incurred by high-fidelity simulators used to get reliable performance. To this end, work by Cutler, et al [11] leveraged the concept of KWIK learners [13]. The fundamental idea of KWIK is to use information about the learned model to provide output indicating its confidence in said model. When the learner is not confident in the model, it reports relevant state-action pairs as unknown (⊥) [13]. Cutler, et al use the number of samples to say a model is known to some degree of confidence specified by the Hoeffding inequality (parameterized by \( \epsilon, \delta \)) [11].

In the multi-fidelity case, the knowledge is used to decide when to switch between the \( D \) fidelities of simulators. The knowledge \( \mathcal{K} \) is defined for this work as both estimates of the transition and reward models \( (\mathcal{K}_T, \mathcal{K}_R) \), as well as the confidence in these models \( (\mathcal{K}_{T, \hat{T}}(s, a), \mathcal{K}_{R, \hat{R}}(s, a)) \). Here, the subscript \( d \) indicates the specific fidelity. The lower fidelities aim to guide search in higher fidelity simulators, while knowledge from higher fidelities is used to improve the accuracy of lower fidelity models. Mathematically, two
simulators are said to be in fidelity to each other if their $Q$ values are sufficiently close according to

$$ f(\Sigma_1, \Sigma_2, \rho_i, \beta_i) = \begin{cases} 
\Delta = -\max_{s,a} [Q^*_1(s,a) - Q^*_2(\rho_j(s), a)] , & \forall s, a \text{ if } \Delta \leq \beta_i 
-\infty, & \text{else.} 
\end{cases} \quad (3) $$

Note, $\beta$ indicates a user specified bound on how dissimilar two simulators can be and $\rho$ is a mapping of states from the fidelity of $s$ to fidelity $j$. In this work, this is applied on a state-by-state basis, as the decision process only needs to model one decision at a time, not the whole system.

D. Knowledge Markov Decision Process

Whereas this learning problem has classically been represented as an MDP, doing so assumes the model is known with sufficient accuracy. This may not always be the case (as in reinforcement learning). When the decision maker can quantify its lack of information about the model, it should be able to use this information to its benefit. Thus, we introduce the KMDP as $(\mathcal{S}, \mathcal{A}, T, \mathcal{R}, \mathcal{K}, \gamma)$. Here, $\mathcal{K}$ represents the knowledge about the decision maker’s model of the system (e.g., estimates of the other terms $(\mathcal{S}, \mathcal{A}, T, \mathcal{R})$, confidence in those estimates, and other assumptions of the learned model). Consequently, $\hat{T}$ and $\hat{R}$ are used to solve the underlying MDP as the agent learns. Note from Sec. II-C these estimates of the model are considered part of $\mathcal{K}$. In doing so, the learner is explicitly aware that it is making decisions based on a model of its environment (rather than the actual environment) and that it has some confidence in these models. Thus this information can factor into decisions, such as through the reward which is now defined as $\hat{R}(s, a, s', \mathcal{K})$. As an example, knowledge about the lack of information for some subset of states could necessitate a reward that encourages exploration in those regions.

The KMDP is related to the concepts of maximum likelihood model MDPs and reward shaping. The maximum likelihood model MDP and KMDP share similar update rules for the learned reward and transition models [14]. However, KMDP generalizes these concepts to include estimates of $\mathcal{S}$ and $\mathcal{A}$, as well as providing a more broad use of the knowledge gained from sampling. As an example, the agent may not be aware of all states if it is required to explore its environment, thus the estimated state space would be a subset of $\mathcal{S}$. Furthermore, knowledge may now capture concepts such as whether a state has been visited, which is not inherently captured by the underlying state. It is important to note that this would break the Markov property, but such an approximation has been used to break explicit temporal dependencies [15]. The second concept is reward shaping, where the additional reward terms guide learning as a function of the system state [16]. The difference being that here the learner is deciding based on learned information or knowledge about the model, not only the underlying state. Following from this, $Q^*$ cannot actually be achieved, and the best $Q$ given $\mathcal{K}$ becomes

$$ Q_{\mathcal{K}}(s, a) = \max_a \sum_{s'} \mathcal{K}.\hat{T}(s' | s, a)[\mathcal{K}.\hat{R}(s, a, s', \mathcal{K}) + \gamma Q_{\mathcal{K}}(s', a')]. \quad (4) $$

III. MF-KWIK-AST

Here, the functionality of the MF-KWIK-AST algorithm (Alg. 1) is described. The approach centers on three functionalities: search, evaluateState, and marginalUpdate. The search function is tasked with iterating through trials and collecting failure information. As part of search, evaluateState simulates trajectories and updates the learned models. Lastly, marginalUpdate, given confidence information, updates the reward function to bias search away from trajectories for which the models are known (sampled a sufficient number of times). MF-KWIK-AST is initialized with a set of simulators, fidelity constants, and state mappings $(\Sigma, \beta, \rho)$, the system under test $\mathcal{M}$, and the planner $P$ of choice (this work uses value iteration). Additionally, the confidence parameters $(\epsilon, \delta)$, and minimum samples to change fidelity $(m_{\text{known}}, m_{\text{unknown}})$ are supplied. For more information on the choice of these parameters, see [11]. With no samples, $\mathcal{K}$ is initialized with a uniform distribution over transitions and rewards set as $R_{\text{max}}$ of the KMDP. The $Q$ values are set to 0 and the current fidelity $d$ is declared as the lowest fidelity. Lastly, it is assumed there are no changes to the model estimate, and no known or unknown samples $(m_k, m_u)$.

A. Search

The search algorithm serves primarily to initiate the learning process. MF-KWIK-AST looks at failures from a single initial condition. As such, a state $s$ is passed in along with a desired number of trajectories $n$. The estimate of the set of failure modes, $\mathcal{F}$ is initialized as empty. For every iteration, the simulators are reset and evaluateState is called. Failure scenarios found by evaluateState are added to $\mathcal{F}$. Due to the nature of sampling, policies may come from more than one fidelity. Thus, after the search has concluded, it is necessary to evaluate whether failure modes are viable in the highest fidelity; those not meeting this condition are rejected. The remaining set is returned to the user.

Plausibility checks may manifest in a few ways: for simulators with relatively simple transition models, these can be checked directly to exist or not; for more complex, stochastic systems, this may involve a Monte Carlo sampling to determine if a transition occurs in the model. Fortunately, due to the Markov assumption, this can be broken down to a state-by-state basis, as opposed to looking at the entire trajectory at once.

B. State Evaluation

At every iteration of the search algorithm, the evaluateState function attempts to find a failure scenario given the current model estimate. The process begins by initializing the trajectory $f$. Then actions are iteratively selected from the current policy $\pi$ and the decision is made to increment
Algorithm 1: MF-KWIKA-AST

input: $(\Sigma, \beta, \rho, M, R_{\text{inc}}, P, (\epsilon, \delta), (m_{\text{known}}, m_{\text{unknown}}))$
initialize: $K, Q_d$
initialize: $\text{change}_d = \text{false}, d = 1, m_k, m_u = 0$

Procedure search($s, n$)

$\mathcal{F} \leftarrow \emptyset$
for $i = 1, \ldots, n$ do
\begin{itemize}
  \item \text{reinitialize}($\Sigma$)
  \item $\mathcal{F} \leftarrow \mathcal{F} \cup \text{evaluateState}(s)$
\end{itemize}
for $f \in \mathcal{F}$ do
\begin{itemize}
  \item if $\neg \text{isPlausible}(f, D)$ then
    \item $\mathcal{F} \leftarrow \mathcal{F} \setminus f$
\end{itemize}
return $\mathcal{F}$

Procedure evaluateState($s$)

initialize: $f \leftarrow \emptyset$
while $\neg \text{terminal}(s)$ do
\begin{itemize}
  \item $a \leftarrow \{P.\pi(s), M.\pi(s)\}$
  \item if $d > 1$ and $\text{change}_d$ and $K_{d-1}(\rho_{d-1}(s), a, \rho) = \bot$ and $m_u \geq m_{\text{unknown}}$ then
    \item $Q_{d-1} \leftarrow \text{plan}(d - 1)$
    \item $d \leftarrow d - 1$
  \item else
    \begin{itemize}
      \item $s', r \sim T_{\Sigma}(s' | s, a)$
      \item if $K_d(s, a) = \bot$ then
        \item $\text{update}(K_d(s, a))$
      \item if $K_d(s, a) \neq \bot$ then
        \item $Q_d \leftarrow \text{plan}(d)$
        \item $f \leftarrow f \cup \langle s, a, s' \rangle$, $s \leftarrow s'$
        \item if $d < D$ and $m_k \geq m_{\text{known}}$ then
          \item $Q_{d+1} \leftarrow \text{plan}(d + 1)$
          \item $d \leftarrow d + 1$
    \end{itemize}
\end{itemize}
if $\text{isConverged}(f)$ then
\begin{itemize}
  \item $\text{marginalUpdate}(f)$
\end{itemize}
if $\text{isFailure}(f)$ then
\begin{itemize}
  \item return $f$
\end{itemize}

Procedure plan($d$)

for all $\langle s, a \rangle$ do
\begin{itemize}
  \item $d' \leftarrow \{\max(d') \mid d' \geq d, K_{d'}(s, a) \neq \bot\}$
  \item $K_d(s, a) \leftarrow K_{d'}(s, a)$
\end{itemize}
return $P.\text{train}(K, \hat{T}, K, \hat{R}, Q_d, Q_{d-1} + \beta_{d-1})$

Procedure marginalUpdate($f$)

$m_{\text{max}} \leftarrow 0$, $t \leftarrow \langle \rangle$
for $(s, a, s') \in f$ do
\begin{itemize}
  \item $m = Q_K(s, a) - Q_K'(s, a')$
  \item if $m > m_{\text{max}}$ then
    \item $m_{\text{max}} \leftarrow m$
    \item $t \leftarrow \langle s, a, s' \rangle$
\end{itemize}
$R(t) \leftarrow R(t) - R_{\text{inc}}$
$Q_d \leftarrow \text{plan}(d)$

the fidelity or sample the current one. This continues until a terminal state is found; terminal states can be a failure event, exceeding the maximum number of time steps, or reaching some state where failure cannot occur. Whenever the fidelity is incremented, both $m_k$ and $m_u$ are reset. Otherwise, if a known state-action is reached, $m_k$ is incremented, while $m_u$ is reset. The opposite occurs when an unknown state is reached. Similarly, $\text{change}_d$ is set to false when the state is incremented and set to true when a state-action becomes known.

The fidelity is decremented if a known state has not recently been reached and the lower fidelity states are also known. This permits more efficient search by using the less expensive simulator to carry out uncertain actions. Along with the decrement, the planner is updated using the current model. Otherwise, a simulation is performed to gather data from the model. If the current state is considered unknown, this information is used to update the learned model and knowledge. The knowledge about the transition is incremented by one (the distribution is generated by scaling the number of samples over all transitions from taking $a$ at $s$). The reward is updated as a weighted average of the reward estimate and sampled reward. Lastly, the knowledge about each, being the number of samples for a state-action, is incremented by one. If a state-action becomes known, the planner is updated. The transition is added to the trajectory and the current state is updated. Subsequently, the fidelity is incremented and re-planning occurs if enough known samples have been visited. The algorithm concludes with the marginal update if the trajectory has converged (all $\langle s, a \rangle \in f$ considered known). If a failure state has been achieved by the trajectory, the trajectory is deemed a failure scenario and returned.

When planning, the algorithm must pass information to lower fidelities to improve the performance of future samples. The plan function does this by finding the highest fidelity ($\geq d$) for which a state-action pair is known. This knowledge is then used to update the knowledge of the current fidelity. Similarly, to pass information up, if states from the current $d$ and next lower $d - 1$ model estimates are deemed to be in fidelity ($\Delta \leq \beta$), the lower fidelity model estimate is used in place of the current fidelity. The planner is then run to update $Q$.

C. Marginal Update

The marginal update incorporates knowledge into the decision making process. Here, the marginal is defined as the difference in the best state-action value $Q_K$ and the second best state-action value $Q_{K'}$ for a given state. When the system is known to be converged, the state having the largest marginal (along the trajectory) is selected. Using this state, the reward is decreased by $R_{\text{inc}}$ and the planner updated.

The underlying idea is that if the learner has converged on a trajectory, it will cease to explore the solution space, exploiting an action repeatedly. By finding the state where the best action is significantly greater than its neighbors, exploration can erode the most functionally important actions for a
Fig. 2. 4×4 Grid worlds under test (left) low-fidelity, (right) high-fidelity. Here, fidelity is demonstrated by accuracy of the model, where the low-fidelity does not capture the impact of puddles on the transition model. The blue dot indicates the system under test, the red the MF-KWIK learner, and the green “×” the system under test’s goal. Puddles are indicated by black and grey regions.

given trajectory. As training continues, this cuts off search in more fragile trajectories (hence less likely trajectories early) and encourages exploration along the entirety of more robust failure trajectories.

IV. RESULTS

A. Simulation

The simulator involved a 4×4 grid world with puddles (Fig. 2). Within the grid world, the system under test was given a goal location; the learner was tasked with causing failures by intercepting the system under test either by blocking it or inducing a collision. Herein, the system under test was assumed to be myopic, taking the closest unobstructed action towards the goal. States are defined by the coordinates of the system under test \((x, y)\)_M and the learner \((x, y)\)_P. Actions for both were to move one cell along the cardinal directions or remain in place. The agents were both assumed to be fully controllable unless in a puddle. In which case, the transition probability of reaching the desired cell was 20%; else the agent would not move. The learner was penalized by the negative of its distance to the system under test, \(-5\) when in a puddle, and \(-25\) when the system under test reached the goal. The learner earned a reward of 50 for inducing a failure. \(\gamma\) was set to 0.95 for experimentation. Note that while in the AST, a term was included to optimize the log-likelihood of a trajectory, here this was forgone given the relatively small state space.

B. Monte Carlo Trials

To evaluate the approach, 25 Monte Carlo trials were conducted for both the single-fidelity (SF) and MF learners. Here, the SF learner captures the behavior of AST using a KWIK learner, dubbed KWIK-AST (Alg. 2, see Appendix for description). To evaluate the performance of the marginal update, each of five values for \(R_{inc} = [0, 0.25, 1, 2, 5]\) at up to 1000 iterations were used in both simulators. Note that \(R_{inc} = 0\) indicates no marginal update. Each trial, the initial coordinates of the system under test and learner were uniformly sampled; samples that obviously could not cause failures or scenarios initialized to failure states were removed. Other parameters were set as follows: \(\beta = 1250, t_{max} = 20, \epsilon = 0.25, \delta = 0.5, m_{known} = 10\), and \(m_{unknown} = 5\).

Given, the relatively simple scenario, it is helpful to understand where the MF simulator converges in the highest fidelity. Much larger examples involve sparse scenarios, so the area of interest are those regions where the ratio of high-fidelity to low-fidelity samples (Fig. 3) is relatively low and likely far below this convergence estimate. For context, the optimal high-fidelity policy could be found at a high-to-low-fidelity sample ratio of less than \(\sim 0.4\). Notice that the marginal update does not appreciably impact the ratio, except for \(R_{inc} = 5\), where the search reached under-explored states in low fidelities. Naturally, as the algorithm converges, return to lower fidelities is prevented and the samples grow at approximately the same rate as the SF case, hence the near linear behavior after convergence (Fig. 4). In The SF scenario, increasing \(R_{inc}\) led to further search, though in the MF scenario the behavior was unclear, likely due to interactions from changing fidelities.

With respect to the number of failures, the MF simulator quickly accumulates failures from low-fidelity samples, while the earliest high-fidelity samples in the SF simulator are wasted (Fig. 5). Around convergence, their performance...
is comparable. Without the marginal update, search in SF ceases at convergence; a large $R_{inc}$ yields $\sim 15\%$ increase in failures found. In the MF scenario, small updates perform similarly to no update, however, $R_{inc} = 5$ yields a significant improvement, as much as $15\%$ with almost no HF samples, supporting use of the marginal update. The improvements of the MF scenario are further realized in Fig. 6. The MF scenario finds as many as 6 times more failures per sample than the SF case early on, with their performance approaching unity after convergence.

C. Limitations

There were two primary limitations with the implementation. The first being that this algorithm was designed for discrete problems; while there are continuous extensions to the KWIK frameworks, representations for knowledge and coverage of continuous domains can be difficult to accomplish. Second, because value iteration provides exact solutions to MDPs, it does not scale well (hence the limited number of Monte Carlo trials and relatively small environment). This could likely be overcome using an anytime solver such as Monte Carlo tree search [17].

V. CONCLUSIONS

The work presented here aimed to improve the ability of reinforcement learning techniques to find failure scenarios for an autonomous system. In particular, this work defined the knowledge MDP, using insights from KWIK learners. The key contribution being the integration of a learner’s knowledge about its model estimates into the decision-making process. This supported the development of a bidi-
rectional algorithm, MF-KWIK-AST. Results demonstrated significant improvements in the MF approach relative to SF. The integration of knowledge into the solution through the marginal update offered promising early results for the KMDP formulation. In particular, it continued exploration in scenarios where the learning converged and bolstered early exploration of the solution space.

Given that the theoretical limit on samples for bidirectional learning is upper bounded by the unidirectional case, this should present a viable competitor to existing techniques. Before these techniques can be directly compared, additional efforts are required to generalize this work. First, to improve scalability, the learned models should be used by anytime planners such as Monte Carlo tree search. Additionally, further work is necessary to apply the framework to continuous state and action spaces. With these innovations in place, the work can be extended to partially observable Markov decision processes. In parallel, the KMDP formulation could be tested in broader contexts to fully explore its utility in reinforcement learning.

APPENDIX

The KWIK-AST formulation (Alg. 2) was generated by stripping MF-KWIK-AST of its dependence on fidelity. As such, supplying MF-KWIK-AST with a SF simulator, will have the same performance. In spite of this, the KWIK-AST algorithm was used for experiments. Notice, there is no longer a need to check the veracity of failures since all samples are from a single fidelity. Furthermore, planning is accomplished using the models for $\hat{T}$ and $\hat{R}$ as there is no longer information to pass. This reduces the plan step to training as would be done in classical approaches.

VI. ACKNOWLEDGMENT

This research was supported in part by the FAA under contract No. 692M15-21-T-00022. This work was also funded in part by the Ruby Distinguished Doctoral Fellowship. Additionally, the authors would like to thank Jared Strader, Nicholas Ohi, Hao Ren, and Peng Wei for their contributions to their efforts in formulating this work.

REFERENCES

[1] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, “A survey of algorithms for black-box safety validation of cyber-physical systems,” Journal of Artificial Intelligence Research, vol. 72, pp. 377–428, 2021.

[2] J. V. Deshmukh and S. Sankaranarayanan, “Formal techniques for verification and testing of cyber-physical systems,” in Design Automation of Cyber-Physical Systems. Springer, 2019, pp. 69–105.

[3] E. Plaku, L. E. Kavraki, and M. Y. Vardi, “Hybrid systems: from verification to falsification by combining motion planning and discrete search,” Formal Methods in System Design, vol. 34, no. 2, pp. 157–182, 2009.

[4] C. E. Tuncali and G. Fainekos, “Randomly-exploring random trees for testing automated vehicles,” in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 661–666.

[5] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, “Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles,” Journal of Systems and Software, vol. 137, pp. 197–215, 2018.

[6] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu, “Testing cyber-physical systems through Bayesian optimization,” ACM Transactions on Embedded Computing Systems (TECS), vol. 16, no. 5x, pp. 1–18, 2017.

[7] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer, “Adaptive stress testing: Finding likely failure events with reinforcement learning,” Journal of Artificial Intelligence Research, vol. 69, pp. 1165–1201, 2020.

[8] M. Koren, S. Alsaiif, R. Lee, and M. J. Kochenderfer, “Adaptive stress testing for autonomous vehicles,” in 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 1–7.

[9] A. Corso, R. Lee, and M. J. Kochenderfer, “Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition,” in 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2020, pp. 1–6.

[10] R. J. Moss, R. Lee, N. Visser, J. Hochwarth, J. G. Lopez, and M. J. Kochenderfer, “Adaptive stress testing of trajectory predictions in flight management systems,” in 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). IEEE, 2020, pp. 1–10.

[11] M. Cutler, T. J. Walsh, and J. P. How, “Real-world reinforcement learning via multifidelity simulators,” IEEE Transactions on Robotics, vol. 31, no. 3, pp. 655–671, 2015.

[12] M. Koren, A. Nassar, and M. J. Kochenderfer, “Finding failures in high-fidelity simulation using adaptive stress testing and the backward algorithm,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 5944–5949.

[13] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl, “Knows what it knows: A framework for self-aware learning,” Machine learning, vol. 82, no. 3, pp. 399–443, 2011.

[14] M. Wiering and J. urgen Schmidhuber, “Learning exploration policies with models,” Proc. CONALD, 1998.

[15] J. J. Beard, Environment Search Planning Subject to High Robot Localization Uncertainty. West Virginia University, 2020.

[16] A. D. Laud, Theory and application of reward shaping in reinforcement learning. University of Illinois at Urbana-Champaign, 2004.

[17] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, “A survey of Monte Carlo tree search methods,” IEEE Transactions on Computational Intelligence and AI in games, vol. 4, no. 1, pp. 1–43, 2012.