Probabilistic Activity Recognition For Serious Games With Applications In Medicine

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

Probabilistic Activity Recognition For Serious Games With Applications In Medicine

Elisabetta De Maria[1], Thibaud L'Yvonnet[2], Sabine Moisan[2], and Jean-Paul Rigault[2]

[1] Université Côte d'Azur, CNRS, I3S, France edemaria@i3s.unice.fr
[2] Université Côte d'Azur, INRIA Sophia Antipolis, France {thibaud.lyvonnet, sabine.moisan, jean-paul.rigault}@inria.fr

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 (PCTL) concerning the dynamic evolution of activities. We illustrate our approach on the model of a *serious game* 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 only the model definition and its suitability to check behavioral properties of interest. Indeed, this is mandatory before envisioning any clinical study.

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 [23] and is employed in many contexts, such as visual surveillance in public places [19,5], smart homes [24], or pedestrian detection for smart cars [22,8]. A recent application in the health domain are "serious games", used to evaluate the performances of patients affected by neuro-degenerative pathologies such as the Alzheimer disease [21]. Behavior, emotions, and performance displayed by patients during these games can give indications on their disease.

A lot has been done, especially in computer vision, on simple *action* recognition [25], whereas we target complex *activities*, including several actions. In our view, an activity consists in a set of scenarios that describe possible behavioral variants. Therefore, recognition means to identify which scenario is running from 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 in monitoring patients with cognitive deficiencies.

All the scenarios of an activity are not equivalent: some are typical (thus 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. 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. Markov chains are deterministic and do not impose to associate a real duration with each action, contrary 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 states in the model and model checking would not be feasible. Our choice for using formal modeling and model checking is mainly motivated by their ability 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 discrete-time Markov chains using the PRISM language [14]. We used temporal logic to encode some relevant properties on their dynamical evolution, and we applied model checking techniques [7] to automatically validate the models with respect to these properties and 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.

We are developing a language for hospital practitioners to describe activities they expect from their patients as programs representing all the envisioned paths (possible combinations of actions from the patient or the environment), both typical behaviors and marginal ones. Some actions will be performed for sure by the patient (or the environment) and need no probabilities. Other ones depend on the stage of Alzheimer of the patient. With these latter actions, practitioners can associate a discrete probability level (e.g., low, medium, high...) or directly a real number or weight. Hence, we can deduce how relevant the scenario played by a patient is. For example, if a patient known to be healthy plays a "medium cognition deficit" scenario, our system is able to spot this information. The same goes if a "severe cognition deficit" patient plays a "healthy" scenario.

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 achieve, which is the focus of this paper.

The paper is organized as follows. Section 2 formally details discrete-time Markov chains and their support in the PRISM model checker. Section 3 presents a serious game case study used as a running example. Section 4 introduces the

PRISM encoding of this game as a discrete-time Markov chain and section 5 applies model checking to it. Finally, section 6 concludes and opens future research directions.

2 The PRISM Model Checker

The probabilistic model checker PRISM [14] is a tool for formal modeling and analysis of systems with random or probabilistic behavior. It has already been used to describe human activity [20]. 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 1. *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 1. In this game, the player has to press a button as many times as she wishes.

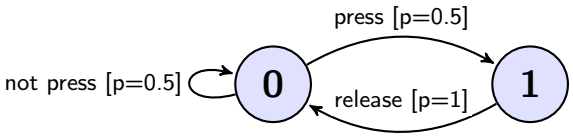


Fig. 1: DTMC representing a simple press button game. Each edge is labelled with both an action and the corresponding probability.

2.1 PRISM Modeling Language

PRISM provides a state-based modeling language inspired from the reactive modules formalism of [2]. A model is composed of a set of *modules* which can 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 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 all the 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 1 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* [15], 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.

Algorithm 1 PRISM code for Figure 1 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
```

2.2 Probabilistic Temporal Logic

The dynamics of DTMCs can be specified in PRISM thanks to the PCTL (Probabilistic Computation Tree Logic) temporal logic [10]. PCTL extends the CTL logic (Computation Tree Logic) [7] with probabilities. The following state quantifiers are available in PCTL: **X** (next time), **F** (sometimes in the future), **G** (always in the future), and **U** (until). Note that the classical path quantifiers **A** (forall) and **E** (exist) of CTL are replaced by probabilities. Thus, instead of 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 [10]. The most important operator in PCTL is **P**, which allows to reason about the probability of event occurrences. 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 properties on the expected values of rewards. 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 accumulated along a path until `t` time units have elapsed. PRISM provides model checking algorithms [7] 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.

3 Motivation and Case Study

For non experts in computer science, we propose a language to describe activities to recognize in real-time. It offers usual instructions such as parallel execution, conditional, or repetition. Most instructions may have associated weights in the form of real numbers between 0 and 1 or using a discrete scale. 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, which is still under development, but we simply illustrate its use with an example of a serious game (see listing 1.1).

Serious games 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 (biometric and external) while playing the game. In the health domain, they can be used to incite patients to practice physical exercises [6], to train medical staff with engaging activities [4], or to help diagnose and treat patients [3,9]. When formally modeling a diagnosis game, a user can associate probabilities with instructions 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.

After discussions with medical doctors, we identified three prospective uses for 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 gives us a fairly good idea of what such a healthy behavior is, as 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 could 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 will allow practitioners to shortlist patients to apply a specific protocol on this cohort.

3.1 Case Study

As a use case, we consider a serious game to analyze the behavior of Alzheimer patients: the *Match Items game* [21]. In this game, 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 (see Figure 2).

Fig. 2: Display of the Match Items game.

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. The game lasts at most five minutes. A simplified pseudo-code program describing this game is given in Listing 1.1.

```
Initial: patient inside game_zone and patient presses_start_button
 during 300s
   console displays_picture
   when [0.0005] patient exits game_zone
   preempt { emit no_player; 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_smileys
            case [0.33] (console displays_sad_smiley)
              // wrong answer: loop keeping current picture
              nothing !! count: sad_smileys
            end switch
          case [0.25] (console notifies_inactivity)
            // patient did not react, continue with same picture
            nothing !! count: non_interactions
          end switch
        end every
      end when
    end during
  emit game_over
```

Listing 1.1: Serious game pseudo code description.

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 awaited event occurs executes its statements. A probability of occurrence may be associated with a branch (indicated within square brackets in the pseudo-code).

Furthermore, the clinicians can indicate (through ***!!*** comments) 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. 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.

4 Serious Game Model

We model the behavior of a patient in this game using a discrete-time Markov chain (DTMC). To the best of our knowledge, DTMC models are barely used for the description of human behavior, although we can cite [11]. In computer vision, Hidden Markov Models (HMMs) are a popular approach for the description of activities [1,12]. However, PRISM and most of the other probabilistic model checkers do not allow to check temporal logic properties over HMMs.

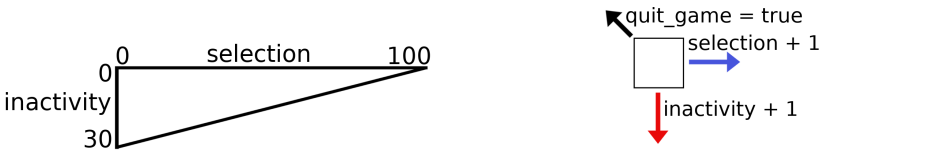
Due to a limitation in PRISM, we explicitly represent all the possible states in the model. This limitation concerns looping through a state: in PRISM Markov

chains, we cannot put a limit on the number of times we can loop through a state. This means that, even if we give a low probability to the loop transition, there will always be a risk for a simulation to never quit this loop (fairness is not automatically imposed). By 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 our 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, we can translate the time constraint of three-hundred seconds in 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 ($300/3 = 100$). 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 1.1). We call this event *inactivity* and it cannot happen more than thirty times in a row($300/10 = 30$).

To represent all combinations of these two events, we picture a right-angle triangle (Figure 3a). The edge of length one hundred (representing the scenario of a succession of *selection*) and the edge of length thirty (representing the scenario of a succession of *inactivity*) 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 3b.



(a) Combinations of events triangle.

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

Fig. 3: Concepts of the model of activity.

According to Figure 3b, 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 "Serious_game"[3]. 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.

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.

The 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):

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

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 in the game by the patient 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 "Serious_game" module:

[3] PRISM code at `https://gitlab.com/ThibLY/activity-recognition-modeling`

1. variable *location* is equal to 0, meaning the patient is in the room;
2. variable *location* is equal to 1, *time_Is_Not_Over* is *true* 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 leaved the game before the end of the maximum duration.

The PRISM code for the command associated with the second guard is given in Listing 1.2, 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 1.2: Excerpt from the Serious_Game module.

The global variable *time_Is_Over* is defined to ease the readability of the module. It contains a Boolean expression to determine if the maximum number of actions that a patient can perform is reached.

The state transitions performed in a simulation describe the patient's behavior in a 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.

In the following section, as a theoretical example, we will assume that these parameters represent a typical patient with mild cognitive impairment (MCI).

5 Temporal Logic Properties and Results

In the previous model, we encoded and tested several properties in PCTL. 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.

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 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 *Match Items game* model. In our case the result is 1.0; it is obtained in 0.002 seconds.

5.2 Medically Oriented Properties

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 last property checks the probability to leave the game before the end of the five minutes.

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 forty-three times (not necessarily consecutively) and not to interact with it eighteen times (not necessarily consecutive)?

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

Property 5. What is the probability for a patient to leave the game before the maximum game duration?

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

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

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	3.1364×10^{-2}	0.058

Table 1: Results from Property 3 to 5.

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. Moreover, this path 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 over 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 state 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.

The probability obtained for Property 5 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 5 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 and Property 5 should be observed quite rarely (respectively 2% and 3% of the cases). The behaviors described in Property 2 and Property 3 must not be observed. 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.

Properties about quality of actions. These properties are relative to the quality of the actions that can be performed. The first one provides an average "score" for the model. The second and third ones give probabilities to follow some specific paths in the model.

Property 6. What is the average amount of good responses given by patients during their game sessions?

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

Property 7. What is the probability for a patient to choose the correct picture exactly one time and to never choose a good one again until the end of the game?

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

Property 8. What is the probability for a patient to directly choose the right picture, without choosing a wrong picture before?

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

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

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 6.

Property	Result	Time(s)
7	3.3012×10^{-12}	2.046
8	6.6622×10^{-1}	0.007

(b) Results of Properties 7 and 8.

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

Property 6 can be written for *Happy_ smiley _ reward*, *Sad_ smiley_ reward* and for *Inactivity_ bool_ reward*. According to its results, the average "score" for a cohort of patients matching this model parameters should be 31 good answers for 15 bad answers and there should be 15 inactivity messages before the end of the session.

Property 7 was the longest one to compute. The complexity of this property comes from the nesting of G operators. Property 8 gives the biggest probability value compared to all others. Indeed, unlike Property 7, 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 6. If the score differs too much from this result, the cohort must be rejected. According to the result of Property 7, 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 8, 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 variable allows the use of the "run experiments" feature of PRISM and the acquisition of graphs of results.

Property 9. What is the amount of happy smileys accumulated within i steps?

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

where i is the number of steps to perform before checking the reward. This property is reused for *Sad_smiley_reward*, *Inactivity_bool_reward*, *Gaming_time* and *Leave_game_reward*.

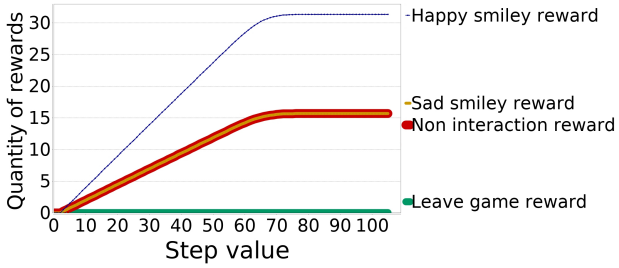


Fig. 4: Average model checking results for rewards related to good answers, bad answers, non-interaction, and game leaving behavior.

In Figure 4, 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 6. 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 5) where a patient may leave the game before its maximum duration.

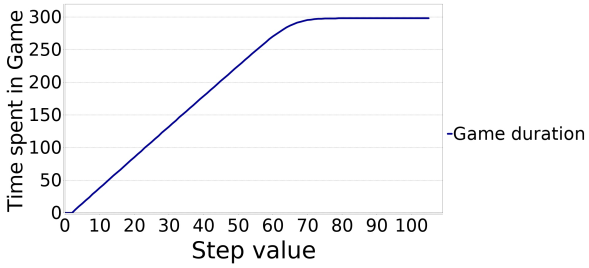
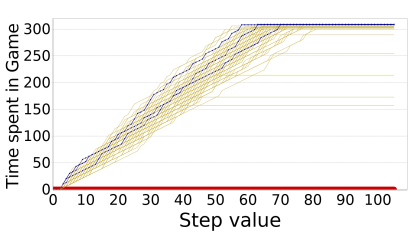


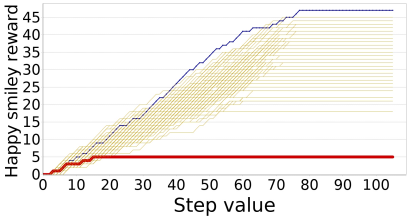
Fig. 5: Average duration of the game obtained with model checking.

In Figure 5, 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 6a, over 100 simulations, some of them (in blue/thin black in the figure) reach a maximum value which is above three-hundred 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 6a) 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 (Property 5 and 6).



(a) Duration of the game over 100 runs.



(b) Accumulation of good answers over 100 runs.

Fig. 6: Experiment results on the accumulation of rewards over 100 runs.

In Figure 6, 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 6 as well as in Figure 4, there is no "preferred" time to act during the game. This can be seen with the linear increase of each reward. 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 see the different runs in Figure 6, a shell and a Python scripts were written to retrieve raw data from simulations. These data are used in Figure 7 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

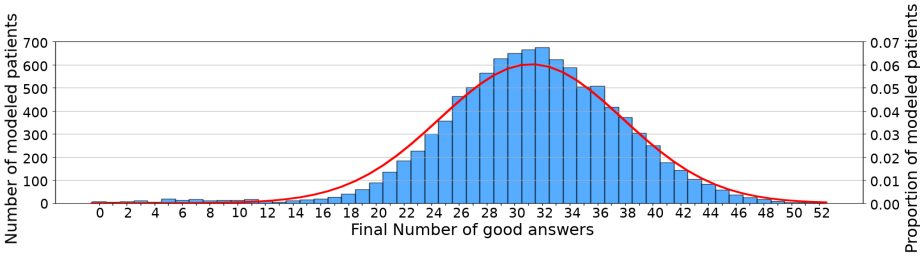


Fig. 7: Frequency of good answers over 10,000 runs (in blue/grey) and its fitting normal distribution with $\mu = 31.2131$ and $\sigma^2 = 43.9271$ (in red/black).

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.

6 Conclusions and Future Work

In this paper, we target complex activity recognition, which remains a challenging research area [13] to obtain viable 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 technique we propose complements the main existing approaches in the field of activity recognition. Indeed, these approaches seldom address formal verification issues. Some work on human activity recognition relies on online model checking [16,17]. Probabilistic model checking can be used to debug activity models [18]. In our case, we use probabilities to explore paths associated with different behaviors.

Thanks to our formal probabilistic modelling approach we can expect three medically interesting outcomes. First, to evaluate a new patient before the first diagnosis of 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 could be monitored. Finally, to preselect a cohort of patients, we can use a reference model to determine, in a fast way, whether a new group of patients belongs to this specific category.

Our models need to be updated according to real experiment results. When creating a reference model of a certain degree of Alzheimer disease, as for instance the "mild cognitive impairment", practitioners may initially configure it with probabilities deduced from their experience. This model will be verified and compared to the average results of several experiments done by 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 encoded a serious game for Alzheimer patients as a DTMC 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.

The next step is to validate our approach as well as to test its scalability on three other serious games selected with the help of clinicians. These games will be represented by PRISM models, similar to the one presented in this paper, and used in clinical experimentation. Once the models created, we will set up different reference profiles (such as mild, moderate or severe Alzheimer) with the participation of clinicians. Then, several groups of patients will play these games. Their results will be recorded and used to adjust our initial models.

The 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 Ph.D. thesis of Thibaud L'Yvonnet. We thank the French Provence-Alpes-Côte d'Azur region for the financial support.

References

1. Ahouandjinou, A.S., Motamed, C., Ezin, E.C.: A temporal belief-based hidden Markov model for human action recognition in medical videos. Pattern Recognition and Image Analysis (2015)
2. Alur, R., Henzinger, T.: Reactive modules. Formal Methods in System Design (1999)
3. Atkinson, S.D., Narasimhan, V.L.: Design of an introductory medical gaming environment for diagnosis and management of parkinson's disease. In: Trendz in Information Sciences Computing(TISC) (2010)
4. Buttussi, F., Pellis, T., Cabas-Vidani, A., Pausler, D., Carchietti, E., Chittaro, L.: Evaluation of a 3D serious game for advanced life support retraining. Int. Journal of Medical Informatics (2013)
5. Chamasemani, F.F., Affendey, L.S.: Systematic review and classification on video surveillance systems. Int. Journal of Information Technology and Computer Science(IJITCS) (2013)
6. Chittaro, L., Sioni, R.: Turning the classic snake mobile game into a location–based exergame that encourages walking. In: Persuasive Technology. Design for Health and Safety (2012)
7. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
8. Du, X., El-Khamy, M., Lee, J., Davis, L.: Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In: 2017 IEEE Winter Conf. on Applications of Computer Vision (WACV) (2017)
9. Fleming, T.M., Bavin, L., Stasiak, K., Hermansson-Webb, E., Merry, S.N., Cheek, C., Lucassen, M., Lau, H.M., Pollmuller, B., Hetrick, S.: Serious games and gamification for mental health: Current status and promising directions. Frontiers in Psychiatry (2017)
10. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal aspects of computing (1994)

11. Hassan, M.: A performance model of pedestrian dead reckoning with activity-based location updates. In: 2012 18th IEEE Int. Conf. on Networks (ICON) (2012)
12. Jalal, A., Kamal, S., Kim, D.: 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 (2017)
13. Kim, E., Helal, S., Cook, D.: Human activity recognition and pattern discovery. IEEE pervasive computing (2009)
14. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Proc. 23rd Int. Conf. on Computer Aided Verification (CAV'11) (2011)
15. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: Int. School on Formal Methods for the Design of Computer, Communication and Software Systems (2007)
16. Magherini, T., Fantechi, A., Nugent, C.D., Vicario, E.: Using temporal logic and model checking in automated recognition of human activities for ambient-assisted living. IEEE Transactions on Human-Machine Systems (2013)
17. Magherini, T., Parente, G., Nugent, C.D., Donnelly, M.P., Vicario, E., Cruciani, F., Paggetti, C.: Temporal logic bounded model-checking for recognition of activities of daily living. In: Proc. of the 10th IEEE Int. Conf. on Information Technology and Applications in Biomedicine (2010)
18. Nyolt, M., Yordanova, K., Kirste, T.: Checking models for activity recognition. In: ICAART (2015)
19. Piciarelli, C., Canazza, S., Micheloni, C., Foresti, G.L.: A network of audio and video sensors for monitoring large environments. In: Handbook on Soft Computing for Video Surveillance. Chapman & Hall/CRC (2012)
20. Sadigh, D., Driggs-Campbell, K., Puggelli, A., Li, W., Shia, V., Bajcsy, R., Sangiovanni-Vincentelli, A.L., Sastry, S.S., Seshia, S.A.: Data-Driven Probabilistic Modeling and Verification of Human Driver Behavior. In: Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium (FVHMS) (2014)
21. Tran, M.K.P., Brémond, F., Robert, P.: Assistance for older adults in serious game using an interactive system. In: 4th Int. Conf. on Games and Learning Alliance (GALA) (2015)
22. Ujjwal, U., Dziri, A., Leroy, B., Bremond, F.: Late Fusion of Multiple Convolutional Layers for Pedestrian Detection. In: 15th IEEE Int. Conf. on Advanced Video and Signal-based Surveillance (AVSS) (2018)
23. Vrigkas, M., Nikou, C., Kakadiaris, I.A.: A review of human activity recognition methods. Frontiers in Robotics and AI (2015)
24. Weerachai, S., Mizukawa, M.: Human behavior recognition via top-view vision for intelligent space. In: Int. Conf. on Control, Automation and Systems (ICCAS) (2010)
25. Zhang, H.B., Zhang, Y.X., Zhong, B., Lei, Q., Yang, L., Du, J.X., Chen, D.S.: A comprehensive survey of vision-based human action recognition methods. Sensors (5) (2019)