Model Learning for Adjusting the Level of Automation in HCPS

Mehrnoush Hajnorouzi   Astrid Rakow   Martin Fränzle

German Aerospace Center (DLR) e.V.   Carl von Ossietzky Universität Oldenburg
Oldenburg, Germany   Oldenburg, Germany

mehrnoush.hajnorouzi@dlr.de   astrid.rakow@dlr.de   martin.fraenzle@uni-oldenburg.de

The steadily increasing level of automation in human-centred systems demands rigorous design methods for analysing and controlling interactions between humans and automated components, especially in safety-critical applications. The variability of human behaviour poses particular challenges for formal verification and synthesis. We present a model-based framework that enables design-time exploration of safe shared-control strategies in human–automation systems. The approach combines active automata learning—to derive coarse, finite-state abstractions of human behaviour from simulations—with game-theoretic reactive synthesis to determine whether a controller can guarantee safety when interacting with these models. If no such strategy exists, the framework supports iterative refinement of the human model or adjustment of the automation's controllable actions. A driving case study, integrating automata learning with reactive synthesis in UPPAAL, illustrates the applicability of the framework on a simplified driving scenario and its potential for analysing shared-control strategies in human-centred cyber-physical systems.

1   Introduction

Formal verification and synthesis methods have achieved significant success for automated systems, where the dynamics and control logic of the system can be precisely modelled. However, this paper focuses on human-centred cyber-physical systems (HCPS) that operate under a shared-control paradigm [46]. In such systems, humans and automation jointly contribute to task execution (e.g. in advanced driver-assistance systems [33], robot-assisted surgery [45], or flight-control systems [21]), and the level of automation must be dynamically adapted according to the evolving system state and the human's behaviour. This poses a major modelling challenge, since the "human component" must also be represented in a form that can be analysed by formal methods.

Many shared-control HCPS are safety-critical and must satisfy stringent safety and performance requirements. A key challenge lies in managing the dynamic and bidirectional interaction between humans and automation, ensuring coordinated actions that maintain safety and efficiency despite behavioural uncertainty. This calls for control strategies that not only optimise the automation's capabilities but also account for the inherent unpredictability of human cognition and decision-making. Because human cognition is variable, context-dependent, and only partially observable, constructing models that support formal safety guarantees remains a major challenge [12, 13, 49].

This work aims to support the design of safety-critical shared-control HCPS. In their design, two central questions arise: *(Q1) "What does the automation need to know about the human component to interact appropriately?"* and *(Q2) "What must the automation do in order to interact appropriately?"*. Naturally, these questions are interrelated. *Consider a driver-support system that intervenes only to prevent collisions. If the automation can detect an inattentive driver early, a slight alert might suffice to avoid an imminent collision; but if the driver fails to respond, emergency braking must be triggered*
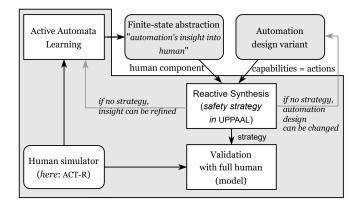
Figure 1: Overview of the approach: Does the insight into the human and the automation capabilities suffice to implement a safe shared-control HCPS?

to ensure safety. We provide a method that aligns the automation's capabilities with the controller's knowledge of the human.

Our approach (cf. Fig. 1) supports this design process through a design-time analysis framework. A human simulation engine—in this paper, the cognitive architecture ACT-R—generates observable human behaviour, which is abstracted via active automata learning into a finite-state model representing the automation's operational view of the human. Given this abstraction and a specific automation design, we apply game-theoretic reactive synthesis to determine whether the automation can guarantee safety and fulfill its mission goals. If no winning strategy is found, we either refine the human model or adjust the automation's capabilities. Since the strategy is synthesised with respect to the learned human abstraction and scenario mode, it is subsequently validated through co-simulation with the full cognitive model. When synthesis and validation both succeed, the derived models specify the level of human state observability and automation capability required for implementation.

Formal methods have been applied to human–automation interaction [12, 49], yet mostly rely on static or normative human models, limiting applicability in dynamic or adaptive contexts. Our framework addresses this limitation by integrating model learning and reactive synthesis to enable reasoning about shared-control with behaviourally derived human abstractions. We demonstrate the approach in a simplified driving scenario that combines active automata learning from a cognitive architecture with reactive synthesis in UPPAAL, to analyse safety properties under varying human behaviour. Rather than focusing on isolated control actions, this work addresses the formal analysis of continuous interaction over unbounded time horizons [4], characterised by dynamically evolving event sequences arising from the coupled dynamics of the human, automation, and environment.

The contributions of this paper are threefold: (1) a design-time framework that integrates model learning and reactive synthesis for analysing shared-control in HCPS; (2) a method for deriving finite-state human abstractions through interaction with a cognitive simulation; and (3) a preliminary evaluation in a driving scenario demonstrating the feasibility of synthesising safety-preserving automation strategies.

Structure of the article. Section 2 introduces the key design considerations for HCPS, Section 3 details the proposed framework. Section 4 presents a brief case study. Related work is reviewed in Section 5. Section 6 summarises the main findings, discusses limitations and outlines directions for future research. Finally, Section 7 concludes the paper.

2   Preliminaries

Figure 1 provides an overview of the proposed framework for analysing shared-control in HCPS. To facilitate understanding of the subsequent sections, this section introduces the main concepts and notations required to comprehend the overall approach. We briefly recall the notions of human–automation interaction, finite-state models, and reactive synthesis, as well as the concept of adjusting the level of automation, which together constitute the methodological foundation of the framework.

2.1   Human–Automation Interaction

Designing HCPS requires explicit consideration of the bidirectional feedback between automation and human operators: the automation's actions influence user behaviour, and human actions in turn affect the automation. To represent the human component of an HCPS, we employ computational models of human decision-making based on cognitive architectures. Architectures such as ACT-R [5] and CASCaS [31] provide general frameworks grounded in psychological theory [6,36] and have been validated empirically across diverse human-factors domains (e.g. [30, 43]). They generate discrete, time-stamped behavioural traces that can serve as observable input–output data for learning and analysis.

The ACT-R architecture models human behaviour as a perception–cognition–action loop operating over a sequence of cognitive cycles (typically 50 ms each). At every cycle, perceptual modules process sensory input from the simulated environment; the central procedural system selects a production rule that determines the next cognitive or motor action; and the resulting action updates both the environment and the internal state. Adjustable parameters such as memory decay, learning rate, and stochastic noise govern inter-individual variability and non-deterministic decision patterns. Consequently, ACT-R produces rich but discrete behavioural sequences that can be interpreted as a mapping between environmental stimuli and human responses.

In the context of this work, the simulated interactions yield traces of human behaviour that serve as the empirical basis for model inference. These traces are abstracted by the active automata-learning procedure into a finite-state representation of the behaviour that the automation *assumes* or *expects* from the human component during interaction. We refer to this internal predictive model simply as the *abstract human model* (*HM*). The *HM* thus embodies the automation's working hypothesis about human responses within the task context and forms the formal interface between cognitive simulation and reactive synthesis, enabling systematic exploration of shared-control strategies.

2.2   Reactive Strategy Synthesis and Safety Games

The objective of reactive synthesis is to automatically construct strategies that ensure system objectives while adapting to variations in human behaviour, rather than to pre-compute fixed action sequences. The interaction between the automation and the human is modelled as a finite-state (timed) two-player game, where the automation aims to maintain system safety and achieve mission goals despite uncertain or variable human actions. Such games are commonly referred to as *safety games*, in which a *winning strategy* [2] for the automation guarantees that no unsafe state can be reached under any admissible human behaviour. Desired system properties are expressed in temporal logics such as LTL or TCTL. In contrast to the traditional *implement-then-verify* workflow, reactive synthesis produces a *correct-by-construction* controller directly from these formal specifications.

Formally, the system is regarded as a discrete-event system [15], whose behaviour is represented as a temporally ordered sequence of events. Finite automata and their (ω-)regular languages provide a

natural formalism for representing and analysing such behaviours. Cognitive architectures such as ACT-R, however, are not expressed in an automata-based formalism. To obtain a representation suitable for synthesis, we employ *active automata learning*.

Active learning techniques based on Angluin's L* algorithm [7] iteratively query the system under learning (SUL), construct hypotheses, and refine them using counterexamples until convergence to a finite automaton that approximates the target behaviour. This approach is particularly suitable for modelling human behaviour: data obtained through passive observation are typically incomplete due to behavioural variability and context dependence, and rare yet safety-critical actions may be absent from logs. Active learning mitigates these limitations by interactively exploring the behavioural space and refining the hypothesis through counterexample analysis.

Active automata learning has seen numerous extensions in recent years, including optimised algorithms such as TTT [25] and NL* [10] that reduce the number of required queries, as well as probabilistic and non-deterministic variants [34, 37, 48] aimed at modelling uncertainty and variability in system behaviour. The development of tools such as AALPY [34] and continued research on applying and extending active model learning [9, 35] underline its ongoing relevance in both theory and practice. Its simplicity, transparency, and established convergence guarantees make it a natural choice for integration with cognitive simulations, while more efficient variants can be adopted within the same framework without conceptual modification.

2.3 Levels of Automation and Shared-Control

The degree of automation in human–machine systems varies depending on how control authority and decision-making are distributed between the human operator and the automated components. The widely adopted taxonomy SAE [42], defines six levels of vehicle automation, ranging from level 0 (no automation) to level 5 (full automation). At lower levels, automation offers assistance functions such as adaptive cruise control; intermediate levels enable partial automation that jointly controls acceleration and steering while requiring continuous human supervision; and higher levels allow conditional automation that manages all driving tasks within defined scenarios, returning control to the human when intervention is required. Across this spectrum, human involvement ranges from hands-on to mind-off interaction.

For such systems, particularly those operating at intermediate levels, the *shared-control* principle becomes essential: both human and automation contribute to task execution, dynamically balancing authority to ensure safety and performance. The automation must recognise when to intervene and when to defer to the human operator, adapting its behaviour to the situational context and the human's state. In our framework, levels of automation are instantiated as design variants, each defined by its controllable action set available to the automation. Through reactive synthesis, we evaluate whether a given variant can satisfy the control objectives while maintaining safety and accommodating the human behaviour. This enables systematic comparison of automation levels within a unified formal framework.

3 Iterative Model Learning and Control Synthesis

Our approach, illustrated in Fig. 1, integrates model learning with reactive strategy synthesis to enable adaptive automation in HCPS. The interaction between automation and human is formalised as a two-player game and control strategies are synthesised to guarantee mission objectives under all admissible human behaviours.

This section presents the methodological core: the construction of finite-state human abstractions
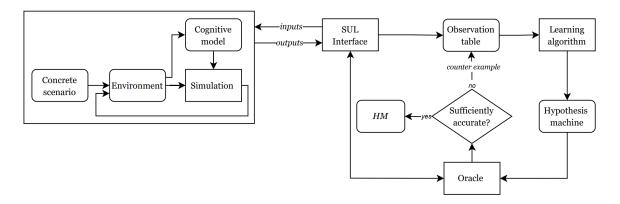
Figure 2: Learning behaviour abstraction automaton (*HM*) from simulation of cognitive model.

and the subsequent synthesis of control strategies. Active automata learning derives a finite-state *abstract human model* of the cognitive architecture, representing the information available to the automation about the human. The learned model, composed with the environment and the formalised objectives, serves as input to game-theoretic synthesis. The process proceeds in three main stages.

1. **Model learning**: construct a finite-state abstraction of the cognitive architecture within its operational context, capturing the information that the automation obtains about the human.

2. **Game construction**: combine the learned human abstraction with the environment and the control objectives, encoding them as winning conditions.

3. **Synthesis**: compute a winning strategy for the automation that satisfies the objectives against admissible human behaviours.

3.1   Generating the Abstract Human Model

Shared-control automation must adapt to variability and limitations of human cognition— intervening when human state may lead into unsafe situations and remaining otherwise unobtrusive. To formalise these conditions, we define abstractions of the human–machine interaction at an appropriate level of granularity and learn corresponding finite-state representations from cognitive-architecture simulations. The resulting automaton represents the automation's predictive model of human's behaviour within the task context.

The cognitive architectures are inherently state-based but structurally complex and concurrent. We use *finite game graphs* [47] as the formal basis. The abstraction is obtained via *active automata learning* based on Angluin's L* algorithm [7]. Learning begins from a coarse abstraction——formed over discrete observation intervals——and is refined iteratively until sufficient behavioural insight is achieved for the target scenario. Each iteration simulates the cognitive model within the task environment, records input–output traces, and updates the learned automaton (*HM*). This process yields a simplified yet behaviourally representative depiction of the human component (see Fig. 2).

The selection of observable parameters follows a pragmatic principle that prioritises variables with measurable correlation to the cognitive state. Typical examples include perceptual indicators such as facial expressions or gaze direction, which can serve as estimators of mental workload or attention focus (cf. [17, 22]). This facilitates the integration of the approach within application-specific domains. The key criteria are (i) observability of parameters and (ii) inference effectiveness for the underlying cognitive state.

Active automata learning incrementally constructs a hypothesis automaton by querying a *system under learning* (SUL) and refining the hypothesis based on counterexamples. The learner maintains an *observation table* comprising input prefixes and suffixes, and their observed outputs. For the table to define a valid hypothesis, it must satisfy closure and consistency conditions, ensuring that equivalent prefixes denote the same abstract state. Once these conditions are met, the learner constructs a hypothesis automaton whose states correspond to equivalence classes of prefixes exhibiting identical output behaviour over all suffixes. Subsequently, the learner issues equivalence queries to assess behavioural conformance with the SUL. In practice, equivalence is approximated through randomised testing, whereby a large set of input sequences is applied to both the hypothesis and the SUL. Any divergence yields a counterexample, which is incorporated into the observation table to refine the hypothesis. The iterative cycle of construction, testing, and refinement continues until no counterexamples are identified within the defined bounds.

We note that the expressive power of cognitive models generally exceeds that of regular languages; exact learning is therefore infeasible. The resulting automaton approximates the behavioural language of the cognitive model, analogous to abstractions learned for recurrent neural networks in related work [11]. Empirical validation across multiple cognitive architectures can be used to assess the adequacy of this finite-state approximation for the intended synthesis tasks. The proposed framework is not restricted to a specific cognitive architecture, such as ACT-R. Different architectures emphasise distinct aspects of human cognition—for instance, emotional–behavioural interaction in MAMID [24]—and may be selectively employed depending on the application focus. Combining or comparing multiple architectures further supports cross-validation of the learned abstractions and enhances the robustness of the resulting automation design. The learned abstraction *HM* thus serves as the finite-state human behaviour within the subsequent game-theoretic synthesis stage.

3.2   Game Graph Construction and Synthesis

Let CPS be a design variant of the automation defined by its controllable action set. The learned human model *HM* is combined with the cyber-physical components (CPS) and the environment models to form the HCPS game arena. The arena is represented as a *timed game automaton* (TGA) [32] $\mathscr{A} = \langle L, l_0, \Sigma_c, \Sigma_u, X, Inv, E \rangle$, where L denotes the set of locations, l_0 the initial location, X the set of clocks, and *Inv* assigns invariants to locations. The transitions E are labelled with actions from the disjoint sets of controllable actions Σ_c (automation) and uncontrollable actions Σ_u (human and environment). The synchronous product $\mathscr{A} = HM \parallel CPS$ defines the joint behaviour, interleaving independent actions and synchronising on shared ones.

During execution, each player chooses an action and an associated delay $t \in \mathbb{R}_{\geq 0}$. The opponent may pre-empt by selecting an enabled action with a shorter delay $t' \leq t$. The resulting interleaving defines the plays of the game. In the case study configuration, a fixed cognitive-cycle delay of 50 ms was adopted, corresponding to the standard temporal resolution in cognitive modelling.

The synthesis problem is to construct a control strategy that selects actions in Σ_c such that the control objective C is satisfied for all behaviours in Σ_u. Control objectives are expressed as winning conditions in a fragment of Timed Computation Tree Logic (TCTL) [14], typically including:

- *Safety*: undesirable states are never reached ($\forall\Box\neg bad$);

- *Reachability*: desirable states are eventually reached ($\forall\Diamond goal$);

- *Response*: whenever a trigger holds, a response eventually follows ($\forall\Box(trigger \Rightarrow \Diamond response)$).
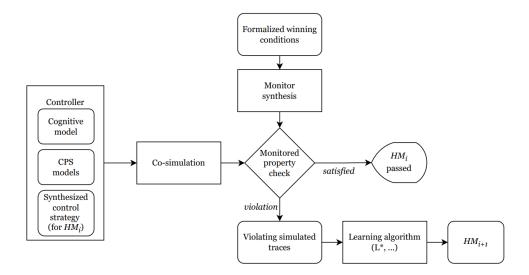
Figure 3: Framework to refine the learned human model *HM*.

These specifications are compiled into acceptance conditions and integrated into the game arena. Solving the game amounts to computing the set of winning states—those from which the controller can enforce the objective regardless of the uncontrolled system parts—via a fixed-point computation [32, 38]. If the initial state q_0 lies in this set, a winning strategy exists; otherwise, the specification is unrealisable.

A winning strategy is a mapping $\pi : Q \to (\Sigma_c \times \mathbb{R}_{\geq 0})$ assigning to each reachable state a controllable action and delay such that the successors remain within the winning region. This strategy constitutes a correct-by-construction controller that guarantees satisfaction of the specified objectives. In practice, we employ the UPPAAL TIGA tool [8] for synthesis and validation. The strategy guarantees that if the human behaves as captured by *HM*, the automation achieves its objectives. Since we coarsely abstracted the human, we validate and adapt the automation, if necessary. A successful synthesis yields an adaptive design [27] that explicitly accounts for the learned human model.

3.3 Refining the Human Model

Following synthesis, the correctness of the derived automation strategy is evaluated through co-simulation with the full human model. As illustrated in Fig. 3 our framework systematically validates the synthesised automation strategies: The full ACT-R model, the environment and the synthesised controller are composed and monitored to verify whether the control objectives are satisfied. Failure to meet these objectives indicates that the learned model *HM* may not yet capture sufficient behavioural fidelity. Since *HM* is inferred from a finite set of coarse traces, its abstraction may omit behaviours relevant to the satisfaction of the control objectives.

To address this limitation, the framework employs iterative refinement (Fig. 3), focusing on traces that lead to specification violations. The traces are fed back into the learning process to yield an updated model HM_{i+1} and a revised strategy CS_{i+1}. Each iteration thus extends the behavioural coverage of the human model and enhances the robustness of the synthesised controller.

Although convergence of this loop is not formally guaranteed, termination can be enforced through predefined criteria such as reaching a performance threshold or achieving stability across successive iterations. Future work will address the formulation of formal termination conditions and the integration of

the full refinement loop into case study evaluations, as this mechanism was not yet applied in the present study. As empirical behavioural data become available, additional refinement can be performed through parameter adaptation, improving the generalisability of the learned model across operator profiles.

3.4 Adapting the Automation Design Variant

If no winning strategy exists for a given design variant, the automation design is revised—typically by expanding the controllable action set—and synthesis is repeated. This iterative design exploration continues until a feasible strategy is identified or all design variants have been exhausted. Each synthesised strategy is validated through co-simulation as described above, completing the design-time loop of model learning, synthesis, and refinement. This process supports the systematic development of robust shared-control automation that explicitly accounts for human variability and task context.

4 Preliminary Evaluation through Case Study

We conducted a case study on a driving task to evaluate the feasibility of deriving reactive automation strategies based on a learned human model and to assess whether the obtained behavioural abstraction provide sufficient fidelity for safe shared-control. The study focuses on *cognitive-band actions* along the activity continuum [36]—decisions unfolding over time scales of a few seconds. In the driving context, these correspond to longitudinal control subtasks, such as adjusting acceleration to maintain a safe headway to a lead vehicle. The simulation setup consists of a single-lane road with a *lead vehicle* whose velocity varies over time and a *following vehicle* controlled by the driver model. The objective is to maintain a safe longitudinal distance from the lead vehicle while adapting to its speed fluctuations. This configuration provides a controlled environment for analysing shared-control strategies.

4.1 HM – the Driver Model

We implemented the *driver model* within the ACT-R architecture (using the *pyactr* implementation [1]), integrating goal, declarative, procedural, visual and manual modules. Its task is to select the longitudinal acceleration of the following vehicle based on the observed time headway (*thw*) to the lead vehicle, i.e. the longitudinal gap divided by the speed of the following vehicle. Specifically, the model evaluates the change in time headway (Δthw) over elapsed time (Δt), and computes the acceleration according to an adapted form of the Salvucci driver model [44]:

$$\Delta a = k_1 \Delta thw + k_2 (thw - thw_{follow}) \Delta t.$$

Here, k_1 penalises abrupt acceleration changes, while k_2 drives *thw* towards the desired value thw_{follow}. At each simulation step, (i) the environment updates vehicle positions and velocities; (ii) the current *thw* is computed and passed to the driver model; (iii) the model selects the acceleration for the next step; and (iv) the internal subgoal is updated to ensure behavioural continuity. This process establishes a closed-loop interaction between the driver model and its environment (Fig. 4). The simulation generates temporal traces of states and actions $(s_0, a_1, s_1, a_2, ..., s_n)$, where states encode cognitive variables such as buffer contents and goals, and actions correspond to perceptual updates, rule firings, or motor responses.

Scope and rationale. The generation of the cognitive model itself is not part of the proposed framework; instead, we modified and used a validated ACT-R-based driver model. The cognitive mechanisms
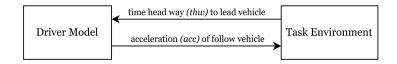
Figure 4: Interaction between driver model and driving environment.

Parameter	Description	Type of parameter
Declared chunks and procedural rules	Internal logic and rule-based system of ACT-R driver model	Internal
k_1, k_2	Coefficients of Eq. 1 tuning driver model behaviour	Internal
thw_follow	Desired temporal gap the driver aims to maintain	Internal
Reaction latency	Delay due to ACT-R processing	Internal
Vehicle dynamics	Update equations for vehicle velocity and position	Task-specific
destination_pos	Target position of follow vehicle	Task-specific
Init_conditions	Initial positions and velocities of two vehicles	Task-specific
lead_profile(t)	Time-varying speed/acceleration profile of the lead vehicle	Task-specific
thw	current time headway	Simulation
Δthw	Change in *thw* compared to previous step	Simulation
Δt	Simulation time step size	Simulation
acc_follow	Acceleration decided by driver model	Output
simulation-trace	Sequence of activated modules, fired production rules and executed actions	Output

Table 1: List of identified parameters.

and empirically grounded representation of decision-making at the cognitive-band level make it a suitable proxy for human control behaviour in the present driving scenario. This allows the evaluation to focus on the subsequent learning and synthesis steps rather than on model construction.

4.2   Learning the Human Behaviour Abstraction

We treat the ACT-R driver model as a grey-box *system under learning* (SUL) and infer a finite-state *HM* using active automata learning. The learned *HM* is a finite Mealy machine to represent the automation's operational view of admissible human input–output behaviour in the given task context.

What is identified.   During simulation, a set of tuneable parameters and observable variables are exposed to the learner, grouped into four categories (Table 1): (i) *Internal parameters*—model-intrinsic variables such as ACT-R production rules, k_1, k_2, thw_{follow} and reaction latency, which shape the underlying *HM* dynamics; (ii) *Task-specific parameters*—scenario-defining elements such as the lead vehicle profile, initial conditions, and destination, which determine contextual stimuli; (iii) *Simulation parameters*—measured variables and timing quantities (e.g. *thw*, Δthw, Δt), forming the input alphabet (Σ) and time context; (iv) *Outputs*—selected accelerations and relevant cognitive-trace features, constituting the output alphabet (Γ). This mapping enables the learner to capture both observable task behaviour and salient internal reasoning features yielding an *HM* that reflects the driver's decision process in its operational context.

Learning configuration.   We define the input alphabet Σ as discrete environmental stimuli delivered to the driver model, such as quantised levels of *thw*. The output alphabet Γ comprises tuples combining the decided acceleration and compact representations of the cognitive trace (e.g. sequences of fired production rules).

Each input stimulus triggers internal deliberation that culminates in an output action; the resulting input–output observations populate the L* observation table, where prefixes form the rows, suffixes the columns, and output tuples the cell entries. An illustrative excerpt is provided in Table 2. For instance, the last row and first column indicate that, given the input sequence $(1, 1, 2, 1)$ and query 1, the learner receives the output *(('attend','read','encode','n_ret'),-2)*. This corresponds to the sequential firing of the four production rules mentioned, whose combined execution results in a decided acceleration of -2.

Rational for active learning. Active automata learning offers a query-driven, sample-efficient approach for identifying finite-state models [7], supported by mature methodology and tooling [23, 34, 48]. Unlike purely passive, log-based methods, it can deliberately explore rare or safety-critical behaviours that may not appear in empirical data—an essential property for HCPS safety analysis. Refinements such as TTT [25] and NL* [10] improve query efficiency and model succinctness, while recent studies demonstrate continued applicability in complex software settings [35]. We therefore adopt an active setup to derive an *HM* that is both behaviourally discriminating and synthesis-ready.

Implementation details. The active learning process was implemented using AALpy [34] employing a deterministic L* learner in combination with a `RandomWalkEqOracle` to approximate equivalence queries. The oracle's reset probability parameter controlled the frequency of simulation restarts during exploration. The resulting *HM* is a deterministic Mealy machine, in which outputs depend on the current state and the provided input. For the configuration with four discretised time-headway levels, the learned *HM* comprised 13 states and 52 transitions.

4.3 Shared-Control Automaton

To operationalise the synthesised strategy, we implement the automation component as a *supervisory automaton* that monitors safety-relevant variables and dynamically regulates control authority. The automaton acts over the *HM*, enforcing safety constraints while maintaining a cooperative shared-control interaction.

The controller monitors the time headway (*thw*) and time to collision (*ttc*), defined respectively as the ratio of inter-vehicle distance to the follower's velocity and to their relative velocity. Both quantities are required to remain above specified thresholds (e.g. *thw* ≥ 1.5s, *ttc* ≥ 2s). At each step, the controller evaluates short-horizon predictions (one or two steps ahead) to assess whether the driver's selected acceleration might violate these thresholds, indicating increased collision risk. Based on this assessment, the controller switches between three modes:

- Nominal (no hazard): the driver's acceleration is accepted without modification;

Prefixes/E set	(1,)	(2,)
()	(('attend','read','encode','retrieve','decide'), 0)	(('attend','read','encode','retrieve','decide'), 0)
(1,)	(('attend','read','encode','n_ret'),0)	(('attend','read','encode','retrieve','decide'),2)
(2,)	(('attend','read','encode','retrieve','decide'),-3)	(('attend','read','encode','n_ret'),0)
(1,2)	(('attend','read','encode','retrieve','decide'),2)	(('attend','read','encode','n_ret'),2)
(2,1)	(('attend','read','encode','n_ret'),-3)	(('attend','read','encode','retrieve','decide'),-1)
(1,2,1)	(('attend','read','encode','n_ret'),2)	None
(2,1,2)	None	(('attend','read','encode','n_ret'),-1)
(1,1,2)	(('attend','read','encode','retrieve','decide'),-2)	(('attend','read','encode','n_ret'),1)
(1,1,2,1)	(('attend','read','encode','n_ret'),-2)	(('attend','read','encode','retrieve','decide'),0)

Table 2: Example of observation table.

- Advisory (low risk): the controller issues an alert signal ("hint") prompting the driver model to re-evaluate the current *thw* before finalising its decision;

- Intervention (high risk): if risk exceeds critical limits, the controller overrides or modifies the driver's selected acceleration, enforcing emergency braking. This represents a transition from advisory support to direct intervention.

Transitions between these modes are guarded by hazard predicates:

$$RiskWarn = (thw < thw_{warn}) \vee (ttc < ttc_{warn}),$$
$$RiskFilter = (thw < thw_{min}) \vee (ttc < ttc_{min}),$$
$$SafeNow = (thw \geq thw_{safe} \wedge (ttc \geq ttc_{safe}).$$

This defines a compact three-mode shared-control policy with recovery transitions (e.g. *Recover-From-Advisory*, *Recover-From-Intervention*) returning the system to lower modes once hazards resolve.

The resulting automaton provides a concrete realisation of the shared-control paradigm explored in this work: it embodies predictive risk awareness, selective intervention, and preservation of human agency. In the broader context of HCPS design, it serves as a design blueprint for instantiating formally synthesised strategies as interpretable supervisory control logic.

4.4 Game-Theoretic Synthesis of Shared-Control

The final step integrates the learned *HM* and the automation's supervisory logic into a unified game-theoretic synthesis framework. The synthesis process formally derives a reactive control strategy that guarantees satisfaction of the specified objectives under all admissible human and environmental behaviours.

The game arena is constructed as the synchronous product of the component automata:

$$\mathscr{A} = Driver \parallel Lead \parallel Follow \parallel SensorError \parallel SimCtrl \parallel Control.$$

Each automaton represents a constituent of the HCPS, contributing its own states, variables, and transition dynamics.

- *Driver*: Derived from the learned Mealy-machine abstraction of the cognitive model and translated into a timed automaton. A Python-based tool extracts states and transitions, while timing guards and invariants encode the 50 ms cognitive-cycle constraint (cf. Sect. 2.1).

- *Lead and Follow*: Represent vehicle dynamics; the lead vehicle follows a predefined motion profile, while the following acceleration is governed by the driver model or overridden by the Control automaton during intervention.

- *SensorError*: Introduces stochastic perturbations to the perceived *thw* to represent sensor and perception noise.

- *SimCtrl*: Serves as a global scheduler, maintaining consistent timing semantics across all components.

- *Control*: Encodes the supervisory shared-control automaton with three modes—*Nominal*, *Advisory*, and *Intervention*—encoding the automation's decision logic for issuing hints or enforcing overrides.

A global system state of $\mathscr{A}$ is defined as

$$q = (l_{Driver}, l_{Lead}, l_{Follow}, l_{SensorError}, l_{SimCtrl}, l_{Control}, v),$$

where l_i denotes the active location of component i and v is the joint valuation of clocks and continuous variables (e.g. positions, velocities, accelerations). Transitions of the *Control* automaton correspond to controllable actions Σ_c, while all other components contribute uncontrollable actions Σ_u, representing human behaviour, environment dynamics, and exogenous disturbances.

Specifications and synthesis. The control objectives are formulated as winning conditions expressed in a fragment of TCTL:

- Safety: the follower must never overtake the lead, $\forall\square(follow\_pos < lead\_pos)$;

- Reachability: the follower eventually reaches the destination, $\exists\lozenge(follow\_pos \geq DEST)$;

- Minimal intervention: the control automaton remains in Nominal or Advisory unless a hazard is detected $\forall\square((safe \vee low\text{-}risk) \rightarrow \neg Intervention)$.

These objectives jointly yield the weak-until condition:

$$\mathtt{control} : A[\neg(follow\_pos \geq lead\_pos) \ W \ (follow\_pos \geq DEST)]$$

requiring that the follower never overtakes the leader until the destination is reached.

We employ the UPPAAL model checker [8] (v5.0.0) to solve the game. If the objectives are satisfiable, the tool synthesises a memoryless strategy $\pi : Q \rightarrow \Sigma_c$, mapping each reachable state to a controllable action. The resulting strategy instantiates the three-mode policy:

$$\pi(q) = \begin{cases} \textit{Nominal:} & \neg RiskWarn \wedge \neg RiskFilter \Rightarrow follow\_acc = driver\_acc, \\ \textit{Advisory:} & RiskWarn \wedge \neg RiskFilter \Rightarrow \mathtt{hint!}, \\ \textit{Intervention:} & RiskFilter \Rightarrow acc\_floor = -3, acc\_cap = -1. \end{cases}$$

Simulation traces confirm the intended behaviour: In *Nominal* mode, the driver retains full authority and the invariant $follow\_pos < lead\_pos$ is preserved. Under *Advisory*, the controller issues a $\mathtt{hint!}$ message, prompting the driver to reassess the situation. *Intervention* occurs only under critical conditions (e.g. near-collision under sensor noise), enforcing braking until safety is restored. Representative trace segments in Table 3 demonstrate that the synthesised strategy satisfies the weak-until property, maintains safe distance, ensures progress to the destination, and minimises unnecessary overrides.

This synthesis step demonstrates how the learned human abstraction, supervisory control logic, and safety-game formulation combine to realise a correct-by-construction shared-control policy for HCPS.

lead_pos	follow_pos	*thw*	control_mode	driver_acc	follow_acc (applied)	Controller action
172	158	3	Intervention	2	−1 (clamped)	Brake override
180	165	2	Nominal → Advisory	−3	1	Hint issued
188	175	2	Nominal	−1	−1	Safe, no action
204	182	4	Nominal	−2	1	Safe, no action
204	189	0	Intervention	2	−1	Brake override
212	200	1	Advisory → Intervention	0	−1	Brake override
220	198	2	Nominal	0	1	Safe, no action

Table 3: Exemplary fragment of simulation trace supervised by the synthesised strategy.

5   Related Work

While the technical aspects of automation within cyber-physical systems (CPS) have been extensively investigated, the impact on human operators remains comparatively unexplored [29]. Recent efforts toward reliable HCPS aim to integrate models of human behaviour with CPS models, extending formal analysis and synthesis to encompass human–machine interaction. Human-in-the-loop control synthesis has been studied, for instance, by [19], in the context of conditional driving automation (SAE Level 3), where advisory controllers are synthesised to mediate authority transfer between driver and automation.

Bridging formal methods and cognitive modelling remains at an early stage. Most existing work adopts automata- or Markov-based abstractions of human behaviour to enable verification or strategy synthesis in human–CPS interaction (e.g. [16, 41]). While computationally tractable, these abstractions lack the psychological grounding of cognitive architectures, such as ACT-R [5], SOAR [40], or CASCaS by [50], which explicitly model perception, decision-making, and learning processes. Incorporating such architectures into formal frameworks represents a promising but still underdeveloped direction toward cognitively grounded and explainable synthesis of shared-control automation.

Initial steps towards bridging this gap have been taken by [28] and [20]. Both groups pioneered translations of fragments of cognitive architectures into formal models, thereby providing effective reductions of correctness problems of cognitive architectures into model-checking or constraint-solving problems for which effective tool support exists. Langenfeld et al. [28] mapped ACT-R mechanisms into network of timed automata [3], enabling automated defect analysis of programmed models. While precise, such encoding often leads to significant computational costs when applied to control synthesis. Current synthesis algorithms do not appear capable of handling timed automata translations of full-fledged ACT-R models, due to both computational scalability limitations and the lack of expressiveness for probabilistic transition selection inherent to ACT-R models. The optimized encoding may alleviate the scalability issues, but the application of less heavy-weight strategy synthesis algorithms like reinforcement learning trade off scalability with weaker formal guarantees.

The cognitive architectures have also served as proxies for human behaviour, particularly in the design of driver-assistance systems (e.g. [50]), and automated risk analysis (e.g. [39]). However, their potential for automatic synthesis of reactive strategies remains largely unexplored. Such a procedure would allow automation to anticipate human behaviour to generate a strategy to fulfil the technical subsystems objectives. It follows model-predictive control paradigm to account for the expected states and behaviour of human as reflected in the cognitive model.

One of the key challenges is state estimation. As discussed in our previous work, the existing control synthesis approaches relying on the synthesis methodology of [32] assume full observability of the underlying timed game. The strategy is synthesised depending on the entire vector of cognitive states which is unrealistic. In practice, human cognitive states are largely not observable and can only be inferred indirectly to a limited extent through correlations with (often weak) neurophysiological signals or behavioural cues. Practical controllers must therefore remain robust under partial observability, reconstructing latent human states from measurable indicators while tolerating uncertainty.

Approaches addressing scalability and limited observability include abstraction-based methods. Ehler et al. [18] proposed abstract timed games that group behaviourally equivalent states with respect to the analysed properties to mitigate state-space explosion. The key idea is to construct high-level representation of relevant behaviour and iterative refinement to increase accuracy and preserve correctness. Similarly, [26] integrated automata learning and statistical model checking to derive user-centred strategies in interactive systems.. Their case study on music streaming service demonstrates how learned user models can be used to guide interaction strategies without restricting the user's freedom of choice. The authors

of [9] developed probabilistic models of human driver behaviour using a hierarchical learning approach that incorporates known system modes. Their model captures highway driving responses to contextual factors such as speed limits and curvature, accurately reproducing observed patterns and generating realistic simulation data. However, these models remain descriptive rather than explanatory. Our approach models behaviour through perception, decision-making, and action processes grounded in psychological theory. Cognitive architectures offer explainability and enabling analysis of why drivers act as they do, though they may be less precise in continuous control reproduction. For evaluating shared-control paradigms in HCPS, such cognitively grounded models are preferable to purely data-driven approaches, as they capture goals and situational awareness.

Our implemented case study illustrates the feasibility of combining human-behaviour learning and reactive synthesis within a shared-control scenario. While the results are encouraging, several aspects of the framework merit further reflection regarding methodological choices, assumptions, and applicability to larger systems. We therefore discuss these points in more detail in the following section.

6   Discussion and Future Work

The proposed framework demonstrates how active automata learning and reactive synthesis can be jointly applied to analyse shared-control in HCPS. While the feasibility study confirms the conceptual soundness of the approach, several aspects deserve further refinement, and extension, particularly regarding abstraction design, timing representation, and empirical validation.

Choice of learning algorithm.   We adopted Angluin's L* algorithm owing to its simplicity, transparency, and well-understood convergence guarantees. Although more recent variants such as TTT or NL* improve query efficiency, they remain conceptually based on L* and are already supported by existing learning libraries such as AALPY. The framework can therefore incorporate these variants without conceptual modification.

Timing and observability assumptions.   The learned Mealy machine (*HM*) captures the discrete decision logic of the ACT-R model, whereas timing aspects are currently introduced through manual annotation. Transitions are enriched with timing parameters derived from ACT-R 's internal processing cycle and observed reaction latencies, ensuring consistency with the cognitive architecture's temporal dynamics. Future work will investigate automatic inference of timing parameters, for example through timed-model learning or statistical estimation of timing distributions from simulation traces.

Abstraction design and human-state observation.   An important consideration concerns how the full cognitive model is abstracted into a form usable for synthesis. The abstract human model (*HM*) represents the controller's belief about the operator's behaviour; its construction must therefore align with the observability of human states. In practice, the controller continually compares predicted and observed human actions to assess whether its abstraction remains accurate. This implies concrete requirements for sensor design and perceptual inference, potentially involving physiological and behavioural indicators such as facial expressions or gaze direction. The feasible level of abstraction must thus be co-designed with the sensing modalities and computational resources available. Future work will explore systematic methods for linking abstraction refinement to observability, enabling the automation to adapt its predictive model using measurable human cues.

Refinement of the human model.   In the current framework, refinement of the human model is triggered by specification violations detected during co-simulation. Although this iterative process improves behavioural coverage, formal termination criteria remain to be defined. Future work will address the formalisation of convergence criteria, potentially using stability measures or bounded performance metrics. Moreover, probabilistic or non-deterministic learning algorithms could capture inter-individual variability and uncertainty, strengthening the robustness of the learned abstractions.

Design variants and levels of automation.   The synthesis process supports systematic exploration of automation levels and design variants. Extending this analysis to more complex interaction settings, such as multimodal human–machine interfaces, cooperative robotics, or higher SAE automation levels, will enable quantitative assessment of trade-offs between safety guarantees, computational scalability, and human engagement. A promising direction is the comparison of design variants under equivalent formal objectives to identify architectures that maximise safety while preserving human agency.

Integration and empirical validation.   Integration of the complete toolchain, from cognitive simulation (ACT-R) through model learning (AALPY) to synthesis (UPPAAL), remains a technical challenge. Automating this pipeline would enable large-scale evaluation across domains and facilitate reproducibility. Ultimately, empirical human-in-the-loop validation is required to assess how well the learned abstractions generalise to real human behaviour and whether the synthesised strategies remain effective under real variability.

Outlook.   Overall, the framework establishes a principled foundation for bridging cognitive modelling and formal synthesis in shared-control systems. Future extensions will aim to close the loop between abstraction, observability, and synthesis—allowing the automation to refine its human model dynamically as behavioural evidence accumulates. By grounding abstraction refinement in measurable human states, the framework may evolve towards adaptive, co-regulative automation that maintains safety while remaining sensitive to human intent.

7   Conclusion

This work demonstrates that combining automata learning of cognitive models with reactive synthesis provides a rigorous foundation for adaptive automation in HCPS. By learning finite-state abstractions of cognitive architectures and integrating them with cyber-physical components in a game-theoretic synthesis framework, the proposed approach enables systematic, correct-by-construction reasoning about shared control and safety in dynamic human–automation interaction. A key contribution is the iterative learning–synthesis–validation loop, which refines both the human abstraction and the control strategy through counterexamples obtained from co-simulation, thereby establishing a principled link between cognitive modelling and formal verification for realising verifiable, adaptive shared control.

References

[1] Available at https://colab.research.google.com/github/abrsvn/pyactr-book/blob/master/notebooks/2_pyactr_implementation_of_ACTR.ipynb#scrollTo=DnNa6als9IC2.

[2] Rajeev Alur (2004): *Games for Formal Design and Verification of Reactive Systems*. In Farn Wang, editor: *Automated Technology for Verification and Analysis: Second International Conference, ATVA 2004, Taipei, Taiwan, ROC, October 31-November 3, 2004. Proceedings, Lecture Notes in Computer Science* 3299, Springer, p. 1, doi:10.1007/978-3-540-30476-0_1.

[3] Rajeev Alur & David L. Dill (1994): *A theory of timed automata*. *Theoretical Computer Science* 126(2), pp. 183–235, doi:10.1016/0304-3975(94)90010-8.

[4] John R. Anderson, Shawn Betts, Daniel Bothell, Cvetomir M. Dimov & Jon M. Fincham (2024): *Tracking the Cognitive Band in an Open-Ended Task*. *Cognitive Science* 48(5), p. e13454, doi:10.1111/cogs.13454.

[5] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere & Yulin Qin (2004): *An integrated theory of the mind*. *Psychological review* 111 4, pp. 1036–60, doi:10.1037/0033-295X.111.4.1036.

[6] John R. Anderson & Christian Lebiere (1998): *The Atomic Components of Thought*. Psychology Press, doi:10.4324/9781315805696.

[7] Dana Angluin (1987): *Learning regular sets from queries and counterexamples*. *Information and computation* 75(2), pp. 87–106, doi:10.1016/0890-5401(87)90052-6.

[8] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen & Didier Lime (2007): *UPPAAL-Tiga: Time for Playing Games!* In Werner Damm & Holger Hermanns, editors: *Computer Aided Verification*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 121–125, doi:10.1007/978-3-540-73368-3_14.

[9] Benjamin von Berg, Bernhard K. Aichernig, Maximilian Rindler, Darko Štern & Martin Tappler (2025): *Hierarchical Learning of Generative Automaton Models from Sequential Data*. In Alexandre Madeira & Alexander Knapp, editors: *Software Engineering and Formal Methods*, Springer Nature Switzerland, Cham, pp. 215–233, doi:10.1007/978-3-031-77382-2_13.

[10] Benedikt Bollig, Peter Habermehl, Carsten Kern & Martin Leucker (2009): *Angluin-style learning of NFA*. In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 1004–1009, doi:10.5555/1661445.1661605.

[11] Benedikt Bollig, Martin Leucker & Daniel Neider (2022): *A Survey of Model Learning Techniques for Recurrent Neural Networks*, pp. 81–97. doi:10.1007/978-3-031-15629-8_5.

[12] Matthew L. Bolton & Ellen J. Bass (2010): *Using Task Analytic Models and Phenotypes of Erroneous Human Behavior to Discover System Failures Using Model Checking*. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54(13), pp. 992–996, doi:10.1177/154193121005401315. PMID: 25382961.

[13] M.L. Bolton, Ellen Bass & Radu Siminiceanu (2013): *Using Formal Verification to Evaluate Human-Automation Interaction: A Review*. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on* 43, pp. 488–503, doi:10.1109/TSMCA.2012.2210406.

[14] Patricia Bouyer, François Laroussinie, Nicolas Markey, Joël Ouaknine & James Worrell (2017): *Timed Temporal Logics*. In Luca Aceto, Giorgio Bacci, Giovanni Bacci, Anna Ingólfsdóttir, Axel Legay & Radu Mardare, editors: *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday, Lecture Notes in Computer Science* 10460, Springer, pp. 211–230, doi:10.1007/978-3-319-63121-9_11.

[15] Christos G Cassandras & Stéphane Lafortune (2008): *Introduction to discrete event systems*. Springer, doi:10.1007/978-0-387-68612-7.

[16] W. Damm, M. Fränzle, A. Lüdtke, J. W. Rieger, A. Trende & A. Unni (2019): *Integrating Neurophysiological Sensors and Driver Models for Safe and Performant Automated Vehicle Control in Mixed Traffic*. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 82–89, doi:10.1109/IVS.2019.8814188.

[17] Moussa Diarra, Jean Theurel & Benjamin Paty (2025): *Systematic review of neurophysiological assessment techniques and metrics for mental workload evaluation in real-world settings*. Frontiers in Neuroergonomics Volume 6 - 2025, doi:10.3389/fnrgo.2025.1584736.

[18] Rüdiger Ehlers, Robert Mattmüller & Hans-Jörg Peter (2010): *Combining Symbolic Representations for Solving Timed Games*. In Krishnendu Chatterjee & Thomas A. Henzinger, editors: *Formal Modeling and Analysis of Timed Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 107–121, doi:10.1007/978-3-642-15297-9_10.

[19] Lu Feng, Clemens Wiltsche, Laura Humphrey & Ufuk Topcu (2016): *Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems*. IEEE Transactions on Automation Science and Engineering 13(2), pp. 450–462, doi:10.1109/TASE.2016.2530623.

[20] Daniel Gall & Thom Frühwirth (2018): *An Operational Semantics for the Cognitive Architecture ACT-R and Its Translation to Constraint Handling Rules*. ACM Trans. Comput. Logic 19(3), doi:10.1145/3218818.

[21] Kenneth H. Goodrich, Paul C. Schutte, Frank O. Flemisch & Ralph A. Williams (2006): *Application of the H-Mode, A Design and Interaction Concept for Highly Automated Vehicles, to Aircraft*. In: *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*, pp. 1–13, doi:10.1109/DASC.2006.313781.

[22] Holly Gorin, Jigna Patel, Qinyin Qiu, Alma Merians, Sergei Adamovich & Gerard Fluet (2024): *A Review of the Use of Gaze and Pupil Metrics to Assess Mental Workload in Gamified and Simulated Sensorimotor Tasks*. Sensors 24(6), doi:10.3390/s24061759.

[23] Falk Howar & Bernhard Steffen (2018): *Active Automata Learning in Practice*, pp. 123–148. Springer International Publishing, Cham, doi:10.1007/978-3-319-96562-8_5.

[24] Eva Hudlicka (2002): *This Time with Feeling: Integrated Model of Trait and State Effects on Cognition and Behavior*. Applied Artificial Intelligence 16, pp. 1–31, doi:10.1080/08339510290030417.

[25] Malte Isberner, Falk Howar & Bernhard Steffen (2014): *The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning*. In Borzoo Bonakdarpour & Scott A. Smolka, editors: *Runtime Verification*, Springer International Publishing, pp. 307–322, doi:10.1007/978-3-319-11164-3_26.

[26] Einar Broch Johnsen, Paul Kobialka, Andrea Pferscher & Silvia Lizeth Tapia Tarifa (2025): *Nudging Strategies for User Journeys: Take a Path on the Wild Side*, pp. 42–63. Springer Nature Switzerland, Cham, doi:10.1007/978-3-031-73751-0_6.

[27] Ioan Doré Landau, Rogelio Lozano, Mohammed M'Saad & Alireza Karimi (2011): *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media, doi:10.1007/978-0-85729-664-1.

[28] Vincent Langenfeld, Bernd Westphal & Andreas Podelski (2019): *On Formal Verification of ACT-R Architectures and Models*. In: *CogSci*, pp. 618–624.

[29] John D Lee & Katrina A See (2004): *Trust in automation: Designing for appropriate reliance*. Human factors 46(1), pp. 50–80, doi:10.1518/hfes.46.1.50_30392.

[30] A Lüdtke, F Frische & JP Osterloh (2011): *Validation of a digital human model for predicting flight crew-aircraft cockpit interaction*. Proceedings of Berliner Werkstatt für Mensch-Maschine Systeme.

[31] Andreas Lüdtke, Lars Weber, Jan-Patrick Osterloh & Bertram Wortelen (2009): *Modeling pilot and driver behavior for human error simulation*. In Vincent G. Duffy, editor: *International Conference on Digital Human Modeling*, Springer, pp. 403–412, doi:10.1007/978-3-642-02809-0_43.

[32] Oded Maler, Amir Pnueli & Joseph Sifakis (1995): *On the Synthesis of Discrete Controllers for Timed Systems*. In: *STACS 95*, Springer Berlin Heidelberg, pp. 229–242, doi:10.1007/3-540-59042-0_76.

[33] Mauricio Marcano, Sergio Díaz, Joshué Pérez & Eloy Irigoyen (2020): *A Review of Shared Control for Automated Vehicles: Theory and Applications*. IEEE Transactions on Human-Machine Systems 50(6), pp. 475–491, doi:10.1109/THMS.2020.3017748.

[34] Edi Muškardin, Bernhard K Aichernig, Ingo Pill, Andrea Pferscher & Martin Tappler (2022): *AALpy: an active automata learning library*. Innovations in Systems and Software Engineering 18(3), pp. 417–426, doi:10.1007/s11334-022-00449-3.

[35] Edi Muškardin, Tamim Burgstaller, Martin Tappler & Bernhard K. Aichernig (2024): *Active Model Learning of Git Version Control System*. In: *2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 78–82, doi:10.1109/ICSTW60967.2024.00024.

[36] Allen Newell (1990): *Unified Theories of Cognition*. Harvard University Press, USA.

[37] Andrea Pferscher & Bernhard K. Aichernig (2020): *Learning Abstracted Non-deterministic Finite State Machines*. In Valentina Casola, Alessandra De Benedictis & Massimiliano Rak, editors: *Testing Software and Systems*, Springer International Publishing, Cham, pp. 52–69, doi:10.1007/978-3-030-64881-7_4.

[38] A. Pnueli & Roni Rosner (1989): *On the synthesis of a reactive module*. Automata Languages and Programming 372, pp. 179–190, doi:10.1145/75277.75293.

[39] Stefan Puch, Bertram Wortelen, Martin Fränzle & Thomas Peikenkamp (2013): *Evaluation of Drivers Interaction with Assistant Systems Using Criticality Driven Guided Simulation*. In: *Digital Human Modeling and Applications in Health, Safety, Ergonomics, and Risk Management. Healthcare and Safety of the Environment and Transport - 4th International Conference, DHM*, Lecture Notes in Computer Science 8025, Springer, pp. 108–117, doi:10.1007/978-3-642-39173-6_13.

[40] Paul S. Rosenbloom, Allen Newell & John Laird (1993): *The SOAR papers: Research on integrated intelligence*. 978-0-262-18152-5. Mit Press Cambridge, MA.

[41] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia & Anca D. Dragan (2016): *Planning for Autonomous Cars that Leverage Effects on Human Actions*. In: *Robotics: Science and Systems*, doi:10.15607/RSS.2016.XII.029.

[42] SAE International (2021): *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE Standard J3016_202104. Available at: https://doi.org/10.4271/J3016_202104.

[43] Dario Salvucci, Erwin Boer & Andrew Liu (2001): *Toward an Integrated Model of Driver Behavior in Cognitive Architecture*. Transportation Research Record 1779, pp. 9–16, doi:10.3141/1779-02.

[44] Dario D Salvucci (2006): *Modeling driver behavior in a cognitive architecture*. Human factors 48(2), pp. 362–380, doi:10.1518/001872006777724417.

[45] Paul Maria Scheikl, Balázs Gyenes, Tornike Davitashvili, Rayan Younis, André Schulze, Beat P. Müller-Stich, Gerhard Neumann, Martin Wagner & Franziska Mathis-Ullrich (2021): *Cooperative Assistance in Robotic Surgery through Multi-Agent Reinforcement Learning*. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1859–1864, doi:10.1109/IROS51168.2021.9636193.

[46] Thomas B Sheridan, William L Verplank & TL Brooks (1978): *Human and computer control of undersea teleoperators*. In: *NASA. Ames Res. Center The 14th Ann. Conf. on Manual Control*.

[47] Wolfgang Thomas (1995): *On the Synthesis of Strategies in Infinite Games*. In Ernst W. Mayr & Claude Puech, editors: *STACS 95, 12th Annual Symposium on Theoretical Aspects of Computer Science, Munich, Germany, March 2-4, 1995, Proceedings*, Lecture Notes in Computer Science 900, Springer, pp. 1–13, doi:10.1007/3-540-59042-0_57.

[48] Frits Vaandrager (2017): *Model learning*. Commun. ACM 60(2), p. 86–95, doi:10.1145/2967606.

[49] Matt Webster, David Western, Dejanira Araiza-Illan, Clare Dixon, Kerstin Eder, Michael Fisher & Anthony G Pipe (2019): *A corroborative approach to verification and validation of human–robot teams*. The International Journal of Robotics Research 39(1), p. 73–99, doi:10.1177/0278364919883338.

[50] Bertram Wortelen, Andreas Lüdtke, Martin Baumann, WG Kennedy, R St Amant & D Reitter (2013): *Integrated Simulation of Attention Distribution and Driving Behavior*. In: *Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling and Simulation*, pp. 69–76.