Synthesis of Shared Control Protocols with Provable Safety and Performance Guarantees

Nils Jansen[1] Murat Cubuktepe[1] Ufuk Topcu[1]

***Abstract*— We formalize synthesis of shared control protocols with correctness guarantees for temporal logic specifications. More specifically, we introduce a modeling formalism in which both a human and an autonomy protocol can issue commands to a robot towards performing a certain task. These commands are blended into a joint input to the robot. The autonomy protocol is synthesized using an abstraction of possible human commands accounting for randomness in decisions caused by factors such as fatigue or incomprehensibility of the problem at hand. The synthesis is designed to ensure that the resulting robot behavior satisfies given safety and performance specifications, e.g., in temporal logic. Our solution is based on nonlinear programming and we address the inherent scalability issue by presenting alternative methods. We assess the feasibility and the scalability of the approach by an experimental evaluation.**

I. Introduction

We study the problem of shared control, where a robot shall accomplish a task according to a human operator's goals and given specifications addressing safety or performance. Such scenarios are for instance found in remotely operated semi-autonomous wheelchairs [11]. In a nutshell, the human has a certain action in mind and issues a *command*. Simultaneously, an *autonomy protocol* provides—based on the available information—another command. These commands are *blended*—also referred to as *arbitrated*—and deployed to the robot.

Earlier work discusses shared control from different perspectives [7], [8], [20], [19], [13], [10], however, *formal correctness* in the sense of ensuring *safety* or optimizing *performance* has not been considered. In particular, having the human as an integral factor in this scenario, correctness needs to be treated in an appropriate way as a human might not be able to comprehend factors of a system and—in the extremal case—can drive a system into inevitable failure.

There are several things to discuss. First, a human might not be sure about which command to take, depending on the scenario or factors like fatigue or incomprehensibility of the problem. We account for uncertainties in human decisions by introducing *randomness* to choices. Moreover, a means of actually interpreting a command is needed in form of a user interface, e. g., a brain-computer interface; the usually imperfect interpretation adds to the randomness. We call a formal interpretation of the human's commands the *human strategy* (this concept will be explained later).

As many formal system models are inherently stochastic, our natural formal model for robot actions inside an environment is a Markov decision process (MDP) where deterministic action choices induce probability distributions over system states. Randomness in the choice of actions, like in the human strategy, is directly carried over to these probabilities when resolving nondeterminism. For MDPs, quantitative properties like "the probability to reach a bad state is lower than 0.01" or "the cost of reaching a goal is below a given threshold" can be formally *verified*. If a set of such *specifications* is satisfied for the human strategy and the MDP, the task can be carried out *safely* and *with good performance*.

Given that the human strategy induces certain critical actions with a high probability, one or more specifications might be refuted. In this case, the autonomy should provide an alternative strategy that—when blended with the human strategy—satisfies the specifications without discarding too much of the human's choices. As in [8], the blending puts weight on either the human's or the autonomy protocol's choices depending on factors such as the confidence of the human or the level of information the autonomy protocol has at its disposal.

The question is now how such a human strategy can be obtained. It seems unrealistic that a human can comprehend an MDP modeling a realistic scenario in the first place; primarily due the possibly very large size of the state space. Moreover, a human might not be good at making sense of probabilities or cost of visiting certain states at all. We employ *learning* techniques to collect data about typical human behavior. This can, for instance, be performed within a simulation environment. In our case study, we model a typical shared control scenario based on a wheelchair [11] where a human user and an autonomy protocol share the control responsibility. Having a human user solving a task, we compute strategies from the obtained data using *inverse reinforcement learning* [16], [1]. Thereby, we can give guarantees on how good the obtained strategy approximates the actual intends of the user.

The design of the autonomy protocol is the main concern of this paper. We define the underlying problem as a *nonlinear optimization problem* and propose a technique to address the consequent scalability issues by reducing the problem to a *linear optimization problem*. After an autonomy protocol is synthesized, guarantees on safety and performance can be given assuming that the user behaves according to the human strategy obtained beforehand. The main contribution is a formal framework for the problem of shared autonomy together with thorough discussions on formal verification, experiments, and current pitfalls. A summary of the approaches and an outline are given in Section II.

All authors are with the University of Texas at Austin, Austin, TX 78751, USA, njansen@utexas.edu

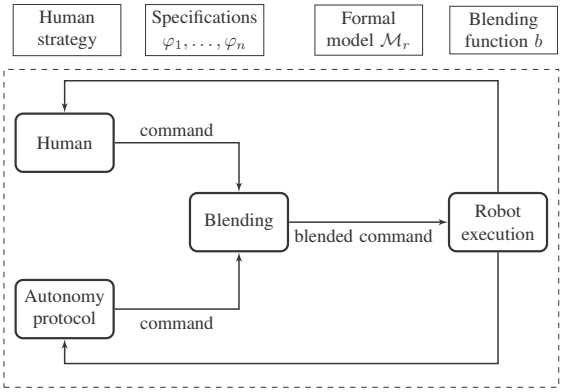


Fig. 1. Shared autonomy architecture.

Shared control has attracted considerable attention recently. We only overview some recent approaches into context with our results. First, Dragan and Srinivasa discussed strategy blending for shared control in [8], [7]. There, the focus was on the *prediction of human goals*. Combining these approaches, e. g., by inferring formal safety or performance specifications by prediction of human goals, is an interesting direction for future work. Iturrate *et al.* presented shared control using feedback based on electroencephalography (a method to record electrical activity of the brain) [13], where a robot is partly controlled via error signals from a brain-computer interface. In [19], Trautman proposes to treat shared control broadly as a random process where different components are modeled by their joint probability distributions. As in our approach, randomness naturally prevents strange effects of blending: Consider actions "up" and "down" to be blended with equally distributed weight without having means to actually evaluating these weights. Finally, in [10] a synthesis method switches authority between a human operator and the autonomy such that satisfaction of linear temporal logic constraints can be ensured.

II. Shared control

Consider first Fig. 1 which recalls the general framework for shared autonomy with blending of commands; additionally we have a set of specifications, a formal model for robot behavior, and a blending function. In detail, a robot is to take care of a certain task. For instance, it shall move to a certain landmark. This task is subject to certain performance and safety considerations, e. g., it is not safe to take the shortest route because there are too many obstacles. These considerations are expressed by a set of *specifications* $\varphi_1, \ldots, \varphi_n$. The possible behaviors of the robot inside an environment are given by a Markov decision process (MDP) $\mathscr{M}_r$. Having MDPs gives rise to choices of certain *actions* to perform and to *randomness* in the environment: A chosen path might induce a high probability to achieve the goal while with a low probability, the robot might slip and therefore fail to complete the task.

Now, in particular, a *human user* issues a set of commands for the robot to perform. We assume that the commands issued by the human are consistent with an underlying randomized *strategy* σ_h for the MDP $\mathscr{M}_r$. Put differently, at design time we compute an abstract strategy σ_h of which the set of human commands is one realization. This modeling way allows to account for a variety of imperfections. Although it is not directly issued by a human, we call this strategy the *human strategy*. Due to possible human incomprehensibility or lack of detailed information, this leads to the fact that the strategy might not satisfy the requirements.

Now, an *autonomy protocol* is to be designed such that it provides an alternative strategy σ_a, the *autonomous strategy*. The two strategies are then *blended*—according to the given blending function b into a new strategy σ_{ha} which satisfies the specifications. The blending function reflects preference over either the decisions of the human or the autonomy protocol. We also ensure that the blended strategy deviates only minimally from the human strategy. At runtime we can then blend decisions of the human user with decisions based on the autonomous strategy. The resulting "blended" decisions are according to the blended strategy σ_{ha}, thereby ensuring satisfaction of the specifications. This procedure, while involving expensive computations at design time, is very efficient at runtime.

Summarized, the problem we are addressing in this paper is then—in addition to the proposed modeling of the scenario—to *synthesize* the autonomy protocol in a way such that the resulting blended strategy meets *all of the specifications* while it only deviates from the human strategy as little as possible. We introduce all formal foundations that we need in Section III. The *shared control synthesis problem* with all needed formalisms is presented in Section IV as being a *nonlinear optimization problem*. Addressing scalability, we reduce the problem to a *linear optimization problem* in Section V. We indicate the feasibility and scalability of our techniques using data-based experiments in Section V and draw a short conclusion in Section VII.

III. Preliminaries

1) Models: A *probability distribution* over a finite or countably infinite set X is a function $\mu : X \to [0,1] \subseteq \mathbb{R}$ with $\sum_{x \in X} \mu(x) = \mu(X) = 1$. The set of all distributions on X is denoted by $Distr(X)$.

Definition 1 (MDP): A *Markov decision process (MDP)* $\mathscr{M} = (S, s_I, A, \mathscr{P})$ is a tuple with a set of states S, a unique initial state $s_I \in S$, a finite set A of actions, and a (partial) probabilistic transition function $\mathscr{P} : S \times A \to Distr(S)$.

MDPs operate by means of *nondeterministic choices* of actions at each state, whose successors are then determined *probabilistically* with respect to the associated probability distribution. The *enabled* actions at state $s \in S$ are denoted by $A(s) = \{\alpha \in A \mid \exists \mu \in Distr(S).\, \mu = \mathscr{P}(s, \alpha)\}$. To avoid deadlock states, we assume that $|A(s)| \geq 1$ for all $s \in S$. A *cost function* $\rho : S \times A \to \mathbb{R}_{\geq 0}$ for an MDP $\mathscr{M}$ adds cost to a *transition* $(s, \alpha) \in S \times A$ with $\alpha \in A(s)$. A *path* in an $\mathscr{M}$ is a finite (or infinite) sequence $\pi = s_0 \alpha_0 s_1 \alpha_1 \ldots$ with $\mathscr{P}(s_i, \alpha, s_{i+1}) > 0$ for all $i \geq 0$. If $|A(s)| = 1$ for all $s \in S$, all actions can be disregarded and the MDP $\mathscr{M}$ reduces to a *discrete-time Markov chain (MC)*.

The *unique probability measure* $\Pr^{\mathscr{D}}(\Pi)$ for a set Π of paths of MC $\mathscr{D}$ can be defined by the usual cylinder set construction, the *expected cost* of a set Π of paths is denoted by $\mathrm{EC}^{\mathscr{D}}(\Pi)$, see [2] for details. In order to define a probability measure and expected cost on MDPs, the nondeterministic choices of actions are resolved by so-called *strategies*. For practical reasons, we restrict ourselves to *memoryless* strategies, again refer to [2] for details.

Definition 2 (Strategy): A *randomized strategy* for an MDP $\mathscr{M}$ is a function $\sigma\colon S \to \mathit{Distr}(A)$ such that $\sigma(s)(\alpha) > 0$ implies $\alpha \in A(s)$. A strategy with $\sigma(s)(\alpha) = 1$ for $\alpha \in A$ and $\sigma(\beta) = 0$ for all $\beta \in A \setminus \{\alpha\}$ is called *deterministic*. The set of all strategies over $\mathscr{M}$ is denoted by $\mathit{Sched}^{\mathscr{M}}$.
Resolving all nondeterminism for an MDP $\mathscr{M}$ with a strategy $\sigma \in \mathit{Sched}^{\mathscr{M}}$ yields an *induced Markov chain* $\mathscr{M}^{\sigma}$. Intuitively, the random choices of actions from σ are transferred to the transition probabilities in $\mathscr{M}^{\sigma}$.

Definition 3 (Induced MC): Let MDP $\mathscr{M} = (S, s_I, A, \mathscr{P})$ and strategy $\sigma \in \mathit{Sched}^{\mathscr{M}}$. The *MC induced by* $\mathscr{M}$ *and* σ is $\mathscr{M}^{\sigma} = (S, s_I, A, \mathscr{P}^{\sigma})$ where

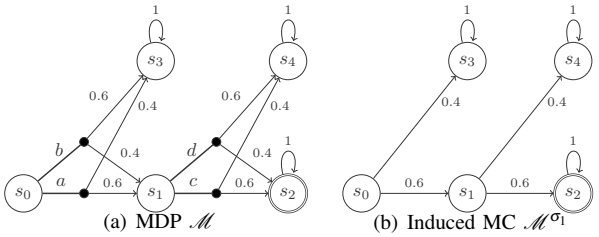
$$\mathscr{P}^{\sigma}(s,s') = \sum_{\alpha \in A(s)} \sigma(s)(\alpha) \cdot \mathscr{P}(s,\alpha)(s') \text{ for all } s,s' \in S\ .$$

2) Specifications: A quantitative reachability property $\mathbb{P}_{\leq\lambda}(\Diamond T)$ with upper probability threshold $\lambda \in [0,1] \subseteq \mathbb{Q}$ and target set $T \subseteq S$ constrains the probability to reach T from s_I in $\mathscr{M}$ to be at most λ. Expected cost properties $\mathbb{E}_{\leq\kappa}(\Diamond G)$ impose an upper bound $\kappa \in \mathbb{Q}$ on the expected cost to reach goal states $G \subseteq S$. Intuitively, bad states T shall only be reached with probability λ (*safety specification*) while the expected cost for reaching goal states G has to be below κ (*performance specification*). Probability and expected cost to reach T from s_I are denoted by $\Pr(\Diamond T)$ and $\mathrm{EC}(\Diamond T)$, respectively. Hence, $\Pr^{\mathscr{D}}(\Diamond T) \leq \lambda$ and $\mathrm{EC}^{\mathscr{D}}(\Diamond G) \leq \kappa$ express that the properties $\mathbb{P}_{\leq\lambda}(\Diamond T)$ and $\mathbb{E}_{\leq\kappa}(\Diamond G)$ are satisfied by MC $\mathscr{D}$. These concepts are analogous for lower bounds on the probability. We also use *until properties* of the form $\Pr_{\geq\lambda}(\neg T\ \mathscr{U}\ G)$ expressing that the probability of reaching G while not reaching T beforehand is at least λ.

An MDP $\mathscr{M}$ satisfies both safety specification ϕ and performance specification ψ, iff *for all* strategies $\sigma \in \mathit{Sched}^{\mathscr{M}}$ it holds that the induced MC $\mathscr{M}^{\sigma}$ satisfies ϕ and ψ, i.e., $\mathscr{M}^{\sigma} \models \phi$ and $\mathscr{M}^{\sigma} \models \psi$. If several performance or safety specifications $\varphi_1, \ldots, \varphi_n$ are given MDP $\mathscr{M}$, the simultaneous satisfaction for all strategies, denoted by $\mathscr{M} \models \varphi_1, \ldots, \varphi_n$, can be formally verified for an MDP using multi-objective model checking [9].

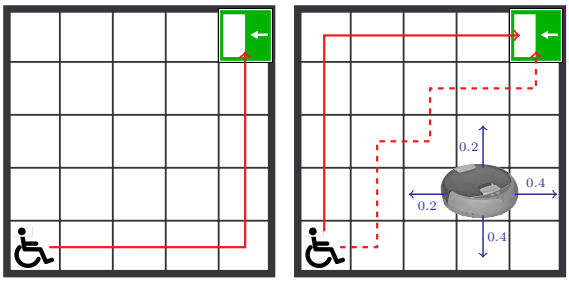
Here, we are interested in the *synthesis problem*, where the aim is to find one particular strategy σ for which the specifications are satisfied. If for $\varphi_1, \ldots, \varphi_n$ and strategy σ it holds that $\mathscr{M}^{\sigma} \models \varphi_1, \ldots, \varphi_n$, then σ is said to *admit* the specifications, also denoted by $\sigma \models \varphi_1, \ldots, \varphi_n$.

Example 1: Consider Fig. 2(a) depcting MDP $\mathscr{M}$ with initial state s_0, where states s_0 and s_1 have choices between actions a or b and c or d, respectively. For instance, action a induces a probabilistic choice between s_1 and s_3 with probabilities 0.6 and 0.4. The self loops at s_2, s_3 and s_4 indicate looping back with probability one for each action.



(a) MDP $\mathscr{M}$ (b) Induced MC $\mathscr{M}^{\sigma_1}$

Fig. 2. MDP $\mathscr{M}$ with target state s_2 and induced MC for strategy σ_{unif}



(a) Human perspective (b) Autonomy perspective

Fig. 3. A wheelchair in a shared control setting.

Assume now, a safety specification is given by $\phi = \mathbb{P}_{\leq 0.21}(\Diamond s_2)$. The specification is *violated* for $\mathscr{M}$, as the deterministic strategy $\sigma_1 \in \mathit{Sched}^{\mathscr{M}}$ with $\sigma_1(s_1)(\alpha) = 1$ and $\sigma_1(s_1)(c) = 1$ induces a probability of reaching s_2 of 0.36, see the induced MC in Fig. 2(b). For the randomized strategy $\sigma_{unif} \in \mathit{Sched}^{\mathscr{M}}$ with $\sigma_{unif}(s_0)(\alpha) = \sigma_{unif}(s_0)(b) = 0.5$ and $\sigma_{unif}(s_1)(c) = \sigma_{unif}(s_1)(d) = 0.5$, which chooses between all actions uniformly, the specification is also violated: The probability of reaching s_2 is 0.25, hence $\sigma_2 \not\models \phi$. However, for the deterministic strategy $\sigma_{safe} \in \mathit{Sched}^{\mathscr{M}}$ with $\sigma_{safe}(s_0)(b) = 1$ and $\mathit{sched}_{safe}(s_1)(d) = 1$ the probability is 0.16, thus $\sigma_{safe} \models \phi$. Note that σ_{safe} minimizes the probability of reaching s_2 while σ_1 maximizes this probability.

IV. Synthesizing Shared Control Protocols

In this section we describe our formal approach to synthesize a shared control protocol in presence of randomization. We start by formalizing the concepts of *blending* and *strategy perturbation*. Afterwards we formulate the general problem and show that the solution to the synthesis problem is correct.

Example 2: Consider Fig. 3, where a room to navigate in is abstracted into a grid. We will use this as our ongoing example. A wheelchair as in [11] is to be steered from the lower left corner of the grid to the exit on the upper right corner of the grid. There is also an autonomous robotic vacuum cleaner moving around the room; the goal is for the wheelchair to reach the exit without crashing into the vacuum cleaner. We now assume that the vacuum cleaner moves according to probabilities that are fixed according to evidence gathered beforehand; these probabilities are unknown or incomprehensible to the human user. To improve the safety of the wheelchair, it is equipped with an autonomy protocol that is to improve decisions of the human or even overwrite them

in case of safety hazards. For the design of the autonomy protocol, the evidence data about the cleaner is present.

Now an obvious strategy to move for the wheelchair, not taking into account the vacuum cleaner, is depicted by the red solid line in Fig. 3(a). As indicated in Fig. 3(b), the strategy proposed by the human is *unsafe* because there is a high probability to collide with the obstacle. The autonomy protocol computes a safe strategy, indicated by the solid line in Fig. 3(b). As this strategy deviates highly from the human strategy, the dashed line indicates a still safe enough alternative which is a compromise or—in our terminology—a blending between the two strategies.

We assume in the following that possible *behaviors of the robot* inside the environment are modeled by MDP $\mathscr{M}_r = (S, s_I, A, \mathscr{P})$. The *human strategy* is given as randomized strategy σ_h for $\mathscr{M}_r$. We explain how to obtain this strategy in Section VI. *Specifications* are $\varphi_1, \ldots, \varphi_n$ being either *safety properties* $\mathbb{P}_{\leq\lambda}(\Diamond T)$ or *performance properties* $\mathbb{E}_{\leq\kappa}(\Diamond T)$.

A. Strategy blending

Given two strategies, they are to be *blended* into a new strategy favoring decisions of one or the other in each state of the MDP. In our setting, the human strategy $\sigma_h \in \mathit{Sched}^{\mathscr{M}_r}$ is blended with the autonomous strategy $\sigma_a \in \mathit{Sched}^{\mathscr{M}_r}$ by means of an arbitrary *blending function*. In [8] it is argued that blending intuitively reflects the *confidence* in how good the autonomy protocol is able to assist with respect to the human user's goals. In addition, factors probably unknown or incomprehensible for the human such as safety or performance optimization also should be reflected by such a function.

Put differently, possible actions of the user should be assigned *low confidence* by the blending function, if he cannot be trusted to make the right decisions. For instance, recall Example 2. At cells of the grid where with a very high probability the wheelchair might collide with the vacuum cleaner, it makes sense to assign a high confidence in the autonomy protocol's decisions because not all safety-relevant information is present for the human.

In order to enable formal reasoning together with such a function we instantiate the blending with a *state-dependent* function which at each state of an MDP weighs the confidence in both the human's and the autonomy's decisions. A more fine-grained instantiation might incorporate not only the current state of the MDP but also the strategies of both human and autonomy or history of a current run of the system. Such a formalism is called *linear blending* and is used in what follows. In [19], additional notions of blending are discussed.

Definition 4 (Linear blending): Given an MDP $\mathscr{M}_r = (S, s_I, A, \mathscr{P})$, two strategies $\sigma_h, \sigma_a \in \mathit{Sched}^{\mathscr{M}_r}$, and a *blending function* $b\colon S \to [0,1]$, *the blended strategy* $\sigma_{ha} \in \mathit{Sched}^{\mathscr{M}_r}$ for all states $s \in S$, and actions $\alpha \in A$ is

$$\sigma_{ha}(s)(\alpha) = b(s) \cdot \sigma_h(s)(\alpha) + (1 - b(s)) \cdot \sigma_a(s)(\alpha) \ .$$

Note that the blended strategy σ_{ha} is a well-defined randomized strategy. For each $s \in S$, the value $b(s)$ represents the confidence in the human's decisions at this state, i. e., the "weight" of σ_h at s.

Coming back to Example 2, the critical cells of the grid correspond to certain states of the MDP $\mathscr{M}_r$; at these states a very low confidence in the human's decisions should be assigned. For instance at such a state $s \in S$ we might have $b(s) = 0.1$ leading to the fact that all randomized choices of the human strategy are scaled down by this factor. Choices of the autonomous strategy are only scaled down by factor 0.9. The addition of these scaled choices then gives a new strategy highly favoring the autonomy's decisions.

B. Perturbation of strategies

As mentioned before, we want to ensure that the blended strategy deviates minimally from the human strategy. To now *measure* such a deviation, we introduce the concept of *perturbation* which was—on a complexity theoretic level—for instance investigated in [5]. Here, we introduce an *additive perturbation* for a (randomized) strategy, incrementing or decrementing probabilities of action choices such that a well-defined distribution over actions is maintained.

Definition 5 (Strategy perturbation): Given MDP $\mathscr{M}$ and strategy $\sigma \in \mathit{Sched}^{\mathscr{M}}$, an (additive) *perturbation* δ is a function $\delta\colon S \times A \to [-1,1]$ with

$$\sum_{\alpha \in A} \delta(s,\alpha) = 0 \text{ for all } s \in S \ .$$

The value $\delta(s,\alpha)$ is called the *perturbation value* at state s for action α. Overloading the notation, the *perturbed strategy* $\delta(\sigma)$ is given by

$$\delta(\sigma)(s,\alpha) = \sigma(s)(\alpha) + \delta(s,\alpha) \text{ for all } s \in S \text{ and } \alpha \in A \ .$$

C. Design of the autonomy protocol

For the *formal problem*, we are given blending function b, specifications $\varphi_1, \ldots, \varphi_n$, MDP $\mathscr{M}_r$, and human strategy $\sigma_h \in \mathscr{M}_r$. We assume that σ_h does not satisfy all of the specifications, i. e., $\sigma_h \not\models \varphi_1, \ldots, \varphi_n$. The *autonomy protocol* provides the *autonomous strategy* $\sigma_a \in \mathit{Sched}^{\mathscr{M}_r}$. According to b, the strategies σ_a and σ_h are blended into strategy σ_{ha}, see Definition 4, i. e., $\sigma_{ha}(s,\alpha) = b(s) \cdot \sigma_a(s,\alpha) + (1 - b(s)) \cdot \sigma_h(s,\alpha)$. The *shared control synthesis problem* is to design the autonomy protocol such that for the blended strategy σ_{ha} it holds $\sigma_{ha} \models \varphi_1, \ldots, \varphi_n$, while *minimally deviating* from σ_h. The deviation from σ_h is captured by finding a perturbation δ as in Definition 5, where, e. g., the *infinity norm* of all perturbation values is minimal.

Our problem involves the explicit computation of a randomized strategy and the induced probabilities, which is inherently nonlinear because the corresponding variables need to be multiplied. Therefore, the canonical formulation is given by a nonlinear optimization program (NLP). We first assume that the only specification is a quantitative reachability property $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$, then we describe how more properties can be included. The program has to encompass defining the autonomous strategy σ_a, the perturbation δ of the human strategy, the blended strategy σ_{ha}, and the probability of reaching the set of target states $T \subseteq S$.

We introduce the following specific set Var of *variables*:

- $\sigma_a^{s,\alpha}, \sigma_{ha}^{s,\alpha} \in [0,1]$ for each $s \in S$ and $\alpha \in A$ define the autonomous strategy σ_a and the blended strategy σ_{ha}.
- $\delta^{s,\alpha} \in [-1,1]$ for each $s \in S$ and $\alpha \in A$ are the perturbation variables for σ_h and σ_{ha}.
- $p_s \in [0,1]$ for each $s \in S$ are assigned the probability of reaching $T \subseteq S$ from state s under strategy σ_{ha}.

Using these variables, the NLP reads as follows:

$$\text{minimize} \quad \max\{|\delta^{s\alpha}| \mid s \in S, \alpha \in A\} \tag{1}$$

$$\text{subject to} \quad p_{s_I} \leq \lambda \tag{2}$$

$$\forall s \in T. \quad p_s = 1 \tag{3}$$

$$\forall s \in S. \quad \sum_{\alpha \in A} \sigma_a^{s,\alpha} = \sum_{\alpha \in A} \sigma_{ha}^{s,\alpha} = 1 \tag{4}$$

$$\forall s \in S. \forall \alpha \in A. \quad \sigma_{ha}^{s,\alpha} = \sigma_h(s)(\alpha) + \delta^{s,\alpha} \tag{5}$$

$$\forall s \in S. \quad \sum_{\alpha \in A} \delta^{s,\alpha} = 0 \tag{6}$$

$$\forall s \in S. \forall \alpha \in A. \quad \sigma_{ha}^{s,\alpha} = b(s) \cdot \sigma_h(s)(\alpha) + (1 - b(s)) \cdot \sigma_a^{s,\alpha} \tag{7}$$

$$\forall s \in S. \quad p_s = \sum_{\alpha \in A} \sigma_{ha}^{s,\alpha} \cdot \sum_{s' \in S} \mathscr{P}(s,\alpha)(s') \cdot p_{s'} \tag{8}$$

The NLP works as follows. First, the infinity norm of all perturbation variables is minimized (by minimizing the maximum of all perturbation variables) (1). The probability assigned to the initial state $s_I \in S$ has to be smaller than or equal to λ to satisfy $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$ (2). For all target states $T \subseteq S$, the probability of the corresponding probability variables is assigned one (3). Now, to have well-defined strategies σ_a and σ_{ha}, we ensure that the assigned values of the corresponding strategy variables at each state sum up to one (4). The perturbation δ of the human strategy σ_h resulting in the strategy σ_{ha} as in Definition 5 is computed using the perturbation variables (5); in order for the perturbation to be well-defined, the variables have to sum up to zero at each state (6). The blending of σ_a and σ_{ha} with respect to b as in Definition 4 is defined in (7). Finally, the probability to reach $T \subseteq S$ from each $s \in S$ is computed in (8), defining a non-linear equation system, where action probabilities, given by the induced strategy σ_{ha}, are multiplied by probability variables for all possible successors.

Note that this nonlinear program is in fact *bilinear* due to multiplying the strategy variables $\sigma_{ha}^{s,\alpha}$ with the probability variables $p_{s'}$ (8). The number of constraints is governed by the number of state and action pairs, i. e., the *size of the problem* is in $\mathscr{O}(|S_r| \cdot |A|)$.

An *assignment* of real-valued variables is a function $\nu\colon \text{Var} \to \mathbb{R}$; it is *satisfying* for a set of (in)equations, if each one evaluates to `true`. A satisfying assignment ν^* is *minimizing* with respect to objective o if for $\nu^*(o) \in \mathbb{R}$ there is no other assignment ν' with $\nu'(o) < \nu^*(o)$. Using these notions, we state the correctness of the NLP in (1) – (8).

Theorem 1 (Soundness and completeness): The NLP is *sound* in the sense that each minimizing assignment induces a solution to the shared control synthesis problem. It is *complete* in the sense that for each solution to the shared control synthesis there is a minimizing assignment of the NLP.

TABLE I
EXAMPLE RESULTS

	b_i	$\sigma_a(a)$	$\sigma_a(b)$	$\sigma_a(c)$	$\sigma_a(d)$	$\sigma_{ha}(a)$	$\sigma_{ha}(b)$	$\sigma_{ha}(c)$	$\sigma_{ha}(d)$	$\Pr_{ha}$
$i=1$	0.5	0.08	0.92	0.08	0.92	0.29	0.71	0.29	0.71	0.209
$i=2$	0.1	0.27	0.73	0.27	0.73	0.29	0.71	0.29	0.71	0.209
$i=3$	0	0.29	0.71	0.29	0.71	0.29	0.71	0.29	0.71	0.209

Soundness tells that each satisfying assignment of the variables corresponds to strategies σ_a and σ_{ha} as well as the perturbation δ as defined above. Moreover, any optimal solution induces a perturbation minimally deviating from the human strategy σ_h. Completeness means that all possible solutions of the shared control synthesis problem can be encoded by this NLP. Unsatisfiability means that no such solution exists; the problem is *infeasible*.

D. Additional specifications

We now explain how the NLP can be extended for further specifications. Assume in addition to $\varphi = \mathbb{P}_{\leq\lambda}(\Diamond T)$, another reachability property $\varphi' = \mathbb{P}_{\leq\lambda'}(\Diamond T')$ with $T' \neq T$ is given. We add another set of probability variables p'_s for each state $s \in S$; (2) is copied for p'_{s_I} and λ', (3) is defined for all states $s \in T \cup T'$ and (8) is copied for all p'_s, thereby computing the probability of reaching T' under σ_{ha} for all states.

To handle an *expected cost property* $\mathbb{E}_{\leq\kappa}(\Diamond G)$ for $G \subseteq S$, we use variables r_s being assigned the expected cost for reaching G for all $s \in S$. We add the following equations:

$$r_{s_I} \leq \kappa \tag{9}$$

$$\forall s \in G. \quad r_s = 0 \tag{10}$$

$$\forall s \in S. \quad r_s = \sum_{\alpha \in A} \left(\sigma_{ha}^{s,\alpha} \cdot \mathrm{r}(s,\alpha) + \sum_{s' \in S} \mathscr{P}(s,\alpha)(s') \cdot r_{s'} \right) \tag{11}$$

First, the expected cost of reaching G is smaller than or equal to κ at s_I (9). Goal state are assigned cost zero (10), otherwise infinite cost is collected at absorbing states. Finally, the expected cost for all other states is computed by (11) where according to the blended strategy σ_{ha} the cost of each action is added to the expected cost of the successors. An important insight is that if all specifications are expected reward properties, the program is *no longer nonlinear* but a linear program (LP), as there is no multiplication of variables.

E. Generalized blending

If the problem is not feasible for the given blending function, optionally the autonomy protocol can try to compute a new function $b\colon S \to [0,1]$ for which the altered problem is feasible. We call this procedure *generalized blending*. The idea is that computing this function gives the designer of the protocol insight on where more confidence needs to be placed into the autonomy or, vice versa, where the human cannot be trusted to satisfy the given specifications.

Computing this new function is achieved by nearly the same NLP as for a fixed blending function while adding variables b^s for each state $s \in S$, defining the new blending function by $b(s) = b^s$. We substitute Equation 7 by

$$\forall s \in S. \forall \alpha \in A. \quad \sigma_{ha}^{s,\alpha} = b^s \cdot \sigma_h(s)(\alpha) + (1 - b^s) \cdot \sigma_a^{s,\alpha} \ . \tag{12}$$

A satisfying assignment for the resulting nonlinear program induces a suitable blending function $b\colon S \to [0,1]$ in addition to the strategies. If this problem is also infeasible, there is no strategy that satisfies the given specifications for MDP $\mathscr{M}_r$.

Corollary 1: If there is no solution for the NLP given by Equations 1 – 12, there is no strategy $\sigma \in Sched^{\mathscr{M}_r}$ such that $\sigma \models \varphi_1, \ldots, \varphi_n$.

As there are no restrictions on the blending function, this corollary trivially holds: Consider for instance b with $b(s) = 0$ for each $s \in S$. This function disregards the human strategy which may be perturbed to each other strategy $\sigma_a = \sigma_{ha}$.

Example 3: Reconsider the MDP $\mathscr{M}$ from Example 1 with specification $\varphi = \mathbb{P}_{\leq 0.21}(\Diamond\{s_2\})$ and the randomized strategy σ_{unif} which takes each action uniformly distributed. As we saw, $\sigma_{unif} \not\models \varphi$. We choose this strategy as the human strategy $\sigma_h = \sigma_{unif}$ and $\mathscr{M}_r = \mathscr{M}$ as the robot MDP. For a blending function b_h putting high confidence in the human, e. g., if $b_h(s) \geq 0.6$ for all $s \in S$, the problem is infeasible.

In Table I we display results putting medium (b_1), low (b_2), or no confidence (b_3) in the human at s_0 and s_1. We list the assignments for the resulting strategies σ_a and σ_{ha} as well as the probability $\Pr_{ha} = \Pr_{s_0}^{\mathscr{M}_r^{\sigma_{ah}}}(\Diamond T)$ to reach s_2 under the blended strategy σ_{ha}. The results were obtained using the NLP solver `IPOPT` [4].

We observe that for decreasing confidence in the human decisions, the autonomous strategy has higher probabilities for actions a and c which are the "bad" actions here. That means that—if there is a higher confidence in the autonomy—solutions farer away from the optimum are good enough. The maximal deviation from the human strategy is 0.21. *Generalized blending* with maximizing over the confidence in the human's decisions at all states $s \in S$ yields $b_h(s) = 0.582$, i. e., we compute the highest possible confidence in the human's decisions where the problem is still feasible under the given human strategy.

V. Computationally Tractable Approach

The nonlinear programming approach presented in the previous section gives a rigorous method to solve the shared control synthesis problem and serves as mathematically concise definition of the problem. However, NLPs are known to have severe restrictions in terms of scalability and suffer from numerical instabilities. The crucial point to an efficient solution is circumventing the expensive computation of optimal randomized strategies and reducing the number of variables. We propose a heuristic solution which enables to use linear programming (LP) while ensuring soundness.

We utilize a technique referred to as *model repair*. Intuitively, an erroneous model is changed such that it satisfies certain specifications. In particular, given a Markov chain $\mathscr{D}$ and a specification φ that is violated by $\mathscr{D}$, a repair of $\mathscr{D}$ is an automated method that transforms it to new MC $\mathscr{D}'$ such that φ is satisfied for $\mathscr{D}'$. Transforming refers to changing probabilities or cost while regarding certain side constraints such as keeping the original graph structure.

In [3], the first approach to automatically repair an MC model was presented as an NLP. Simulation-based algorithms were investigated in [6]. A heuristic but very scalable technique called *local repair* was proposed in [17]. This approach greedily changes the probabilities or cost of the original MC until a property is satisfied. An upper bound δ_r on changes of probabilities or cost can be specified; correctness and completeness can be given in the sense that if a repair with respect to δ_r exists, it will be obtained.

Take now the MC $\mathscr{D}_r^{\sigma_h}$ which is induced by the robot MDP $\mathscr{M}_r$ and the human strategy σ_h. We perform model repair such that the repaired MC $\mathscr{D}' = (S, s_I, P')$ satisfies the specifications $\varphi_1, \ldots, \varphi_n$. The question is now, how from the repaired MC $\mathscr{D}'$, the strategy $\sigma' \in Sched^{\mathscr{M}_r}$ can be extracted. More precisely, we need σ' inducing exactly $\mathscr{D}'$, i. e., $\mathscr{D}_r^{\sigma'} = \mathscr{D}'$, when applied to MDP $\mathscr{M}_r$.

First, we need to make sure that the repaired MC is *consistent* with the original MDP such that a strategy σ' with $\mathscr{D}_r^{\sigma'} = \mathscr{D}'$ actually exists. Therefore, we define the maximal and minimal possible transition probabilities $P_{\max}$ and $P_{\min}$ that can occur in any induced MC of MDP $\mathscr{M}_r$:

$$P_{\max}(s,s') = \max\{\mathscr{P}_r(s,\alpha)(s') \mid \alpha \in A\} \tag{13}$$

for all $s \in S$; $P_{\min}$ is defined analogously. Now, the repair is performed such that in the resulting MC $\mathscr{D}' = (S, s_I, P')$ for all $s, s' \in S$ it holds that

$$P_{\min}(s,s') \leq P(s,s') \leq P_{\max}(s,s') \ . \tag{14}$$

While obtaining $\mathscr{D}'$, model checking needs to be performed intermediately to check if the specifications are satisfied; once they are, the algorithm terminates. In fact, for each state $s \in S$, the probability of satisfaction is computed. We assign variables mc_s for all $s \in S$ with exactly this probability:

$$mc_s = \Pr(s \models \varphi_1, \ldots, \varphi_n) \ . \tag{15}$$

Now recall the NLP from the previous section, in particular Equation 8 which is the only nonlinear equation of the program. We replace each variable p_s by the concrete model checking result mc_s for each $s \in S$:

$$mc_s = \sum_{\alpha \in A} \sigma_{ha}^{s,\alpha} \cdot \sum_{s' \in S} \mathscr{P}(s,\alpha)(s') \cdot mc_{s'} \ . \tag{16}$$

As (16) is affine in the variables σ_{ah}, the program resulting from replacing (8) by (16) is a linear program (LP). Moreover, (2) and (3) can be removed, reducing the number of constraints and variables. The LP gives a feasible solution to the shared control synthesis problem.

Lemma 1 (Correctness): The LP is sound in the sense that each minimizing assignment induces a solution to the shared control problem.

The correctness is given by construction, as the specifications are satisfied for the blended strategy which is derived from

the repaired MC. However, the minimal deviation from the human strategy as in Equation 1 is dependent on the previous computation of probabilities for the blended strategy. Therefore, we actually compute an *upper bound* on the optimal solution. Let δ^* be the minimal deviation possible for any given problem and δ be the minimal deviation obtained by the LP resulting from replacing (8) by (16). Let $\|\delta\|_\infty$ and $\|\delta^*\|_\infty$ denote the infinity norms of both perturbations.

Corollary 2: For the perturbations δ and δ^* of σ_h it holds that $\|\delta^*\|_\infty \leq \|\delta\|_\infty$.

As we mentioned before, the local repair method can employ a bound δ_r on the maximal change of probabilities or cost in the model. If a repair exists for a given δ_r, the resulting deviation δ is then bounded by this δ_r.

VI. Case study and experiments

Defining a formal synthesis approach to the shared control scenario requires a precomputed estimation of a human user's intentions. As explained in the previous chapter, we account for inherent uncertainties by using a *randomized strategy* over possible actions to take. We discuss how such strategies may be obtained and report on benchmark results.

A. Experimental setting

Our setting is the wheelchair scenario from Example 2 inside an interactive `Python` environment. The size of the grid is variable and an arbitrary number of stationary and randomly moving obstacles (the vacuum cleaner) can be defined. An agent (the wheelchair) is moved according to predefined (randomized) strategies or interactively by a human user.

From this scenario, an MDP with states corresponding to the position of the agent and the obstacles is generated. Actions induce position changes of the agent. The safety specification ensures that the agent reaches a target cell without crashing into an obstacle with a certain high probability $\lambda \in [0,1]$, formally $\mathbb{P}_{\geq\lambda}(\neg\texttt{crash}\ \mathscr{U}\ \texttt{target})$. We use the probabilistic model checker `PRISM` [15] for verification, in form of either a worst–case analysis for each possible strategy or concretely for a specific strategy. The whole toolchain integrates the simulation environment with the approaches described in the previous sections. We use the NLP solver `IPOPT` [4] and the LP solver `Gurobi` [12]. To perform model repair for strategies, see Section V, we implemented the greedy method from [17] into our framework augmented by side constraints ensuring well-defined strategies.

B. Data collection

We ask five participants to perform tests in the environment with the goal to move the agent to a target cell while never being in same cell as the moving obstacle. From the data obtained from each participant, an individual randomized human strategy σ_h for this participant can be obtained via Maximum Entropy Inverse Reinforcement Learning (MEIRL) [22]. Inverse reinforcement learning has—for instance—also been used in [14] to collect data about human behavior in a shared control scenario (though without any formal guarantees) or in [18] to distinguish human intents

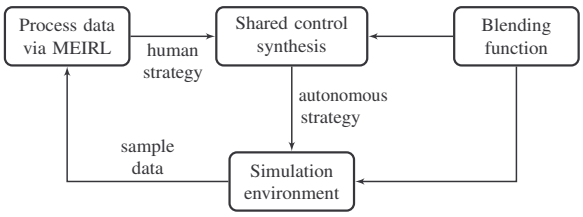


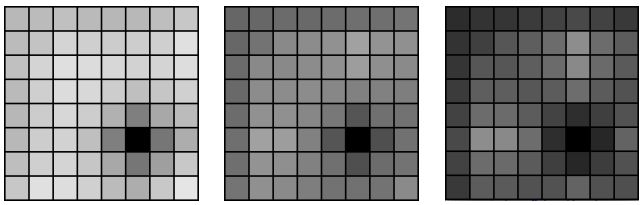
Fig. 4. Experimental setting for the shared control simulation.

with respect to different tasks. In our setting, each *sample* is one particular command of the participant, while we have to assume that command is actually made with the intent to satisfy the specification of safely reaching a target cell. For the resulting strategy, the probability of a possible deviation from the actual intend can be bounded with respect to the number of samples using Hoeffding's inequality, see [21] for details. On the other hand, we can determine the number of samples needed to get a reasonable approximation of typical behavior.

The concrete probabilities of possible deviation depend on $\mathscr{O}(\exp(-n\varepsilon))$, where n is the number of samples and ε is the desired upper bound on the deviation between the true probability of satisfying the specification and the average obtained by the sampled data. Here, in order to ensure an upper bound $\varepsilon = 0.1$ with probability 0.99, the required amount of samples is 265.

C. Experiments

The work flow of the experiments is depicted in Figure 4. First off, we discuss sample data for one particular participant using a 8×8 grid with one moving obstacle inducing an MDP of 2304 states. In the synthesis, we employ the model repair procedure as explained in Section V because the approach based on NLP is only feasible for very small examples. We design the blending function as follows: At states where the human strategy induces a high probability of crashing, we put low confidence in the human and vice versa. Using this function, the autonomous strategy σ_a is created and passed (together with the function) back to the environment. Note that the blended strategy σ_{ah} is ensured to satisfy the specification, see Lemma 1. Now, we let the same participant as before do test runs, but this time we blend the human commands with the (randomized) commands of the autonomous strategy σ_a. Then the actual action of the agent is determined stochastically. We obtain the following results. Our safety specification is instantiated with $\lambda = 0.7$, ensuring that the target is safely reached with at least probability 0.7. The human strategy σ_h has probability 0.546, violating the specification. With the aforementioned blending function we compute σ_a which induces probability 0.906. Blending these two strategies into σ_{ah} yields a probability of 0.747. When testing the synthesized autonomy protocol for the individual participant, we observe that his choices are mostly corrected if intentionally bad decisions are made. Also, simulating the blended strategy leeds to the expected result that the agent



(a) strategy σ_h (b) strategy σ_{ah} (c) strategy σ_h

Fig. 5. Graphical representation of the obtained human, blended, and autonomous strategy in the grid.

does not crash in roughly 70% of the cases.

To make the behavior of the strategies more accessible, consider Figure 5. For each σ_a, σ_{ah}, and σ_h we indicate for each cell of the grid the worst-case probability to safely reach the target. This probability depends on the current position of the obstacle, which is again probabilistic. The darker the color, the higher the probability; thereby black indicates a probability of 1 to reach the target. We observe that the human's decisions are rather risky even near the target, while for the blended strategy—once the agent is near the target—there is a very high probability of reaching it safely. This representation also shows that with our approach the blended strategy improves the human strategy while not changing it too much. Specifically, the maximal deviation from the human strategy is 0.27, which is the result of the infinity norm as in Equation 1.

To finally assess the scalability of our approach, consider Table II. We generated MDPs for several grid sizes, number of obstacles, and human strategies. We list the number of reachable MDP states (states) and the number of transitions (trans.). We report on the time the synthesis process took (synth.), which is basically the time of solving the LP, and the total time including model checking times using `PRISM` (total) measured in seconds. To give an indication on the quality of the synthesis, we list the deviation from the human strategy (δ_∞). A memory out is indicated by "–MO–". All experiments were conducted on a 2.3GHz machine with 8GB of RAM. Note that MDPs resulting from grid structures are very strongly connected, resulting in a large number of transitions. Thus, the encoding in the `PRISM`-language [15] is very large, rendering it a very hard problem. We observe that while the procedure is very efficient for models having a few thousand states and hundreds of thousands of transitions, its scalability is ultimately limited due to memory issues. In the future, we will utilize efficient symbolic data structures internal to `PRISM`. Moreover, we observe that for larger benchmarks the computation time is governed by the solving time of `Gurobi`.

TABLE II
SCALABILITY RESULTS.

grid	obst.	states	trans.	synth.	total	δ_∞
8×8	1	2.304	36.864	6.30	14.12	0.15
8×8	2	82.944	5.308.416	–MO–	–MO–	–MO–
10×10	1	3.600	57.600	12.29	23.80	0.24
12×12	1	14.400	230.400	157.94	250.78	0.33

VII. CONCLUSION

We introduced a formal approach to synthesize autonomy protocols in a shared control setting with guarantees on quantitative safety and performance specifications. The practical usability of our approach was shown by means of data-based experiments. Future work will concern experiments in robotic scenarios and further improvement of the scalability.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[3] Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, CR Ramakrishnan, and Scott A Smolka. Model repair for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 326–340. Springer, 2011.

[4] Lorenz T. Biegler and Victor M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[5] Taolue Chen, Yuan Feng, David S. Rosenblum, and Guoxin Su. Perturbation analysis in verification of discrete-time Markov chains. In *CONCUR*, volume 8704 of *LNCS*, pages 218–233. Springer, 2014.

[6] Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Kwiatkowska, Hongyang Qu, and Lijun Zhang. Model repair for Markov decision processes. In *TASE*, pages 85–92. IEEE CS, 2013.

[7] Anca D. Dragan and Siddhartha S. Srinivasa. Formalizing assistive teleoperation. In *Robotics: Science and Systems*, 2012.

[8] Anca D. Dragan and Siddhartha S. Srinivasa. A policy-blending formalism for shared control. *I. J. Robotic Res.*, 32(7):790–805, 2013.

[9] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008.

[10] Jie Fu and Ufuk Topcu. Synthesis of shared autonomy policies with temporal logic specifications. *IEEE Trans. Automation Science and Engineering*, 13(1):7–17, 2016.

[11] F. Galán, M. Nuttin, E. Lew, P. W. Ferrez, G. Vanacker, J. Philips, and J. del R. Millán. A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, 2016/05/28.

[12] Gurobi Optimization, Inc. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2013.

[13] Iñaki Iturrate, Jason Omedes, and Luis Montesano. Shared control of a robot using eeg-based feedback signals. In *Proceedings of the 2Nd Workshop on Machine Learning for Interactive Systems: Bridging the Gap Between Perception, Action and Communication*, MLIS '13, pages 45–50, New York, NY, USA, 2013. ACM.

[14] Shervin Javdani, J Andrew Bagnell, and Siddhartha Srinivasa. Shared autonomy via hindsight optimization. In *Proceedings of Robotics: Science and Systems*, 2015.

[15] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[16] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[17] Shashank Pathak, Erika Ábrahám, Nils Jansen, Armando Tacchella, and Joost-Pieter Katoen. A greedy approach for the efficient repair of stochastic models. In *NFM*, volume 9058 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2015.

[18] Constantin A Rothkopf and Dana H Ballard. Modular inverse reinforcement learning for visuomotor behavior. *Biological cybernetics*, 107(4):477–490, 2013.

[19] Pete Trautman. Assistive planning in complex, dynamic environments: a probabilistic approach. *CoRR*, abs/1506.06784, 2015.

[20] Pete Trautman. A unified approach to 3 basic challenges in shared autonomy. *CoRR*, abs/1508.01545, 2015.

[21] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[22] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.