Pre-print of accepted manuscript.
Presented at the 17th ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI), 2022.

# Leveraging Non-Experts and Formal Methods to Automatically Correct Robot Failures

Sanne van Waveren
KTH Royal Institute of Technology
Stockholm, Sweden
sannevw@kth.se

***Abstract*—State-of-the-art robots are not yet fully equipped to automatically correct their policy when they encounter new situations during deployment. We argue that in common everyday robot tasks, failures may be resolved by knowledge that non-experts could provide. Our research aims to integrate elements of formal synthesis approaches into computational human-robot interaction to develop verifiable robots that can automatically correct their policy using non-expert feedback on the fly. Preliminary results from two online studies show that non-experts can indeed correct failures and that robots can use the feedback to automatically synthesize correction mechanisms to avoid failures.**

***Index Terms*—robot failure, policy repair, non-experts, shielded reinforcement learning**

## I. Introduction

When we deploy robots in the real world, automatic failure recovery is needed to succeed in new situations. This typically requires costly expert intervention, or substantial time or data to retrain the policy. Yet, certain failures, especially when they happen in everyday contexts, can also be understood and corrected by people who do not necessarily have technical expertise, i.e., non-experts (NEs), see Fig. 1a. Our research leverages NEs to automatically correct high-level robot actions on the fly in sequential decision-making tasks. We synthesize repair mechanisms from NE feedback, which can circumvent delays when experts are not available, let end-users tune their robot to match their personal preferences, and exploit diverse feedback from a large number of individuals.

In prior work, NEs have successfully taught robots cooking [1]–[4], navigation [5], [6], and object sorting tasks [4], [7]. Often, the teacher was required to be present throughout the entire training process. However, during deployment it becomes impractical to require NEs to constantly monitor the robot [8]. This inspired us to investigate: **How NEs can correct robots only *after* a robot failure occurred?**

Existing approaches typically also only consider feedback pre-deployment. If we want to correct robot policies or plans on the fly, automated corrections and formal guarantees on the system's behavior become crucial. Formal synthesis approaches can automatically synthesize correction mechanisms, referred to as shields [9], [10], from a given specification to enforce that the robot's behaviors always satisfy the specification. To date, such formal guarantees focus primarily on safety aspects, and have received less attention in HRI approaches [11], e.g., to correct robot failures and define

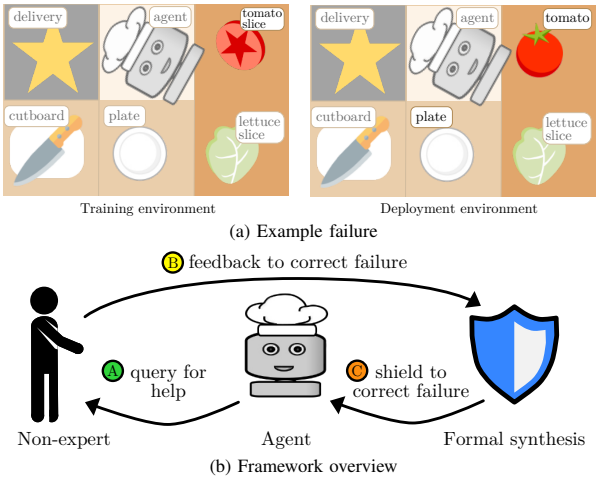


Fig. 1. (a) The robot needs to make a salad with a tomato slice, but in the deployment environment, it can only find a whole tomato. NEs can suggest to resolve the failure by instructing the robot to cut the tomato. (b) Framework to automatically correct robot failures using NEs and formal synthesis.

appropriate alternatives when the original policy fails. This raises our second research question: **How can we bring formal synthesis approaches into HRI to develop verifiable robots that automatically correct their policy using NEs?**

Fig. 1b illustrates our approach, with three main components: a) the robot's query to request NE feedback, b) the feedback to correct failures, and c) the shield generated from this feedback and applied to the robot's behavior. In the next sections, we discuss our efforts to realize our approach.

## II. Non-Expert-Generated Robot Specifications

We focus primarily on NE feedback that can be translated into Temporal Logic (TL) specifications for verifying system behaviors [12], [13]. Due to their resemblance to natural language, TLs balance the trade-off between rigorousness and simplicity for NEs. While existing approaches usually require experts to manually define specifications or to turn them into shields, our work explored if NEs can author specifications that are rich enough to be used in formal synthesis while structured enough to be useful for robots to automatically synthesize specifications. In addition to rigor and simplicity, time-efficiency is important; NEs should be able to help the robot without having to go through substantial training.

Our first online study investigated whether NEs can create specifications in the form of simple robot programs for a navigation and pick-and-place task, with as little as one minute of instructions [14]. In both tasks, the robot may fail at some point due to missing rules. Participants were asked to provide these rules in a block-based programming interface. In the navigation task, a cat may randomly appear that would block the robot's way and the robot might get stuck. Participants could suggest *if the cat is blocking the way, make the cat go away*. In the pick-and-place task, the robot was tasked to only stack pink cylinders, not other colored cylinders. Participants could specify an alternative action: "*if a cylinder is not pink, place it in the bin*". Initial results suggested that people were generally able to successfully write specifications for the robot to handle such exceptions. However, when NEs had to use a larger number of programming statements (i.e., use more advanced programming blocks), task load increased and task usability decreased. Our second work explored natural language, which is intuitive for NEs [15] and machine-translatable [16], and reduces the risk of having the visual programming interface as a confounding factor on task performance.

## III. Formally Enforcing Robot Behaviors

While crucial for robot safety and trustworthiness, formal verification has received less attention in HRI [11]. Formal methods-based approaches have been primarily used to enforce safe robot behaviors, e.g., constrained RL approaches [17]–[19], TL-based approaches [20]–[22], or shielded RL [9], [23], [24]. To date, shields have been used to enforce safety properties, but not to repair policies in case of failures.

In our second online study, we extended shielding to repair policies and investigated whether NE feedback can be used to automatically synthesize shields to repair failures of an RL agent [25]. We assume that the robot can detect when a failure occurs, but might not know the cause. NEs would suggest alternative actions or items, or define forbidden actions, in three cooking tasks. Participants were instructed to provide natural language feedback to the simulated robot using TL operators, e.g., always, never (i.e., always not) and conditional statements, e.g., if X then Y (i.e., implies). For brevity, we elaborate on the action refinement for failures that can be corrected by proposing alternative actions instead of executing the failed action $a_F$. The other two shields are synthesized similarly. For example, in Fig. 1a, NEs may suggest to chop the tomato first. This feedback indicates that the original action $\text{fetch}^{\text{tomatoSlice}}$ will result in a failure state. We denote a set of desired states $\mathcal{S}_{\text{desired}}$ that does not include such failure states. If the agent were to leave $\mathcal{S}_{\text{desired}}$, we correct the chosen action $a$ with the corrective actions as suggested by the NE. The function $c(s, a)$ returns corrective action(s), e.g., $\text{fetch}^{\text{tomato}}$ and $\text{chop}^{\text{tomato}}$ instead of $\text{fetch}^{\text{tomatoSlice}}$. The $\text{refine}$ shield minimally interferes with the learning by only correction actions $a \in \mathcal{A}$ if they lead to failures:

$$\text{refine}_s(a) = \begin{cases} a, & \text{if } \delta(s,a) \in \mathcal{S}_{\text{desired}} \\ c(s,a) \in \mathcal{A}, & \text{if } \delta(s,a) \notin \mathcal{S}_{\text{desired}} \end{cases}, \quad (1)$$

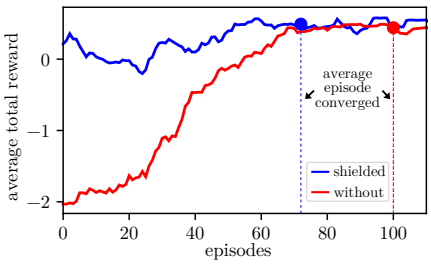


Fig. 2. Average total reward with and without NE-generated shielding.

where $\delta(s, a)$ obtains the next state when applying action $a$ in state $s$. Retraining with our shield resulted in an average of 30% faster learning compared to retraining without a shield (see Fig. 2). In this work, we showed for all three cooking tasks that we can synthesize shields from NE feedback that correct failures of robot policies while increasing data-efficiency and reducing the need for experts. Our experiments also show that our shields ensure that the robot does not enter failure states, whereas non-shielded learning is still prone to failures.

## IV. Future Work

Our work shows that 1) NEs can create simple robot programs, and 2) we can successfully generate shields from NE feedback that automatically correct the robot's actions. Our studies show the potential of combining formal synthesis with HRI to correct robot failures, specifically after deployment.

So far, we have not yet conducted a fine-grained analysis of the type of information needed to convey the robot failure. Diagnosis information and action recommendations can help remote operators to resolve failures [8]. For action or item refinement, NEs need to know what alternatives are available. For example, in Fig. 1a, NEs need to know that $\text{chop}$ and $\text{tomato}$ are an available action and item, respectively. Diagnosis information may include robot's dynamics, e.g., a robot that is stuck on carpet or unable to lift a heavy object. Currently, we investigate which information, e.g., velocity, exerted forces, intended motion [26], or context [27], robots need to provide so that NEs can identify and correct failures.

We will validate our approach in a real-world kitchen task with a manipulator robot that is tasked to set a dinner table. We want to explore if NEs can specify desired robot high-level pick-and-place (e.g., 'move object to' and 'close gripper') behavior using an object-centric TL. This TL defines spatial (e.g., left, right, above, below, close) and temporal (e.g., always, eventually, first) relations between the objects, e.g., the fork should always be placed left of the plate. From the TL specifications, we synthesize the robot controller. The TL tells us if/how much we satisfy these specifications. In case of violation (i.e., robot failure), NEs can correct the robot's behavior on the fly by refining existing or providing additional specifications. We will artificially induce failures, e.g., we occupy the space left of the plate with another object, after which we query remote NEs to correct the specification to include additional constraints or exceptions, e.g., 'if there is no space left of the plate, place the fork on the right instead'.

## REFERENCES


[1] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *AAAI*, vol. 6, 2006, pp. 1000–1005.

[2] ——, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, 2008.

[3] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, "Supervised autonomy for online learning in human-robot interaction," *Pattern Recognition Letters*, vol. 99, pp. 77–86, 2017.

[4] D. Koert, M. Kircher, V. Salikutluk, C. D'Eramo, and J. Peters, "Multi-channel interactive reinforcement learning for sequential tasks," *Frontiers in Robotics and AI*, vol. 7, 2020.

[5] W. B. Knox, C. Breazeal, and P. Stone, "Learning from feedback on actions past and intended," in *in Proc. of the Int. Conf. on Human-Robot Interaction, Late-Breaking Reports Session (HRI 2012)*. Citeseer, 2012.

[6] N. Wilde, A. Blidaru, S. L. Smith, and D. Kulić, "Improving user specifications for robot behavior through active preference learning: Framework and evaluation," *The Int. Journal of Robotics Research*, vol. 39, no. 6, pp. 651–667, 2020.

[7] H. B. Suay and S. Chernova, "Effect of human guidance and state space size on interactive reinforcement learning," in *Proc. of the IEEE Int. Symposium on Robot and Human Interactive Communication*, 2011, pp. 1–6.

[8] S. Banerjee, M. Gombolay, and S. Chernova, "A tale of two suggestions: Action and diagnosis recommendations for responding to robot failure," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 398–405.

[9] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Int. Conf. on Artificial Intelligence*, 2018.

[10] H. Hu, K. Nakamura, and J. F. Fisac, "SHARP: Shielding-aware robust planning for safe and efficient human-robot interaction," *arXiv preprint arXiv:2110.00843*, 2021.

[11] H. Kress-Gazit, K. Eder, G. Hoffman, H. Admoni, B. Argall, R. Ehlers, C. Heckman, N. Jansen, R. Knepper, and J. Křetínskỳ, "Formalizing and guaranteeing human-robot interaction," *arXiv preprint arXiv:2006.16732*, 2020.

[12] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[13] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[14] S. van Waveren, E. J. Carter, O. Örnberg, and I. Leite, "Exploring non-expert robot programming through crowdsourcing," *Frontiers in Robotics and AI*, p. 242, 2021.

[15] J. Y. Chai, M. Cakmak, C. Sidner, and J. Lupp, "Teaching robots new tasks through natural interaction," in *Interactive Task Learning: Agents, Robots, and Humans Acquiring New Tasks through Natural Interactions, Strüngmann Forum Reports, J. Lupp, series editor*, vol. 26, 2017.

[16] W. Zaremba, G. Brockman, and OpenAI. (2021) OpenAI codex. [Online]. Available: https://openai.com/blog/openai-codex/

[17] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Int. Conf. on Machine Learning*, 2017, pp. 22–31.

[18] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[19] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *Int. Conf. on Machine Learning*, 2020, pp. 9797–9806.

[20] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Authoring and verifying human-robot interactions," in *Proc. of the ACM Symposium on User Interface Software and Technology*, 2018, pp. 75–86.

[21] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-constrained reinforcement learning," *arXiv preprint arXiv:1801.08099*, 2018.

[22] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an RL agent using LTL," in *Proc. of the Int. Conf. on Autonomous Agents and MultiAgent Systems*, 2018, pp. 452–461.

[23] N. Jansen, B. Könighofer, S. Junges, and R. Bloem, "Shielded decision-making in MDPs," *arXiv preprint arXiv:1807.06096*, 2018.

[24] B. Könighofer, J. Rudolf, A. Palmisano, M. Tappler, and R. Bloem, "Online shielding for stochastic systems," in *NASA Formal Methods Symposium*, 2021, pp. 231–248.

[25] S. van Waveren, C. Pek, J. Tumova, and I. Leite, "Correct me if I'm wrong: Using non-experts to repair reinforcement learning policies," in *Proc. of the 2022 ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2022.

[26] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing robot incapability," in *Proc. of the ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2018, pp. 87–95.

[27] D. Das, S. Banerjee, and S. Chernova, "Explainable AI for robot failures: Generating explanations that improve user assistance in fault recovery," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 351–360.