Safe Human-Interactive Control via Shielding

Jeevana Priya Inala[1], Yecheng Jason Ma[2], Osbert Bastani[2], Xin Zhang[3], Armando Solar-Lezama[1]

***Abstract*— Ensuring safety for human-interactive robotics is important due to the potential for human injury. The key challenge is defining safety in a way that accounts for the complex range of human behaviors without modeling the human as an unconstrained adversary. We propose a novel approach to ensuring safety in these settings. Our approach focuses on defining *backup actions* that we believe human always considers taking to avoid an accident—e.g., brake to avoid rear-ending the other agent. Given such a definition, we consider a safety constraint that guarantees safety as long as the human takes the appropriate backup actions when necessary to ensure safety. Then, we propose an algorithm that overrides an arbitrary given controller as needed to ensure that the robot is safe. We evaluate our approach in a simulated environment, interacting with both real and simulated humans.[1]**

I. Introduction

Robots are increasingly operating in environments where they must interact with humans, such as collaborative grasping [1], [2] and autonomous driving [3], [4], [5], [6]. Thus, there has been much interest in designing planning and control algorithms for human-robot interaction. Ensuring safety for such robots is paramount due to the potential to inflict harm on humans [7]. These challenges are particularly salient in settings such as autonomous driving, where robots and humans may have disjoint or conflicting goals—e.g., a self-driving car making an unprotected left turn [5].

The key challenge is how to define safety for human-interactive robots. We could model the human as an adversary, but this approach is typically prohibitively conservative. Another approach is to learn a model to predict human actions [8], [9], [10], and ensure safety with respect to this model. If the model captures all actions exhibited by humans, then this approach ensures safety. However, different humans may exhibit very different behaviors [6]—e.g., people in a lab may act differently than people on a street. Collecting data from all possible settings can be very challenging. If a behavior is not exhibited in the training data, then the model may not account for it. More fundamentally, even the best machine learning models make errors, which can correspond to actions missed by the model. Finally, *responsibility sensitive safety (RSS)* [11] is an approach where that manually specifies the range of acceptable robot actions in various scenarios. That is, the designer of the robot controller is responsible for ensuring that acceptable actions only include safe actions. However, manually defining acceptable robot actions for all possible scenarios is challenging, especially for robots operating in open-world environments. For instance, [11] only formally defines acceptable actions for a limited number of scenarios such as changing lanes.

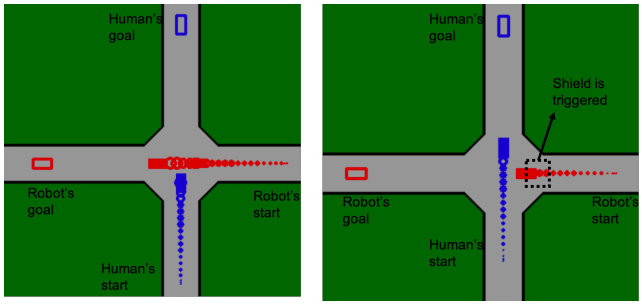


Fig. 1. Trajectories showing a robot (red) and a human (blue) interacting at an intersection (for 25 timesteps). Left: The robot passes before the human, leveraging the fact that a responsible human would slightly brake to allow the robot to cross safely. Right: Human arrives at the intersection first; the robot triggers the shield to brake and allow the human to cross first.

We propose a novel approach for ensuring safety in human-interactive systems that accounts for all human behaviors in some bounded set. There are two challenges: (i) how to define the set of human behaviors, and (ii) how to ensure safety with respect to this set. We address these challenges as follows:

- **Bounding human behavior via backup actions:** Rather than specify the set of all actions the robot is *allowed* to take (as in RSS), the designer specifies *backup actions* that we believe the human always considers taking to avoid an accident (e.g., braking while steering in some direction). In particular, we assume the human may take any action in general, but that they take these actions when necessary to ensure safety.
- **Ensuring safety via abstract interpretation:** We use *abstract interpretation* [12] to conservatively overapproximate the reachable set of the system for the above model of human behavior, and then ensure safety with respect to this overapproximation.

First, our notion of backup actions captures the idea that we reasonably believe the human will take a limited range of evasive maneuvers to avoid an accident—e.g., if the robot gradually slows to a stop, then we may expect the human

[1]Jeevana Priya Inala and Armando Solar-Lezama are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {jinala, asolar}@csail.mit.edu

[2]Yecheng Jason Ma and Osbert Bastani are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA {jasonyma,obastani}@seas.upenn.edu

[3]Xin Zhang is with the Department of Computer Science and Technology at Peking University, Beijing, 100871, China xin@pku.edu.cn

[1]Our full paper (with appendix): https://https://obastani.github.io/icra22.pdf.

to also slow down to avoid rear-ending it.[2] If the robot is on a highway, coming to a stop is more dangerous, so we may require that the robot pull over to the shoulder before coming to a stop. Similarly, we may require that the robot avoid coming to a stop in an intersection. Specifying backup actions provides a way to define safety; we refer to such a safety constraint as *safety modulo fault*.[3]

At a high level, to instantiate our framework, the designer of the robot controller needs to design the following:

- **Robot backup action:** An action the human anticipates the robot may take to ensure safety—e.g., brake without changing directions. It should be chosen based on intuition about what actions the human driver anticipates the robot may take (e.g., based on traffic rules).
- **Human backup action set:** A set of actions that includes *at least one* action the human considers taking to ensure safety—e.g., braking while steering in some direction. It should be chosen based on intuition about safety maneuvers the human driver considers taking.

In particular, whereas RSS requires the control designer to specify the set of *all* actions the robot is allowed to take, we only require the designer to specify actions that they believe the human considers taking to use to avoid an accident.

Next, we propose an algorithm for ensuring safety modulo fault. Our algorithm builds on *shielding* [13], [14], [15], [16]; in particular, rather than designing a specific controller, our algorithm takes as input an arbitrary controller designed to achieve the goal, and then combines it with backup controller in a way that tries to use the given controller as frequently as possible while ensuring safety modulo fault. At a high level, it does so by using on-the-fly verification based on abstract interpretation to determine whether it is safe modulo fault to use the given controller; if so, it uses the given controller, but otherwise, it uses a backup controller.

Finally, we empirically evaluate our approach in a simulation, including both settings where the humans are simulated and settings where the human is controlled by an real person via keyboard inputs. We demonstrate that our MPS modulo fault algorithm enables the robot to avoid accidents both with real and simulated humans, even when combined with a naïve controller that altogether ignores the humans.

Figure 1 illustrates how our algorithm ensures safety while interacting with a human driver without being overly cautious. It assumes that the human will at least slightly brake to avoid an accident (left). If it still cannot guarantee safety, then it allows the human to go first (right).

II. Preliminaries

a) Human-robot system: We consider a system with a robot R and human H. Following prior work [5], we assume the human and robot act in alternation, which is reasonable as long as the time steps are sufficiently small. For a state x_t where it is the robot's turn to act, we have

$$x'_t = f_R(x_t, u_{R,t}) \qquad \text{and} \qquad x_{t+1} = f_H(x'_t, u_{H,t}),$$

where $f_R : \mathcal{X} \times \mathcal{U}_R \to \mathcal{X}$ is the robot dynamics, $f_H : \mathcal{X} \times \mathcal{U}_H \to \mathcal{X}$ is the human dynamics, $\mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the joint state space, $\mathcal{U}_R \subseteq \mathbb{R}^{n_{U,R}}$ are the robot actions, and $\mathcal{U}_H \subseteq \mathbb{R}^{n_{U,H}}$ are the human actions. Given an initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ where it is the robot's turn to act, along with two action sequences

$$\vec{u}_R = (u_{R,0}, u_{R,1}, ...) \in \mathcal{U}_R^\infty$$
$$\vec{u}_H = (u_{H,0}, u_{H,1}, ...) \in \mathcal{U}_H^\infty$$

for R and H, respectively, the *trajectory* from x_0 using $\vec{u}_R, \vec{u}_H$ is the sequence of states

$$\zeta_R(x_0, \vec{u}_R, \vec{u}_H) = (x_0, x'_0, x_1, ...) \in \mathcal{X}^\infty,$$

where $x'_t = f_R(x_t, u_{R,t})$ and $x_{t+1} = f_H(x'_t, u_{H,t})$. Similarly, given an initial state $x'_0 \in \mathcal{X}$ where it is the human's turn to act along with $\vec{u}_R \in \mathcal{U}_R^\infty$ and $\vec{u}_H \in \mathcal{U}_H^\infty$, the trajectory from x_0 using $\vec{u}_H, \vec{u}_R$ is the sequence of states

$$\zeta_H(x'_0, \vec{u}_H, \vec{u}_R) = (x'_0, x_0, x'_1, ...) \in \mathcal{X}^\infty,$$

where $x_t = f_H(x'_t, u_{H,t})$ and $x'_{t+1} = f_R(x_t, u_{R,t})$. Also, given robot and human policies $\pi_R : \mathcal{X} \to \mathcal{U}_R$ and $\pi_H : \mathcal{X} \to \mathcal{U}_H$, respectively, we define the trajectory

$$\zeta_R(x_0, \pi_R, \pi_H) = (x_0, x'_0, x_1, ...) \in \mathcal{X}^\infty$$

by $x'_t = f_R(x_t, \pi_R(x_t))$ and $x_{t+1} = f_H(x'_t, \pi_H(x'_t))$, and

$$\zeta_H(x_0, \pi_H, \pi_R) = (x'_0, x_0, x'_1, ...) \in \mathcal{X}^\infty$$

by $x_t = f_H(x'_t, \pi_H(x'_t))$ and $x'_{t+1} = f_R(x_t, \pi_R(x_t))$.

Given a given safe region $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$, our goal is to ensure the system stays in $\mathcal{X}_{\text{safe}}$ for the entire trajectory.

Definition II.1. A trajectory $\zeta = (x_0, x'_0, x_1, ...)$ (or $\zeta = (x'_0, x_0, x'_1, ...)$) is *safe* if $x_t, x'_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$.

b) Example: Consider an autonomous driving robot R interacting with a human driver H. The state $x \in \mathcal{X} = \mathbb{R}^8$ is a vector $x = (x_R, y_R, v_R, \theta_R, x_H, y_H, v_H, \theta_H)$ representing the positions $(x_R, y_R), (x_H, y_H)$, velocities v_R, v_H, and angles θ_R, θ_H of R and H, respectively. The actions $u \in \mathcal{U}_R = \mathcal{U}_H = \mathbb{R}^2$ are vectors $u = (\phi, a)$ representing the steering angle ϕ and acceleration a; we assume $|\phi| \le \phi_{\max}$ and $|a| \le a_{\max}$ are bounded. Given time step $\tau \in \mathbb{R}_{>0}$, the dynamics are

$$f_A(x, u_A) = x + g_A(x, u_A) \cdot \tau \qquad (\forall A \in \{R, H\})$$
$$g_R(x, u_R) = (v_R \cos\theta_R, v_R \sin\theta_R, a_R, v_R\phi_R, 0, 0, 0, 0)$$
$$g_H(x, u_H) = (0, 0, 0, 0, v_H \cos\theta_H, v_H \sin\theta_H, a_H, v_H\phi_H).$$

Safety means the robot and the human have not collided:

$$\mathcal{X}_{\text{safe}} = \{x \in \mathcal{X} \mid \|(x_R, y_R) - (x_H, y_H)\| \ge d_{\text{safe}}\},$$

for some constant $d_{\text{safe}} \in \mathbb{R}_{\ge 0}$.

[2]We do not assume the human will take evasive maneuvers such as swerving to avoid an accident (though can do so if it is safe). Also, we assume a reasonable amount of time for them to brake and come to a stop.

[3]The term "fault" is used simply to mean that safety ignores cases where the human does not act according to our model, not to blame an accident on the human driver. The controller designer is responsible for designing the backup actions to obtain a reasonable notion of safety.

III. Safety Modulo Fault

Ensuring safety in the presence of an adversarial human would be impossible or at least significantly degrade performance—e.g., to avoid an accident with an adversarial human driver, the robot would have to maintain a very large distance. Thus, to ensure safety, we must make assumptions about the behavior of the human. Ideally, we want to make the minimal possible assumptions about the behavior of the human while still accounting for all possible behaviors of a human acting in a responsible way. Then, we want to ensure safety for any human acting according to these assumptions.

The key challenge is devising a reasonable set of assumptions on the human. Intuitively, our assumptions are based on the idea that if the human can act in a safe way to avoid an accident, then they do so (Assumption III.1). We need to formalize what it means for the human to "be able to act in a safe way". We give the human great leeway in what safe actions they consider—for instance, we could assume the human always considers slowing down in *some* manner to ensure safety (Assumption III.3). Finally, just as we make assumptions about how the human acts, we expect the human to make assumptions about how the robot may act. We give the human great leeway in doing so—for instance, we could assume that the human always accounts for the possibility that the robot may take a safe action such as gradually braking to avoid an accident (Assumption III.2). We formalize our assumptions and safety notion below.

a) Human objective: Our definition of safety is based on a model of a human acting according to a maximin objective. In this objective, the "min" is the worst-case over a set of action sequences that the human anticipates the robot may take, and the "max" is over the human's own actions. That is, the human plans optimally according to their objective, while conservatively accounting for actions they anticipate the robot might take. In particular, suppose that at state x, the robot takes action u_R and transitions to state $x' = f_R(x, u_R)$; then, the human takes an action $\pi_H(x') = u^*_{H,0}$ such that

$$\vec{u}^*_H \in \arg\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(x', \vec{u}_H), \tag{1}$$

where the $\arg\max$ denotes the set of all optimal values, and

$$J_H(\vec{u}_H) = \min_{\vec{u}_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H, \vec{u}_R))$$

$$J_H((x'_0, x_0, x_1, ...)) = \sum_{t=0}^{\infty} \gamma^t r_H(x'_t, u_{H,t}, x_t),$$

where $\hat{\mathcal{U}}_R \subseteq \mathbb{R}^{n_{U,R}}$ is the set of actions the human anticipates the robot may take, $r_H : \mathcal{X} \times \mathcal{U}^H \times \mathcal{X} \to \mathbb{R} \cup \{-\infty\}$ is the human reward function, and $\gamma \in (0, 1)$ is a discount factor.

The key challenge for the robot to plan safely is that it does not know the human reward function r_H, the human action set $\mathcal{U}_H$ or the human-anticipated robot action set $\hat{\mathcal{U}}_R$. Assuming we know these values exactly is implausible. Instead, we assume access to minimal knowledge about each of these objects. Intuitively, our assumptions are:

- **Human reward function:** The human always prioritizes avoiding an unsafe state—in particular, their reward function r_H equals $-\infty$ for any unsafe state.
- **Human-anticipated robot actions:** We are given a conservative robot action u^0_R (e.g., braking) in $\hat{\mathcal{U}}_R$.
- **Human actions:** We are given a set of human actions $\mathcal{U}^0_H$ such that the human takes *some* action $u_H \in \mathcal{U}^0_H$ whenever they cannot ensure safety.

We formalize these assumptions in the next section.

b) Assumption on human objective: First, we assume that the human reward for reaching an unsafe state is $-\infty$.

Assumption III.1. For any $x', x \in \mathcal{X}$ and $u_H \in \mathcal{U}_H$, we have $r_H(x', u_H, x) = -\infty$ if $x' \notin \mathcal{X}_{\text{safe}}$ or $x \notin \mathcal{X}_{\text{safe}}$.

That is, the human driver always acts to avoid an accident. Other than Assumption III.1, r_H can be arbitrary.

With this assumption, there are two reasons accidents may happen: (i) there was a safe action sequence $\vec{u}_H \in \mathcal{U}_H^\infty$ that the human driver failed to take, or (ii) if the robot takes an action $u_R \notin \hat{\mathcal{U}}_R$ that the human driver failed to anticipate. Thus, we can always conservatively take $\hat{\mathcal{U}}_R$ to be smaller than it actually is. Conversely, we can always take $\mathcal{U}^0_H$ to be larger than it actually is. Thus, we make minimal assumptions about what actions are contained in $\hat{\mathcal{U}}_R$ and $\mathcal{U}_H$.

First, we make the following assumption on the set of actions $\vec{\mathcal{U}}_R$ that the human anticipates the robot may take:

Assumption III.2. We are given a *robot backup action* $u^0_R \in \mathcal{U}_R$ that is anticipated by the human—i.e., $u^0_R \in \hat{\mathcal{U}}_R$.

That is, the human always accounts for the possibility that the robot might take action u^0_R. For example, we might assume that u^0_R is gradually braking and coming to a stop.

Next, we make the following assumption about the human:

Assumption III.3. We are given a *human backup action set* $\mathcal{U}^0_H \subseteq \mathcal{U}_H$ such that if $\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(\vec{u}_H) = -\infty$, then the human takes an action $\pi_H(x') \in \mathcal{U}^0_H$.

That is, if the human is unable to guarantee safety (i.e., their objective value is $-\infty$), then they take *some* action in $\mathcal{U}^0_H$. For example, $\mathcal{U}^0_H$ may contain all actions where the human driver decelerates by at least some rate (so they can slow down more quickly or steer in any direction).

Remark III.4. Our approach can easily be extended to the case where u^0_R, $\mathcal{U}^0_H$, and $\hat{\mathcal{U}}_R$ are time varying.

Remark III.5. We can weaken (1) as follows. We only need to assume that the human chooses an action sequence $\vec{u}^*_H$ with a reward $> -\infty$. Then, Assumption III.1 say that the human chooses actions $\vec{u}^*_H$ that will avoid an accident assuming the robot takes actions in $\hat{\mathcal{U}}_R$. For instance, we can replace (1) with a probabilistic model [8] where the human samples action sequence $\vec{u}_H$ with probability $p(\vec{u}_H \mid x_0, u_{R,0}) \propto \exp(J_H(\vec{u}_H))$ and takes action $u_{H,0}$; then, Assumption III.1 says the human never considers trajectories that may reach an unsafe state.

Algorithm 1 Model predictive shielding modulo fault.

procedure $\pi_R(x)$
  $x' \leftarrow f_R(x, \hat{\pi}_R(x))$
  return if IsRec(x') **then** $\hat{\pi}_R(x)$ **else** u_R^0 **end if**
end procedure
procedure IsRec(x')
  $X'_0 \leftarrow \{x'\}$
  for $t \in \{0, ..., k-1\}$ **do**
    if $X'_t \not\subseteq \mathcal{X}_{\text{safe}}$ **then return false end if**
    $U_{R,t} \leftarrow$ **if** $t = 0$ **then** $\{u_R\}$ **else** $\{u_R^0\}$ **end if**
    $U_{H,t} \leftarrow \mathcal{U}_H^0$
    $X'_{t+1} \leftarrow F(X_t, U_{R,t}, U_{H,t})$
  end for
  return if $X_k \subseteq \mathcal{X}_{\text{eq}}$ **then true else false end if**
end procedure

c) Problem formulation: Our goal is to ensure that the robot acts in a way that ensures safety for an infinite horizon for any human that satisfies our assumptions.

Definition III.6. A robot policy $\pi_R : \mathcal{X} \to \mathcal{U}_R$ is *safe modulo fault* for initial states $\mathcal{X}_0 \subseteq \mathcal{X}$ if for any human policy π_H satisfying Assumptions III.1, III.2, & III.3, and any $x_0 \in \mathcal{X}_0$, the trajectory $\zeta_R(x_0, \pi_R, \pi_H) \in \mathcal{X}^\infty$ is safe.

That is, π_R that ensures safety as long as the human acts in a way that satisfies our assumptions. Our goal is to design a policy π_R that is safe modulo fault. We use the term "fault" to indicate that this property only guarantees safety with respect to humans that satisfy our assumptions; we are not assigning blame to the human in case of an accident. The controller designer is responsible for ensuring the assumptions are satisfied by reasonable human drivers.

Finally, we cannot guarantee safety starting from an arbitrary state x_0. For instance, if the robot is about to crash into a wall, no action can ensure safety. We assume that the initial states $\mathcal{X}_0$ are ones where we can guarantee safety.

Definition III.7. A *safe equilibrium state* $x \in \mathcal{X}$ satisfies (i) $x \in \mathcal{X}_{\text{safe}}$, and (ii) $x = f(x, u_R^0, u_H)$ for all $u_H \in \mathcal{U}_H^0$.

We denote the set of safe equilibrium states by $\mathcal{X}_{\text{eq}}$. At a state $x \in \mathcal{X}_{\text{eq}}$, the robot and human can together ensure safety for an infinite horizon by taking actions u_R^0 and u_H for any $u_H \in \mathcal{U}_H^0$. In our driving example, $\mathcal{X}_{\text{eq}}$ contains states where both agents are at rest (i.e., their velocity is zero).

Assumption III.8. We have $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{eq}}$.

In other words, the system starts at a safe equilibrium state where we can ensure safety for an infinite horizon.

IV. Model Predictive Shielding Modulo Fault

We describe our algorithm for constructing a robot controller $\pi_R : \mathcal{X} \to \mathcal{U}_R$ that is safe modulo fault. Our approach is based on *shielding* [14]—it takes as input an arbitrary controller $\hat{\pi}_R : \mathcal{X} \to \mathcal{U}_R$ and modifies it to construct π_R. Intuitively, π_R overrides $\hat{\pi}_R$ when it cannot ensure it is safe.

The challenge is checking whether it is safe to use $\hat{\pi}_R$. Model predictive shielding (MPS) is an approach to shielding that checks safety online based on the following [15], [16]. These approaches are designed to handle deterministic or stochastic environments where the model is known, whereas we do not have any such model of the human driver. We show how to extend these algorithms to our setting. The idea is to maintain the invariant that the current state is *recoverable*—intuitively, that there is some sequence of actions each agent can take that safely brings the system to a stop. In particular:

Definition IV.1. Given hyperparameter $k \in \mathbb{N}$, a state $x' \in \mathcal{X}$ is *recoverable* (denoted $x' \in \mathcal{X}_{\text{rec}}$) if for any $\vec{u}_H \in (\mathcal{U}_H^0)^\infty$ and for $\vec{u}_R = (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty$, the trajectory $\zeta_H(x', \vec{u}_H, \vec{u}_R) = (x'_0, x_0, x'_1, ...)$ satisfies (i) $x'_t, x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$.

That is, x' is recoverable if the human and robot can ensure safety by using their respective backup actions $\mathcal{U}_H^0$ and u_R^0. By definition, if $x_k \in \mathcal{X}_{\text{eq}}$, then $x_k = x'_{k+1} = x_{k+1} = ...$ Since $x_k \in \mathcal{X}_{\text{safe}}$, it follows that $x'_t, x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$.

Now, our MPS modulo fault algorithm for computing π_R is shown in Algorithm 1. IsRec checks whether $x' = f_R(x, \hat{\pi}_R(x))$ is recoverable. If so, π_R returns $\hat{\pi}_R(x)$; otherwise, it returns u_R^0. The key novelty is how IsRec checks recoverability. Existing approaches [15], [16] do so by simulating the model of the environment. In contrast, we need to guarantee safety with respect to *all* $\vec{u}_H \in (\mathcal{U}_H^0)^\infty$. To do so, IsRec *conservatively overapproximates* recoverability—i.e., if it says that x' is recoverable, then it must be recoverable, but it may say x' is not recoverable even if it is.

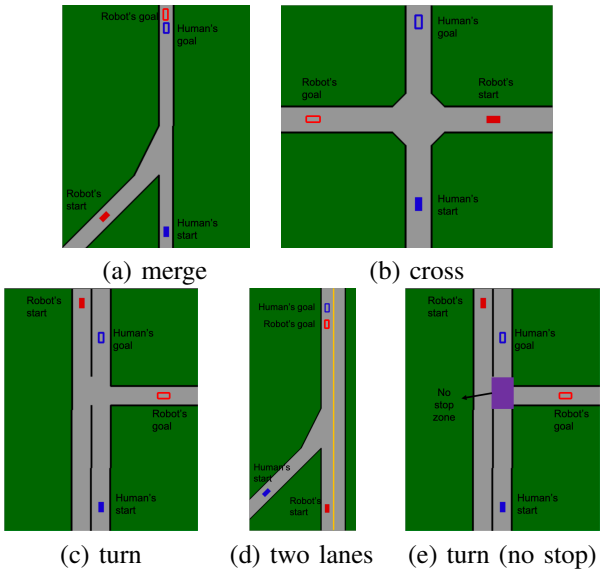
To do so, IsRec overapproximates the reachable set of states after t steps as a subset $X_t \subseteq \mathcal{X}$. More precisely, it assumes given a *dynamics overapproximation* $F : 2^{\mathcal{X}} \times 2^{\mathcal{U}_R} \times 2^{\mathcal{U}_H} \to 2^{\mathcal{X}}$ mapping sets of states $X \subseteq \mathcal{X}$, sets of robot actions $U_R \subseteq \mathcal{U}_R$, and sets human action $U_H \subseteq \mathcal{U}_H$ to sets of states $F(X, U_R, U_H) \subseteq 2^{\mathcal{X}}$, which satisfies

$$f(x, u_R, u_H) \in F(X, U_R, U_H) \tag{2}$$

for all $x \in X$, $u_R \in U_R$, and $u_H \in U_H$—i.e., $F(X, U_R, U_H)$ contains *at least* the states reachable from $x \in X$ by taking actions $u_R \in U_R$ and $u_H \in U_H$. Intuitively, computing F in a way that this property holds with equality may be computationally intractable, but there exist tractable overapproximations—e.g., based on polytopes [17], [18] or ellipsoids [19], [20]. This approach fits in the general framework of *abstract interpretation* [12]; here, the sets X, U_R, and U_H are represented implicitly rather than explicitly for computational tractability. We describe the overapproximation we use for our autonomous driving example in Section I.

Finally, IsRec checks whether (i) safety holds for every state $x_t \in X_t$ (i.e., $X_t \subseteq \mathcal{X}_{\text{safe}}$), and (ii) every state $x_k \in X_k$ is a safe equilibrium state (i.e., $X_k \subseteq \mathcal{X}_{\text{eq}}$). If both these properties hold, then x is guaranteed to be recoverable. We have the following guarantee (see Appendix II for a proof):

Theorem IV.2. *Assuming (2) holds, then our policy π_R is safe modulo fault (i.e., it satisfies Definition III.6).*



(a) merge (b) cross

(c) turn (d) two lanes (e) turn (no stop)

Fig. 2. Visualizations of tasks along with the initial positions and goals of the robot and human. The red (resp., blue) box is the robot (resp., human).

V. Evaluation

We have implemented our approach in a simulation for three robotics tasks. For the robot, we consider an aggressive controller with and without the shield as well as a cross entropy method controller (CEM) that is designed to avoid humans. For the human, we use both simulated humans based on a social forces model of pedestrians [21], as well as real humans interacting with the simulation via keyboard inputs.

Our goal is to understand how our approach can ensure safety in aggressive driving scenarios. Thus, we focus on settings where the human (simulated or real) and robot must compete to reach their goals. We tune the parameters of our MPS modulo fault algorithm (i.e., the robot backup action u_R^0 and the human backup action set $\mathcal{U}_H^0$) to be as aggressive as possible while still ensuring safety on the simulated humans. Furthermore, for our experiments with real-world humans, we strongly encourage them to try and reach their goal before the robot, albeit keeping safety as the top priority. Then, our results are designed to answer the following:

- Can MPS modulo fault can be used to ensure safety with real and simulated humans?
- Can MPS modulo fault outperform a handcrafted MPC based on CEM in terms of performance?

A. Experimental Setup

a) Robotics tasks: We consider three non-cooperative robotics tasks (depicted in Figure 2). In the first task ("merge"), there are two lanes that merge—i.e., the robot is coming in from one lane and the humans from another; the robot and human goals are to navigate the merge and reach their goal. The second task ("cross") has both the human and the robot moving towards an intersection from different directions—i.e., the robot is moving horizontally and the human is moving vertically; the robot and human goals are to get to their goal on the other side of the intersection. The third task ("turn") is an unprotected left turn—i.e., the humans are driving without turning and the robot needs to make a left turn that crosses the human path.

b) Safety property: We assume the robot and human are each a rectangle; then, the safety property is that the the robot and human rectangles should not intersect.

c) Robot dynamics: The robot dynamics are the ones in our running example—i.e., its state is (x, y, v, θ), where (x, y) is position, v is velocity, and θ is orientation, and its actions are (a, ϕ), where a is acceleration and ϕ is steering angle. We assume $|a| \leq a_{\max}$, $|\phi| \leq \phi_{\max}$, and $0 \leq v \leq v_{\max}$.

d) Simulated humans: For simulated humans, we use the social force model [21], which includes potential forces that cause each human to avoid the robot, other humans, and walls, while trying to reach their goal.

e) Real humans: We also considered real human users interacting with the simulation via keyboard. They control the human using the up/down arrows to control acceleration and the left/right arrows to control steering angle. We asked the human users to prioritize safety first, but to drive aggressively to try and reach their goal before the robot.

f) Controllers: We consider three controllers for the robot: (i) an aggressive controller, (ii) a handcrafted MPC based on the cross-entropy method (CEM) designed to ensure safety without a shield, and (iii) our MPS modulo fault algorithm used in conjunction with the aggressive controller.

The first controller is an "aggressive controller" that ignores the humans and moves directly towards the goal as quickly as possible. If nonlinear trajectories are required, we manually specify a sequence of subgoals; once the robot reaches its current subgoal, it continues to the next one.

The second is a model-predictive controller (MPC) that aims to avoid colliding with the human. We use a planning algorithm based on the cross-entropy method (CEM). Then, it chooses the action that attempts to optimize its objective over the planning horizon. We use a handcrafted objective that provides a positive reward for progressing towards its goal and a large negative penalty for collisions. To predict collisions, it forecasts the behavior of the human over the planning horizon by extrapolating their position based on their current velocity (i.e., constant velocity assumption). Finally, for the goal-reaching portion of the objective, we use subgoals the same way we do for the aggressive controller.

The third controller is our MPS modulo fault algorithm used with the aggressive controller. The robot backup action is $u_R^0 = (0, -1)$ where $\phi = 0$ is the steering angle and $a = -1$ is the acceleration. The human backup action set is

$$\mathcal{U}_H^0 = \left\{ (\phi, a) \;\middle|\; \phi \in \left[-\frac{\pi}{10}, \frac{\pi}{10}\right],\ a \in \left[-1, -\frac{1}{2}\right] \right\}.$$

That is, the human predicts that the robot may gradually brake without changing direction, and the human considers braking gradually (or hard) while steering up to some angle.

B. Experimental Results

We describe our experimental results. For simulated humans, all results shown are averaged over 100 simulations. For real humans, the results are based on 18 users.

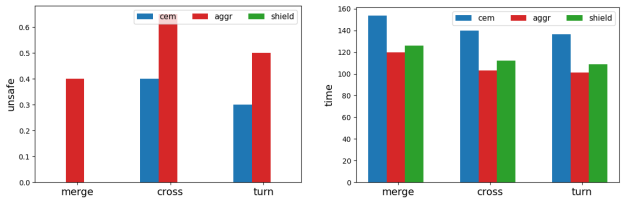


Fig. 3. Results with simulated humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded controller (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

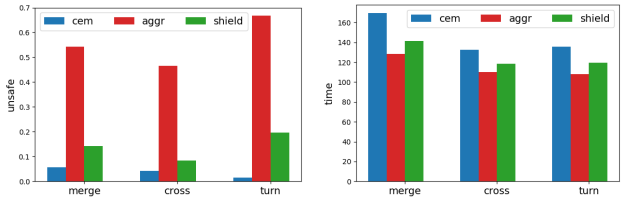


Fig. 4. Results with real humans, for the aggressive controller (red), the CEM MPC (blue), and our shielded controller (green). Left: Fraction of unsafe runs. Right: Time the robot takes to reach its goal in seconds.

a) MPS modulo fault ensures safety for simulated humans: In Figure 3, we show both the fraction of unsafe runs (left), and the time taken by the robot to reach the goal (right), including the aggressive controller (red), the MPC based on CEM (blue), and our shielded aggressive controller (green). As can be seen, for the aggressive controller, the rate of unsafe runs is very high since the robot ignores the human to get to its goal. Next, for the MPC based on CEM, the rate of unsafety is lower but still not zero. However, the CEM policy takes significantly longer to reach its goal compared to the aggressive policy. Finally, our shielded aggressive controller is always safe, yet only takes a small amount of time longer to reach its goal compared to the aggressive policy; in particular, it is significantly faster than the MPC. These comparisons demonstrate that our approach greatly improves safety without significantly reducing time to goal.

b) MPS modulo fault ensures safety for real humans: Next, we had real human users interact with our simulated robot via keyboard input. we show both the fraction of unsafe runs (left), and the time taken by the robot to reach the goal (right), including the aggressive controller (red), the MPC based on CEM (blue), and our shielded aggressive controller (green). As can be seen, for the aggressive controller, the robot gets to its goal the fastest, but is frequently unsafe. The MPC based on CEM is significantly safer; in this case, it is somewhat safer than our shielded aggressive controller. On the other hand, our shield controller reaches its goal significantly faster than the MPC. As described above, we set the shield parameters aggressively based on the simulated humans to ensure it could reach its goal; in practice, we could ensure safety by setting these parameters more conservatively and by tuning them to the real human driver data.

c) Alternative robot backup actions: A key feature of our approach is that we can flexibly design the robot backup action to ensure safety. To demonstrate this flexibility, we

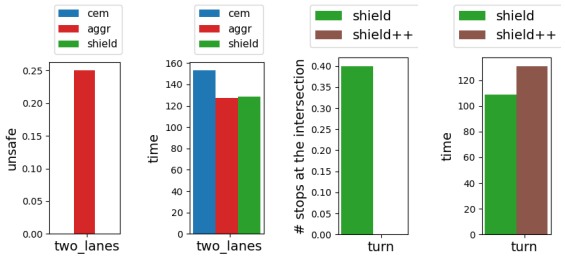


Fig. 5. Results for alternative robot backup actions with simulated humans. For the "pull over" backup action, we show the fraction of unsafe runs (leftmost) and the time the robot takes to reach its goal in seconds (second from the left), for the aggressive controller (red), the CEM MPC (blue), and our shielded aggressive controller (green). For the "no-stop zone" backup action, we show the number of stops in the intersection (second from the right) and the time the robot takes to reach its goal in seconds (rightmost), for the original (green, "shield") and the new (brown, "shield++") shielded controllers; both controllers are always safe.

design an alternative backup action that pulls the robot over to the shoulder of a highway. This backup policy is time varying—i.e., the robot steering depends on the current state. We test this backup policy with simulated humans on the task in Figure 2 (d), where there are two lanes on the highway and an on-ramp that merges onto the highway. The human is on the on-ramp and the robot is on the highway. To avoid collisions, the robot can pull over to the right-most lane. Figure 5 shows the fraction of the unsafe runs (leftmost), and the time the robot takes to reach its goal (second from left) for all three controllers—aggressive (red), the MPC based on CEM (blue), and our shielded aggressive controller with the pull over backup policy (green). Our shielded controller is always safe and is significantly faster than the MPC.

We also design a robot backup action that avoids stopping in the middle of an intersection and blocking it, which is often illegal. To this end, we modify the turn task to include a no-stop zone (shown in Figure 2 (e)) where the robot is prohibited from stopping. In this zone, the robot backup action does not come to a stop immediately; instead, it drives through the zone and only brakes once it has fully cleared the intersection. The results for this experiment using simulated humans are shown in Figure 5 (right). We compare the original shielded controller ("shield") that may stop in the intersection with the new one that adheres to the no-stop zone in Figure 2 (e) ("shield++"). In this case, both the controllers were always safe; instead, we show the fraction of runs where the robot stops in the intersection (second from the right), and the time the robot takes to reach its goal (rightmost). The new shielded controller takes slightly longer to reach the goal, but never stops in the intersection.

VI. Conclusion

We have proposed an approach for ensuring safety in human-interactive robotics systems. We define a notion of safety that models human behavior by specifying their backup behaviors, and propose our MPS modulo fault algorithm for ensuring safety with respect to this model. We have validate our approach on both real and simulated humans.

REFERENCES

[1] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, "Toward seamless human-robot handovers," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 112–132, 2013.

[2] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 301–308.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[4] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.

[5] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016.

[6] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[7] K. Eder, C. Harper, and U. Leonards, "Towards the safety of human-in-the-loop robotics: Challenges and opportunities for safety assurance of robotic co-workers'," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 660–665.

[8] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[9] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," in *RSS*, 2018.

[10] D. Sadigh, S. S. Sastry, S. A. Seshia, and U. Berkeley, "Verifying robustness of human-aware autonomous cars," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 131–138, 2019.

[11] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[12] P. Cousot, "Abstract interpretation," in *In POPL*. Citeseer, 1977.

[13] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1424–1431.

[14] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 7130–7135.

[16] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *ICRA*, 2019.

[17] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[18] S. Sadraddini and R. Tedrake, "Linear encodings for polytope containment problems," *arXiv preprint arXiv:1903.05214*, 2019.

[19] L. Asselborn, D. Gross, and O. Stursberg, "Control of uncertain nonlinear systems using ellipsoidal reachability calculus," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 50–55, 2013.

[20] T. F. Filippova, "Ellipsoidal estimates of reachable sets for control systems with nonlinear terms," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15 355–15 360, 2017.

[21] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

APPENDIX I
OVERAPPROXIMATING THE DYNAMICS

We describe our approach for overapproximating the multi-agent dynamics f in our autonomous car example. In particular, we describe representations U_R, U_H, and X of sets of robot actions, human actions, and states, respectively, along with a function $F : (U_R, U_H, X) \mapsto X'$ that satisfies (2). First, we consider a representation of U_R of the form

$$U_R = \{(-a_R, 0)\}, \tag{3}$$

where $a_R \in \mathbb{R}_{>0}$ is the rate at which the robot decelerates; we assume it corresponds to the robot gradually braking to give a trailing human driver sufficient time to respond. Also, we consider a representation of U_H of the form

$$U_H = (-\infty, -a_{\min}] \times [-\phi_{\max}, \phi_{\max}] \tag{4}$$

where $a_{\min} \in \mathbb{R}_{>0}$ is the minimum deceleration rate—i.e., the minimum rate at which we expect the human driver to brake to avoid an accident. Then, both U_R and U_H have form

$$U_A = [a_{A,\min}, a_{A,\max}] \times [-\phi_{\max}, \phi_{\max}]$$

for $a_{A,\min}, a_{A,\max} \in \mathbb{R}$ and $\phi_{\max} \in \mathbb{R}_{>0}$, and where $A \in \{R, H\}$. Next, we consider X of the form

$$\begin{aligned} X &= X_R \times X_H \\ X_A &= [x_{A,\min}, x_{A,\max}] \times [y_{A,\min}, y_{A,\max}] \\ &\quad \times [v_{A,\min}, v_{A,\max}] \times [-\theta_{A,\max}, \theta_{A,\max}], \end{aligned} \tag{5}$$

where $A \in \{R, H\}$. Then, we use

$$\begin{aligned} F(X, U_R, U_H) &= F_R(X, U_R, U_H) \times F_H(X, U_R, U_H) \\ F_A(X, U_R, U_H) &= X_A^x \times X_A^y \times X_A^v \times X_A^\theta, \end{aligned}$$

where

$$\begin{aligned} X_A^x &= [x_{A,\min} - v_{A,\max}, x_{A,\max} + v_{A,\max}] \\ X_A^y &= [y_{A,\min} - v_{A,\max}, y_{A,\max} + v_{A,\max}] \\ X_A^v &= [\max\{v_{A,\min} + a_{A,\min}, 0\}, \max\{v_{A,\max} + a_{A,\max}, 0\}] \\ X_A^\theta &= [-\theta_{A,\max} - v_{A,\max}\phi_{A,\max}, \theta_{A,\max} + v_{A,\max}\phi_{A,\max}] \end{aligned}$$

where again $A \in \{R, H\}$. It is easy to see that this choice of F satisfies (2). Finally, we note that (5), (3), and (4) are satisfied for all time steps. In particular, X_0 satisfies (5), $U_{R,t}$ satisfies (3) for all time steps t, and $U_{H,t}$ satisfies (4) for all time steps t according to our choices in Section V. Thus, $X_{t+1} = F(X_t, U_R, U_H)$ satisfies (5) by induction.

Remark I.1. Formally, the abstract domain of U_H is $a \in \mathbb{R} \cup \{-\infty, \infty\}$ (with the lattice structure $a \sqsubseteq a'$ if and only if $a \leq a'$, $\top = \infty$, and $\bot = -\infty$), and its abstraction and concretization functions are $\alpha : U_H \mapsto \sup_{a \in \mathbb{R}} U_H$ and $\gamma : a \mapsto (-\infty, -a]$, respectively. The setup for U_R is similar. Then, (2) says that F is an abstract transformer for f.

APPENDIX II
PROOF OF THEOREM IV.2

First, we have the following guarantee on F.

Lemma II.1. *Given initial state $x_0 \in \mathcal{X}$, and robot and human action set sequences $(U_{R,0}, U_{R,1}, ...) \in (2^{\mathcal{U}_R})^\infty$ and $(U_{H,0}, U_{H,1}, ...) \in (2^{\mathcal{U}_H})^\infty$, respectively, let the state set sequence $(X_0, X_1, ...) \in (2^{\mathcal{X}})^\infty$ be defined by $X_0 = \{x_0\}$ and $X_{t+1} = F(X_t, U_{R,t}, U_{H,t})$ for $t \in \mathbb{N}$. Then, for any $\vec{u}_R \in \mathcal{U}_R^\infty$ and $\vec{u}_H \in \mathcal{U}_H^\infty$ such that $u_{R,t} \in U_{R,t}$ and $u_{H,t} \in U_{H,t}$ for all $t \in \mathbb{N}$, the trajectory $(x_0, x_1, ...) \in \mathcal{X}^\infty$ from x_0 using $\vec{u}_R, \vec{u}_H$ satisfies $x_t \in X_t$ for all $t \in \mathbb{N}$.*

Proof: We prove by induction on t. The base case $t = 0$ follows since $x_0 = x \in X_0$. For the inductive case, note that

$$x_{t+1} = f(x_t, u_{R,t}, u_{H,t}) \in F(X_t, U_{R,t}, U_{H,t}) = X_{t+1},$$

since $x_t \in X_t$ (by induction), since $u_{R,t} \in U_{R,t}$ and $u_{H,t} \in U_{H,t}$ (by assumption), and by (2). □

Next, we have the following guarantee on IsRec.

Lemma II.2. *If* IsRec(x') *= true, then* $(x') \in \mathcal{X}_{rec}$.

Proof: Suppose IsRec(x') returns true. We need to show that for any $\vec{u}_H = \mathcal{U}_H^\infty$ and for $\vec{u}_R = (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty$, the trajectory $\zeta_H(x', \vec{u}_H, \vec{u}R) = (x'_0, x_0, x'_1, ...) \in \mathcal{X}^\infty$ satisfies (i) $x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$. For (i), $x_t \in X_t \subseteq \mathcal{X}_{\text{safe}}$, where the first inclusion follows by Lemma II.1 and the second follows by the definition of IsRec. For (ii), as in the previous case, $x_k \in X_k \subseteq \mathcal{X}_{\text{eq}}$. □

Next, we prove the existence of a safe invariant set—i.e., one preserved by π_R and any π_H satisfying our assumptions.

Definition II.3. A state $x' \in \mathcal{X}$ where it is the human's turn to act is *invariant* (denoted $x' \in \mathcal{X}_{\text{inv}}$) if either (i) $J_H(x') > -\infty$, or (ii) $x'_+ \in \mathcal{X}_{\text{rec}}$, where we have $x = f_H(x', \pi_H(x'))$ and $x'_+ = f_R(x, \pi_R(x))$.

That is, either the human currently has objective value $> -\infty$, or on their next step, the human will be in a recoverable state. Next, we show that $\mathcal{X}_{\text{inv}}$ is an invariant set. We begin with three technical lemmas. First, we show that if the current human state x' (i.e., it is the human's turn to act) is recoverable and the human uses one of their backup actions $u_H \in \mathcal{U}_H^0$, then the next human state x'_+ is also recoverable.

Lemma II.4. *If $x' \in \mathcal{X}_{rec}$ and $u_H \in \mathcal{U}_H^0$, then $x \in \mathcal{X}_{safe}$ and $x'_+ \in \mathcal{X}_{rec}$, where $x = f_H(x', u_H)$ and $x'_+ = f_R(x, \pi_R(x))$.*

Proof: To show $x'_+ \in \mathcal{X}_{\text{rec}}$, we need to show that for any

$$\begin{aligned} \vec{u}_H &= (u_{H,0}, u_{H,1}, ...) \in (\mathcal{U}_H^0)^\infty \\ \vec{u}_R &= (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty, \end{aligned}$$

the trajectory $\zeta_H(x'_+, \vec{u}_H, \vec{u}_R) = (x'_0, x_0, x'_1, ...)$ satisfies (i) $x'_t, x_t \in \mathcal{X}_{\text{safe}}$ for $t \in \{0, ..., k\}$, and (ii) $x_k \in \mathcal{X}_{\text{eq}}$. Let

$$\begin{aligned} \vec{\tilde{u}}_H &= (u_H, u_{H,0}, u_{H,1}, ...) \in (\mathcal{U}_H^0)^\infty \\ \vec{\tilde{u}}_R &= (u_R^0, u_R^0, ...) \in \mathcal{U}_R^\infty, \end{aligned}$$

where the first line follows since by assumption, we have $u_H \in \mathcal{U}_H^0$. Also by assumption, we have $x' \in \mathcal{X}_{\text{rec}}$, so by definition or recoverability, the trajectory $\zeta_H(x', \vec{\tilde{u}}_H, \vec{\tilde{u}}_R) = (\tilde{x}'_0, \tilde{x}_0, \tilde{x}'_1, ...)$ satisfies $\tilde{x}'_t, \tilde{x}_t \in \mathcal{X}_{\text{safe}}$ for $t \in \{0, ..., k\}$ and $\tilde{x}_k \in \mathcal{X}_{\text{eq}}$. By definition of equilibrium, $\tilde{x}'_{k+1} = f_H(\tilde{x}_k, \tilde{u}_{H,k}) = \tilde{x}_k$ (since $\tilde{u}_{H,k} \in \mathcal{U}_H^0$) and $\tilde{x}_{k+1} = f_R(\tilde{x}'_{k+1}, \tilde{u}_{R,k+1}) = \tilde{x}'_{k+1} = \tilde{x}_k$. Thus, we have $\tilde{x}'_{k+1}, \tilde{x}_{k+1} \in \mathcal{X}_{\text{safe}}$ and $\tilde{x}_{k+1} \in \mathcal{X}_{\text{eq}}$. Finally, note that $x'_t = \tilde{x}'_{t+1}$ and $x_t = \tilde{x}_{t+1}$. By this fact and the above, both conditions (i) and (ii) holds, so it follows that $x'_+ \in \mathcal{X}_{\text{rec}}$. Finally, to see $x \in \mathcal{X}_{\text{safe}}$, note that $x = \tilde{x}_0 \in \mathcal{X}_{\text{safe}}$. □

Next, we show that if our algorithm uses $\hat{\pi}$ at state x, then the next human state x' is recoverable

Lemma II.5. *If $\pi_R(x) \neq u_R^0$, then $f_R(x, \pi_R(x)) \in \mathcal{X}_{rec}$.*

Proof: By definition of π_R, if $\pi_R(x) \neq u_R^0$, then we have $\textsc{IsRec}(f_R(x, \hat{\pi}_R(x))) = \text{true}$. Thus, by Lemma II.2, $f_R(x, \hat{\pi}_R(x)) \in \mathcal{X}_{\text{rec}}$, as claimed. □

Third, letting $J_H(x') = \max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(x', \vec{u}_H)$ be the the human objective value at human state x', we show that if $J_H(x') > -\infty$ and the robot uses its backup action u_R^0, then we also have $J_H(x'_+) > -\infty$ at the next human state x'_+.

Lemma II.6. *If $J_H(x') > -\infty$, then $J(x'_+) > -\infty$, where $x = f_H(x', \pi_H(x'))$ and $x'_+ = f_R(x, u_R^0)$.*

Proof: If $J_H(x') > -\infty$, then $J_H(x', \vec{u}_H^*) > -\infty$, where $\vec{u}_H^* \in \arg\max_{\vec{u}_H \in \mathcal{U}_H^\infty} J_H(x', \vec{u}_H)$ is the optimal action sequence chosen by π_H—i.e., $\pi_H(x') = u_{H,0}^*$. Then, we have

$$\begin{aligned}
-\infty < J_H(x', \vec{u}_H^*) &= \min_{\vec{u}_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H^*, \vec{u}_R) \\
&\leq \min_{\vec{u}_R \in \{u_R^0\} \times \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H^*, \vec{u}_R)) \\
&= r_H(x', u_{H,0}^*, x) + \gamma \min_{\vec{u}'_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x'_+, \vec{\tilde{u}}_H, \vec{u}'_R)) \\
&= r_H(x', u_{H,0}^*, x) + \gamma J_H(x'_+, \vec{\tilde{u}}_H), \qquad (6)
\end{aligned}$$

where $\vec{\tilde{u}}_H = (u_{H,1}^*, u_{H,2}^*, ...)$. The first line holds by assumption, and the second and last lines hold by definition. The third line follows since by Assumption III.2, we have $u_R^0 \in \hat{\mathcal{U}}_R^0$, so $\{u_R^0\} \times \hat{\mathcal{U}}_R^\infty \subseteq \hat{\mathcal{U}}_R^\infty$. To see why the fourth line follows, note that for any $\vec{u}_R \in \{u_R^0\} \times \hat{\mathcal{U}}_R^\infty$, we have[4]

$$\begin{aligned}
\zeta_H(x', \vec{u}_H, \vec{u}_R) &= (x'_0, x_0) \circ (x'_1, x_1, x'_2, ...) \\
&= (x'_0, x_0) \circ \zeta_H(x'_+, \vec{\tilde{u}}_H, \vec{\tilde{u}}_R),
\end{aligned}$$

where the second line follows since $\vec{u}_R \in \{u_R^0\} \times \hat{\mathcal{U}}_R^\infty$ implies that $u_{R,0} = u_R^0$, so $x_0 = f_H(x'_0, u_{H,0}^*) = f_H(x', \pi_H(x')) = x$ and $x'_1 = f_R(x_0, u_R^0) = f_R(x, u_R^0) = x'_+$, and since we have $\vec{u}_H = (u_{H,0}^*) \circ \vec{\tilde{u}}_H^*$ and $\vec{u}_R = (u_R^0) \circ \vec{\tilde{u}}_R$, where $\vec{\tilde{u}}_R = (u_{R,1}, u_{R,2}, ...)$. As a consequence, we have

$$\begin{aligned}
&J_H(\zeta_H(x', \vec{u}_H, \vec{u}_R)) \\
&= \sum_{t=0}^{\infty} \gamma^t r_H(x'_t, u_{H,0}^*, x_t) \\
&= r_H(x', u_{H,0}^*, x) + \gamma \sum_{t=0}^{\infty} \gamma^t r_H(x'_{t+1}, u_{H,t+1}^*, x_{t+1}) \\
&= r_H(x', u_{H,0}^*, x) + \gamma J_H(\zeta_H(x'_+, \vec{\tilde{u}}_H, \vec{\tilde{u}}_R).
\end{aligned}$$

Then, the fourth line above holds since for any $\vec{\tilde{u}}_R \in \hat{\mathcal{U}}_R^0$, we have $(u_R^0) \circ \vec{\tilde{u}}_R \in \{u_R^0\} \times \hat{\mathcal{U}}_R^0$, so

$$\begin{aligned}
&\min_{\vec{u}_R \in \{u_R^0\} \times \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x', \vec{u}_H, \vec{u}_R)) \\
&= r_H(x', u_{H,0}^*, x) + \gamma \min_{\vec{\tilde{u}}_R \in \hat{\mathcal{U}}_R^\infty} J_H(\zeta_H(x'_+, \vec{\tilde{u}}_H, \vec{\tilde{u}}_R).
\end{aligned}$$

Now, we prove the lemma. Since $r_H(x', u_{H,0}^*, x) < \infty$ and $\gamma > 0$, by (6), we have $J_H(x'_+, \vec{\tilde{u}}_H) > -\infty$. Thus, we have

$$J_H(x'_+) = \max_{\vec{u}'_H \in \mathcal{U}_H^\infty} J_H(x'_+, \vec{u}'_H) \geq J_H(x'_+, \vec{\tilde{u}}_H) > -\infty,$$

as claimed. □

Next, we show that $\mathcal{X}_{\text{inv}}$ is an invariant set.

Lemma II.7. *Suppose π_R is as in Algorithm 1 and π_H satisfies Assumptions III.1, III.2, & III.3. If $x' \in \mathcal{X}_{inv}$, then $x'_+ \in \mathcal{X}_{inv}$ and $x'_+, x_+ \in \mathcal{X}_{safe}$, where $x = f_H(x', u_H)$, $x'_+ = f_R(x, \pi_R(x))$, and $x_+ = f_H(x'_+, \pi_H(x'_+))$.*

Proof: Since $x' \in \mathcal{X}_{\text{inv}}$, either $x'_+ \in \mathcal{X}_{\text{rec}}$ or $J(x') > -\infty$. First, suppose that $x'_+ \in \mathcal{X}_{\text{rec}}$. Let $x'_{++} = f_R(x_+, \pi_R(x_+))$. If $\pi_H(x'_+) \in \mathcal{U}_H^0$, then by Lemma II.4, we have $x'_{++} \in \mathcal{X}_{\text{rec}}$, so $x'_+ \in \mathcal{X}_{\text{inv}}$. Also, in this case, $x'_+ \in \mathcal{X}_{\text{rec}} \subseteq \mathcal{X}_{\text{safe}}$, and by Lemma II.4, $x_+ \in \mathcal{X}_{\text{safe}}$. Otherwise, if $\pi_H(x'_+) \notin \mathcal{U}_H^0$, then by Assumption III.3, we have $J_H(x'_+) > -\infty$, so $x'_+ \in \mathcal{X}_{\text{inv}}$. Since $J_H(x'_+) > -\infty$, we also have $r_H(x'_+, \pi_H(x'_+), x_+) > -\infty$, so by Assumption III.1, we have $x'_+, x_+ \in \mathcal{X}_{\text{safe}}$. Thus, the claim follows for the case $x'_+ \in \mathcal{X}_{\text{rec}}$.

Next, suppose that $J(x') > -\infty$. Then, we must have $\pi_R(x) = u_R^0$—otherwise, by Lemma II.5, we would have $x'_+ = f_R(x, \pi_R(x)) \in \mathcal{X}_{\text{rec}}$. Thus, by Lemma II.6, we have $J(x'_+) > -\infty$, so $x'_+ \in \mathcal{X}_{\text{inv}}$. As before, $J_H(x'_+) > -\infty$ also implies that $x'_+, x_+ \in \mathcal{X}_{\text{safe}}$. Thus, the claim follows. □

Finally, we prove Theorem IV.2.

Proof: We can equivalently consider the setting where the human starts at state $x'_0 = x$ and takes any action $\pi_H(x'_0) \in \mathcal{U}_H^0$; in particular, since $x \in \mathcal{X}_{\text{eq}}$, we have $x = f_H(x'_0, \pi_H(x'_0)) = x'_0$. Now, let $\zeta_H(x'_0, \pi_H, \pi_R) = (x'_0, x_0, x'_1, ...)$, where $x_0 = x$. Then, note that $x'_1 \in \mathcal{X}_{\text{rec}}$, since either $\pi_R(x) \neq u_R^0$, in which case $x'_1 = f_R(x_0, \pi_R(x_0)) \in \mathcal{X}_{\text{rec}}$ by Lemma II.5, or $\pi_R(x) = u_R^0$, in which case $x'_1 = f_R(x_0, u_R^0) = x_0 \in \mathcal{X}_{\text{eq}} \subseteq \mathcal{X}_{\text{rec}}$. As a consequence, by definition, we have $x'_0 \in \mathcal{X}_{\text{inv}}$.

Now, by Lemma II.7, we have $x'_t \in \mathcal{X}_{\text{inv}}$ and $x'_{t+1}, x_{t+1} \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$. By definition, $x'_0 = x_0 \in \mathcal{X}_{\text{eq}} \subseteq \mathcal{X}_{\text{safe}}$. Thus, $x'_t, x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$, as claimed. □

[4]Here, $\vec{z} \circ \vec{z}'$ denotes the concatenation of sequences $\vec{z}$ and $\vec{z}'$.