Probabilistic Model Checking for Human Activity Recognition in Medical Serious Games

Thibaud L'Yvonnet, Elisabetta de Maria, Sabine Moisan, Jean-Paul Rigault

Probabilistic Model Checking for Human Activity Recognition in Medical Serious Games

Thibaud L'Yvonnet[a], Elisabetta De Maria[b], Sabine Moisan[a], Jean-Paul Rigault[a]

[a] *Université Côte d'Azur, INRIA, Sophia Antipolis, France*
[b] *Université Côte d'Azur, CNRS, I3S, Sophia Antipolis, France*

Abstract

Human activity recognition plays an important role especially in medical applications. This paper proposes a formal approach to model such activities, taking into account possible variations in human behavior. Starting from an activity description enriched with event occurrence probabilities, we translate it into a corresponding formal model based on discrete-time Markov chains (DTMCs). We use the PRISM framework and its model checking facilities to express and check interesting temporal logic properties concerning the dynamic evolution of activities. We illustrate our approach with the models of several *serious games* used by clinicians to monitor Alzheimer patients. We expect that such a modeling approach could provide new indications for interpreting patient performances. This paper addresses the definition of patient's models for three serious games and the suitability of this approach to check behavioral properties of medical interest. Indeed, this is a mandatory first step before clinical studies with patients playing these games. Our goal is to provide a new tool for doctors to evaluate patients.

Keywords: activity description, probabilistic model, model checking, serious games, bio-medicine

1. Introduction

In the last decades human behavior recognition has become a crucial research axis [1] and has been employed in many contexts, such as visual surveillance in public places [2, 3], smart homes [4], or pedestrian detection for smart cars [5, 6]. Recently, serious games have been introduced in the health domain to help evaluate the performances of patients affected by neuro-degenerative pathologies such as the Alzheimer disease [7]. Behavior, emotions and performance displayed by patients during these games can give indications on their disease.

Much has been done on simple *action* recognition [8], especially in computer vision, whereas we target complex *activities* that embed several actions. In our view, an activity consists in a set of scenarios that describe possible behavioral variants. Therefore, the purpose of recognition is to identify which scenario is running by analyzing the inputs produced by different types of sensors. Currently, we mostly use video cameras but also binary sensors or audio signals. Our ultimate aim is to propose a general (human) activity recognition system that helps medical practitioners to monitor patients with cognitive deficiencies.

Not all scenarios of an activity are equivalent: some are typical (usually frequent) while others seldom happen; this is due to variations in the behavior of the actors involved in the activity. To improve the analysis and interpretation of an activity (e.g., a patient playing a serious game), we propose to quantify the likelihood of these variations by associating probabilities with the key actions of the activity description. Some actions will be performed for sure by the patient (or the environment) and need no probabilities. Other ones depend on the Alzheimer stage of the patient. With these latter actions, practitioners will associate (a

Email addresses: thibaud.lyvonnet@inria.fr (Thibaud L'Yvonnet), edemaria@i3s.unice.fr (Elisabetta De Maria), sabine.moisan@inria.fr (Sabine Moisan), jean-paul.rigault@inria.fr (Jean-Paul Rigault)

priori) a discrete probability level (e.g., low, medium, high...) or directly a real number or weight. Hence, we can deduce how relevant to the Alzheimer stage the scenario played by a patient is. For example, if a patient known to be healthy plays a "mild cognitive impairment" (MCI) scenario, our system is able to spot this information. The same goes if a "severe cognition deficit" patient plays a "healthy" scenario. The recognition process remains deterministic since, at recognition time, only one scenario at a time will be played and recognized.

Our first contribution is a formal modeling framework where activities are represented by (hierarchical) discrete-time Markov chains whose edges can be decorated with probabilities [9]. Markov chains are deterministic and do not impose to associate a real duration with each action, contrarily to, e.g., timed automata. We can thus "master" the time in our activity models, restricting it to the instants when some significant events occur, hence reducing the duration of simulations or model checking.

Furthermore, in the games that we address we can have non homogeneous delays between actions and we do not want to consider the smallest delay as the (minimal) time unit, since that would generate a huge number of (useless) states in the model and model checking would not be feasible. The choice of using formal modeling and model checking is mainly motivated by the ability of these techniques to directly provide probabilities associated with *classes* of paths and to test universal properties on the model, contrary to simulation techniques which only deal with existential properties.

As a second contribution, we have implemented serious game activities as discrete-time Markov chains using the PRISM language [10, 9]. We used temporal logic to encode some relevant properties on their dynamical evolution, and we applied model checking techniques [11] to automatically validate the models with respect to these properties as well as to infer the probabilities of some interesting paths. When applied to the recognition of serious games for Alzheimer patients, this technique can provide medical doctors with indications to interpret patients' performance.

In parallel, we are developing a formal computer language[1] to allow hospital practitioners to write programs describing the activities that they expect from their patients. These programs will represent all the envisioned paths (possible combinations of actions from the patient or the environment), both for typical behaviors and marginal ones.

After discussions with medical doctors, we identified three prospective applications of our approach:

- *Evaluate a patient.* If a patient comes for the first time to get a diagnosis, we can compare her results to a reference model representing a "healthy" patient behavior. Our approach provides a fairly good idea of what such a healthy behavior is: for example, the approximate number of good and bad answers at the end or at a certain point of the game, the type of errors made, or the probability for the patient to quit the game before its end. If the patient's results differ too much from the simulation results, it may be due to a disease and the patient might need a full diagnosis from a doctor.

- *Monitor a patient.* For a given patient, a customized profile can be created according to the results obtained during the first tests. Thus, from one session to the next, her health improvement or deterioration can be monitored. If the ratio of good/bad answers is increasing while the number of answered questions is not decreasing, it may show an improvement. On the other hand, if the ratio is decreasing or if the number of answered questions is decreasing, it may show that the disease is progressing.

- *Create a cohort of patients.* Once a reference profile is validated, we can use it to determine whether a new group of patients belongs to this specific category. This process is similar to a screening test on a population as it would only be a step before a definitive diagnosis; it is cheaper compared to a full diagnosis for the whole population and faster thanks to the automation of the process. For example, such tests may allow practitioners to shortlist some patients who correspond to some medical criteria and to apply a specific protocol on this cohort.

In this paper, we study three serious games used by our medical partners to analyze the behavior of Alzheimer patients. These three games target different brain functions and raise different formal modeling

[1]The details of this formal language are out of the scope of this paper.

issues. Before performing clinical tests on real patients, it is necessary to validate our approach and to explore the kind of properties that model checking can address, which is the focus of this paper.

The present work relies on model-checking but not as in the classical engineering way. Generally formal methods are applied in various domains, including the medical one, to develop and validate new software and artifacts. This usage is of great help to designers. Our perspective is different: we do not address serious game *design*. In contrast we rather apply model checking to *existing* games in order to qualify the behavior of their players. We consider serious games that have already demonstrated their utility in medical practice to provide practitioners with significant information to complete patient diagnosis.

The paper is organized as follows. Section 2 describes our approach and introduces the three serious games that we choose as case study: the *Match Items game*, the *Recognition game*, and the *Inhibitory Control game*. Section 3 formally details discrete-time Markov chains and their support in the PRISM model checker. Section 4 introduces the PRISM model of the Match Items game as a discrete-time Markov chain, Section 5 applies model checking to this model and Section 6 introduces a variant model using pre-computed time dependent probabilities. Similarly, Section 7 and Section 8 are respectively devoted to the Recognition game model and its properties, and Section 9 and Section 10 detail the Inhibitory Control game and its properties. Finally, Section 11 concludes and opens future research directions.

This paper is an updated and extended version of [9]. It largely extends the conference paper with

- A new version of the Match Items game taking into account the patient's fatigue;
- The addition of two serious games, the Recognition game and the Inhibitory Control one, together with their activity models as Discrete-Time Markov Chains in PRISM, and a dozen of temporal logic properties;
- A comparison between the three game models.

2. Context and Formal Approach

2.1. Methodology

Our methodology to study the behavior of patients playing serious games is depicted on Figure 1. In Step 1, the medical staff (psychiatrists, psychologists, nurses) selects a serious game that already proved its medical interest in their practice and we discuss with them to understand precisely the rules of the game. Step 2 consists in translating these rules into a conceptual model (in the future, a program in the above mentioned formal language) that facilitates their encoding into a PRISM model. Together with the medical staff we choose *a priori* probabilities that are incorporated in the PRISM model (Step 3) and that represent the likelihood of patients' actions. This phase also offers an opportunity to talk with practitioners about the relevant actions to consider in the model and the interesting properties to check on modeled patient behaviors. In Step 4 we run PRISM to check the sanity of the model implementation and also to verify logical properties related to patients' medical assessment. These properties are expressed in PCTL (Probabilistic Computation Tree Logic). Then we can implement a clinical protocol where real patients play the game and the resulting data are used to calibrate the model, especially to adjust the probabilities. In the long run, we expect that this kind of model-checking based approach could become part of routine clinical practice.

2.2. Model Checking and Medical Activity Recognition

As seen in Step 4 of the methodology, we use model checking techniques but not in the classical engineering way, where it is applied to verify that a system design meets its specifications. In particular, in the medical domain, such techniques have been used to verify safety critical software or artifacts. Indeed, as technology keeps on improving, software systems find their place in medical tools. The use of formal methods thus becomes a crucial step in the engineering of these tools [12] and skipping it may lead to dangerous health issues. Model checking of medical tools can be applied either offline (at design phase before system deployment) or online (during system utilization). Regarding offline model checking, we can cite [13] and [14] in which formal methods are used to detect inconsistencies of software components. In [13], such

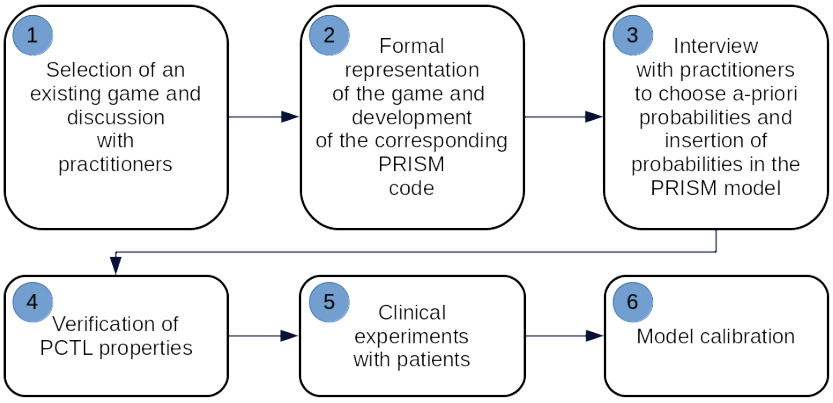


Figure 1: Workflow displaying the methodology adopted in the paper.

methods are applied to prove that an hemodialysis machine design can be considered as defect-free, that it correctly implements its specification, and that it complies with safety, security, and reliability standards. In [14], the authors use model checking to validate the design of a control algorithm for a surgical robot. This algorithm and its environment are modeled with DTMCs in PRISM. The authors could verify the following properties: deadlock freedom, absence of collision between surgical tools, and reachability of a position in the operating area. Even more importantly they could analyze the impact of different control algorithms on the out-of-boundary detection. A more methodological article [15] concentrates on operator's actions modeling. The authors succeed in generating Hollnagel's zero-order phenotypes of erroneous manipulations such as omissions, jumps, repetitions, and intrusions. To demonstrate the use of this methodology, the authors choose as example a radiation therapy machine based on the Therac-25. Despite limitations due to complexity, the generated model can be verified and reveals system problems due to erroneous human behavior. This "action phenotypes" approach could be an interesting track to explore, in our case to model patient's actions. Regarding online model checking, in [16] the authors specify a network of timed automata in UPPAAL to model patients respiratory motions during radiation therapy. Since these external motions are correlated with tumor internal motions, the goal is to detect movements that may lead to misalignment between the beam and the tumor and thus requires to stop irradiation. Online model-checking processes real time input data to predict such a problem and to adjust the model if necessary. The authors demonstrate that this approach outperforms classical state-of-the-art methods in term of number of problems detected.

In these applications, although the patients are modeled in some way, they are mainly described by their physiological data (blood pressure, pulse rate, respiratory rhythm...) that may influence the modeled device operation or the ongoing medical process.

Contrarily to this designer oriented model checking usage, our aim is to model and verify patients' actions and behavior in order to help physicians refine their diagnosis about patients' condition. We do not design serious games and thus we do not address the model checking of these software tools. Moreover, our target games are already in use and have been verified and debugged by their developers. As in [16], we perform model-checking during the game sessions with respect to real-time inputs coming from sensors. However, the goal is not to detect software problems nor to influence the game progress (nor the player of course) but only to observe and record data on the patient's interaction with the game. The type of interaction and its differences with a "normal" game session will help physicians diagnose a possible disease.

Our use of model checking is related to human activity recognition. It is close to [17] where model checking is used to detect differences between actual medical actions and ideal ones as described in clinical guidelines. The guidelines are represented as state-transition systems and the model checking of (CTL) temporal logic properties investigate the consistency of the formalized guidelines and the actual treatment. In [18] the authors apply an extended temporal logic and model checking to support automated real-time recognition of activities of daily living of elderly people. These papers apply model checking as we do, both to verify properties of a predefined formal model of activity and to confront it with real data coming from

patients or doctors activities. A first difference is that these two works use deterministic logics and model checking whereas we rely on probabilistic ones. This is essential to fulfill our aim which is not only to recognize people activities but also to evaluate patient impairment level and thus to use model checking as an aid to diagnose and characterize patients. Note that probabilistic model checking is also used to debug PRISM *models* of activities [19]. In our case, we use probabilities to explore paths associated with different behaviors.

2.3. Case Studies: Serious Games for Alzheimer Patients

Observing patients when they perform simple activities provides interesting clues on their cognitive status. To this aim, several "behavioral" tests have been used by medical doctors for years. Among these tests, serious games are interesting software tools that combine entertainment with a medical purpose. They constitute a domain in which real-time activity recognition is particularly relevant: the expected behavior is well identified and it is possible to rely on different sensors (bio-metric and external) while playing the game. In the health domain, serious games can be used to incite patients to practice physical exercises [20], to train medical staff with engaging activities [21], or to help diagnose and treat patients [22, 23]. When formally modeling a patient playing a diagnosis game, one can associate probabilities with actions to represent a healthy or a pathological behavior. These probabilities are initially defined according to physicians' past experience. Properties can then be written to extract relevant data, to be compared, first with experimental results in order to refine the model and ultimately, with real patients results.

As use cases, we consider three serious games currently used to analyze the behavior of Alzheimer patients: the *Match Items game* [7], the *Recognition game* [24], and the *Inhibitory Control game* [25].

All these games were designed to provide playful training tools for patients with MCI (Mild Cognitive Impairment), but each game targets a different brain function. The *Match Items* game targets selective and sustained visual attention functions. The *Recognition* game targets episodic memory. The *Inhibitory Control* game targets the inhibitory control function whose role is to inhibit some of our reflexes.

Since these games are selected by practitioners (Step 1 of Figure 1), they constitute a good panel of tests that can complement the set of validated neuro-psychological tests usually conducted by clinicians to establish their diagnosis. The use of such games may assist practitioners during the diagnosis process. It will provide a new non-invasive screening method based on observation of patient performance.

To model the behavior of a patient playing these games we use discrete-time Markov chains (DTMCs). To the best of our knowledge, DTMC models are barely used for the description of human behavior, although, in the context of pedestrian localization, we can cite [26] who uses DTMC to describe pedestrian trajectories. In computer vision, Hidden Markov Models (HMMs) are a popular approach for the description of activities [27, 28]. However, PRISM and most of the other probabilistic model checkers do not allow to check temporal logic properties over HMMs.

The three games, their models, and some interesting properties on these models are detailed in the next sections. The tests were run on a computer with eight processors (Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz) and 32GB RAM, running under the Fedora Linux operating system.

3. The PRISM Model Checker

Several probabilistic model checkers exist (such as UPPAAL [29], STORM [30], or PAT [31]). We decided to rely on PRISM [10], developed as a probabilistic model checker since the beginning, well established in the literature, and compatible with many supplementary tools (such as parameter synthesis tools and other model checkers, e.g., STORM [30]). More precisely, PRISM is a tool for formal modeling and analysis of systems with random or probabilistic behavior. It has already been used to describe human activity [32]. It supports several types of probabilistic models, discrete as well as continuous. In this work we rely on discrete-time Markov chains (DTMC), which are transition systems augmented with probabilities. Their set of states represents the possible configurations of the system being modeled, and the transitions between states represent the evolution of the system, which occurs in discrete-time steps. Probabilities to transit between states are given by discrete probability distributions. Markov chains are memoryless, that is, their current state contains all the information needed to compute future states. More precisely:

Definition A Discrete-Time Markov Chain over a set of atomic propositions AP is a tuple (S, S_{init}, P, L) where S is a set of states (state space), $S_{init} \subseteq S$ is the set of initial states, $P : S \times S \to [0, 1]$ is the transition probability function (where $\sum_{s' \in S} P(s, s') = 1$ for all $s \in S$), and $L : S \to 2^{AP}$ is a function labeling states with atomic propositions over AP.

An example of DTMC of a simple two-state game is depicted in Figure 2. In this game, the player has to press a button as many times as she wishes. This figure shows labels (press and release) over transitions. Even though they are not part of the definition of DTMCs, they are implemented by PRISM. This usual extension does not change the semantics of DTMCs, but it makes the models easier to read and to understand.

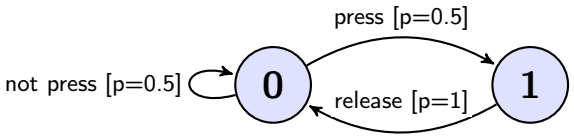


Figure 2: DTMC representing a simple game. Each edge is labeled with both an action and the corresponding probability.

Note that the models presented in this paper have a *finite* set of states. Indeed, the behaviors that they describe can be represented by finite state machines. Furthermore we observe them over a bounded and discrete time. Once unwound to feed PRISM model checkers, the resulting overall state space remains finite.

3.1. PRISM Modeling Language

PRISM provides a state-based modeling language inspired from the reactive module formalism of [33]. A model is composed of a set of *modules* which interact with each other. The state of a module is given by the values of its *local variables* and the global state of the whole model is determined by the union of the local states of all its modules. The dynamics of each module is described by a set of commands of the form:

$$[\,]guard \to prob_1 : update_1 + ... + prob_n : update_n;$$

where *guard* is a predicate over variables of the model, corresponding to a condition to be verified in order to execute the command, and each *update* indicates a possible transition of the model, achieved by giving new values to variables. Each *update* is assigned a probability and, for each command, the sum of probabilities must be 1. The square brackets at the beginning of each command can either be empty or contain labels representing *actions*. These actions can be used to force two or more modules to transit simultaneously. The PRISM code for the DTMC of Figure 2 is shown in Algorithm 1. In this code, the unique integer variable y represents the state of the player, it ranges over $\{0, 1\}$. Its initial value is 0. When the guard $y = 0$ is true, the updates $(y' = 0)$ and $(y' = 1)$ and their associated probabilities state that the value of y remains at 0 with probability 0.5 and switches to 1 with probability 0.5. When $y = 1$, the update $(y' = 0)$ with probability 1 states that y switches back to 0.

Finally, PRISM models can be extended with *rewards* [34], associating real values with model states or transitions. An example of reward is given at the end of Algorithm 1: each time $y = 1$ (button pressed), the reward is incremented by 1.

3.2. Probabilistic Temporal Logic

The dynamics of DTMCs can be specified in PRISM owing to Probabilistic Computation Tree Logic (PCTL) [35]. PCTL extends the Computation Tree Logic (CTL) [11] with probabilities. The following state quantifiers are available in PCTL: **X** (next time), which specifies that a property holds at the next state of a given path, **F** (sometimes in the future), which requires a property to hold at some state on the path, **G** (always in the future), which imposes that a property is true at every state on the path. The until operator, **U**, is such that *p1* **U** *p2* means that property *p1* is *true* until property *p2* becomes *true*. Note that the classical path quantifiers **A** (forall) and **E** (exists) of CTL are replaced by probabilities. Thus, instead of

Algorithm 1 PRISM code for Figure 2 DTMC.

```
dtmc //Discrete-Time Markov Chain
module good_answer_game
y: [0..1] init 0;
//Commands
[ ] y=0 -> 0.5:(y'=0) + 0.5:(y'=1); // y' corresponds to y in the next instant
[ ] y=1 -> 1:(y'=0);
endmodule
rewards "y"
y=1: 1;
endrewards
```

saying that some property holds for all paths or for some paths, we say that a property holds for a certain fraction of the paths [35].

The most important operator in PCTL is **P**, which allows to reason about the probability of event occurrences. The property **P** *bound* [*prop*] is true in a state s of a model if the probability that the property *prop* is satisfied by the paths from state s satisfies the bound *bound*. As an example, the PCTL property `P= 0.5 [X (y = 1)]` holds in a state if the probability that $y = 1$ is true in the next state equals 0.5. All the state quantifiers given above, with the exception of **X**, have bounded variants, where a time bound is imposed on the property. Furthermore, in order to compute the actual probability that some behavior of a model occurs, the **P** operator can take the form `P=?`. For instance, the property `P =? [G (y = 0)]` expresses the probability that `y` always equals 0.

PRISM also supports the notion of integer user-defined "rewards" which can be seen as counters that do not impact the number of states and transitions of the model nor its behavior. The **R** operator allows to retrieve reward values. Additional operators have been introduced to deal with rewards; we mainly use **C** (cumulative-reward). The property `C<=t` corresponds to the reward value accumulated along a path until `t` time units have elapsed. PRISM provides model checking algorithms [11] to automatically validate DTMCs over PCTL properties and reward-based ones. On demand, the algorithms compute the actual probability of some behavior of a model to occur.

PRISM proposes four different model checking engines. Three of them ("MTBDD", "sparse", and "hybrid") use data structures, as for example binary decision diagrams (BDDs) and multi-terminal BDDs (MTBDDs). This use of data structures allows to qualify these engines as wholly or partly symbolic. These three engines represent models as MTBDDs but each one has its own model checking algorithm. The fourth engine uses explicit-state data structures, hence it is known as the "explicit" engine. Since this engine does not use symbolic data structures for model construction, it can perform model checking in cases where other engines fail. For example, it can handle models with a potentially very large state space but in which only a fraction of this space is actually reachable.

In addition to the classical model checking tools, PRISM offers the possibility to run "experiments". The PRISM authors describe this feature as a "way of automating multiple instances of model checking". Thanks to this feature, users can obtain curves describing the results of a property with respect to one or several variables. Besides, PRISM also proposes "statistical model-checking", a way to verify properties through simulations.

4. Match Items Game

In the Match Items game [7] patients interact with a touch-pad. They are asked to match a random picture displayed in the center of the touch-pad with the corresponding element in a list of pictures at the bottom of the screen (see Figure 3). The game lasts at most five minutes.

If the patient chooses the right picture, a happy smiley is displayed and a new picture is proposed. Otherwise a sad smiley is displayed and the patient is asked to try again. If the patient does not interact quickly enough with the touch-pad (more than 10 seconds of inactivity), the game prompts her to choose a picture. Whenever the patient exits the game zone, the game is aborted.

Figure 3: Display of the Match Items game.

For non experts in computer science, we intend to propose a language to describe activities to be recognized in real-time. Since human activities are usually event-driven, this language will belong to the well-known class of *reactive* languages in which events from the environment trigger actions. This language will offer usual instructions such as parallel execution, conditional, or repetition. Most instructions will have associated weights either in the form of real numbers or symbolic discrete values. These weights will be digitized (if they are discrete) and normalized to obtain probabilities. In this paper we do not provide a full description of the language. This language is still under development, indeed such a formal description is necessary to avoid plain text ambiguity. Here we simply use a tentative syntax to illustrate its application in listing 1 with a simplified pseudo-code program for the Match Items game.

```
Initial: patient inside game_zone and patient presses_start_button
 during 300s
   console displays_picture
   when [0.0005] patient exits game_zone preempt { exit }
      // main loop on each occurrence of the asks_to_choose event
      every console asks_to_choose patient
         switch
         case [0.75] (patient selects_picture) // patient selected something
            switch
            case [0.66] (console displays_happy_smiley)
               // correct answer: new picture and continue loop
               console displays_picture !! count: happy_smiley
            case [0.33] (console displays_sad_smiley)
               // wrong answer: loop keeping current picture
               nothing !! count: sad_smiley
            end switch
         case [0.25] (console notifies_inactivity)
            // patient did not react, continue with same picture
            nothing !! count: non_interaction
         end switch
      end every
   end when
 end during
emit game_over
```

Listing 1: Match Items serious game pseudo code description. Green numbers inside square brackets are probabilities. Red comments starting with !! suggest PRISM rewards.

The game starts when the patient has been detected in the game zone and presses the start button. The **when** clause introduces a preemption: the game may abort prematurely, whatever its execution state is, if

the patient leaves the game zone before the normal end of the game; this is possible with Alzheimer patients who may suffer from attention deficiency. The core of the game is described via the probabilistic **switch case**s. The branches of a **switch** are exclusive and their order is a priority order: the first branch whose trigger event occurs executes its statements (in the listing, trigger events are the ones between parentheses following the **case** keyword). A probability of occurrence may be associated with a branch (indicated within square brackets in the pseudo-code). For example, the probability of receiving a sad smiley (meaning that the patient chose a wrong picture) is 0.33. In this case, the console asks again to choose the (correct) picture.

Furthermore, clinicians can indicate (through ***!!*** annotations) significant events that should be remembered and counted. For instance, the number of happy smileys displayed during the game gives an interesting information about a patient's performance. In this example, practitioners gave a priori probabilities that should represent patients with mild cognitive impairment (Step 3 of Figure 1).

Note that, in this example, the sum of the weights in the probabilistic switch case and in the preemptive condition is not 1. A normalization will be applied to obtain the probabilities for the formal model. Thus, the user does not have to bother with numeric computations.

As stated in the previous section, we use a DTMC to model the behavior of a patient playing this game. Due to a limitation in PRISM, we must explicitly represent all the possible states in the model. Indeed, in PRISM Markov chains, it is not possible to limit the number of times the model can loop over the same state. This means that, even with a low probability on the loop transition, there will always be a risk for a simulation to never quit the loop (fairness is not automatically imposed). By unwinding the model, i.e. explicitly representing all possible states of the game, we avoid this issue. Since the game activity lasts at most five minutes (or three-hundred seconds), we know that there will be a finite number of states in the chain. Thus, in the PRISM model, we made the assumption that a patient needs at least three seconds to select a picture (minimum time needed to think of which picture to choose and to touch the screen to select it).

4.1. Model Design

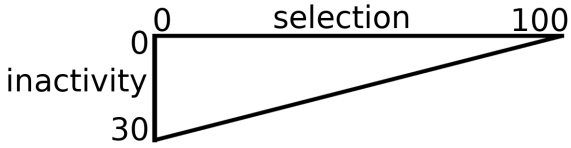
With the previous assumption, Step 2 of Figure 1 can be completed. Indeed, the time constraint of three-hundred seconds (and three seconds minimum for a selection) can be translated into a maximum number of actions (or events) that can happen in a scenario. If the patient keeps on selecting pictures, a smiley (happy or sad) is displayed. We call this event *selection* and it cannot happen more than a hundred times in a row ($300s/3s = 100$). On the other hand, if the patient does not interact with the game for ten seconds, the system displays a message (event $notifies\_inactivity$ in listing 1). We call this event *inactivity* and it cannot happen more than thirty times in a row ($300s/10s = 30$).

To represent all combinations of these two events, we picture a right-angle triangle (Figure 4a). The edge of length one hundred (representing the scenario of a succession of *selections*) and the edge of length thirty (representing the scenario of a succession of *inactivities*) form the perpendicular sides of the triangle. Each state of this triangle, except those on the hypotenuse, have three different possible transitions, represented in Figure 4b. According to this figure, a state can either increment *selection* and move on the *selection* axis, or increment *inactivity* and move on the *inactivity* axis. To represent the action of the patient leaving the game before the end of the five minutes (which could be detected by a camera) we use a Boolean variable called *quit_game*. If this variable is true, the state previously reached in the triangle is considered as the final state of the game session.

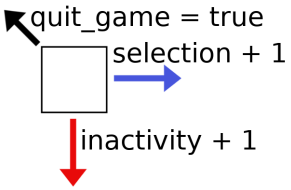
All states on the hypotenuse represent the end of the five minutes of the game. The only possible transition from them is equivalent to *quit_game*.

4.2. PRISM Implementation

The model is composed of a single module called "Match_Items_game". In this module, the location of the patient is represented by an integer variable with range [0..2] called *location*: 0 represents the patient being in the room before playing, 1 the patient being in the gaming area, and 2 the patient being outside this area.



(a) Combinations of events triangle.



(b) Possible state transitions for states symbolizing the game session.

Figure 4: Concepts of the model of activity.

As previously described, the interaction of a patient with the game is represented as an integer variable with range [0..100] called *selection*. A value i represents the fact that the patient had i interaction(s) with the game.

The event of the game displaying a message after ten seconds of inactivity is represented as an integer variable with range [0..30] called *inactivity*. A value i represents the fact that the game displayed the message i time(s).

To ease readability and re-usability of the module, each of the previous variables gets its maximum value defined outside the module in a global variable: *location_max*, *selection_max*, and *inactivity_max*, respectively.

Variables *selection_max* and *inactivity_max* are also used to determine if a state belongs to the hypotenuse of the triangle mentioned before. To do so, we solve the following equation (where $\lceil x \rceil$ is the application of the ceiling function to x, denoting the smallest integer greater or equal to x):

$$inactivity = \left\lceil \left(-\frac{inactivity\_max}{selection\_max} \right) \times action + inactivity\_max \right\rceil \tag{1}$$

To take advantage of the rewards of PRISM, we use Boolean variables to represent the other concepts.

- The event "a happy (resp., sad) smiley is displayed" for a good (resp., bad) answer is represented by the variable *happy_smiley* (resp., *sad_smiley*).
- The event "the patient leaves the game area before the end of the five minutes" is represented by *quit_game*.
- The event "the console displays a message after ten seconds of inactivity" is represented by *non_interaction*.

Only one of these variables at a time can be *true*. Each time a variable is *true*, it means that the event it represents happened and the associated reward is incremented. The rewards associated with these Boolean variables are the following: *happy_smiley* is associated with *Happy_smiley_reward*, *sad_smiley* with *Sad_smiley_reward*, *non_interaction* with *Non_interaction_reward*, and *quit_game* with *Leave_game_reward*; the amount of time spent by the patient in the game is represented by *Gaming_time*.

The *Gaming_time* reward is more complex than the others because it increases by three units for each good or bad answer and by ten units for each inactivity message displayed by the console.

The state of the patient can go through different transitions only if it matches one of the four different guards of the "Match_Items_game" module (note that the global variable *time_Is_Over* contains a Boolean expression to determine if the maximum number of actions that a patient can perform is reached).

1. variable *location* is equal to 0, meaning the patient is in the room;
2. variable *location* is equal to 1, *time_Is_Over* is *false* and *quit_game* is *false*, meaning the patient is playing the game;
3. variable *location* is equal to 1 and *time_Is_Over* is *true*, meaning the patient has played for the maximum time;

4. variable *location* is equal to 1 and *quit_game* is *true*, meaning the patient left the game before the end of the maximum duration.

The PRISM code for the command associated with the second guard is given in listing 2, where $p1 = 0.5/sum$, $p2 = 0.25/sum$, $p3 = 0.25/sum$, and $p4 = 0.0005/sum$, with $sum = 0.5 + 0.25 + 0.25 + 0.0005$.

```
[acts] location=1 & !time_Is_Over & quit_game=false ->
  // good answer
  p1 : (selection'=selection+1) & (happy_smiley'=true) &
       (sad_smiley'=false) & (inactivity_bool'=false) +
  // bad answer
  p2 : (selection'=selection+1) & (happy_smiley'=false) &
       (sad_smiley'=true) & (inactivity_bool'=false) +
  // inactivity
  p3 : (inactivity'=inactivity+1) & (happy_smiley'=false)&
       (sad_smiley'=false) & (inactivity_bool'=true) +
  // game left
  p4 : (quit_game'=true) & (happy_smiley'=false) &
       (sad_smiley'=false) & (inactivity_bool'=false);
```

Listing 2: Excerpt from the Match_Items_game module.

The state transitions taken along a path describe the patient's behavior in a specific scenario. Some of these transitions have attached probabilities. The different possible transitions for a patient are the following:

- if the first guard is *true*, *location* is updated to 1, meaning the patient enters the gaming area;
- if the second guard is *true*, four different transitions can be taken with different probabilities: (i) the patient gives a good answer (with a weight of 0.5 for our tests); (ii) the patient gives a bad answer (weight 0.25); (iii) the system asks the patient to choose a picture after ten seconds of inactivity (weight 0.25); (iv) the patient leaves the game (weight 0.0005);
- if the third or fourth guard is *true*, *location* is updated to 2, meaning the patient leaves the gaming area.

To complete the previous model, we asked medical practitioners to provide *a priori* probabilities corresponding to a typical patient suffering from mild cognitive impairment (MCI). These probabilities are used in the following section.

5. Temporal Logic Properties for Match Items Game and Results

For the model described in the previous section, we encoded and tested several properties in PCTL (Step 4 of Figure 1).

Two kinds of properties may be defined: those to verify the model and those oriented toward the medical domain, which may give indications to a practitioner regarding a patient's behavior.

5.1. Model Verification

One typical property of the model itself is that all the model scenarios must reach the final state, which means in this case that the variable *location* must eventually be updated to 2. The following property verifies that this update occurs:

Property 1. What is the probability to reach the final state of the Markov chain?

$$P =?[F\ (location = location\_max)]$$

If the result is below 1, there exists a possibility to never reach the final state. This possibility only occurs if there is an error in the *Match Items game* model. In our case the result is 1.0; it is obtained in 0.002 seconds.

5.2. Medically Oriented Properties

5.2.1. Properties about interactions.

The following properties evaluate the probability for a path to go through i occurrences of *selection* and j occurrences of *inactivity*. The first three properties check the probability to end the game with $i = selection\_max$ or $j = inactivity\_max$ or i in between 0 and $selection\_max$ and j in between 0 and $inactivity\_max$. The two last properties check the probability for the patient to leave the game unexpectedly.

Property 2. What is the probability for a patient to never interact with the game until the end of the duration of the game?

$$P =?[F\ (selection = 0)\ \&\ (inactivity = inactivity\_max)]$$

Property 3. What is the probability for a patient to interact with the game until the end of the game without any interruption?

$$P =?[F\ (selection = selection\_max)\ \&\ (inactivity = 0)]$$

Property 4. What is the probability for a patient to start the game and to interact with it, e.g., 43 times (not necessarily consecutively) and not to interact with it 18 times (not necessarily consecutive)?

$$P =?[F\ (selection = 43)\ \&\ (inactivity = 18)]$$

Property 5. What is the probability for a patient to leave the game just after pressing the start button?

$$P =?[F\ ((selection = 0)\ \&\ (inactivity = 0))\ U\ (location = location\_max)]$$

Property 6. What is the probability for a patient to leave the game before the maximum game duration?

$$P =?[F\ (quit\_game)]$$

Discussion. The results for these properties are displayed in table 1, together with their computing time.

The probability obtained for Property 2 is rather low. This is due to the fact that there is only one path leading to the state satisfying this property and this path itself only goes through low probability transitions.

Two observations can be made on the results of Property 3: (i) the probability is higher than the one of Property 2; (ii) this probability is low. The first observation is due to the fact that the transition taken and repeated when this property is verified has three times more chances to be taken than the one taken to satisfy Property 2. The probability of Property 3 is pretty low because there is only one path made of three hundred transitions that satisfies this property.

Property 4 checks the probability to reach one of the states representing the end of the five minutes of the game. To give an example, a state which can only be reached with paths composed of 43 transitions representing an interaction and 18 transitions representing a non-interaction was chosen. The probability for this property is higher than the one of Property 3. This is due to the fact that this state can be reached by a large amount of paths.

Property 5 determines the probability for the patient to leave the game just after pressing the start button. This property can be seen as a specialized version of Properties 2, 3, and 4. Indeed, in its architecture, we see the same structure checking the values of *selection* and *inactivity* together with another structure checking the value of the variable *location*. The use of this third variable is used to discriminate between the instant before the start of the game and after the game started. Indeed, for these latter instants, *selection* and *inactivity* are both equal to 0. The probability for this property is low, which is consistent with the

Property	Result	Time(seconds)
Property 2	8.5445×10^{-19}	0.026
Property 3	3.0508×10^{-13}	0.049
Property 4	2.3188×10^{-2}	0.03
Property 5	4.9975×10^{-4}	0.054
Property 6	3.1364×10^{-2}	0.058

Table 1: Results from Property 2 to 6.

expected behavior of a MCI patient who usually has enough motivation to stay at least some time in the game.

The probability obtained for Property 6 is approximately 3% even though the probability for the path to go through "$quit\_game{=}true$" is five hundred times lower than the probability to take the non-interaction transition. To satisfy this property, all paths in which the transition $quit\_game$ is taken are considered. Note that if one increases the maximum duration of the game but keeps the parameters of the model as they are, the result of Property 6 increases.

Possible medical significance. The results obtained from the above properties give several indications. In the case of a cohort selection based on this model, the behavior described in Property 4, 5, and 6 should be observed quite rarely (respectively 2% and 3% of the cases). These behaviors may indicate that patients have a more degraded health condition than the desired patient profile. The behaviors described in Properties 2 and 3 must not be observed for MCI patients. If a cohort differs too much on the frequency of these behaviors, the practitioners must discard or deeply change it. Otherwise, the risk to perform a clinical test on the wrong sample of population is too high.

5.2.2. Properties about quality of actions.

These properties are relative to the quality of the actions (correct or not) that can be performed. The first one provides an average "score" for the model. The following ones give probabilities to follow some specific paths in the model.

Property 7. Is the average amount of good responses given by a patient greater than or equal to 30?

$$R\{"Happy\_smiley\_reward"\} >= 30[F\ (location = location\_max)]$$

Property 8. What is the average amount of good responses given by patients during their game sessions?

$$R\{"Happy\_smiley\_reward"\} =?[F\ (location = location\_max)]$$

Property 9. What is the probability for a patient to choose the correct picture exactly once and to never choose a good one again until the end of the game?

$$P =?[(F\ happy\_smiley)\ \&\\ (G\ (happy\_smiley\ =>\ (X\ G\ !happy\_smiley\ \&\ !quit\_game)))]$$

where $(!happy\_smiley)$ means *(not happy_smiley)*, that is $(happy\_smiley = false)$.

Property 10. What is the probability for a patient to directly choose the right picture, without choosing a wrong picture before?

$$P =?[F\ (selection = 1\ \&\ happy\_smiley)]$$

Discussion. The results for these properties are displayed in tables 2a and 2b.

Property 7 returns a Boolean[2]. For this model, the result is *true* and it is confirmed by the result of Property 8. This second property can also be written for *Happy_smiley_reward*, *Sad_smiley_reward* and

[2]According to the PRISM manual, "the total reward for a path is the sum of the state rewards for each state along the path plus the sum of the transition rewards for each transition between these states". In the case of bounded properties, the result is a Boolean indicating whether the target value is reached.

Reward	Result	Time(s)
Happy_ smiley_ reward	31	0.044
Sad_ smiley_ reward	15	0.019
Inactivity_ bool_ reward	15	0.042

(a) Results of Property 8.

Property	Result	Time(s)
7	*true*	0.047
9	3.3012×10^{-12}	2.046
10	6.6622×10^{-1}	0.007

(b) Results of Properties 7, 9 and 10.

Table 2: Results for the properties concerning the quality of actions.

Inactivity_ bool_ reward. According to its results, the average "score" for a cohort of patients matching this model parameters should be 31 good answers against 15 bad answers and there should be about 15 inactivity messages before the end of the session.

Property 9 was the longest to compute. The complexity of this property comes from the nesting of G operators. Property 10 gives the biggest probability value compared to all others. Indeed, unlike Property 9, there is a huge amount of scenarios that can validate it.

Possible medical significance. Still in the case of a cohort pre-selection, the group of patients should obtain an average "score" similar to the one obtained in Property 8. If the score differs too much from this result, the cohort must be rejected. According to the result of Property 9, a patient from this group is not expected to choose only one right answer and then stay without exiting until the end of the game. On the other hand, according to the result of Property 10, in this same group, it should be common to observe patients choosing the right picture on the first try (66% of the cohort).

5.3. Cumulative Rewards and Simulations

This subsection gives an example of a property which shows the interest to perform simulations of the model. We use the PRISM "cumulative reward" facilities to track how the model accumulates rewards over time. Properties using rewards can include variables such as the one indicating the number of steps to perform before checking the reward. This kind of variables allows the use of the "experiments" feature of PRISM and the creation of graphs of results.

Property 11. What is the amount of happy smileys accumulated within i steps?

$$R\{\text{"}Happy\_smiley\_reward\text{"}\} =?[C <= i]$$

where i is the number of steps to perform before checking the reward and C is the "cumulative-reward" operator presented in Section 3. This property is also applied to *Sad_ smiley_ reward*, *Inactivity_ bool_ reward*, *Gaming_ time*, and *Leave_ game_ reward*.

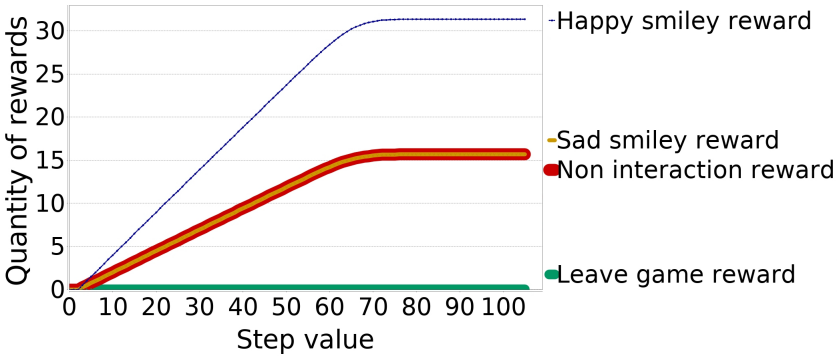


Figure 5: Average model checking results for rewards related to good answers, bad answers, non-interaction, and game leaving behavior.

In Figure 5, the rewards for good answers, bad answers, and non-interactions have a linear increase until they reach a plateau. The values reached by the rewards are the ones obtained in Property 8. The reward for the action of leaving the game is almost equal to zero. This is because this reward can be incremented only once in a run and that there is only 3% of the paths (see Property 6) where a patient may leave the game before its maximum duration.

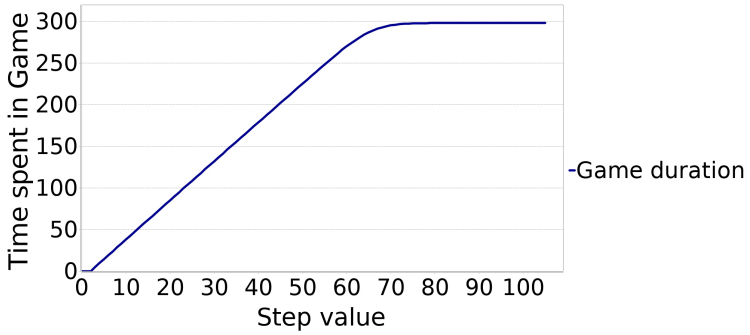


Figure 6: Average duration of the game obtained with model checking.

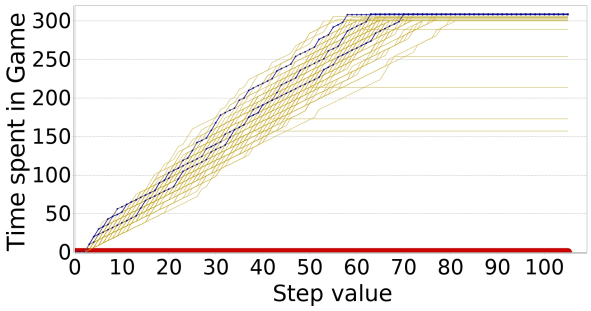
In Figure 6, the average game duration is slightly under 300 seconds. This is due to the paths where a patient may leave the game before the maximum duration. This shows that, although Equation 1 in section 4 implies an approximation with the ceiling function, the patients leaving the game are lowering the average enough to bring it just under the maximum expected value. As a final observation, the game duration reaches the plateau around the seventy-fifth step. This is due to the fact that most of the paths go through non-interaction transitions several times. Should they not go through these transitions at all, the plateau might have been reached around the 100^{th} step.

In Figure 7a, over 100 simulations, some of them (in blue in the figure) reach a maximum value which is above 300 seconds (still due to the approximation in Equation 1). Among these 100 simulations, some do not reach 300 seconds, one of them (in red in Figure 7a) even never increases and stays at 0. These simulations follow the paths where a modeled patient leaves the game before the end of the maximum duration. This experiment illustrates the results obtained with model checking (Properties 6 and 8).

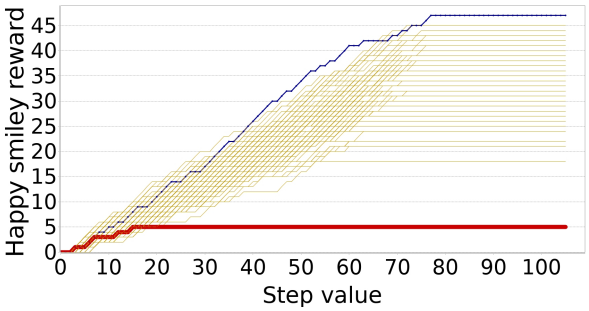
In Figure 7, over the 100 simulations, the results present a high variability which cannot be foreseen with model checking. In this experiment, a maximum value of 47 good answers for a minimum of 5 good answers is reached.

Globally, in Figure 7 as well as in Figure 5, there is no "preferred" time to act during the game. This can be seen with the linear increase of each reward value. This is due to the current version of the model; in fact, the states representing the game have homogeneous probabilities of transitions.

Due to the difficulty to distinguish the different runs in Figure 7, a shell and a Python scripts were



(a) Duration of the game over 100 runs.



(b) Accumulation of good answers over 100 runs.

Figure 7: Experiment results on the accumulation of rewards over 100 simulation runs.

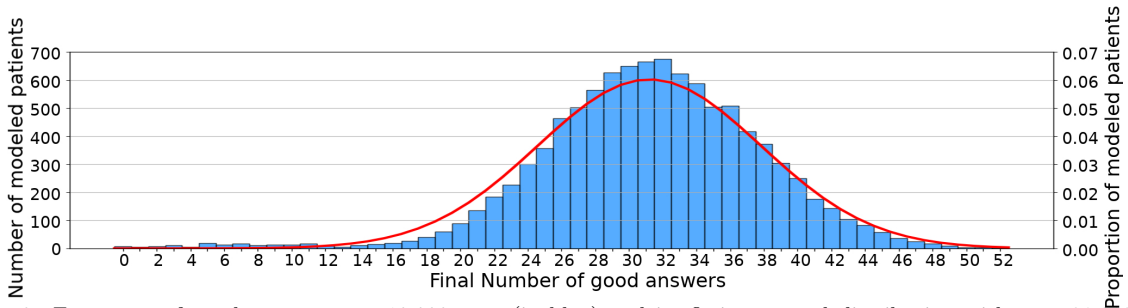


Figure 8: Frequency of good answers over 10,000 runs (in blue) and its fitting normal distribution with $\mu = 31.2131$ and $\sigma^2 = 43.9271$ (in red).

written to retrieve raw data from simulations. These data are used in Figure 8 to display the frequency of good answers over 10,000 runs. In this figure, the distribution of the frequency of good answers at the end of the game can be approximated by a normal distribution of mean $\mu = 31.2131$. This result is coherent with the result of Property 2. It can be stated that a patient represented by this model is more likely to give around 31 good answers rather than 40 or 25 ones.

For medical doctors to use these results, a range of acceptance must be defined experimentally for the game. A patient supposedly represented by this model who gets results that are out of the range of acceptance can be interpreted in two different ways: either the patient is not matching the model at all (improvement in the patient's behavior or wrong categorization of the patient) or the patient actually belongs to the group of patients represented by this model, but the model itself needs adjustments to better represent this group.

Remarks on the Match Items game model. This model was our first attempt to model and validate serious game activities with PRISM. This first experience has shown that a tool like PRISM is suitable to represent human activities and to automatically check properties of medical interest. Of course, the model parameters have to be later calibrated with respect to forthcoming clinical studies.

6. Pre-computed Time Dependent Probabilities for Match Items Game Model

In the previous model, regardless of the time elapsed in the game, the chances of the patients to give a good or bad answer are always the same. In reality, performances can vary within a single game session, mainly because the patient may become tired as the game proceeds. Such decrease in attention is documented for both healthy elders [36] and early Alzheimer's subjects [37]. The way sustained attention decreases over age is still under investigation but can be observed by practitioners through cognitive tests. Thus it would be more realistic that some probabilities may vary with the game time. However a "true" dynamic model where probabilities evolve in real-time would violate the Markov's hypotheses. Therefore our approach is to propose a simplified fatigue profile that enables us to *pre-compute* the needed probabilities at each time step of the game. So the PRISM unwound model mimics a dynamic behavior while respecting the memoryless property of DTMCs.

This section details the corresponding modifications to the Match Items game model and the results obtained with this new "fatigue-aware" version of the model. The probabilities are as follows.

```
// probability to give a good answer
p1 = (act_one_weight_one/sum_weight_one) - ((7/3000)*((selection+inactivity*10/3)));
// probability to give a bad answer
p2 = (act_one_weight_two/sum_weight_one) + ((7/5000)*((selection+inactivity*10/3)));
// probability to stay inactive
p3 = (act_one_weight_three/sum_weight_one) + ((3/5000)*((selection+inactivity*10/3)));
// probability to leave the game
p4 = (act_one_weight_four/sum_weight_one) + ((1/3000)*((selection+inactivity*10/3)));
```

Listing 3: Probability definition of the new Match_Items_Game module.

In these definitions, the first pair of parentheses contains the original probabilities from the previous implementation and the second pair of parentheses introduces the time dependent part. In this model, we consider that, as time passes in the session, the risk increases for the patient to give bad answers, to stay inactive, or to leave the game. To represent this, we introduce *a priori* a "fatigue constant" for each possible action of the patient. These constants are based on practitioners' experience [36, 37] and are to be updated with the results of future clinical studies. As *selection* transitions represent an elapsed time of 3s and *inactivity* transition represents 10s, we multiply *inactivity* by 10/3 to really take this time ratio into account.

To compare with our previous implementation, we checked this model with the same properties presented in Section 5. The new results from Property 2 to Property 6 are shown in table 3.

Property	Previous result	New result	New time(s)
Property 2	8.5445×10^{-19}	2.1732×10^{-17}	0.025
Property 3	3.0508×10^{-13}	4.8966×10^{-16}	0.032
Property 4	2.3188×10^{-2}	1.2294×10^{-2}	0.166
Property 5	4.9975×10^{-4}	4.9975×10^{-4}	0.068
Property 6	3.1364×10^{-2}	6.3952×10^{-1}	0.245

Table 3: Results from Property 3 to 6 in the new model, compared with previous results (first column).

Discussion on probabilistic properties. The results of Property 2 and Property 6 are far superior in this version, while the results from Property 3 and Property 4 are inferior (compared to table 1). The result of Property 5 is similar to the one obtained in the previous version.

Possible medical significance. The profile of patient obtained with these modifications is less stable than the previous one. Indeed, the risk for this patient to leave the game doubled and the probability for other actions such as non-interaction greatly increased compared to table 1.

The next tables show the results for Property 7 to Property 10.

Reward	Previous result	New result	New time(s)
Happy_ smiley_ reward	31	18	0.234
Sad_ smiley_ reward	15	13	0.247
Inactivity_ bool_ reward	15	12	0.233

(a) Results of Property 8.

Property	Previous result	New result	New time(s)
7	*true*	*false*	0.228
9	3.3012×10^{-12}	1.6475×10^{-9}	2.046
10	6.6622×10^{-1}	6.6333×10^{-1}	0.008

(b) Results of Properties 9 and 10.

Table 4: Results for the properties concerning the quality of actions with the new model, compared with previous results (first column).

Discussion on reward based properties. Table 4a shows that the accumulation of *Happy_ smiley_ reward* is far under the previous version, but the accumulated values of *Sad_ smiley_ reward* and *Inactivity_ bool_ reward* did not grow, indeed, they decreased.

Logically, the result of Property 7 is now *false*, since the reward accumulation does not reach 30. Even though the accumulation of *Sad_ smiley_ reward* and *Inactivity_ bool_ reward* rewards did not increase, the result of Property 9 is a thousand time higher than in the previous version of the model. The result of Property 10 is similar to what it was.

Possible medical significance. The low accumulation of rewards is due to the fact that the patient tends to leave the game before the end of the timer. Property 10 also shows that the behavior of the patient at the beginning of the game is really close to the previous profile, but this new profile presents more risks to achieve a poor performance at the end.

As for the previous model, we can track the changes in reward accumulation by using the "run experiment" tool of PRISM.

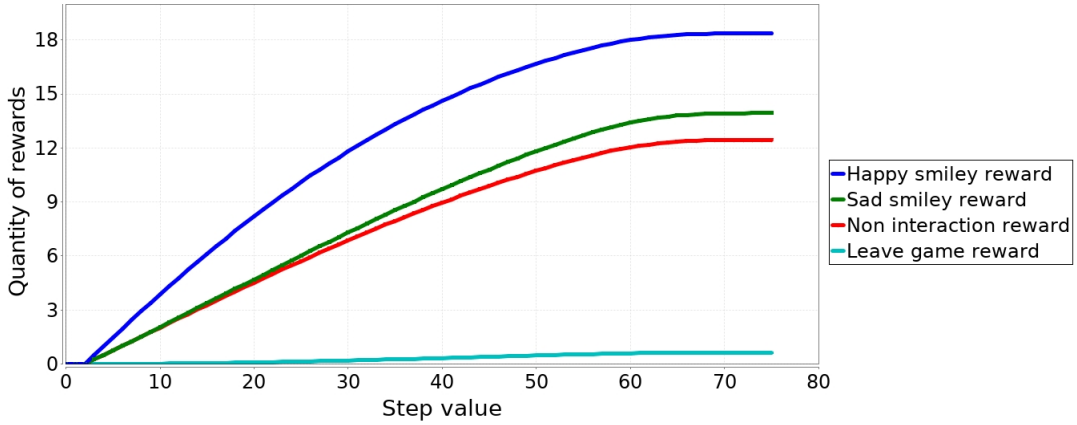


Figure 9: Average model checking results for rewards related to good answers, bad answers, non-interaction, and game leaving behaviors in the new version of the model.

Discussion on Figures 9 and 6. There are several changes in Figure 9 compared to Figure 6. The change of maximum reward accumulation is similar to the one already depicted in table 4a, but the shape of the curves changes. The curve of happy smiley reward is the most noticeable as the more time goes, the less rewards get accumulated. We can say that the accumulation decelerate. Sad smiley and non interaction rewards are now clearly separated. The *leave game* reward does not stay as low as in the previous version of the model.

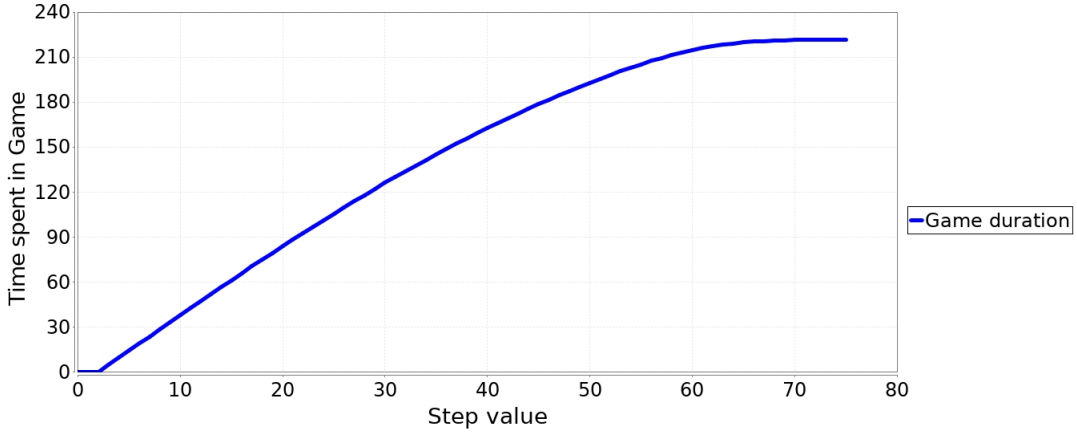


Figure 10: Average duration of the game obtained with model checking in the new version of the model.

This last observation explains Figure 10. In this figure the maximum game duration is below 220s, this is a direct consequence of the accumulation of *leave game* reward values.

Possible medical significance. These results indicate that most of the patients represented by this model will leave the game before the timer reaches 220s.

Remarks on the fatigue-aware Match Items game model. Since, according to the medical staff, this version is suitable to realistically represent the behavior of patients, the remaining two games have directly been modeled with this approach.

7. Recognition Game Model

The Recognition game is part of the MeMo Web platform (`https://games.memory-motivation.org/?lang=en`) which provides several serious games to train different neuro-cognitive functions. This game trains the memory functions. A set of pictures is displayed in front of the patient one by one. Most pictures of this set are unique, others are duplicated (one original and one duplicate). Pictures are randomly displayed to the patient. The duplicate of a picture must appear within a given distance from its original. This distance is a number of pictures varying in a range defined before the beginning of the game. The goal for the patient is to classify each picture as a "first-seen" picture (unique or original) by dragging it into a folder or as a duplicate by dragging it into a garbage bin.

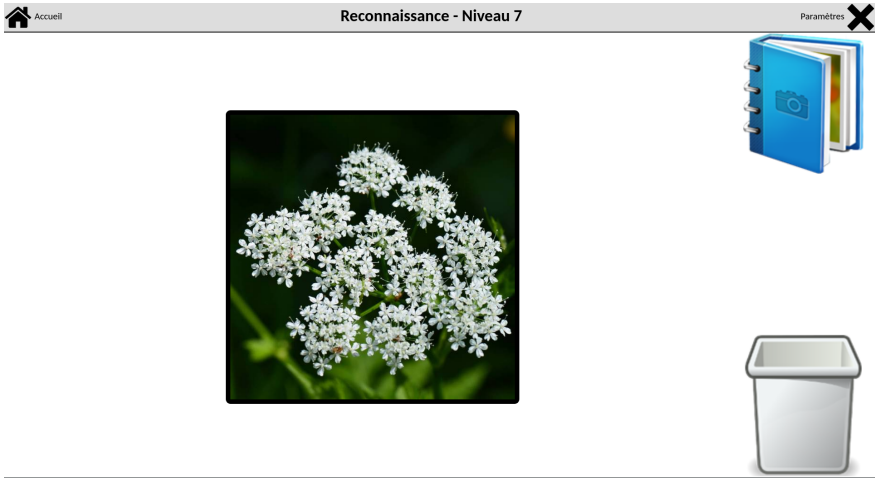


Figure 11: Display of the Recognition game, with a flower example of an intermediate complexity picture.

The difficulty increases with the number of unique, original, and duplicate pictures and with the parameters of the distance range (start point and size). The nature of the displayed pictures (which is defined by their subject, their details, their color range, etc.) also impacts the difficulty.

7.1. Model Design

To simplify the modeling task on this game we designed, with the help of our medical partners, nine exclusive levels of the game. The advantage of these customized levels is that we know the exact sequences of pictures for each of them, whereas in the original game the sequence is randomly generated for each game session. Each customized level includes 28 unique pictures, 6 originals and 6 duplicates and is associated with two tags. The first tag describes the subject of the pictures which can be "animals", "landscapes", or "flowers". The second tag describes three degrees of difficulty. The difficulty of a level takes into account the distance range between originals and duplicates and the similarity between the pictures as well as their significant details (both latter features will be referred as the picture complexity). Each level has a different combination of these two tags. To avoid a learning mechanism from the patients, different sets of levels have been created, each one containing different tag combinations. Moreover, tested patients should play this game only once a month, thus minimizing the memorization risk. In this paper, we will focus on the third level of difficulty, using flower pictures.

To model this game we decided to categorize each picture depending on its complexity. Two labels are thus associated with a picture. The first one corresponds to the information "unique", "original", or

(a) Easy picture.

(b) Intermediate picture.

(c) Difficult picture.

Figure 12: Example of pictures of various complexity. Picture 12c is categorized as difficult because it is displayed some time after 12b which is close in color and in flower shape.

"duplicate". The second describes the complexity of the picture and can be "easy", "intermediate", or "difficult" (see Figure 12).

The model of the game is a deterministic discrete finite automaton in which states correspond to the behavior of the patient for the current picture. The model takes into consideration four potential actions for each picture. The patient can: (1) give a good answer (rightfully drag the picture into the folder or into the bin); (2) give a bad answer (wrongfully drag the picture into the folder or into the bin); (3) hesitate before answering; (4) leave the game. The time spent hesitating can vary for a given patient depending on the picture. To represent such behavior, all states of the model representing hesitation are equipped with three different transitions: one is a loop back to the same state, whereas the others go to an "answering state" (good or wrong answer). In this model, each state representing an action of the patient is linked to other states by a probabilistic transition. Not only do these probabilities depend on the picture displayed but they also depend on the time elapsed since the beginning of the game. Thus, as in the Match Items model, the time dependent probabilities of the unwound model are pre-computed to take into account the patient's fatigue.

7.2. PRISM Implementation

To ensure that the PRISM implementation behaves as expected, we first created a simplified model containing fewer details than necessary. The advantage is that this first version is simple enough to be verified by all model checker engines available in PRISM. Afterwards, we implemented a second version with slightly more details. This second version could be verified using the explicit and the exact model checker engines. These two intermediate models were useful to verify the feasibility and the adequacy of this type of model to the properties we want to prove. The verification of the final version, presented below, requires the PRISM explicit engine.

Contrary to the previous game, the patient behavior is very dependent on the behavior of the game, that is, which picture is shown at what point during the game. It is a good software engineering practice to separate the two concerns and to design two different modules. In the same spirit we added an observer module to facilitate the computation of probabilities. This version is thus implemented with three PRISM modules. The first one describes the game, the second the patient, and the third is an observer to compute probabilities. The interactions between these modules are illustrated in Figure 13.

The **game module** has two variables, a Boolean variable and an integer counter. The Boolean variable (*game_on*) is initialized to *false* and is updated if one of the two following guards is *true*. The first guard, labeled *pressStart*, is *true* if, in the current state, *game_on* is *false*. In such a case, *pressStart* updates *game_on* to *true*. The second guard, labeled *endGame*, is *true* if, in the current state, *game_on* is *true* and the counter variable, called *nb_pictures*, reaches its maximal value. In this case, *endGame* updates *game_on* to *false*. The counter *nb_pictures* represents the number of pictures that have been displayed so far. Its range goes from 0 to 40 and it is updated with the validation of one of several guards. In this module, these guards are useful to: (i) label the updates differently depending on the picture displayed by the game; (ii) keep a code structure easy to read and to modify if needed. All updates are similar and

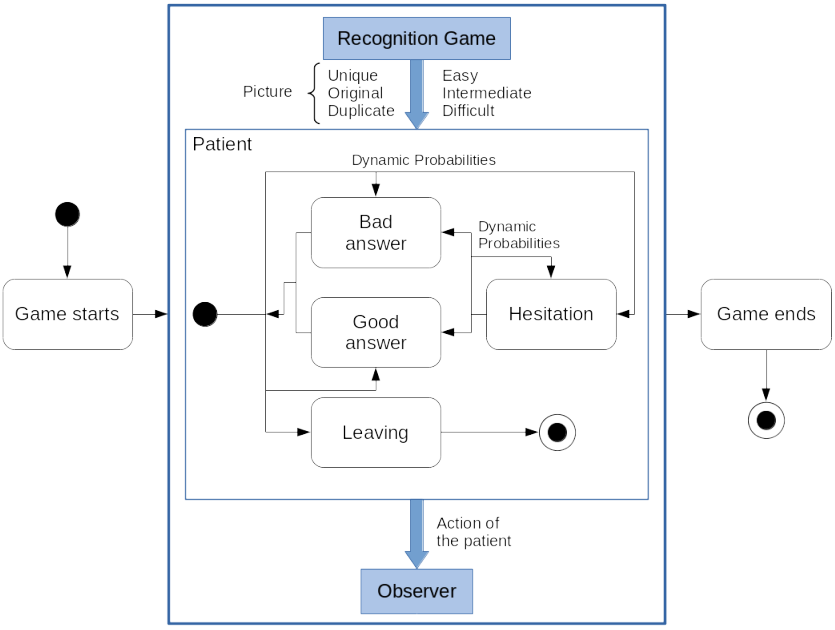


Figure 13: Recognition game module interactions.

increase the value of *nb_pictures* by one. For these updates we use five labels: *easyFolder*, *medFolder*, and *diffFolder* corresponding to (easy, medium or difficult) unique/original pictures, to be put in the folder and *medBin* and *diffBin* corresponding to duplicate pictures, to be put in the bin. A transition labeled *endLoop* is used to lead to a final state. This transition is triggered for sure when *nb_pictures* reaches 40 thanks to the synchronization with the patient module. Thus, an infinite loop between $game\_on = true$ and $game\_on = false$ is avoided. For instance, if the game module triggers the transition labeled *easyFolder*, the patient module (described below) transits with pre-computed probabilities that depend on the time elapsed in the game session.

The **patient module** is synchronized with the game module. That is why some of its labels are the same as the game module ones. This synchronization allows us to assign different probabilities to the transitions of the patient module, depending on the transition taken in the game module. The patient module has six Boolean variables describing the state of the patient: *finish_play* which indicates that the patient has stopped playing or has not finished yet; *front_screen* which indicates whether the patient is facing the screen or not; *choose_folder*, *choose_bin*, *quit_game*, and *hesitate* which indicate the action chosen by the patient for the current picture. If *hesitate* becomes *true*, a counter named *hesi_count* provides a measure of the "logical" hesitation time[3]; it increases its value until the patient makes a choice for the current picture or until the counter reaches its maximal value. In this model, for implementation simplicity, we assume that in this case a patient *has to* make a choice. Moreover, with the chosen parameters, if the patient enters an hesitation state she will have a better chance to succeed on the current picture. When the choice is given, *hesitate* becomes *false* and *hesi_count* is reset to 0. To store the nature of the picture during hesitation, we added five Boolean variables: *easy_fold*, *med_fold*, *diff_fold*, *med_bin* and *diff_bin*. Each one becomes *true* when the corresponding label of the game module is activated. For example, if the game module triggers a transition labeled *easyFolder*, variable *easy_fold* is updated to *true*. The patient module uses probabilities stored in external variables. These probabilities are associated with specific guards. This module is equipped with ten labels. Labels *pressStart*, *easyFolder*, *medFolder*, *diffFolder*, *medBin*, *diffBin*, *endGame*, and *endLoop* are used to synchronize with the game module. Labels *patientHesitate* and *quitGame*

[3]The time unit represented by this counter depends on the patient's wellness and on the difficulty of the game. It will be refined after statistical analysis of the results of real games.

are not synchronized with the game module.

The third module is the **observer**, it has only one variable, a counter ranging from 0 to the maximal number of actions that the patient can do. For each of the 40 pictures, the patient can hesitate from 0 to 5 times before executing an other action (good answer, bad answer or leave). Thus, the maximal number of actions per picture is 6 and for a whole session it is $40 * 6 = 240$. This counter is increased each time one of the labels *easyFolder*, *medFolder*, *diffFolder*, *medBin*, *diffBin* or *patientHesitate* is activated. The value of this counter is used by the external variables that compute the different probabilities corresponding to the actions of the patient. The code presented in listing 4 shows an example of how the probabilities are computed in this model. Note that all numerical values result from discussion with practitioners (Step 3 of Figure 1).

```
// Probability to drag the picture in the folder
formula p_easy_folder_f =
    0.90 - (4/30*(counter/240) + 2/30*(nb_pictures/40))
        + (0.02*hesi_count);
// Probability to drag the picture in the bin
formula p_easy_folder_b =
    0.10 + (4/30*(counter/240) + 2/30*(nb_pictures/40))
        - (0.02*hesi_count);

// Probability when taking into account
// hesitation or quitting the game
formula p_easy_folder_f_tot =
    p_easy_folder_f*(1 - (p_start_hesi + p_quit_game));
formula p_easy_folder_b_tot =
    p_easy_folder_b*(1 - (p_start_hesi + p_quit_game));

// Probability to start hesitating
formula p_start_hesi = 0.10
    +(1/(4*pow(2*0.3,1/2)))
        *pow(1.359,-pow(((counter+nb_pictures)-26)/4,2));

// Probability to quit the game
formula p_quit_game = 0.0002;
```

Listing 4: Probability calculation for an easy unique/original picture.

Listing 4 shows the probabilities for the four different actions. Here, the probabilities for giving a good or a bad answer are specific to an easy first seen (unique or original) picture. The results of formulas `p_easy_folder_f` and `p_easy_folder_b` give the order of magnitude of how much more (or fewer) chances the player has to give a good answer over a bad one. To normalize these results with other probabilities they are used in formulas `p_easy_folder_f_tot` and `p_easy_folder_b_tot` to compute the probabilities for the patient to give respectively a good or a bad answer over the other actions (which are leaving the game and hesitating). The results of formulas `p_easy_folder_f` and `p_easy_folder_b` are also used to compute the probabilities in the specific case where the maximum hesitation time has been reached (remember that in this case the patient must answer).

8. Temporal Logic Properties for Recognition Game and Results

This section presents some of the properties tested on the Recognition Game model as well as their results and analysis. To ease the comprehension of the following section, table 5 summarizes the correspondence between the model variables and what they represent.

Variable	Representation
finish_play	Boolean indicating if the patient is done playing the game or if she has not finished yet.
game_on	Boolean indicating if the game is on or off.
front_screen	Boolean indicating if the patient is in front of the screen or if she left.
quit_game	Boolean indicating if the patient has left the game before classifying the last picture.
counter	Integer variable of the observer module that gives the time spent in the current game.
nb_pictures	Integer variable counting the number of pictures that have been displayed.
choose_bin	Boolean variable indicating that the patient has classified the picture as a duplicate.
hesi_count	Integer variable counting the amount of time the patient hesitates on a given picture.

Table 5: Summary of Recognition Game model variables.

8.1. Model Verification

To verify this model and ensure it is sensibly implemented, a reachability property has been defined and tested. As the approach is similar to the Match Items game, it is not detailed any further. The model checker gave the expected result (a probability of 1 to reach the final state) within 0.038s.

8.2. Medically Oriented Properties

The following properties check the probability of various combinations of actions from the patient that may have an interesting medical interpretation.

Property 1. What is the probability for the patient to finish the game within 40 instants?

$$P =? \ [F \ (counter = 40 \ \& \ !front\_screen \ \& \ !quit\_game)]$$

Property 2. What is the probability to put all 40 pictures in the folder?

$$P =? \ [G \ ((!game\_on \mid (game\_on \ \& \ nb\_pictures = 0) \\ \mid (nb\_pictures >= 1 \ \& \ !choose\_bin)) \ \& \ !quit\_game)]$$

In this property, each term separated by OR operators (| in PRISM) corresponds to a state of the game. The first term specifies that the property is *true* before the game starts; the second specifies that the property is *true* when the game has started but has not displayed any picture yet; the third one specifies that the property is *true* only if, for each picture, the patient does not drag it into the bin. The last term specifies that the property is *true* only if the patient does not quit the game before its end.

Property 3. What is the probability for the patient to hesitate the longest possible time on at least one picture?

$$P =? \ [F \ (hesi\_count = 5)]$$

Property 4. What is the probability for the patient to hesitate the longest possible time on all the 40 pictures?

$$P =? \ [F \ (counter = 240)]$$

Property 5. What is the probability for the patient to stay in the game at least i instants?

$$P =? \ [F \ (counter = i)]$$

Property	Result	Time(seconds)
Property 1	1.0229×10^{-3}	0.011
Property 2	9.3047×10^{-4}	0.053
Property 3	2.9351×10^{-2}	0.043
Property 4	0	0.01
Property 5	1.5258×10^{-13}	0.03

Table 6: Results from Property 1 to 5. Result of Property 5 is given for $i = 85$.

Discussion. The results for these properties are displayed in table 6, together with their computing time.

Property 1 evaluates the probability to finish the game as quickly as possible, that is with no hesitations on any picture. This property uses variables from the *patient module* and from the *observer module*. As described in the previous section, *counter* gives an idea of the amount of time spent in the game. Combined with variables from the *patient module* (*front_screen* and *quit_game*), it allows us to check the probability to finish the game within a given amount of time. Since there are 40 pictures, the minimum amount of instants to finish the game (without wasting any instant in hesitation) is 40. The result for this property is close to 0.001 which is a rather high probability given that there are other options for the patient (leave the game or take more time to finish the game).

Property 2 checks the probability for the patient to put all the 40 pictures in the folder (including those supposed to go into the bin) thus ignoring the duplicates. This property is *true* if the game has not started yet, or if it has ended, or if it has started but no picture has been displayed yet, or if the game has started and if (regardless of the picture) the patient never dragged it into the bin. The goal of the last condition (*!quit_game*) is to consider only the case in which the patient has processed every single picture. The result for this property is close to 0.001, which is again a high probability.

Properties 3, 4, and 5 check combinations of actions related to the *hesitate* transitions.

Property 3 only uses variable *hesi_count*. This variable represents how long the patient has been hesitating on a given picture. The chosen value of 5 is the maximum value that *hesi_count* can take. The result of this property is close to 0.03. This result is high compared to the previous ones which can be explained by a less constrained property. Indeed, the previous properties checked a repetition of a behavior whereas Property 3 only checks if a behavior can be observed at least once in a game session. Therefore, this property indicates that the occurrence of a long hesitation within a single session is rather uncommon.

To verify this rarity, we implemented Property 4 that checks the probability for the patient to hesitate on each picture for the maximum amount of time (i.e., reaching *hesi_count*= 5 for each picture with the current configuration parameters). The reason why we used *counter* alone in this property is because its highest value is only reached if the *patient module* went through all transitions leading *hesi_count* to be equal to 5 for all the 40 pictures of the game. This maximum amount of logical time is $40 * 6 = 240$. This formula expresses the fact that, for each picture, *hesi_count* first takes the value 0 before being incremented 6 times by 1 at each step where the patient hesitates. The result obtained for Property 4 is 0.0 which was unexpected. Indeed, using the guided simulation tool of PRISM, such a state can be reached. Thus, we further investigated this phenomenon, by varying the value of *counter* in this property.

This led to Property 5 that uses variable i, an integer value. We found that, according to the model checker, there is no chance that the patient stays in the game for more than 85 instants. Indeed, for $i = 85$ the probability computed in property 5 is 1.52×10^{-13} but for $i = 86$ the result becomes 0.0. These results may indicate that the probability to have the variable *counter* equal to 86 is so low that the model checking engine considers it as a 0.

Possible medical significance. All these results can be interpreted as the ones of a patient who does not take enough time to reflect on the picture classification, therefore making more mistakes than a healthy subject. This behavior could be induced by a lack of motivation of the patient, which is part of an MCI diagnosis. Thus, the patient would need a deeper examination from medical practitioners.

The next property is a reward-based one to learn more about the general behavior of a patient represented by this model.

Property 6. What is the amount of good answered pictures accumulated during a game session?

$$R\{"good\_answered"\} =? [F\ finish\_play\ \&\ !front\_screen]$$

Discussion. Property 6 checks the average amount of good responses given by a patient. The results for this property and for other behavior-related rewards are displayed in table 7, together with their computing time.

Reward	Result	Time(seconds)
"good_answered"	30.0026	0.181
"good_answer_fold"	28.7624	0.186
"good_answer_bin"	1.2402	0.17
"total_time_hesitate"	7.523	0.158

Table 7: Results of Property 6 for various rewards.

The model checker finds an average of 30 good answers out of 40 for this model. Among these 30 good answers, 29 are unique or original pictures placed in the folder for only 1 duplicate placed in the bin. The "total_time_hesitate" reward shows that the modeled patient takes approximately only 7 instants of hesitation in the whole session, which is a very small amount.

Possible medical significance. These results are consistent with our previous interpretation. The modeled patient makes mistakes on nearly all duplicate pictures and even on unique and original pictures. This may indicate some memorization as well as motivation issues.

8.3. Evolution of probabilities and cumulative rewards

This subsection presents two diagrams that provide a global view of a typical game session along time, contrarily to the previous properties that gave a result at a given point in time. These diagrams were obtained with the "experiment" feature of PRISM. They summarize the evolution of the model concerning the ability of the patient to give good answers or to hesitate on a picture.

Figure 14 presents the evolution of the probabilities during a game session, based on the results of the two following properties.

Property 7. What is the probability to hesitate on picture number i?

$$P =?[F(nb\_pictures = i\ \&\ hesitate)]$$

Property 8. What is the probability to give a good answer on picture number i?

$$P =?[F(nb\_pictures = i\ \& \\ ((choose\_folder\ \&\ (nb\_pictures = x))\ |\ (choose\_bin\ \&\ (nb\_pictures = y))))]$$

In this property, x is an abstract representation for all unique or original pictures and y is the same for all duplicate pictures. This formula uses a simplified version of the syntax of PRISM. The real formula is too big to be displayed and not easy to read. In the real syntax, all values of x and y are known and must be explicitly specified.

Discussion. Figure 14 shows how probabilities are distributed and how they interact with each other. The probability for the patient to hesitate takes a Gaussian-like form centered on picture 13. Picture 12 and 31 both correspond to difficult duplicate pictures. The probability to give a good answer on picture 12 is higher because it appears sooner than picture 31, but also because the patient has a high chance to hesitate thus taking time to think about the answer.

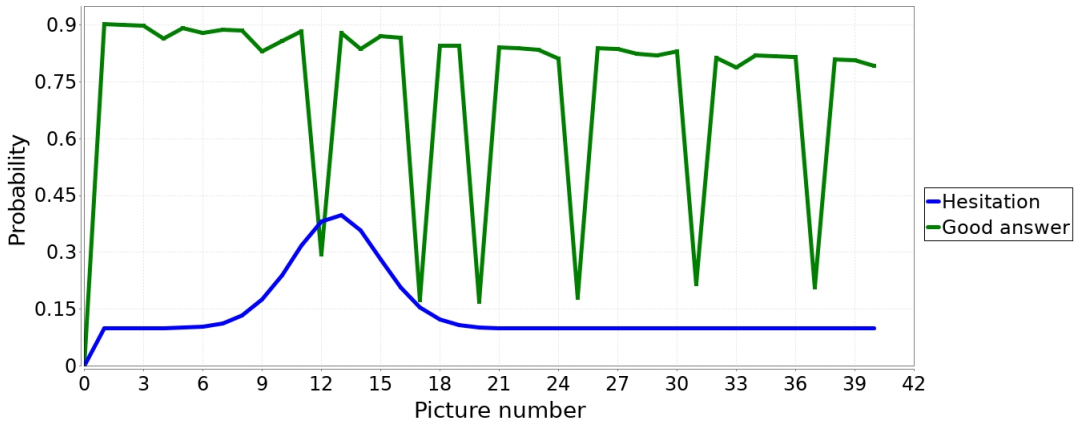


Figure 14: Probabilities to give a good answer and to hesitate per picture.

Possible medical significance. In this figure, we clearly observe the behavior described earlier: a patient who is too hasty to finish the game and gives too many bad answers.

The second diagram in Figure 15 shows the accumulated values of rewards (using the C operator defined in section 3) during a game session. For this purpose, we use the following property:

Property 9. What is the amount of good answers within i steps?

$$R\{\text{"}good\_answered\text{"}\} =? [C <= i]$$

We applied this property to other rewards: *"good_answer_fold"* for the number of good answers on unique and original pictures, *"good_answer_bin"* for the number of good answers on duplicate pictures, and *"total_time_hesitate"* for the number of logical instants when the patient hesitates.

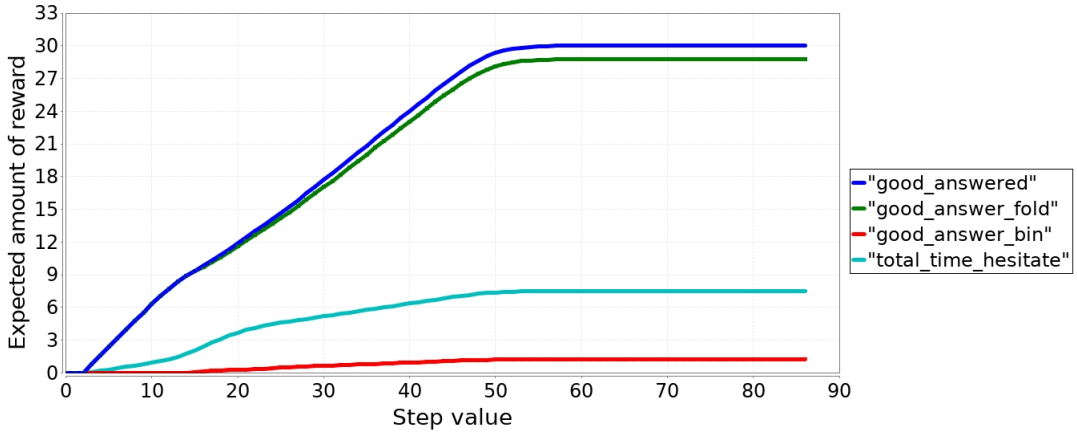


Figure 15: Average model checking results for rewards related to good answers and hesitation.

Discussion. Figure 15 shows the influences of the probabilities on the reward values. Indeed, there is a high increase of the value of *"total_time_hesitate"* reward within instants 10 and 20 preceding a weak increase until the end of the game. For the accumulated value related to good answers, the behavior is different: the highest increase takes place within instants 3 and 14. The accumulation of rewards *"good_answered"* and *"good_answer_fold"* have exactly the same evolution until instant 15. After this instant, the *"good_answer_bin"* reward value starts to accumulate.

Possible medical significance. This figure (15) shows at which point in time a patient is expected to accumulate good answers or when she is more likely to hesitate. Another thing that can be observed is that this patient finishes the game rather quickly around instant 55 which is a consequence of a low hesitation rate.

Remarks on the Recognition game model. This model was the first directly implemented with several modules. This separation of concerns is a good engineering practice. The previous Match Items Game was rather simple and did not require this kind of separation, but for heavier models, it eases implementation, readability, debugging, and extensibility of the code. This model was also the first to be directly implemented with pre-computed time dependent probabilities. An observer module collects information about the current game session to compute these time dependent probabilities. They are computed by functions more complex than in the Match Items game and this is a challenge in the implementation of the model. It was also the first model to require the use of the PRISM explicit model-checking engine due to its large state space. The other model checking engines utterly fail to give a result.

9. Inhibitory Control Game Model

The Inhibitory Control game is part of the TAE Web platform (`https://cmrr-nice.fr/lab/tae/`), which provides several serious games to train and evaluate various cognitive functions going from reflexes to motivation. Among these games, the one we study targets the inhibitory control, one of the cognitive function affected in several neuro-degenerative diseases such as Parkinson or Alzheimer. Two versions of this game are available on this platform and differ only in their settings.

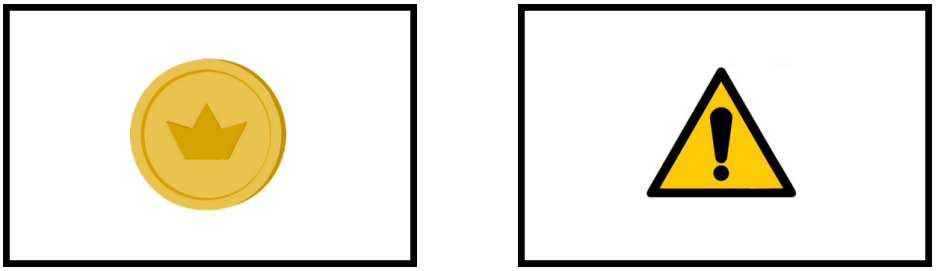


Figure 16: Display of the Inhibitory Control game, with both signals. The coin is the target and the warning sign is the decoy.

In this game, the patient is asked to click exactly one time, as fast as possible, on a target that appears for a short period of time at the center of the screen (see Figure 16 left). Knowing that human's visual reaction time for a healthy adult performing a "go/no-go" task is around 259 ms [38], the game takes into consideration a possible anticipation from the patients and refuses to award a point if they clicked before the end of a lap of 259 ms after the occurrence of the target. According to the practitioners, this delay is suitable in our case because reaction time increases with age and our patients are significantly older than the panel studied in [38]. As long as the patient did not click and as long as a new signal did not appear, the patient can click and score a point. Since this game is inspired by the go/no-go task, a second type of signal exists: a "decoy" (Figure 16 right). The decoy is a warning sign that may appear instead of the target. In this case the patient must *not* click on it, despite a color similar to the target one. This signal tests the inhibitory control function. The difficulty relies on both the brevity of the occurrence of signals and on their common color. A patient who clicks on every decoy may have an inhibitory control problem. The time between signal occurrences is random (within a given range, defined at game configuration).

Due to the difficulty of the task, the developers of the game included a short training phase at the beginning of the game session. During this phase, three targets and a decoy are presented to the patient, but the game does not register the patient's results. Then the "real" game starts and the results are stored.

The settings we chose for the Markov chain model are the following: 300 ms of display time for each signal, 2000 ± 1500 ms separating two signals, 10 target occurrences, 3 training occurrences of target, 5 decoy occurrences, and 1 training instance of decoy.

9.1. Model Design

To model this game we consider it as a succession of events, each one corresponding to the occurrence of a signal; remember that the amount of time between two events is not constant. Since this model only transits on signal occurrences, to determine the current patient scenario we need to know the previous action of the patient. Thus, for each event, we take into account what was the action of the patient since the last event.

We consider six potential actions: (1) no click; (2) one click but with anticipation; (3) one fast click; (4) one slow click; (5) one double click; (6) several clicks regardless of the frequency and the reason. In this model, we do not consider the possibility to prematurely leave the game because we assume that the patient will not want to leave within the rather short duration of the game (a simple computation shows that the maximum duration is 72.2 s). The initial probabilities associated with the different actions depend on two factors: the nature of the previous signal and the time elapsed between the occurrence of the new signal and the occurrence of its predecessor. To represent this time in a Markov chain we consider that the time elapsed between two signals can be short, medium, or long. This information is reflected in the label names of the five transitions between states. The nature of the signal is reflected by both the label and the state variables. The probabilities of the different actions also depend on the time spent in the game so far. As for the two previous games, these probabilities are pre-computed.

9.2. PRISM Implementation

The implementation follows the same organization as in the previous game: a game module, a patient module, and an observer module.

The **game module** is composed of seven Boolean and two integer variables. The integer variables, named *num_ targ* and *num_ deco*, are counters which are incremented for each target, respectively decoy, that appears during the game. The first Boolean variable is *game_ on*, it indicates whether the game started or ended. The next three Boolean variables (*fast_ occurrence*, *medi_ occurrence* and *slow_ occurrence*) determine if the next signal will appear in a short, medium, or long time span. When one of these Boolean variables is *true*, the next transition will hold the corresponding label, e.g., if the variable *fast_ occurrence* is *true* then the next transition will hold a label containing the term *fast* in its name. The last three Boolean variables work in the same way. They are named *next_ targ*, *next_ deco*, and *next_ end*. As their name suggest, they determine if the next transition will lead to a target or to a decoy or if the game will reach its end. Both the nature of the next signal and its speed are decided using probabilities. For each signal, the probabilities for it to be fast, medium, or slow are identical ($\frac{1}{3}$). The probability for the next signal to be a target or a decoy depends on the number of remaining signals. The formulas in listing 5 compute these probabilities:

```
formula proba_next_targ = (num_targ_max - num_targ) / (num_sign_max - num_sign);

formula proba_next_deco = (num_deco_max - num_deco) / (num_sign_max - num_sign);
```

Listing 5: Computation of the probability of a target or a decoy to appear.

where *num_ targ_ max*, *num_ deco_ max*, and *num_ sign_ max* are respectively the maximum number of targets, the maximum number of decoys, and their sum; *num_ targ*, *num_ deco*, and *num_ sign* are respectively the number of targets, the number of decoys and the total number of signals that have appeared so far in the session.

In this implementation, we decided to create twenty-four transition labels, to consider all possible combinations of features related to patient actions and signals (such as occurrence speed, type of signal, etc.). Some labels are common with the previous model (*pressStart*, *endGame*, and *endLoop*) and others are specific to this game. Among these specific labels, the *transiting* one is associated with transitions updating some of the Boolean variables previously introduced (*fast_ occurrence*, *medi_ occurrence*, *slow_ occurrence*, *next_ targ*, *next_ deco*, and *next_ end*). The next labels are associated with transitions leading either to a new signal or to the end of the game.

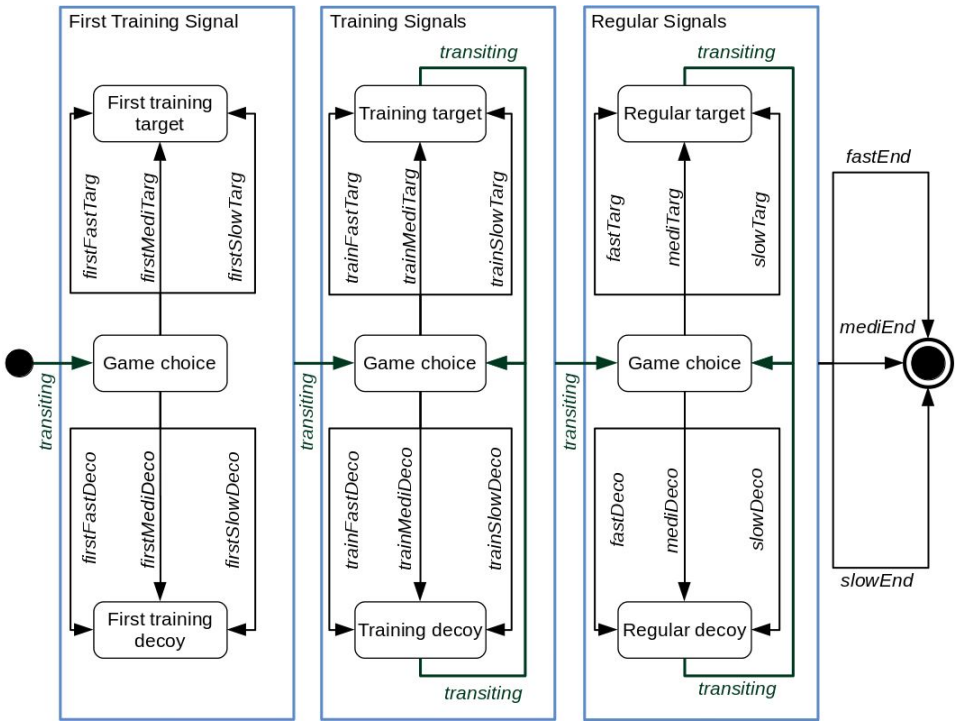


Figure 17: Simplified automaton of the Inhibitory Control game.

Figure 17 depicts the three phases of the game and the main transitions of the model. When starting the game, the speed and the nature of the first signal are automatically selected as one of the transitions associated with labels *firstFastTarg*, *firstMediTarg*, *firstSlowTarg*, *firstFastDeco*, *firstMediDeco*, or *firstSlowDeco*. These label names show the combination of several keywords: *fast*, *medi*, and *slow* for a fast, medium, or slow occurrence of the signal; *targ* and *deco* for the occurrence of a target or of a decoy. These combinations are reused for twelve of the other labels: *trainFastTarg*, *trainMediTarg*, *trainSlowTarg*, *trainFastDeco*, *trainMediDeco*, *trainSlowDeco*, *fastTarg*, *mediTarg*, *slowTarg*, *fastDeco*, *mediDeco*, and *slowDeco*. The six labels containing the keyword *train* correspond to the training phase, while the six others correspond to the "regular" game phase. Finally, after the last signal appeared, some time is left to give the patient enough time to react. This span of time is similar to the one between signals. Therefore, all transitions leading to the end of the game hold one of the labels *fastEnd*, *mediEnd*, or *slowEnd*. All the labels introduced in this game module are used to synchronize with the patient module. Figure 17 summarizes these labels.

The **patient module** is composed of fifteen Boolean variables. The first two ones are *finish_play* and *front_screen*. Depending on their combination, we can know if the patient: (i) has not yet played ($finish\_play = false \ \& \ front\_screen = false$); (ii) is playing ($finish\_play = false \ \& \ front\_screen = true$); (iii) has finished ($finish\_play = true \ \& \ front\_screen = false$).

The next six Boolean variables are used to describe the actions of the patient between two consecutive signals. They are named after their corresponding actions: *not_click*, *anticipate*, *click_fast*, *click_slow*, *double_click*, and *several_click*.

Remember that the patient has to click once when a target appears, and must not click at all if no signal was sent before (which is the case at the beginning of the game) or if a decoy appears. In this model, for each new signal, we check how the patient behaved for the previous one. It is thus crucial to memorize what was the previous signal to determine whether the patient displayed the correct behavior or not. That is why the goal of the seven remaining Boolean variables is to memorize the previous signal. This information is transmitted from one state to the other to validate the correctness of the patient's action. These seven variables are *curr_train*, *curr_targ*, *curr_deco*, *prev_train*, *prev_targ*, *prev_deco*, and *prev_none*. Variables

curr_ targ and *curr_ deco* are *true* if the model goes through transitions leading to a target or a decoy. The same goes for *curr_ train* if the model goes through transitions leading to a training signal. *prev_ none* is *true* only at the beginning of the game and is set to *false* when the game takes a transition associated with the *transiting* label. Every transition with this *transiting* label leads to an update of the variables *prev_ train*, *prev_ targ*, and *prev_ deco* with the values of *curr_ train*, *curr_ targ*, and *curr_ deco*. To manage every transition of this model, we had to spell them out for each possible combination of the following elements: nature of the previous signal, nature of the new signal, and speed of occurrence of the new signal. When the end of the game is about to be reached, we consider the combination of these elements: nature of the previous signal and speed at which the end screen will appear. Depending on the transition, the probabilities associated with the actions vary.

The probability for the *anticipate* action is computed taking into account the time lap preceding the previous signal. The longer it was, the higher the probability, inducing a global decrease of the probabilities of all the other actions. The probabilities for *unique click* (fast or slow), *several click*, *double-click* and *no click* actions mainly depend on the nature of the signal and on the elapsed time since the beginning of the game. These probabilities are computed with a different function depending on the phase of the game (first training signal, training phase, regular phase). These functions refer to variables belonging to the observer module.

The **observer module** is composed of eight integer variables and a Boolean one. The variables *time_ betw_ sign* and *prev_ time_ betw_ sign* both range from 1 to 3. Depending on the speed of a signal, they take one of the values of this range. If the signal is fast, the value is 1; if it is medium, the value is 2, and if it is slow, the value is 3. These values correspond to a logical time, that will be mapped on real time values, thanks to configuration constants (see the beginning of this section).

When going through transitions that hold the label *transiting*, *prev_ time_ betw_ sign* takes the current value of *time_ betw_ sign*. The variable *total_ time* ranges from 0 to the maximum duration of the game. This maximum is obtained when all signals occur slowly. Since there are 19 signals plus the end of the game (thus 20 signals) and since each of them has a non-zero probability to be slow (represented by the value 3), the maximum (logical) duration is 60 (20×3).

Variable *total_ time* is incremented by 1, 2, or 3 logical time units depending on the speed of the signal. Variable *num_ action* is an integer ranging from 0 to the total number of signals plus one (because we consider also the potential action performed before the occurrence of the first signal). *num_ act_ targ* and *num_ act_ deco* are both integer variables ranging from 0 to the maximum number of occurrences of the corresponding signal. Although these three variables are not mandatory for the model definition, they are necessary to express some of the behavioral properties in the next section. To increment *num_ act_ targ* and *num_ act_ deco* we use variables *memo_ targ* and *memo_ deco*, which can take the value 0 or 1 depending on the nature of the previous signal. For example, if the current signal is a target, *memo_ targ* is updated to 1 while *memo_ deco* is updated to 0. In the next transition, *num_ act_ targ* will be updated to $num\_act\_targ + memo\_targ$ and *num_ act_ deco* will be updated to $num\_act\_deco + memo\_deco$.

The Boolean variable *transiting* is *true* when the model enters a transiting state. The purpose of this variable is to facilitate the expression of interesting properties in the next section.

To compute the probabilities of the patient actions, we use three variables: one from the game module (*num_ sign*), the others from the observer module (*total_ time* and *prev_ time_ betw_ sign*). This paragraph only presents probabilities expected after the end of the training phase. Listing 6 shows the probabilities associated with most of the actions that a patient can perform on the apparition of a target.

```
//probability of anticipation for the previous target
formula proba_anticipate = 0.05
    +(0.05 * prev_time_betw_sign/time_betw_sign_max)
    +(0.10 * total_time/total_time_max)
    +(0.05 * num_sign/num_sign_max);

const mu = 34;
const sigma = 9;
```

```
//probability of click exactly once for the previous target
formula gaussian_probability_targ = (1/(sigma * pow(2 * 0.2, 1/2)))
    * pow(1.359, -pow(((num_sign + total_time)- mu)/sigma, 2));

//probability of fast click exactly once for the previous target
formula proba_good_click_fast = (0.25 + (gaussian_probability_targ)
    -(0.08 *((num_sign + total_time)/50)))
    *(1 - proba_anticipate);

//probability of slow click exactly once for the previous target
formula proba_good_click_slow = ((0.5 - gaussian_probability_targ)
    - (0.05*((num_sign + total_time)/50)))*(1 - proba_anticipate);

//probability of more than two clicks for the previous target
formula proba_several_click_targ = 0.175
    -(0.0075*(150/(num_sign + total_time)));

//probability of no click at all for the previous target
formula proba_bad_not_click = 1/3*(
    1 - proba_good_click_fast - proba_good_click_slow
    - proba_several_click_targ - proba_anticipate
    );
```

Listing 6: Probability computing for a target appearing after the training phase.

Formula *proba_ anticipate* gives the probability for the patient to click on the screen in anticipation. This probability depends on three factors: the time elapsed between the previous signal and the signal that just appeared, the total time that has elapsed since the beginning of the game, and the number of signals that have appeared since the beginning of the game. Formula *gaussian_ probability_ targ* helps compute the Gaussian-like functions corresponding to the actions of a good fast and a good slow click (respectively *proba_ good_ click_ fast* and *proba_ good_ click_ slow*). In these three formulas, many constants are used so that the two last equations may fit the expected behaviors. This massive use of constants, necessary in PRISM to represent such Gaussian-like functions, may introduce difficulties for the future step of model calibration (Step 6 of Figure 1). Formulas *proba_ good_ click_ fast* and *proba_ good_ click_ slow* are both normalized with respect to the probability for the patient to anticipate (*proba_ anticipate*). Formula *proba_ several_ click_ targ* also depends on several constants to fit the expected behavior: in this case, practitioners foresee, as the game goes, a slight increase of repeated clicks. Formula *proba_ bad_ not_ click* depends on all probabilities previously computed.

Once again, all formulas (especially their numerical parameters) have been defined *a priori* to correspond to behaviors expected by medical practitioners (see Step 1 of Figure 1). All the parameters will be adapted with respect to the results of clinical experimentation.

10. Temporal Logic Properties for Inhibition Control Game and Results

To ease the comprehension of the following section, table 8 summarizes the correspondence between the model variables and what they represent.

Since the reachability property is similar to the one of the first model (see section 5), we will not describe it in this section. For this model, the model checker takes 1.225s to check this reachability property.

Variable	Representation
prev_ targ *prev_ deco*	Boolean indicating if the previous signal was a target or a decoy.
prev_ signal	Variable name used in properties to refer to one of the following variables: *prev_ targ*, *prev_ deco*.
transiting	Boolean indicating if the model is in a transition state
click_ fast *click_ slow*	Boolean indicating if the patient clicked in a fast or slow way for the previous signal.
several_ click	Boolean indicating if the patient clicked more than once for the previous signal (excluding a single double clicking).
not_ click	Boolean indicating if the patient did not click at all for the previous signal.
double_ click	Boolean indicating if the patient double clicked once for the previous signal.
action	Variable name used in properties to refer to one or more of the following variables: *click_ fast*, *click_ slow*, *several_ click*, *not_ click* or *double_ click*
num_ sign	Integer variable counting the number of signals that have been displayed.
num_ action	Integer variable counting the number of actions that have been done by the patient.
num_ act_ targ *num_ act_ deco*	Integer variables counting the number of actions done by the patient for a target or for a decoy.
num_ act	Variable name used in properties to refer to one of the following variables: *num_ action*, *num_ act_ targ* or *num_ act_ deco*.
next_ end	Boolean variable indicating that the end of the game is reached.
game_ on	Boolean indicating if the game is on or off.

Table 8: Summary of Inhibitory Control Game model variables.

10.1. Medically Oriented properties

The following properties check how the patient performs on this game.

Property 1. What is the probability for the patient to click exactly once on each target?

$$P =? [G\ (prev\_targ\ \&\ !transiting) => (click\_fast \mid click\_slow)]$$

Property 2. What is the probability for the patient to click several times on the same decoy?

$$P =? [F\ (prev\_deco\ \&\ !transiting\ \&\ several\_click)]$$

Property 3. What is the probability for the patient to click several times on each decoy?

$$P =? [G\ (prev\_deco\ \&\ !transiting) => (several\_click)]$$

Property 4. What is the probability for the patient to click several times on each signal?

$$P =? [G\ ((prev\_deco \mid prev\_targ)\ \&\ !transiting) => (several\_click)]$$

Property 5. What is the probability for the patient to do the right action on each signal until the thirteenth?

$$P =? [(((prev\_targ\ \&\ !transiting) => (click\_fast \mid click\_slow))\ \&$$

$$((prev\_deco \ \& \ !transiting) => (not\_click))) \ U \ (num\_sign = 13)]$$

Property 6. What is the probability for the patient to do the right action on each signal from the thirteenth until the last signal?

$$P =? \ [(((num\_sign >= 13 \ \& \ prev\_targ \ \& \ !transiting) \ => \ (click\_fast \mid click\_slow)) \ \&$$
$$((num\_sign >= 13 \ \& \ prev\_deco \ \& \ !transiting) \ => \ (not\_click))) \ U \ (next\_end \ \& \ !game\_on)]$$

Discussion. The results for these properties are displayed in table 9, together with their computing time.

Property	Result	Time(seconds)
Property 1	4.7734×10^{-4}	0.446
Property 2	2.5042×10^{-1}	0.474
Property 3	5.4716×10^{-7}	0.415
Property 4	7.2736×10^{-8}	0.339
Property 5	1.1900×10^{-4}	0.272
Property 6	2.1314×10^{-4}	0.53

Table 9: Results from Property 1 to 6.

The probability obtained for Property 1 is rather low (of the order of 5×10^{-4}): this is due to the repetition of the action on each target.

Property 2 computes the probability for the patient to click several times on at least one decoy, which is an incorrect action. The result is 0.25 which is a high probability. To draw a parallel with Property 1 we consider the probability to click several times on each decoy, which is Property 3. This property gives 5.47×10^{-7} which is far lower than the result of Property 1 by a factor of 10^3 . This was to be expected since the probability of clicking several times on each decoy is close to 0.05.

Property 4 goes further by checking the probability for the patient to click several times on each signal. The result for this property is 7.27×10^{-8}, which is, as expected, lower than Property 3.

Properties 5 and 6 check the probabilities for the patient to perform the right action, whatever the signal is, within a range of signals. Property 5 checks this probability for the range going from the beginning of the game until the apparition of the thirteenth signal. The answer is 1.19×10^{-4}, which is consistent with the result obtained for Property 1. Property 6 checks the probability to perform the right actions from the thirteenth signal until the end of the game. The probability for this property is 2.13×10^{-4}. The probability to perform a correct action before or after the thirteenth signal are really close. Given that after this signal there are six signals left, the chances to give good answers after the thirteenth signal are lower than before.

Possible medical significance. Even though the patient has low chances to perform one of the worst behaviors (clicking several time on each picture would indicate some severe damages to the inhibitory control functions), she also has low chances to give a good performance. According to medical practitioners, this behavior could be observed on patients suffering from early stages of dementia diseases such as Alzheimer or Parkinson.

The next property is reward-based and gives an idea of the overall performance of a patient during the game.

Property 7. What is the average accumulation of good answers on targets at the end of the game?

$$R\{"good\_on\_target"\} =? \ [F \ (!game\_on \ \& \ next\_end)]$$

Discussion. Property 7 checks the average amount of good actions done by a patient on a target. The results for this property and for other behavior-related rewards are displayed in table 10, together with their computing time.

The model checker takes approximately 1.5 seconds to give the result of these properties. The reward *"good_on_signal"* shows that the average amount of good answers for a patient represented by this model is 6 out of the 15 non-training signals, which is pretty low.

Property	Result	Time(seconds)
"good_on_target"	5.55	1.637
"good_on_decoy"	0.86	1.459
"good_on_signal"	6.42	1.785
"bad_on_target"	4.44	1.448
"bad_on_decoy"	4.13	1.855
"bad_on_signal"	8.57	1.437

Table 10: Results of Property 7 for various rewards.

Possible medical significance. The reward related results are consistent with all other results obtained with this game. The modeled patient has issues playing this game and probably needs further examination regarding the inhibitory control function.

10.2. Evolution of probabilities and cumulative rewards

In this subsection, we present several diagrams obtained with the "run experiment" tool of PRISM. These diagrams summarize the evolution of the model along time concerning the ability of the patient to perform good or bad actions and provide a better understanding on how the fatigue can influence the patient's behavior.

The first three diagrams present the evolution of the probabilities during the game. Figures 18, 19, and 20 present the results of the following property.

Property 8. What is the probability to perform "action" for the game output number i?

$$P =? [\ F\ (num\_act = i\ \&\ prev\_signal\ \&\ (action)\ \&\ !transiting)]$$

In this property, *num_act* and *action* are generic variable names that may refer to any variable described in table 8.

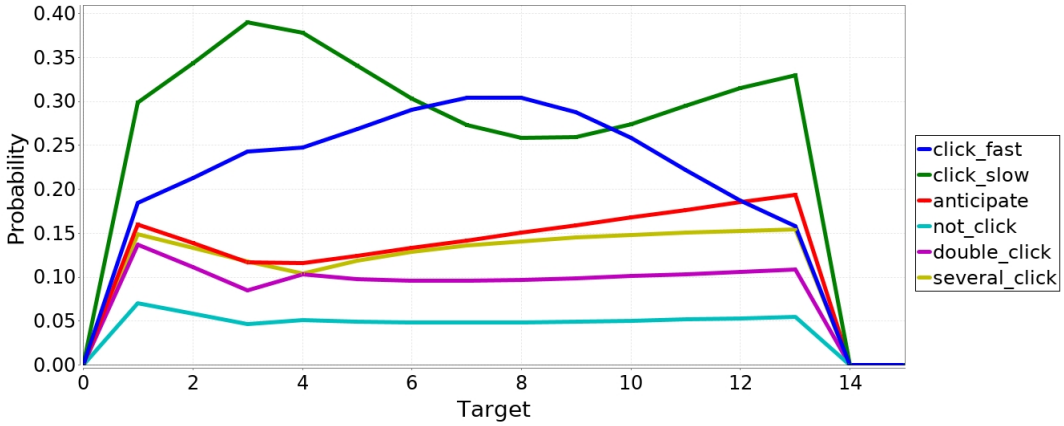


Figure 18: Probability to perform an action for a specific target.

Figure 18 shows the results for Property 8 with *num_act* as *num_act_targ*, *prev_signal* as *prev_targ* and *action* as a Boolean presented in the legend of the figure. This figure depicts the training ability of the patient. Indeed, after the last training target (target number 3), the patient still progresses and has a higher probability to click fast on the following targets. However, only the speed improves, not the correctness.

After a certain point in time (the eighth target), the fatigue of the patient shows its effect with a drastic decrease of the probability to click fast for the benefit of clicking slow or of performing a bad action. The probability to click several times on a target reaches a plateau around 0.15, which means that, even though this probability increased with the fatigue, it is still a rare phenomenon at the end of the game.

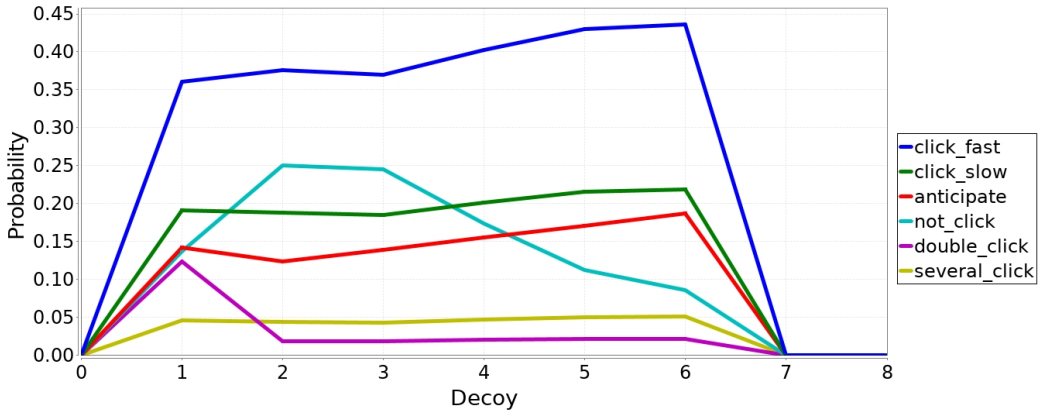


Figure 19: Probability to perform an action for a specific decoy.

Figure 19 shows the results for Property 8 with *num_act* as *num_act_deco* and *prev_signal* as *prev_deco*. In this figure the peak performance of the patient is around the second and third decoy. After that, there is a drastic decrease of the patient's performance in term of inhibitory control.

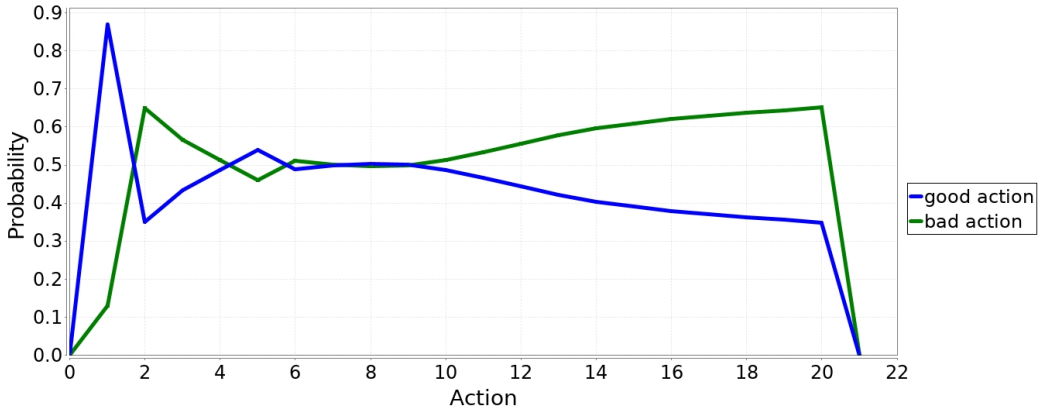


Figure 20: Probability to perform a good or a bad action for each instant when an action is expected from the patient. Here, the action number 1 on the horizontal axis is the one recorded before the occurrence of the first signal.

Figure 20 describes the results for Property 8 with *num_act* as *num_action* and *prev_signal* and *action* as a combination of *prev_none*, *prev_targ* and *prev_deco* with their corresponding good or bad actions. This figure gives a global idea of the patient's behavior and we clearly see the good effect of the training signals between action 2 and 6 but, starting from action 10, the patient does more bad actions than good ones.

The next figures present the result for the following cumulative reward property:

Property 9. What is the amount of good answers within i steps?

$$R\{\text{"}good\_on\_target\text{"}\} =? [C <= i]$$

We applied the same property to the following rewards: *"good_on_target"*, *"good_on_decoy"*, *"good_on_signal"* (which is the sum of the previous ones), *"bad_on_target"*, *"bad_on_decoy"*, and *"bad_on_signal"* summing the results of *"bad_on_target"* and *"bad_on_decoy"*.

These two figures display "stair-like" diagrams. This is due to the presence of transition states in which the model does not accumulate any reward. We see the consequences of the results from Figure 20.

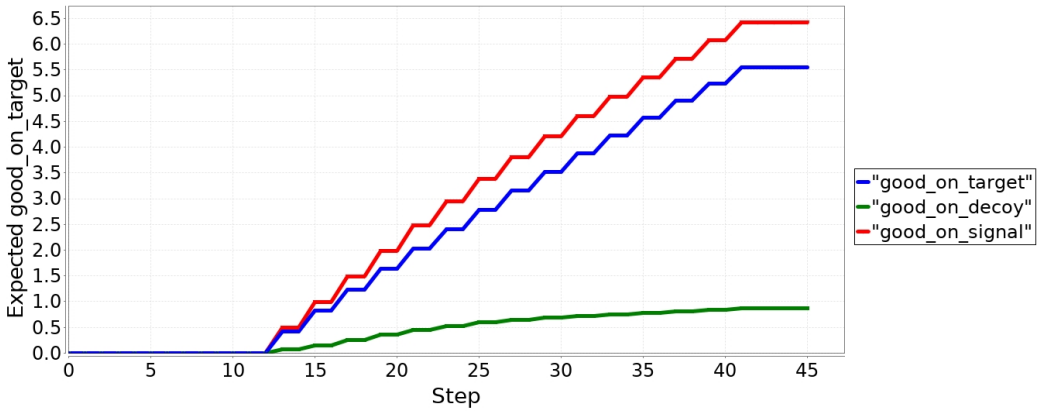


Figure 21: Average model checking results for rewards related to good actions.

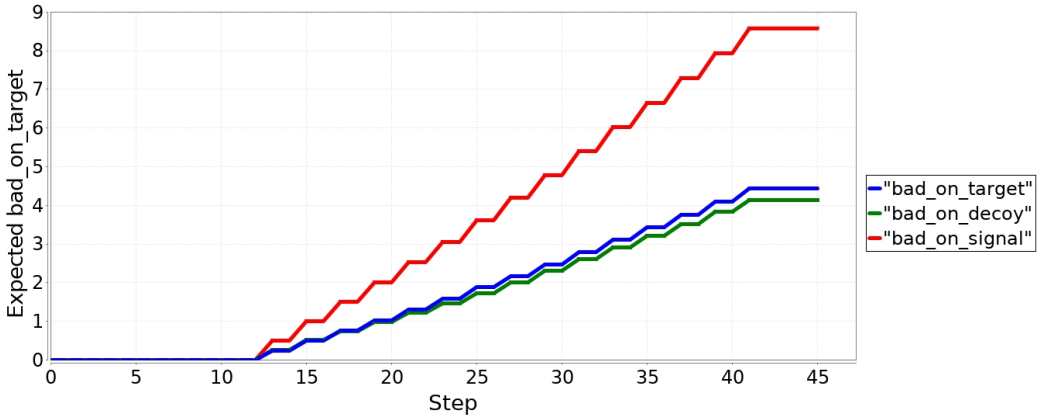


Figure 22: Average model checking results for rewards related to bad actions.

Indeed, the accumulation of *"good_on_decoy"* value around Step 22 decelerates while the accumulation of *bad_on_signal* value accelerates. To summarize, while Figure 20 gives a clear and precise idea of the probability distribution within a whole game session, Figures 21 and 22 give views on the consequences of this distribution. These three figures can thus be used to verify and calibrate the model. Indeed, in a first step, Figures 21 and 22 can be compared to the average behavior of patients to simplify fault-finding. If they do not match, they give a hint on the position of the error. Then, this position can be approximately found on Figure 20 and helps to understand where and how the probability distribution is erroneous.

Remarks on the Inhibitory Control game model. Surprisingly, although this game is very simple to describe and only involves two signals, it was more difficult to model than the previous ones. This is mainly due to the many ways it may fail as well as succeed. This difficulty also pushed PRISM to its limits as the time for model-checking is rather longer than for the two first games (up to 1 or 2 seconds compared to a few hundreds of seconds for the other games). This may not seem significant, but adding one parameter or modifying one (for example, setting num_targ_max to $20 + num\_targ\_training$ instead of $10 + num\_targ\_training$) leads to a full night computation and does not even terminate (the PRISM process consumes most of the CPU time, yet far from saturating the memory).

11. Conclusions and Future Work

This paper targets complex human activity recognition. This remains a challenging research area [39], especially to build effective recognition systems. We propose a formal approach based on discrete-time

Markov chains to model human activities. Important properties of such models can be automatically verified thanks to model checking. The proposed approach complements the main existing ones in the field of activity recognition, which seldom address formal verification issues.

The application presented all along this paper is in the medical field. The Alzheimer disease (and other associated diseases) is difficult to detect early. However, researches showed [40] that the analysis of patients performing daily life activities makes it possible to differentiate their stage of Alzheimer severity. In the same spirit we decided to develop a tool to analyze activities and behaviors of patients performing specific tasks to differentiate the level of damage among different cognitive functions.

Thanks to our formal probabilistic modeling approach we can expect three medically interesting outcomes. First, to evaluate a new patient before the first diagnosis by doctors, we can compare her game performance to a reference model representing a "healthy" behavior. Second, to monitor known patients, a customized model can be created according to their first results, and, over time, their health improvement or deterioration can be monitored. Finally, to pre-select a cohort of patients, we can use a reference model to determine, in a fast way, whether a new group of patients belongs to a specific category.

According to the methodology described in Figure 1, the models need to be updated owing to real experiment results (Step 5 and 6). When creating a reference model of a certain degree of Alzheimer disease, as for instance the "mild cognitive impairment", practitioners initially configure it with probabilities deduced from their experience. This model will be verified and compared to the average results of several experiments done with a known population of "moderate cognitive deficits" patients. We will then use the results to adjust the model probabilities to obtain a more realistic model, providing a more accurate prediction.

As a first step, we selected three serious games for Alzheimer patients with the help of clinicians, we encoded them as DTMCs in PRISM, and we tested meaningful PCTL properties thanks to the PRISM model checker. These properties include the use of rewards to quantify the performances of patients. Modeling these three games allowed us to validate our approach and to test its scalability for rather different applications.

The Match Items game tests visual attention. Its model was our first attempt to create a probabilistic human activity model. After implementing a first version with fixed probabilities associated with patient actions along a whole game session, we implemented another version in which time dependent probabilities are pre-computed to take into account the patient's fatigue over time. This version is more realistic to represent the behavior of a patient. To explore the capacity of PRISM and obtain more lifelike models, we used more complex functions to describe the probabilities in the two other models. The Recognition game targets episodic memory. Its model was directly implemented with pre-computed time dependent probabilities and we also divided it into several modules to facilitate the modeling task. This model requires the use of the explicit model-checking engine of PRISM, the only one able to cope with its big state space (because not all states are attainable). Regardless of this constraint, we succeeded in obtaining meaningful properties for medical monitoring. The third game, which addresses the inhibitory control function, was the most difficult to model. This difficulty also pushed PRISM to its limits as the time for model-checking is rather longer than for the two first games. However, we obtained readable and meaningful results depicting the variants that can be found in a patient behavior.

These results encourage us to pursue our research on behavior modeling for patient analyses. This paper demonstrates that the complexity of a game is not directly impacting the difficulty to model the player's associated behavior. Indeed, in some games, variants of valid or invalid actions from the player can have different relevant meanings. Such variants greatly increase the number of states and transitions in a model.

To understand if this difficulty is due to PRISM limitations and constraints, we intend to implement these models using the Storm model checker [30]. We chose Storm because, as described in [41], its model checker can take several modeling languages as input (including the PRISM language) and it is more suitable for various kinds of properties.

A major difficulty is the conceptual gap between the descriptions of the rules of a game and of the corresponding player behavior on the one hand, and the models and input languages of model-checkers, on the other hand. It would be useful and less error prone to translate automatically a game description to the model-checker syntax. Such translation demands formal descriptions on both sides. Thus plain text is not an option on the game side. As mentioned earlier, we are working on a formal activity description language (see listing 1) that could be used to model games and players. However, the paradigm difference prevents a

direct translation from game descriptions to model-checker inputs. Human intervention is required to bridge the gap, taking the form of code annotations, associations between concepts on both sides, directives on the structure of the state model, etc. Moreover, we may need to restrain the target language to a subset of the model-checker one.

Finally it is essential to test our models in real clinical experimentation. We set up different reference profiles (such as mild, moderate or severe Alzheimer) with the help of clinicians. To test the models, we proposed a medical protocol that started recently. Two groups of patients will play these games over a period of nine months: a witness group will include people with no known cognition deficit, and a patient group will gather people with an identified medium cognition deficit. Two game sessions are planned, repeated after one month; the second session is similar to the first one. Its objective is to minimize the impact of external events in the patient's life that might modify the measured performance. All game data (including scores, responses, and response times) as well as video recordings (focused only on the hands of the participants above the screen) will be recorded and anonymized. These results will then be used to adjust the models presented in this paper (model calibration, Step 6 of Figure 1). The calibration of the model probabilities has not been tackled yet, since it depends on both the quality and the quantity of the clinical data that will be gathered. Once these data acquired, we will consider the possibility to use parameter synthesis (e.g., PROPhESY [42] accepts PRISM format as input) or other classical optimization methods [43]. We hope that these adjustments will lead to a better representation of the behavior variants and make our models effective. In fact, our ultimate goal is to integrate the model-checking approach proposed in this paper into a medical monitoring system designed with the help of clinicians.

Acknowledgements. This work is part of the PhD. of Thibaud L'Yvonnet. We thank the French Provence-Alpes-Côte d'Azur region for the financial support. We also thank our medical partners from the CoBTeK (Cognition Behavior Technology) laboratory of Université Côte d'Azur for our fruitful interactions all along the different stages of this work.

References

[1] M. Vrigkas, C. Nikou, I. A. Kakadiaris, A Review of Human Activity Recognition Methods, Frontiers in Robotics and AI 2 (2015) 28, Frontiers.

[2] C. Piciarelli, S. Canazza, C. Micheloni, G. L. Foresti, A Network of Audio and Video Sensors for Monitoring Large Environments, in: Handbook on Soft Computing for Video Surveillance., Chapman & Hall/CRC, 2012, pp. 287–315.

[3] F. F. Chamasemani, L. S. Affendey, Systematic Review and Classification on Video Surveillance Systems, Int. Journal of Information Technology and Computer Science(IJITCS) 5 (7) (2013) 87–102, MECS Press.

[4] S. Weerachai, M. Mizukawa, Human behavior recognition via top-view vision for intelligent space, in: Int. Conf. on Control, Automation and Systems (ICCAS), IEEE, 2010, pp. 1687–1690.

[5] U. Ujjwal, A. Dziri, B. Leroy, F. Bremond, Late Fusion of Multiple Convolutional Layers for Pedestrian Detection, in: 15th IEEE Int. Conf. on Advanced Video and Signal-based Surveillance (AVSS), IEEE, 2018, pp. 1–6.

[6] X. Du, M. El-Khamy, J. Lee, L. Davis, Fused DNN: A Deep Neural Network Fusion Approach to Fast and Robust Pedestrian Detection, in: 2017 Winter Conf. on Applications of Computer Vision (WACV), IEEE, 2017, pp. 953–961.

[7] M. K. P. Tran, F. Brémond, P. Robert, Assistance for Older Adults in Serious Game Using an Interactive System, in: 4th Int. Conf. on Games and Learning Alliance (GALA), Springer, 2015, pp. 286–291.

[8] H.-B. Zhang, Y.-X. Zhang, B. Zhong, Q. Lei, L. Yang, J.-X. Du, D.-S. Chen, A comprehensive survey of vision-based human action recognition methods, Sensors 19 (5) (2019) 1005, Multidisciplinary Digital Publishing Institute.

[9] E. De Maria, T. L'Yvonnet, S. Moisan, J.-P. Rigault, Probabilistic Activity Recognition for Serious Games with Applications in Medicine, in: Formal Techniques for Safety-Critical Systems, Springer, 2020, pp. 106–124.

[10] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of Probabilistic Real-time Systems, in: Proc. 23rd Int. Conf. on Computer Aided Verification (CAV'11), Vol. 6806, Springer, 2011, pp. 585–591.

[11] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, MIT Press, 1999.

[12] S. Bonfanti, A. Gargantini, A. Mashkoor, A systematic literature review of the use of formal methods in medical software systems, Journal of Software: Evolution and Process 30 (5) (2018) e1943, Wiley Online Library.

[13] A. Mashkoor, A. Egyed, Analysis of Experiences with the Engineering of a Medical Device Using State-Based Formal Methods, in: 2018 International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2018, pp. 75–82.

[14] M. S. Ayub, O. Hasan, Formal Probabilistic Analysis of a Virtual Fixture Control Algorithm for a Surgical Robot, in: Verification and Evaluation of Computer and Communication Systems, Springer, 2017, pp. 1–16.

[15] M. L. Bolton, E. J. Bass, R. I. Siminiceanu, Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking, International Journal of Human-Computer Studies 70 (11) (2012) 888 – 906, Science Direct.

[16] S.-T. Antoni, J. Rinast, X. Ma, S. Schupp, A. Schlaefer, Online model checking for monitoring surrogate-based respiratory motion tracking in radiation therapy, International journal of computer assisted radiology and surgery 11 (11) (2016) 2085–2096, Springer.
[17] P. Groot, A. Hommersom, P. Lucas, R.-J. Merk, A. ten Teije, F. van Harmelen, R. Serban, Using model checking for critiquing based on clinical guidelines, AI in Medicine 46 (1) (2008) 19–36, Elsevier.
[18] T. Magherini, A. Fantechi, C. D. Nugent, E. Vicario, Using Temporal Logic and Model Checking in Automated Recognition of Human Activities for Ambient-Assisted Living, Transactions on Human-Machine Systems 43 (6) (2013) 509–521, IEEE.
[19] M. Nyolt, K. Yordanova, T. Kirste, Checking Models for Activity Recognition, in: Proceedings of the International Conference on Agents and Artificial Intelligence, Vol. 2, SciTePress, 2015, pp. 497–502, SciTePress.
[20] L. Chittaro, R. Sioni, Turning the Classic Snake Mobile Game into a Location–Based Exergame that Encourages Walking, in: Persuasive Technology. Design for Health and Safety, Springer, 2012, pp. 43–54.
[21] F. Buttussi, T. Pellis, A. Cabas-Vidani, D. Pausler, E. Carchietti, L. Chittaro, Evaluation of a 3D serious game for advanced life support retraining, Int. Journal of Medical Informatics 82 (9) (2013) 798 – 809, Elsevier.
[22] S. D. Atkinson, V. L. Narasimhan, Design of an introductory medical gaming environment for diagnosis and management of Parkinson's disease, in: Trends in Information Sciences Computing (TISC), IEEE, 2010, pp. 94–102.
[23] T. M. Fleming, L. Bavin, K. Stasiak, E. Hermansson-Webb, S. N. Merry, C. Cheek, M. Lucassen, H. M. Lau, B. Pollmuller, S. Hetrick, Serious Games and Gamification for Mental Health: Current Status and Promising Directions, Frontiers in Psychiatry 7 (2017) 215, Frontiers.
[24] P. Robert, V. Manera, A. Derreumaux, M. F. Y. Montesino, E. Leone, R. Fabre, J. Bourgeois, Efficacy of a Web App for Cognitive Training (MeMo) Regarding Cognitive and Behavioral Performance in People With Neurocognitive Disorders: Randomized Controlled Trial, Journal of medical Internet research 22 (3), JMIR Publications Inc. (2020).
[25] Tâches Attentionnelles Exécutives, `https://cmrr-nice.fr/lab/tae/`, CoBTeK lab, Université Côte d'Azur (2018).
[26] M. Hassan, A performance model of pedestrian dead reckoning with activity-based location updates, in: 18th Int. Conf. on Networks (ICON), IEEE, 2012, pp. 64–69.
[27] A. S. Ahouandjinou, C. Motamed, E. C. Ezin, A temporal belief-based hidden Markov model for human action recognition in medical videos, Pattern Recognition and Image Analysis 25 (3) (2015) 389–401, Springer.
[28] A. Jalal, S. Kamal, D. Kim, A Depth Video-based Human Detection and Activity Recognition using Multi-features and Embedded Hidden Markov Models for Health Care Monitoring Systems., Int. Journal of Interactive Multimedia & Artificial Intelligence 4 (4) (2017) 54–62, UNIR-Universidad Internacional de La Rioja.
[29] G. Behrmann, A. David, K. G. Larsen, A Tutorial on Uppaal, in: Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, no. 3185 in LNCS, Springer, 2004, pp. 200–236.
[30] C. Dehnert, S. Junges, J.-P. Katoen, M. Volk, A storm is coming: A modern probabilistic model checker, in: Computer Aided Verification, Springer, 2017, pp. 592–600.
[31] J. Sun, Y. Liu, J. S. Dong, J. Pang, PAT: Towards Flexible Verification under Fairness, in: 21th International Conference on Computer Aided Verification, Vol. 5643, Springer, 2009, pp. 709–714.
[32] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. L. Sangiovanni-Vincentelli, S. S. Sastry, S. A. Seshia, Data-Driven Probabilistic Modeling and Verification of Human Driver Behavior, in: Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium (FVHMS), University of California Berkeley, 2014, pp. 1–6.
[33] R. Alur, T. Henzinger, Reactive Modules, Formal Methods in System Design 15 (1) (1999) 7–48, Springer.
[34] M. Kwiatkowska, G. Norman, D. Parker, Stochastic model checking, in: Int. School on Formal Methods for the Design of Computer, Communication and Software Systems, Springer, 2007, pp. 220–270.
[35] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal aspects of computing 6 (5) (1994) 512–535, Springer.
[36] K. L. Votruba, C. Persad, B. Giordani, Cognitive deficits in healthy elderly population with "normal" scores on the Mini-Mental State Examination, Journal of Geriatric Psychiatry and Neurology 29 (3) (2016) 126–132, Sage Publications Sage CA: Los Angeles, CA.
[37] J. D. Huntley, A. Hampshire, D. Bor, A. M. Owen, R. J. Howard, The importance of sustained attention in early Alzheimer's disease, International journal of geriatric psychiatry 32 (8) (2017) 860–867, Wiley Online Library.
[38] J. T. Eckner, J. K. Richardson, H. Kim, D. B. Lipps, J. A. Ashton-Miller, A novel clinical test of recognition reaction time in healthy adults., Psychological assessment 24 (1) (2012) 249, American Psychological Association.
[39] E. Kim, S. Helal, D. Cook, Human activity recognition and pattern discovery, Pervasive Computing 9 (1) (2009) 48–53, IEEE.
[40] A. König, C. F. Crispim Junior, A. Derreumaux, G. Bensadoun, P.-D. Petit, F. Bremond, R. David, F. Verhey, P. Aalten, P. Robert, Validation of an automatic video monitoring system for the detection of instrumental activities of daily living in dementia patients, Journal of Alzheimer's disease : JAD 44 (2) (2015) 675—685, IOS Press.
[41] E. M. Hahn, A. Hartmanns, C. Hensel, M. Klauck, J. Klein, J. Křetínskỳ, D. Parker, T. Quatmann, E. Ruijters, M. Steinmetz, The 2019 comparison of tools for the analysis of quantitative formal models, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2019, pp. 69–92.
[42] C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, E. Ábrahám, PROPhESY: A PRObabilistic ParamEter SYnthesis Tool, in: Computer Aided Verification, Springer, 2015, pp. 214–231.
[43] N. Hansen, The CMA Evolution Strategy: A Comparing Review, in: Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, Springer, 2006, pp. 75–102.