Model Checking For Safe Navigation Among Humans

Sebastian Junges[1], Nils Jansen[2], Joost-Pieter Katoen[1],
Ufuk Topcu[3], Ruohan Zhang[3], Mary Hayhoe[3]

[1] RWTH Aachen University, Aachen, Germany
[2] Radboud University, Nijmegen, The Netherlands
[3] The University of Texas at Austin, Austin, TX, USA ⋆

Abstract. We investigate the use of probabilistic model checking to synthesise optimal strategies for autonomous systems that operate among uncontrollable agents such as humans. To formally assess such uncontrollable behaviour, we use models obtained from reinforcement learning. These behaviour models are, e.g., based on data collected in experiments in which humans execute dynamic tasks in a virtual environment. We first describe a method to translate such behaviour models into Markov decision processes (MDPs). The composition of these MDPs with models for (controllable) autonomous systems gives rise to stochastic games (SGs). MDPs and SGs are amenable to probabilistic model checking which enables the synthesis of strategies that provably adhere to formal specifications such as probabilistic temporal logic constraints. Experiments with a prototype provide (1) systematic insights on the credibility and the characteristics of behavioural models and (2) methods for automated synthesis of strategies satisfying guarantees on their required characteristics in the presence of humans.

1 Introduction

The control of autonomous agents like robots often necessitates to account for other, potentially uncontrollable, agents such as humans. Examples of such *partially controlled multi-agent systems* [1] include self-driving cars [2], autonomous trading agents in the stock market [3], and service robots in presence of humans [4]. Physical interaction with humans makes them safety-critical.

Dependability requirements in safety-critical autonomous systems call for guarantees beyond confidence intervals with soft bounds, such as statistically obtained by simulations or experiments. For example, a self-driving car has to safely operate among pedestrians and human-operated cars. Repeatedly running physical experiments is costly and may put humans at risk.

To address the need for certified bounds on safety requirements in autonomous systems, we synthesise strategies to move a controllable agent towards a goal while limiting the probability of crashing into other dynamic (and uncontrollable) agents, such as humans. We exemplify the synthesis by means of a

⋆ Supported by the CDZ project CAP (GZ 1023), the DFG RTG 2236 "UnRAVeL", and the NSF grants 1652113, 1651089, and 1550212.

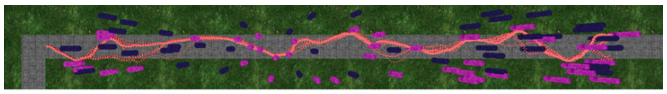


Fig. 1. A multi-task sidewalk environment in simulation. The objective is split into three modular tasks (i) approach targets (purple), (ii) avoid obstacles (blue), and (iii) follow walkway (grey).

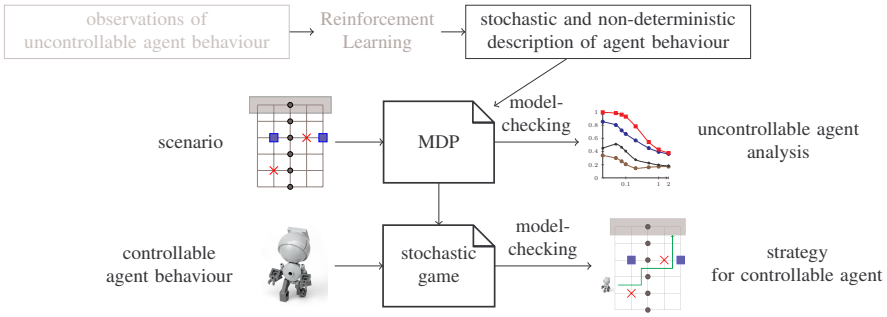


Fig. 2. The conceptual framework of our approach for the partially controlled multi-agent problem.

case study in which a human moves along a board walk, see Fig. 1. We extract a Markov Decision Process (MDP) [5] from an existing *behavioural model*, for instance of humans, obtained by *reinforcement learning* (RL) [6]. We refer to such a model for uncontrollable agents as an RL model. Roughly, states in the MDP are a Cartesian product of the state of the environment and the location of the agent. The stochastic transitions are based on observed experimental data, as computed by RL. Due to lack of observations in corner cases and abstraction of complicated scenarios, accurate stochastic descriptions of the behaviour are not always available, that is, the agent behaviour is *under-specified*. Instead of guessing, we explicitly model the agents' behaviour as non-deterministic in such situations. MDPs are a suitable modelling formalism, as they allow the co-existence of the stochastic and non-deterministic elements in the RL model.

Two key tasks are to (1) *analyse the learned behaviour of uncontrollable agents* and to (2) *synthesise a strategy for the controllable agents* that provably adheres to safety and/or performance specifications. We propose to use *probabilistic model checking* (PMC) for these tasks. PMC is a formal verification technique tailored to systems exhibiting randomness and uncertainty [7], and fully automated tools—e.g., `PRISM` [8] or `Storm` [9]—are readily available to provide an off-the-shelf solution. Figure 2 shows an overview of our approach.

First, given an RL behaviour model and an arbitrary fixed concrete scenario, we generate an MDP *automatically*. To address task (1), we phrase measures that assess the behaviour of the uncontrollable agent in PCTL, a probabilistic tempo-

ral logic [10]. For instance, we consider the minimum/maximum probability—depending on how the non-determinism is resolved—to not bump into obstacles, or the expected number of steps to reach a region in the environment. As a side product, PMC computes strategies which resolve non-determinism that obtain extremal probabilities. These strategies yield insights where the underspecification is critical. *PMC is thus provides quantifiable feedback about the behaviour of the uncontrollable agent, as modelled by the MDP for an RL model.*

To address task (2), we add a controllable agent which moves in the presence of the aforementioned uncontrollable agents. We describe the actions of the controllable agent, together with probabilities for possible uncertain outcomes of these actions, as an MDP. Composing such an MDP with the MDP for the RL models, yields a two-player *stochastic game* (SG) [11]. One player takes the actions of the controllable agent, and one player resolves the (non-deterministic) uncertainty in the behaviour of the uncontrollable agents.

PMC [12] *automatically synthesises strategies for the controllable agent in the SG that, if possible, provably adhere to temporal logic constraints, for any resolution of the non-determinism in the uncontrollable agents.* Such constraints may enforce the controllable agent to reach, with high probability, a certain area without a collision, or impose a bound on the expected number of steps. PMC supports such automatic strategy synthesis. In addition, PMC yields Pareto-curves to analyse the trade-off between several, potentially conflicting, measures.

A practical challenge is that the resulting MDPs and SGs are necessarily large in the context of real-world applications. Moreover, the data obtained from observations may induce models that do not exhibit the common features (e.g., structural symmetries) that usually help suppress the size of the MDPs and SGs. They also contain a large number of different probabilities. Consequently, off-the-shelf abstraction techniques employed in PMC tools [13] barely reduce the state space of such models. We alleviate this challenge by several optimisations in the encoding of the models, and adequately invoking available tools.

Contributions. This paper presents a framework to employ PMC for partially controlled multi-agent systems where a behaviour model for the uncontrollable agents is present and obtained by RL. The main technical contributions are: (1) a general recipe to cast behaviour models from RL into MDPs, (2) the use of PMC to obtain dependable quantitative insights of such models, (3) a technique to combine MDP models from controllable and uncontrollable agents into an SG, and (4) the synthesis of strategies by applying PMC on SGs. Either the resulting strategies are guaranteed to be safe and/or efficient, or the approach reports on the impossibility to synthesise such strategies.

Case study. We apply the proposed approach to an existing case study in human-robot interaction, a visiomotor multi-task model from [14,15]. Here, RL describes

quantitative aspects of human behaviour such as the likelihood to take an action in a certain situation. To account for the limitation in human cognition, global tasks are decomposed into modular sub-tasks. This modelling of multi-task behaviour is called *modular (inverse) RL* [15–18]. Behaviour in such a setting may be observed in mixed-reality environments and cast into a general behaviour model. The goal is to synthesise a *strategy* for a robot navigating safely, i. e., without interfering with the human, through the same environment.

Related work. PMC has been applied to autonomous robot systems, see e.g. [19–23]. Trade-off analysis has been advocated in [22, 24]. These applications all consider MDP models of robots. This paper instead considers MDP model checking of behaviour models obtained from RL. SGs have been used to model multi-agent learning problems [25, 26]. Finally, [27] uses statistical model checking to evaluate self-assembly strategies of autonomous systems. We reflect on the usage of statistical model checking for our setting in Section 6.

2 Preliminaries

Definition 1 (Probabilistic models). *A* stochastic game (SG) *is a tuple* $\mathfrak{M} = (S, s_I, Act, \mathcal{P})$ *with a finite set* S *of states with* $S = S_\circ \uplus S_\square$ *where players* $\circ$ *and* $\square$ *control the corresponding states, an initial state* $s_I \in S$*, a finite set* Act *of actions, and a transition function* $\mathcal{P}\colon S \times Act \times S \to [0,1]$ *with* $\sum_{s' \in S} \mathcal{P}(s, \alpha, s') \in \{0, 1\} \quad \forall s \in S, a \in Act$*. For each state* $s \in S$*, the set of* enabled *actions is* $Act(s) = \{\alpha \in Act \mid \exists s' \in S.\, \mathcal{P}(s, \alpha, s') \neq 0\}$.

- $\mathfrak{M}$ *is a* Markov decision process (MDP) *if* $S_\circ = \emptyset$ *or* $S_\square = \emptyset$.
- *MDP* $\mathfrak{M}$ *is a* Markov chain (MC) *if* $|Act(s)| = 1$ *for all* $s \in S$.

Players *nondeterministically* choose an action at their state; successor states are determined probabilistically according to transition probabilities. MDPs and MCs are one- and zero-player SGs, respectively. Probabilistic models are commonly specified in a guarded command language developed for `PRISM` [8].

Nondeterministic choices of actions in SGs are resolved by so-called *strategies*; resolving all nondeterminism yields *induced MCs*. In general, strategies (for each player) are functions $\sigma\colon S^* \to Distr(Act)$, such that finite paths of probabilistic models are mapped to distributions over actions. For many properties, however, simpler strategies suffice [28].

3 Partially controlled multi-agent setting

The considered setting consists of an environment with a set of (controllable or uncontrollable) agents. Here, we define an environment to be a 2D-grid with

features of different types from a set of types Tp which influence the agents' behaviour. One example for such a feature type is a (potentially moving) obstacle.

Definition 2 (Environment). *An* environment $\mathit{Env} = (\mathit{Loc}, \mathit{Feat})$ *consists of a finite set of* locations $\mathit{Loc} \in [0, \mathit{Grid}_x] \times [0, \mathit{Grid}_y]$ *with* $\mathit{Grid}_x, \mathit{Grid}_y \in \mathbb{N}$*, and a set of* features $\mathit{Feat} \subseteq Tp \times \mathit{Loc}$*. A* feature $f = (\mathit{tp}_f, \ell_f) \in \mathit{Feat}$ *consists of a* type *and a* (feature-)location*.*

An agent h is represented by its *position* $\mathsf{pos}_h = (\ell_h, \alpha_h)$, with *location* ℓ_h and an *orientation* $\alpha_h \in \mathbb{R}$, e.g., north (0) or south (π). The set of all positions is denoted Pos. Two agents crash, if they have the same location. Agent h starts in position init_h, and has an associated set M_h of *movements*[4]. A movement $m \in M_\mathrm{h}$ deterministically updates the position of agent h as described by a partial function $\mathsf{post}_m \colon \mathsf{Pos} \rightharpoonup \mathsf{Pos}$; m is defined in pos_h only if $\mathsf{post}_m(\mathsf{pos}_h)$ describes a *valid* position, i.e, a position on the grid. While moving, an agent can perceive *knowledge* (as described by a set of atomic propositions) about the features or other agents. The *state* s of agent h consists of its position and the perceived knowledge, $s = (\mathsf{pos}_h, \mathsf{know}_h)$. The set of all states is denoted S_h. Let $\mathsf{eff}_m \colon S_h \rightharpoonup S_h$ computes the successor state of taking movement m in a state by updating the position and the perceived knowledge, exemplified later.

Reinforcement Learning. Intuitively, RL lets a controllable agent *explore* its environment by sequential decision-making [6]. The objective is to (approximatively) optimize the expected reward for the agent in the underlying MDP or SG [25]. During the exploration, a strategy may be *unsafe* in the sense that it harms the agent or the environment. This shortcoming restricts the application of RL mainly to application areas where safety is not a concern and has triggered the particular direction of *safe RL* [29]. Most approaches rely on "tweaking" the reward functions such that a learning agent behaves in a desired, potentially safe, manner. As rewards are often specialized for particular environments, such reward engineering runs the risk of triggering negative side effects or hiding potential bugs [30]. Therefore, we separate the concerns by first using RL to obtain a general behaviour model which may then be analysed with respect to both expected rewards and safety in concrete scenarios.

Behaviour of uncontrollable agents. We assume an *estimation of the behaviour* of uncontrollable agents, derived from an existing RL model in two steps.

The first step determines a set $V(s)$ of *movement-value-vectors* for each state s. Each movement-value-vector $\boldsymbol{q} \in V(s)$ contains a *score* for every movement, i. e., $\boldsymbol{q} \colon M_\mathrm{h} \to \mathbb{R}_\infty$. Roughly, scores correspond to likelihoods for the movements, obtained from the RL model. Sometimes $|V(s)| > 1$, as either:

[4] We use the term movements to clarify the distinction from *actions* in MDPs.

– A specific situation may have never been accounted for. Because of this *lack of data*, the learned model is short of dependable estimations of specific movements scores. Or,
– the learned behaviour model is based on *relative information* regarding the features of the setting. For instance, the distance to an obstacle may influence the score of a movement. As the closest obstacle is not necessarily unique, several different scores for a state are possible.

The second step defines the stochastic behaviour of the agent by translating any movement-value-vector $\boldsymbol{q} \neq -\boldsymbol{\infty}$ to a *distribution over movements*. For the translation, we use a *softmax*-function $\mathbb{R}^{|M_{\mathrm{h}}|_\infty} \to [0,1]^{|M_{\mathrm{h}}|}$, which attributes most, but not all probability to the maximum [6]. The distribution is defined as:

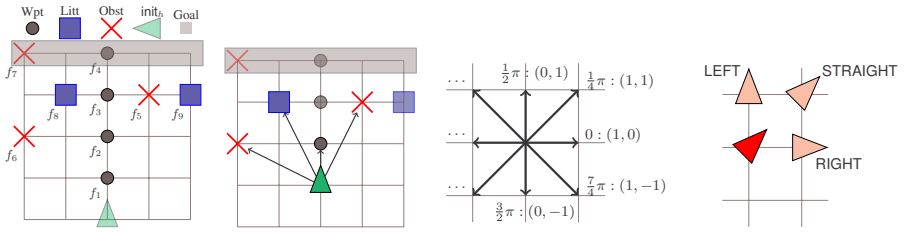
$$\mathsf{softmax}_\tau(\boldsymbol{q})_i = e^{\boldsymbol{q}_{i/\tau}} / \textstyle\sum_{i \leq |\boldsymbol{q}|} e^{\boldsymbol{q}_{i/\tau}} \qquad (\text{with } e^{-\infty} := 0, \text{ and } \tau > 0).$$

Invalid movements ($\boldsymbol{q}_i$) have zero probability due to the numerator. The parameter τ is called the *temperature*. For a temperature close to zero, the function yields a (hard) maximum; for a high temperature it yields an almost uniform distribution over the valid movements.

MDP for one uncontrollable agent. The behaviour can be captured in an MDP by using the position and the perceived knowledge as a state. The obtained distributions over movements yield the transition function. In particular, each movement-value-vector $\boldsymbol{q}$ corresponds to an action in the MDP. The successor states of a an action corresponding to $\boldsymbol{q}$ is determined by applying the associated movements, and the distribution is given by $\mathsf{softmax}_\tau(\boldsymbol{q})$ as described above. Due to the nondeterminism, the behaviour MDP can be seen as a partial strategy for uncontrollable agents provided by the behaviour model.

MDP for multiple (uncontrollable) agents. We assume that all agents move in alternating fashion. This abstraction yields a pessimistic over-approximation of the non-determinism over synchronous movements, and allows a straightforward mapping from movement-vectors to actions. The over-approximation compared to synchronous movements diminishes to less non-determinism the model exhibits. A state of the joint MDP is given by the Cartesean product, together with an entry which agent is moving. The actions and their effects are determined by the transition function of the MDP of the current agent, as in, e.g., [31].

SG for controllable and uncontrollable agents. We assume descriptions of the controllable agent are available, either via RL or directly modelled. The joint SG is similar to the joint MDP. States where a controllable agent is moving correspond to Player $\circ$, the uncontrollable non-determinism to $\Box$, respectively. The controllable agents move first, to prevent them from taking the upcoming move by $\Box$ into account. A strategy for the controllable agent adhering to a



(a) Toy-scenario (b) Example state (c) Direction vectors (d) Movement effects

Fig. 3. Grid worlds, orientations, and human movement.

given property is robust against both stochastic and nondeterministic actions of the uncontrollable agents. If the MDP for the uncontrollable agents does not contain any non-determinism, the joint SG collapses: all choices belong to the controllable player, therefore, the joint agent SG is an MDP.

Analysis. PMC supports a wide range of properties for MDPs and SGs to analyse [28]. We focus on determining strategies that induce maximal or minimal probabilities to safely reach target states and the expected cost until the target is reached. Richer specifications such as probabilistic temporal logic constraints or ω-regular properties are commonly reduced to such reachability objectives [28].

To analyse the MDP, we compute the minimal and maximal values for a property, depending on how non-determinism is resolved. A large difference between these values indicates that the uncertainty (underspecification) in the data is significantly affecting results. For the SG, we consider the best-case behaviour of the controllable agent. Often, the best-case behaviour for one property induces suboptimal behaviour with respect to another property. To illustrate these trade-offs, we compute Pareto-curves via multi-objective PMC [32]. The model, however, can be checked for other properties without adaptions.

4 Case Study

The specific example we study is a visiomotor multi-task model from [14, 15]. We describe the concrete adaption of this model towards a behaviour MDP and the inclusion of a controllable agent in line with the concepts of Section 3.

Scenario. We consider a scenario involving an uncontrollable human agent going along a sidewalk encountering *obstacles* and *litter*, see Fig. 1. The human is given three modular objectives: while FOLLOW a sidewalk (represented as a set of *waypoints*) to get to the other end, she should AVOID walking into obstacles and aim to COLLECT litter. Waypoints, obstacles and litter are the *features*. Features are initially present, but disappear when visited, run over, or collected.

We mark regions of the environment as *target areas*. A toy scenario is given in Fig. 3(a). The scenario is abstracted to a discrete-state model to characterise human behaviour in the model. Recall that the behaviour is determined by movement-values, which are translated into distributions over movements.

We specify both the environment and the human behaviour. Throughout the section, $\mathsf{Env} = (\mathsf{Loc}, \mathsf{Feat})$ is an environment with an arbitrarily sized grid and a finite set of features. Features have a type in $Tp = \{\text{Obst}, \text{Litt}, \text{Wpt}\}$.

Example 1. Consider Fig. 3(a). The environment is $\mathsf{Env} = (\mathsf{Loc}, \mathsf{Feat})$ with $\mathsf{Loc} = \{(x, y) \mid x \in [0, 4]\ y \in [0, 4]\}$, and

$$\begin{aligned}\mathsf{Feat} = \{f_i = (\text{Wpt}, (2, i)) \mid i \in \{1 \dots 4\}\ \} \\ \cup \{f_5 = (\text{Obst}, (3, 3)), f_6 = (\text{Obst}, (0, 2)), f_7 = (\text{Obst}, (0, 4))\} \\ \cup \{f_8 = (\text{Litt}, (1, 3)), f_9 = (\text{Litt}, (4, 3))\}\end{aligned}$$

The human's orientation has 8 possible values, $\alpha_h \in \mathsf{Orient} = \{i \cdot \frac{1}{4}\pi \mid i \in [0, 7]\}$, and is associated with a direction $\mathsf{Dir}\colon \mathsf{Orient} \to \{-1, 0, 1\}^2 \setminus \{(0, 0)\}$, see Fig. 3(c). The possible movements are $M_\text{h} = \{\mathsf{LEFT}, \mathsf{STRAIGHT}, \mathsf{RIGHT}\}$, we depict the associated position updates in Fig. 3(d). Formally, we have

$$\mathsf{post}_m(\mathsf{pos}_h) = (\ell_h + \mathsf{Dir}(\alpha_h + \beta_m), \quad (\alpha_h + \beta_m) \mod 2\pi)$$

for $\mathsf{pos}_h = (\ell_h, \alpha_h)$ and movement m with angle β_m, where $\beta_m \in \{-\frac{1}{4}\pi, 0, \frac{1}{4}\pi\}$. Thus, for some movement, we first update the position based on the current orientation, and then the orientation.

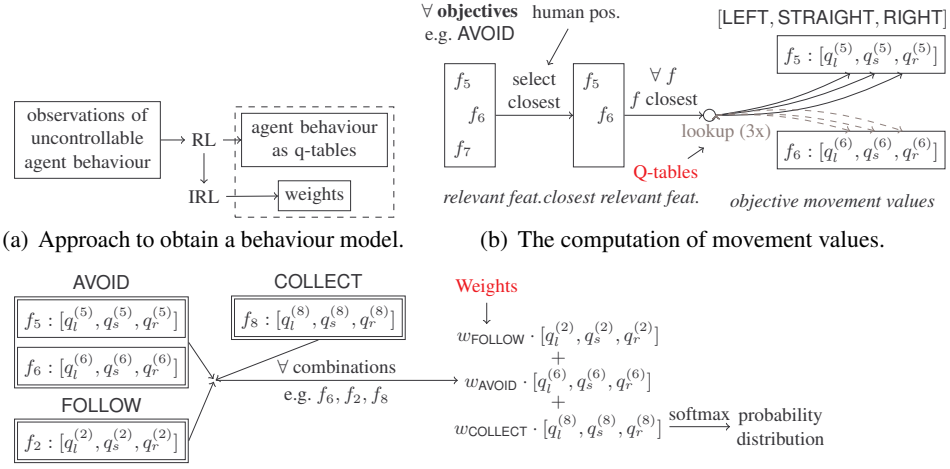
The *perceived knowledge* know_h concerns the presence of features, i. e., if they have not been collected or run over. Here, the set $\mathsf{know}_h = \mathsf{PFeat} \subseteq 2^{\mathsf{Feat}}$ contains the present features. Consequently, a state for a human is a tuple $(\mathsf{pos}_h, \mathsf{PFeat})$. Features that coincide with the updated human location disappear and are removed from the set of present features:

$$\mathsf{fu}_m(s) = \mathsf{PFeat} \setminus \{(\mathsf{tp}, \mathsf{post}_m(\mathsf{pos}_h))\} \subseteq \mathsf{Feat}.$$

Updating the position and the perceived knowledge yields the successor state.

Definition 3 (Successor state for human). *The successor state of movement m in state $s = (\mathit{pos}_h, \mathit{PFeat})$ is $\mathit{eff}_m(s) = (\mathit{post}_m(\mathit{pos}_h), \mathit{fu}_m(s))$.*

Human behaviour. We discuss how to obtain a distribution over different movements within a modular RL framework, where the weights are obtained via inverse RL. From [15], we obtain Q-tables and weights describing human behaviour, cf. 4(a). Fig. 4(b) outlines how to obtain values for the possible movements for each objective. Fig. 4(c) describes how to combine values for different



(a) Approach to obtain a behaviour model.

(b) The computation of movement values.

(c) The computation of distribution

Fig. 4. How to construct sets of distributions.

objectives and their translation into distributions. The estimation depends on the relative location with respect to some features. Details of the steps are below.

Distance and angle. The distance $d_h(f)$ between the human at location ℓ_h and a feature f at location ℓ is the Euclidean norm of their locations. $\theta_h(f)$ denotes the signed angle between the human orientation and $\ell_h - \ell$.

Relevant features. For each objective, only a subset of the features are relevant for the human behaviour in a state. First, for each objective o, we have exactly one *corresponding feature-type* $F(o) \in Tp$; Waypoints, obstacles and litter correspond to FOLLOW, AVOID and COLLECT, respectively. The set $\mathsf{RelFeat}_o(s) = \{(F(o), \ell) \in \mathsf{PFeat}\}$ contains *the relevant features for objective o in state $s = (\mathsf{pos}_h, \mathsf{PFeat})$*. Second, we adopt the assumption from [15] stating that for each objective o, only the closest feature of type $f(o)$ is relevant for the behaviour with respect to o.

Remark. This rather strong assumption reduces the number of hidden parameters a learning method has to estimate. The *closest relevant features* $\mathsf{Close}(s, o)$ for objective o in state s are:

$$\mathsf{Close}(s, o) = \{f \in \mathsf{RelFeat}_o(s) \mid \forall f' \in \mathsf{RelFeat}_o(s).\ d_h(f) \leq d_h(f')\}.$$

Example 2. Recall state s in Fig. 3(b). The closest relevant features are, e.g., $\mathsf{Close}(s, \mathsf{AVOID}) = \{f_5, f_6\}$, and $\mathsf{Close}(s, \mathsf{FOLLOW}) = \{f_2\}$.

As we observe from the example, the human behaviour model is under-specified: It is not clear which feature is relevant, i. e., the set $\mathsf{Close}(s, o)$ is not necessarily

a singleton, as several nearby objects may be located at similar distances. Any feature in the set may be the one that is being considered.

Movement-values. For each movement m and each objective o, we assume we are given a Q-table $Q_o^m \colon \mathbb{R} \times \mathbb{R}_+ \to \mathbb{R}$ which maps the angle $\theta_h(f)$ and distance $d_h(f)$ between a human and relevant feature f to an *objective-movement-value*, constructed via [15]. We map the closest relevant features into a set of objective-movement-vectors $V^o(s)$ by a lookup in the Q-table, and store the feature:

$$V^o(s) = \{(f, \boldsymbol{q} = [q_l, q_s, q_r]) \mid f \in \mathsf{Close}(s, o), q_l = Q_o^{\mathsf{LEFT}}(\theta_h(f), d_h(f)), \\ q_s = Q_o^{\mathsf{STRAIGHT}}(\theta_h(f), d_h(f)), q_r = Q_o^{\mathsf{RIGHT}}(\theta_h(f), d_h(f)) \} .$$

Vector entries collect movement-values for a fixed feature.

Combining movement values. A probability distribution over objectives is given as a *objective-weight vector* $w = [w_{\mathsf{AVOID}}, w_{\mathsf{COLLECT}}, w_{\mathsf{FOLLOW}}]$, e.g. obtained by modular IRL as in [15]. The objective-movement-vectors are translated into a set $V(s)$ of movement vectors $\boldsymbol{q}$ annotated with sets of relevant features F by calculating a weighted sum over all combinations and annotating them with the union of the features. For invalid movements, we set $-\infty$ as movement-value. If no movement is possible, we remove the vector from the movement values.

Distribution. The last step (cf. Fig. 4(c)) is to transfer movement values into a distribution over the movements.

4.1 Operational Model

Behaviour MDP. We automatically generate an MDP for the human behaviour. The human starts in its initial position and all features are present. Recall that the nondeterminism is caused by underspecification of the model, resulting in multiple movement vectors, and that each vector is reflected by an action.

Definition 4. *The MDP $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ reflecting the human behaviour starting in init_h on an environment $\mathsf{Env} = (\mathsf{Loc}, \mathsf{Feat})$ with temperature τ is $S = \{(\mathsf{pos}_h, P) \in (\mathsf{Loc} \times \mathsf{Orient}) \times 2^{\mathsf{Feat}}\}$ with $s_I = (\mathsf{init}_h, \mathsf{Feat})$, $Act = \mathsf{Feat} \times \mathsf{Feat} \times \mathsf{Feat}$ and*

$$\mathcal{P}(s, a, s') = \begin{cases} \mathsf{softmax}_\tau(\boldsymbol{q})_i & \text{if } (a, \boldsymbol{q}) \in V(s),\ s' = \mathsf{eff}_{(M_h)_i}(s) \text{ for an } i \\ 0 & \text{otherwise.} \end{cases}$$

Joint human-robot SG. As controllable agent, we consider a robot either turning 90-degrees (left or right) in place, or moving forward. The human-robot SG is the parallel composition of the MDP for the human and the MDP for the robot.

We consider two cases for the human behaviour in the presence of a robot. First, we assume the behaviour is not influenced by the robot, which yields a strategy for the robot not counting on the human to actively evade the robot. Second, we model the robot as an obstacle-feature: The human avoids the robot, and the robot strategy takes the human 'aversion' of the robot into account. For the latter case, we assume that the human treats the robot as a static obstacle.

Joint human-robot MDP. The SG collapses to an MDP if there is no non-determinism in the description of the uncontrollable agents. One way of *eliminating* underspecification is to create *unique* closest relevant features by selecting the one with the smallest absolute angle, and if tied, left-of-the-human over right. This abstraction allows to treat larger benchmarks on a less precise model.

Example 3. Eliminated underspecification yields for state s from Fig. 3(b), $\mathsf{Close}(s, \mathsf{AVOID}) = \{\ f_5\ \}$, as $|\theta_h(f_5)| \approx 26^\circ < |\theta_h(f_6)| \approx 63^\circ$.

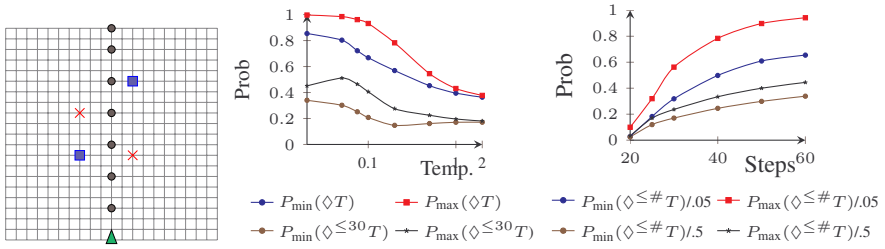
4.2 Experimental Setup

Our prototype realises the framework as in Fig. 2 using `PRISM-Games` [12] for evaluation of the human and synthesising strategies, and `storm` for trade-off analysis of multiple specifications. Results are obtained on a HP BL685C G7, using 48 cores, 2.0GHz each, and 192GB of RAM.

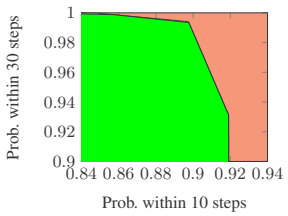
State and transition encoding. We encode the models using the `PRISM`-language, where *parallel composition* of modules yields concrete MDP or SG models[5]. We define MDPs describing human and robot behaviour in form of such modules; parallel composition yields the SG describing joint behaviour. Roughly, the two MDP modules are encoded as follows. A global flag indicates whether the human or the robot moves next; preconditions in both modules ensure a turn-based movement. The module for the human consists of 3 integers to represent the human position and flags b_f indicating the presence of feature f. The transition relation for each human position is listed explicitly. Although the number of reachable states is exponential in the number of features, the encoding is cubic as the behaviour depends on the nearest features only. The robot module defines any given model over the same environment as for the human. The SG state space is the product of the two modules, the size of the encoding equals their sum.

Optimizations. The encoding in the `PRISM` language enables using a variety of tools, but the *size of the encoding* grows rapidly with the size of the environment. In fact, the model cannot be represented as succinctly (up to 100K lines) as typical examples (up to 1K lines); therefore *parsing and model building* take significant time. We employed some performance improvements, among them:

[5] Available at `https://github.com/moves-rwth/human_factor_models`



(a) Environment (b) Different variances (c) Bounded steps



(d) Pareto curve

grid	feat.	states	act's
11×11	2/2/3	9.4E4	2.4E5
11×11	4/4/4	4.9E6	3.3E6
16×16	2/2/3	1.1E5	1.2E5
16×16	4/4/4	3.5E6	4.0E6
21×21	2/2/3	2.0E5	2.1E5
21×21	4/4/4	6.4E6	6.7E6
21×21	2/2/7	3.2E6	3.4E6

(e) MDP sizes

grid	feat.	states	act's
8×8	1/1/2	2.5E6	4.8E6
8×8	1/1/3	4.9E6	9.5E6
8×8	2/1/3	8.4E6	1.6E7
8×8	2/2/3	1.6E7	3.2E7
11×11	2/2/3	7.4E7	1.4E8
11×11	4/4/4	2.7E9	5.6E9
16×16	2/2/7	3.1E9	6.4E9

(f) Stochastic game sizes

Fig. 5. MDP analysis using PMC.

(1) Only the Q-table for obstacle avoidance shows equal values for the far-away columns—indicating that human behaviour does not consider far-away obstacles. It is not necessary to distinguish which obstacle is nearest if all induce the same score. (2) As every location in a concrete scenario occurs in a large number of commands, a symbolic substitution of similar positions reduces the overhead of specifying locations repeatedly. These improvements reduce the parsing time by 40%. It is important to choose the right tool configuration, in particular the engine and preprocessing.

5 Results

To give an indication on the sizes of the MDPs and SGs we handle, consider Table 5(e–f) presenting the grid size, the number of features (obstacles/litter/waypoints), the number of states, and global actions of the models using $\tau = 0.075$.

Evaluating the behaviour model. For the behaviour MDP, we compute *minimum and maximum* probabilities for reaching a target area. These probabilities depend on how the nondeterminism concerning underspecification is resolved. The gap between these probabilities indicates the actual relevance of underspecification. Moreover, we analyse the *number of steps* a human needs to reach the target area to get an indication if the behaviour is realistic. We discuss results for an MDP induced by a 20×20 grid with 2 pieces of litter, 2 obstacles, and 7 waypoints (2/2/7), see Fig. 5(a). The human is only told to follow the waypoints.

Relevance of underspecification. Fig. 5(b) plots the min/max probability to reach the target area against the temperature (controlling the variability in the softmax function). It shows that, with low variability, the fragment of executions indeed reaching the other side without leaving the grid is high. With higher variability (where features are mostly ignored), this number quickly drops.

Performance of the human. Fig. 5(c) indicates a similar behaviour for step-bounded reachability for different number of steps (x-axis) and temperatures 0.05 and 0.5. Most executions take more than 30 steps to reach the goal, indicating that, based on the given data, humans *very unlikely walk in straight lines*. This phenomenon occurs due to the lack of a *notion of progress* in visiting waypoints—it does not penalize walking in circles, as only positive reward is earned on visiting waypoints. The gap due to underspecification is significant as long as the variability is not too high. With low variability, most executions reach the goal within 60 steps. Detailed analyses considering the obtained strategies show where in the model underspecification has the largest effect.

Robot strategy synthesis. For robot strategies, we use PMC to maximise measures regarding *safety*, where the robot never occupies the same cell as the human, and *performance*, where a target area is reached within a certain number of steps. However, these objectives may conflict with each other: If the goal has to be reached in a few steps, there is not much time for precautions, and the probability of safety is affected negatively. In contrast, safer strategies reduce the performance. To obtain insights, we perform a *trade-off analysis*.

Analysing joint human-robot MDPs. On a 8×8 grid with 2/2/3 features, we considered two objectives: To reach the other side within 10 steps without crashing and to reach the other side within 30 steps without crashing. The first gives only few opportunities to evade the human, while the latter gives plenty opportunities. The two objectives thus may conflict with each other. Fig. 5(d) has been automatically generated with PMC. It shows in green combinations of objectives which can be achieved by some strategy. We see that optimising the first condition yields a strategy with 92% chance to satisfy the former condition and a 93% chance to satisfy the latter, while another strategy yields a 99% chance to satisfy the latter and a 90% chance to satisfy the former condition.

Scalability of analysing SGs. Using our aforementioned optimisations, we are able to analyse all SGs in Tab. 5(f). The symbolic engine of `PRISM-Games` finishes (model) building within 30 seconds for all these models, but (model) checking to find a good strategy takes significantly more time: between 270 and 4800 seconds for the 8x8 grids, respectively, and 70 hours for the smaller 11x11 grid. Using an explicit engine, checking is faster, but building suffers from the

large size of the encoding: For the 8x8 grids, checking requires between 300 and 2100 seconds, but building takes between 40 minutes and 20 hours, respectively. The smaller 11x11 grid takes up to 200 hours of building. Once a model is built, multiple properties can be checked on this model efficiently. Currently, only the explicit engine allows to actually extract an optimal strategy. Therefore, we only obtain the strategies for the 8x8 grids and the smaller 11x11 grid. For MDPs, the performance is superior on the sparse engine, which does model-building via the symbolic engine and checking via an explicit representation.

6 Discussion

The model. Based on the PMC results, we obtained five lessons about the weighted Q-table model. **(1)** According to the provided model, humans most likely walk in wavy lines. Following a line and giving a penalty for any diverging move (as in [15]) would provide a notion of progress. **(2)** As the Q-tables do not *take into account the border of the environment*, they are not avoiding a deadlock (or unspecified behaviour): The probability for leaving the grid is substantial in many cases. **(3)** For the discretised model we use, there is a potentially significant difference in behaviour based on how the underspecification is resolved; *any analysis on the learned model has to take this underspecification into account*. **(4)** Modelling variability over human behaviour by a single softmax and *using a memoryless model* are rough estimates. Therefore it is quite likely that the human model contains behaviours which contradict statements about the observed behaviour from e.g. [14]. **(5)** Finally, the Q-tables contain some unexpected *outliers* which in some configurations lead to unexpected behaviour.

The method. The description for the human behaviour including variability can be translated into a formal model; allowing a variety of properties to be easily analysed—in particular, it allows for analysing underspecifications and ill-defined data. The generalization to a robot planning scenario (and to a SG) is straightforward, and enables to compute plans fulfilling specific properties. Probabilistic verification of this model raised *five challenges*: **(1)** The *large number of different probabilities* in learned data blows up the encoding and the symbolic state-space representation, and prevents successful application reduction techniques such as bisimulation. It would be interesting to *regularise the model* on the learning side. **(2)** The softmax function serves the only purpose of introducing variability; its differentiability is not utilised here. *Sensitivity analysis over Q-table values would be of interest*. **(3)** Despite the lack of typical symmetries in the scenarios, its encoding is significantly smaller than enumerating all states, as (here) the behaviour only depends on the nearest obstacles. Even for three features of every kind, a reduction of a factor over 50 is possible. **(4)** While the approach yields

promising results for MDPs, the SG suffers from a state space explosion. The regularity of the turn-based game is exploited by a *symbolic engine*, but suffers from the large symbolic description, see (1). **(5)** Finally, strategies are computed for the full state space, whereas *we are often only interested in a fragment of the full state space*; for many states we can take any action.

Using simulation-based alternatives to PMC such as statistical model checking [33] bears the problem that it is unclear how to resolve two levels of nondeterminism. Moreover, the model we investigate in our case study contains probabilities which are rather small. Therefore, rare-event simulation has to be used, and will increase the simulation effort.

7 Conclusion

We have laid the framework for translating a (learned) behaviour model to a formal setting, and used it to compute control strategies for agents moving among uncontrollable agents handling complex tasks. We discussed a concrete model as well as several (open) challenges. For future work, we propose to investigate abstraction techniques such as [34] and restrict exploration of the model via [35].

References

1. Brafman, R.I., Tennenholtz, M.: On partially controlled multi-agent systems. Journal of Artificial Intelligence Research **4** (1996) 477–507
2. Dresner, K., Stone, P.: A multiagent approach to autonomous intersection management. Journal of Artificial Intelligence Research **31** (2008) 591–656
3. Wellman, M.P., Wurman, P.R., O'Malley, K., Bangera, R., Reeves, D., Walsh, W.E., et al.: Designing the market game for a trading agent competition. IEEE Internet Computing **5**(2) (2001) 43–51
4. Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V., et al.: Bwibots: A platform for bridging the gap between AI and human–robot interaction research. The International Journal of Robotics Research (2017)
5. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons (1994)
6. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
7. Kwiatkowska, M.Z.: Model checking for probability and time: from theory to practice. In: LICS, IEEE Computer Society (2003) 351
8. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. Volume 6806 of LNCS, Springer (2011) 585–591
9. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A STORM is coming: A modern probabilistic model checker. In: CAV. Volume 10427 of LNCS, Springer (2017) 592–600
10. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing **6**(5) (1994) 512–535
11. Condon, A.: The complexity of stochastic games. Inf. Comput. **96**(2) (1992) 203–224
12. Kwiatkowska, M., Parker, D., Wiltsche, C.: PRISM-Games 2.0: A tool for multi-objective strategy synthesis for stochastic games. In: TACAS. Volume 9636 of LNCS, Springer (2016) 560–566

13. Dean, T.L., Givan, R.: Model minimization in Markov decision processes. In: AAAI/IAAI, AAAI Press / The MIT Press (1997) 106–111
14. Tong, M.H., Zohar, O., Hayhoe, M.M.: Control of gaze while walking: Task structure, reward, and uncertainty. Journal of Vision **17**(1) (2017) 28–28
15. Rothkopf, C.A., Ballard, D.H.: Modular inverse reinforcement learning for visuomotor behaviour. Biological Cybernetics **107**(4) (2013) 477–490
16. Sprague, N., Ballard, D.: Multiple-goal reinforcement learning with modular sarsa (0). In: IJCAI. (2003) 1445–1447
17. Ballard, D.H., Kit, D., Rothkopf, C.A., Sullivan, B.: A hierarchical modular architecture for embodied cognition. Multisensory Research **26**(1-2) (2013) 177–204
18. Leong, Y.C., Radulescu, A., Daniel, R., DeWoskin, V., Niv, Y.: Dynamic interaction between reinforcement learning and attention in multidimensional environments. Neuron **93**(2) (2017) 451–463
19. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. Robotics and Autonomous Systems **60**(2) (2012) 199–213
20. Johnson, B., Kress-Gazit, H.: Analyzing and revising synthesized controllers for robots with sensing and actuation errors. Int. J. Robotics Res. **34**(6) (2015) 816–832
21. Giaquinta, R., Hoffmann, R., Ireland, M., Miller, A., Norman, G.: Strategy synthesis for autonomous agents using prism. In: NFM. LNCS, Springer (2018) To appear.
22. Chen, T., Kwiatkowska, M.Z., Simaitis, A., Wiltsche, C.: Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In: QEST. Volume 8054 of Lecture Notes in Computer Science, Springer (2013) 322–337
23. Feng, L., Wiltsche, C., Humphrey, L., Topcu, U.: Synthesis of human-in-the-loop control protocols for autonomous systems. IEEE Trans. Automation Science and Engineering **13**(2) (2016) 450–462
24. Lacerda, B., Parker, D., Hawes, N.: Optimal policy generation for partially satisfiable co-safe LTL specifications. In: IJCAI, AAAI Press (2015) 1587–1593
25. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: ICML. Volume 157 (1994) 157–163
26. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artificial Intelligence **136**(2) (2002) 215–250
27. Bruni, R., Corradini, A., Gadducci, F., Lafuente, A.L., Vandin, A.: Modelling and analyzing adaptive self-assembly strategies with maude. In: WRLA. Volume 7571 of LNCS, Springer (2012) 118–138
28. Katoen, J.P.: The probabilistic model checking landscape. In: LICS, ACM (2016) 31–45
29. Garcıa, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(1) (2015) 1437–1480
30. Sculley, D., Phillips, T., Ebner, D., Chaudhary, V., Young, M.: Machine learning: The high-interest credit card of technical debt. (2014)
31. Winterer, L., Junges, S., Wimmer, R., Jansen, N., Topcu, U., Katoen, J., Becker, B.: Motion planning under partial observability using game-based abstraction. In: CDC, IEEE (2017) 2201–2208
32. Etessami, K., Kwiatkowska, M.Z., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Logical Methods in Computer Science **4**(4) (2008)
33. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1) (2018) 6:1–6:39
34. Wachter, B., Zhang, L., Hermanns, H.: Probabilistic model checking modulo theories. In: QEST, IEEE CS (2007) 129–140
35. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Verification of Markov decision processes using learning algorithms. In: ATVA. Volume 8837 of LNCS, Springer (2014) 98–114