Relationships between human activity models and brain models : application to clinical serious games

Thibaud L'Yvonnet

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Relations entre modèles d'activités humaines et modèles du cerveau : application aux jeux sérieux en clinique

Thibaud L'YVONNET

INRIA - STARS
CoBTeK Lab

Présentée en vue de l'obtention du grade de docteur en Informatique
d'Université Côte d'Azur
Dirigée par : Sabine Moisan
Co-dirigée par : Elisabetta De Maria
Soutenue le : 28/03/2022

Devant le jury, composé de :
Fabio Buttussi, RTD-B, Università degli Studi di Udine
Julien Deantoni, MCF, Université Côte d'Azur, I3S
Elisabetta De Maria, MCF, Université Côte d'Azur, I3S
Ines Di Loreto, MCF, Université de technologie de Troyes
François Fages, DR1, INRIA Saclay
Stefan Klöppel, PU-PH, Universitäre Psychiatrische Dienste Bern
Morgan Magnin, PR, Laboratoire des sciences du numérique de Nantes
Frédéric Mallet, PR, Université Côte d'Azur, I3S

Inria

RÉGION SUD PROVENCE ALPES CÔTE D'AZUR

iAU

Relations entre modèles d'activités humaines et modèles du cerveau : application aux jeux sérieux en clinique

Relationships Between Human Activity Models and Brain Models: Application to Clinical Serious Games

Jury :

Rapporteurs

Ines Di Loreto, MCF, Université de technologie de Troyes
Morgan Magnin, PR, Laboratoire des sciences du numérique de Nantes

Examinateurs

Fabio Buttussi, RTD-B, Università degli Studi di Udine
Julien Deantoni, MCF, Université Côte d'Azur, I3S
Elisabetta De Maria, MCF, Université Côte d'Azur, I3S
François Fages, DR1, INRIA Saclay
Stefan Klöppel, PU-PH, Universitäre Psychiatrische Dienste Bern
Frédéric Mallet, PR, Université Côte d'Azur, I3S

Invitée

Sabine Moisan, CRHC, Université Côte d'Azur, INRIA

"The imposing edifice of science provides a challenging view of what can be achieved by the accumulation of many small efforts in a steady objective and dedicated search for truth."

Charles Hard Townes

Résumé

Cette thèse explore la capacité des méthodes formelles, et plus particulièrement du model-checking, à analyser le comportement de patients lors de tests neuro-cognitifs en clinique. À cette fin, elle étudie l'utilisation de « jeux sérieux » d'entraînement cognitif par des patients seniors atteints de troubles neuro-cognitifs mineurs. Quatre jeux sérieux ont été sélectionnés par les chercheurs cliniciens de l'équipe CoBTeK, chacun ciblant une fonction cognitive différente. Une première partie décrit la modélisation sous forme de chaines de Markov des comportements des joueurs pour les trois jeux qui étaient déjà déployés au début de la thèse à l'institut Claude Pompidou (ICP) de Nice. Les modèles développés sont probabilistes afin de représenter au mieux l'ensemble des comportements que peuvent exprimer les patients. Les probabilités initiales sont issues des estimations des praticiens de l'ICP. Les résultats obtenus lors des vérifications de ces premiers modèles avec le model-checker Prism sont encourageants pour la poursuite de cette démarche de modélisation. La comparaison des performances de Prism et d'un autre model-checker, Storm, sur la vérification de ces modèles a montré que de manière générale, Storm est plus performant que Prism, cependant il est plus contraignant au niveau de l'expression des propriétés logiques. La deuxième partie concerne un protocole clinique mis en place afin d'obtenir des données d'utilisation réelles de patients seniors. Pour cela, un total de 50 volontaires ont été inclus sur une durée de 10 mois : 30 présentaient des troubles neuro-cognitifs mineurs et 20 avaient des plaintes cognitives isolées. Au cours de ce protocole, chaque volontaire participait à deux séances individuelles de 30 à 90 minutes pendant lesquelles il leur était demandé de compléter les tests cliniques manquant à leur dossier et de jouer aux quatre jeux sérieux précédemment mentionnés. Ces résultats ont été analysés avec des méthodes classiques d'analyse statistique pour évaluer leur intérêt dans l'identification de troubles cognitifs. Ils ont permis à la fois de trouver des différences de performances significatives entre les deux groupes ainsi qu'une corrélation entre ces performances et les résultats des tests cliniques. Par la suite, une analyse plus en profondeur des relevés d'utilisation a permis de comprendre quelles étaient les séries d'actions qui permettaient au mieux de catégoriser chaque type de patient. Ces dernières informations ont été utilisées pour calibrer les modèles afin d'obtenir des modèles plus réalistes et fiables. La troisième et dernière partie détaille l'implémentation d'un modèle de neurones artificiels décrivant l'activité des ganglions de la base, région cérébrale impliquée dans le contrôle inhibiteur. Cette fonction cognitive, ciblée par l'un des jeux sérieux étudié dans la thèse, se retrouve dégradée chez des patients atteint de maladies neuro-dégénératives comme la maladie d'Alzheimer ou le syndrome de Parkinson. La vérification de ce modèle a montré une activité cohérente avec celle décrite dans la littérature sur cette fonction cognitive. La finalité de ce modèle est d'être couplé au modèle d'activité associé au jeu évaluant cette fonction afin d'explorer des modifications sur le réseau de neurones pouvant engendrer un comportement caractéristique des patients atteints de troubles neuro-cognitifs.

Mots clés : Modèles de comportements humains, Modèles probabilistes, Méthodes formelles, Jeux sérieux, Réseaux de neurones biologiques, Maladies neuro-dégénératives.

Abstract

This thesis explores the ability of formal methods, and more particularly model-checking, to analyze the behavior of patients during neuro-cognitive tests in clinical settings. To this end, it studies the use of "serious games" for cognitive training of senior patients with minor neurocognitive disorders. Four serious games were selected by clinical researchers from the CoBTeK team, each targeting a different cognitive function. A first part describes the modeling, in the form of Markov chains, of player's behaviors associated with the three games that were already deployed at the beginning of the thesis at the Claude Pompidou Institute (ICP) in Nice. The models developed are probabilistic in order to best represent all the behaviors that patients can express. Initial probabilities are derived from ICP practitioner estimates. The results obtained during the verifications of these first models with the Prism model-checker are encouraging to continue this modeling approach. Comparing the performances of Prism and of another model-checker, Storm, on the verification of these models showed that in general, Storm is more efficient than Prism, however it is more restrictive concerning the expression of logical properties. The second part concerns a clinical protocol set up to obtain real data from senior patients. For that, a total of 50 volunteers were included over a 10-month period: 30 had minor neurocognitive disorders and 20 had isolated cognitive complaints. During this protocol, each volunteer participated in two individual sessions of 30 to 90 minutes during which they were asked to complete the clinical tests missing from their record and to play the four serious games previously mentioned. These results were analyzed using classical statistical analysis methods to assess their interest in identifying cognitive disorders. They made it possible to find significant differences in performance between the two groups as well as a correlation between these performances and the clinical tests results. Subsequently, a more in-depth analysis of the usage records led to understand the series of actions that best allow to categorize each type of patient. This latter information was used to calibrate the models in order to obtain more realistic and reliable models. The third and last part details the implementation of an artificial neuron model describing the activity of the basal ganglia, a brain region involved in inhibitory control. This cognitive function, targeted by one of the serious games studied in the thesis, is impaired in patients suffering from neurodegenerative diseases such as Alzheimer's disease or Parkinson's syndrome. Verification of this model showed an activity consistent with the one described in the literature on this cognitive function. The purpose of this model is to be coupled to the activity model associated with the game evaluating this function in order to explore modifications in the neural network that can generate a behavior characteristic of patients with neurocognitive disorders.

Keywords: Human behavior models, Probabilistic models, Formal methods, Serious games, Biological neural networks, Neurodegenerative diseases

Acknowledgements

This work would have never been made possible without my PhD directors, Elisabetta De Maria and Sabine Moisan. I would like to express my deepest appreciation to Elisabetta De Maria for all the the work and troubles she got through to get me into a PhD and for believing in me. I would also like to extend my deepest gratitude to Sabine Moisan for accepting me as her student, solving lots of administrative issues we have been going through, and trusting me all along this project. I thank them both for the amount of time they dedicated to my training and my manuscripts corrections. I would also like to express my profound gratitude to Jean-Paul Rigault who participated as much as my PhD directors in my work and my writings in the past three years.

I thank all the members of my jury, including my reviewers, for accepting my invitation and for all their valuable commentaries.

I wish to deeply thank François Brémond for accepting me within the team he supervises. I wish to express my most sincere gratitude to Philippe Robert for supporting this PhD project as Head of the Association Innovation Alzheimer and as Head of the CoBTeK lab team. The valuable time he spent allowed me to get helpful hindsight on patients profiles understanding. I'm deeply indebted to Valeria Manera for helping me getting through two protocol review committees and analyzing the data obtained in our experimental protocol. The PhD project could never had reach its end in due time without her generous helping hand. I thank Radia Zeghari for kindly helping me during her manuscript writing and for all the hindsight she gave me on neuropsychology. I gratefully acknowledge the work and adjustments on the serious games of Alexandre Derreumaux. I also wish to thank him for sharing his office with me. I would like to extend my gratitude to Justine Lemaire for her patience and her help in the protocol administrative tasks. Many deep thanks to all the staff I met and work with at the Institut Claude Pompidou of Nice. Thank you for helping me so kindly to go through my clinical protocol whether it was to include new volunteers, administer a test or to set appointments.

I thank my family who coped with me for the past ten years through my high studies. Thank you for encouraging me in every projects whether they led to failure or success. Thank you for helping me getting back on my feet after each fall and celebrating each victory. Thanks for never letting me down and offering so much through childhood and adulthood. I know that wherever I am and whatever challenges I face, I may always come back to you to find unconditional rampart and a safe haven to rest.

Finally, I want to thank all my closest friends that encouraged me and supported me in this project. Thank you to all fellow students of the STARS team and thank you to my music band for letting me have some fun despite the work and the covid-19 crisis.

Contents

List of Figures

List of Tables

Dedicated to my family and my beloved cat Lamia for their constant support and unconditional love.
I love you all dearly.

Chapter 1

Cognitive Impairment, Serious Games, and Activity Recognition

This PhD thesis was partially funded by Région Sud (région Provence Alpes Côte d'Azur, France) in partnership with the Innovation Alzeihmer association, located at ICP (Institut Claude Pompidou) in Nice. The PhD work was done in the Stars team of INRIA (focusing on human activity recognition systems) and, for the clinical protocol part, at ICP (part of the Centre Hospitalier Universitaire of Nice). Both ICP and INRIA are members of the Cognition Behaviour Technology lab (CoBTeK lab), a pluridisciplinary team of the French Université Côte d'Azur directed by Pr. Philippe Robert and Dr. François Brémond. This team groups together specialists in childhood cognition disorders and elder cognition disorders as well as specialists in computer science. As this manuscript focuses on elder cognitive disorders, the main interactions were with cognition experts of CoBTeK at ICP. This institute is specialized in the diagnosis and care of elder patients with memory losses induced by diseases such as the Alzheimer's one.

This thesis explores the ability of formal methods, and more particularly model-checking, to analyze the behavior of patients during neurocognitive tests in clinical settings. The first goal is to specify a new tool that would add to practitioners' toolkit for early diagnosis of neurodegenerative diseases. To this end, the thesis studies the use of "serious games" for cognitive training of senior patients with minor neurocognitive disorders. Formal methods to describe serious game activities should allow both a better understanding of the variety of behaviors and a better prediction of patient diagnosis based on their behavior. They provide a framework for model verification and analysis. This formal approach may predict combination of behaviors if they are not observed in experiments. The models coupled with a recognition system would lead to a reliable detection of risks of underlying neurocognitive disorders in the patients. The second goal is to study models of biological neuron networks involved in patient behaviors. We selected the inhibitory control cognitive function that is targeted by one of the modeled serious games. The study of this neural model coupled with the serious game activity model should lead to a better understanding of patient behaviors and to new explorations on disorder origins.

1.1 Cognitive Impairments and Dementia

To perform daily living activities, human brain displays a set of mental abilities including learning, thinking, reasoning, remembering, problem solving, decision making, and attention. These abilities are called cognitive functions [49]. In one's life, these functions may decline due to normal ageing or to neuro-degenerative diseases. Such declines can lead to cognitive impairments. According to the latest DSM-5 [47]

classification, cognitive impairments are characterized by a cognitive decline as well as behavioral disorders that can interfere in one's daily life. Depending on the severity of these deficits and on their impact, this classification discriminates mild Neuro-Cognitive Disorder (mild NCD) and Major Neuro-Cognitive Disorder (major NCD). Patients suffering from mild NCD need to be supervised by medical practitioners.

The World Health Organization (WHO) defines dementia as a syndrome in which cognitive functions decline beyond what is expected as a consequence of normal ageing. According to the WHO, around 50 million people have dementia worldwide with nearly 10 million new cases every year. Alzheimer may be the cause to $60-70\%$ of these cases. Alzheimer's prevalence varies with the age of the population. A prevalence under 2% for people under 60 years old was observed while it reaches beyond 20% for elders over the age of 85 years old [103]. According to the WHO, in 2019, the estimated global societal cost of dementia was US$ 1.3 trillion, a motivation to work on this peculiar topic.

Present Solutions for Diagnosis and Treatment for Alzheimer's Disease

There are no treatment yet to restore the lost capacity of a patient. Moreover, drugs reducing cognitive decline have yet to prove their efficiency. Indeed, even though a new drug, named Aducanumab, received the agreement of the Food and Drug Administration on June 7^{th} 2021 (first Alzheimer Disease drug approval since 2003 [10]) as a drug reducing cognitive decline, its results are contradictory from a clinical study to another [77]. Nowadays the main therapy consists in cognitive training to maintain the patient capacities. Cognitive training is defined in a Cochrane review [13] as a set of standardized tasks targeting specific cognitive functions (memory, focus, reasoning ...). This training would improve, or at least maintain, the efficiency of a cognitive function and would even apply to other activities. According to the definition cited above, cognitive training includes a large number of activities. Thus, when confronted to cognitive impairments, a practitioner goal is to detect the signs of these impairments as soon as possible. To do so, practitioners have access to a large battery of neuropsychological tests as well as bio-marker tests to establish a highly reliable diagnosis. These tests can be laborious to perform for the practitioner as well as for the patient. To strengthen the diagnosis, complementary data can be acquired with the analysis of bio-markers in the cerebrospinal fluid [45]. The CoBTeK team has a long experience in the development of new diagnosis techniques using numeric technologies.

1.2 From Games to Serious Games

Serious games represent a way to deal with cognitive decline associated with aging. A serious game can be defined as "a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objective" [130]. Thus one can consider serious games as applications that give "attractive shapes or plots (game) to didactic contents (serious)" [4].

When games became serious

This definition of serious games is quite recent in its form but the oxymoron "serious game" is far older [42]. Indeed, one can track it back to the European Renaissance [91] and far earlier [19]. Since that time, the oxymoron has been used

for different purposes (e.g., the professional practice of sport [42]). In their article, Djaouti et al. trace back the first use of the "modern" definition of serious games in the book "Serious Games" by Clark Abt first published in 1970 [1]. In his work, the author attempted to use games for training and education in the context of the Cold War. Nowadays, with the development of technologies in the past decades, digital serious games are more and more affordable and usable for the general public.

Serious Games and Cognitive Impairment

The work presented in this manuscript focuses on the use of serious games in medical applications. Serious games have already proved their interest to evaluate cognitive impairment throughout several feasibility studies [116, 120, 73]. Other researches have shown their potential in Alzheimer's disease therapy as cognitive training activities [5, 70, 98] or for neurocognitive disorder assessment [16, 90]. Moreover, studies have shown that elders prefer games over classical cognitive exercises [94]. All these facts make serious games a tool of choice in patient diagnosis and care.

The CoBTeK team has already developed several serious games [118, 107]. In this manuscript, a selection of these games are used for the validation of a new assessment method thanks to formal activity modeling.

1.3 Activity and Biological Formal Modeling

Modeling, and more particularly conceptual modeling, is the process of making an abstract representation of a real or proposed system [108]. Models are used in various domains (e.g., meteorology, biology, industry) for a variety of purposes (e.g.,to predict, to understand, to design, to verify, to validate ...). In this manuscript, we aim to both predict and understand patient behavior through formal model verification.

Formal Models

Formal methods refer to "mathematically based techniques for describing system properties" [127]. Formal models are conceived and implemented with such formal methods. They are also used in domains other than industry, notably in sciences, to study the properties of a given phenomenon (e.g., verification of the behavior of genetic logic gates [52]). These methods allow a mathematical verification of a system. This verification can either be automated with *model checking* frameworks or partially automated with theorem provers.

In this PhD thesis, we rely on model checking frameworks, that take as input a modeled system and a property written in temporal logic. The framework performs automatically the verification of the property and provides as output both the results (property verified or not, and also, possibly, a counter example if the property does not hold) and the computation times.

Human Activity Formal Modeling in Clinic

Models of human activities are used in recognition systems going from pedestrian location [60] to daily living activity recognition [87]. Human activity formal models for the verification, analysis, and classification of patient behaviors have been less documented. This manuscript focuses on a medical application and more precisely

on patient activities playing serious games. To model such activities, we chose a probabilistic formalism that allows to represent a wide range of behaviors.

Biological Formal Modeling

Formal models are also used to study biological systems. Indeed, biology offers many complex systems to analyze. Modeling and verifying these systems is well suited to validate theories or to find errors in theoretical pathways [32, 28]. In our work, we focus on the inhibitory control loop in the brain. Our goal is to study the relations between its biological neural model and the human activity it monitors.

1.4 Road-map

The PhD thesis manuscript is organized as follows.

Chapter 2: Formal Methods for Modeling Serious Game Activity

This chapter mainly tackles two research questions:

- Are formal approaches suited and effective to model human activities?
- Are clinical serious games suited for such models?

To answer these questions, this chapter focuses on our first attempts to formally model patient activity during serious games. It proposes a feasibility study of activity modeling with Markov chains for three serious games targeting different cognitive functions. The corresponding models were verified with probabilistic model checkers. The games were chosen with practitioners based on their potential clinical relevance. This study showed that probabilistic formal models provide enough flexibility to model serious games activity.

Chapter 3: Clinical Protocol and Experimental Results

This chapter mainly tackles two other research questions:

- Can formal modeling of these activities be of interest for diagnosis?
- Can experimental data validate these models as well as their discriminant capabilities?

To answer these questions, this chapter describes a clinical experiment to collect data on the use of the selected games by patients with mild Neuro-Cognitive Disorders (mild NCD) and patients with Subjective Cognitive Disorders (SCD). The patients were recruited in the Institut Claude Pompidou of Nice. The collected data were analyzed with classical statistical tools. The results are to be used for the tuning of the models developed in the previous chapter. The serious games used in this work were good candidates to differentiate mild NCD and SCD patients. After modification and calibration, the probabilistic models can mimic the observed behaviors of mild NCD patients.

Chapter 4: Model of the Inhibitory Control Circuit in the Brain

This chapter mainly tackles a final question:

- Is it possible to establish a relation between a human behavior model and a neuronal brain model?

To answer this question, this chapter proposes a formal modeling of the inhibitory control loop. This cognitive function was the target of one of the modeled serious games of chapter 2. In the brain, it involves mainly basal ganglia. As basal ganglia are a limited group of anatomical structures, the inhibitory control function was a good candidate to investigate the relations between brain and activity models. We propose the notion of Leaky Integrate & Fire (LI&F) neuron box that is is a generalization of the LI&F neuron and it appears as a good method to model complex neural network behavior while limiting the complexity of the model.

Chapter 2

Formal Methods for Modeling Serious Game Activity

To model the activity of patients playing clinical serious games, we choose a formal approach to set up behavioral models the properties of which can be automatically verified. Model checking appears as a suitable and promising method. Moreover, to represent the variability of human behavior, probabilistic modeling seems essential. In this way, we could implement serious game activity models and verify their expected properties.

This chapter presents, first, the proposed methodology into which such formal methods can be integrated. Second, it details the models developed for three selected games targeting different cognitive functions and the verification of various properties of these models. Using efficient and available formal tools, these models showed their acceptability for human behavior modeling.

2.1 Formal Modeling and Verification

Model checking is a method developed as early as 1980 [33, 105]. It introduces the use of automatic verification of software models. The benefit of such method is that it facilitates the identification of software design problems before implementation. The crashes of the Boeing 737 MAX aircraft give a recent and dramatic example of undetected design problems. The aircraft flight control computer is endowed with MCAS [1], an automatic software system, unbeknownst to the pilots, that automatically compensates the trend of the aircraft nose to pitch up in some situations of flight. Unfortunately, this software was not resilient to sensor failures, leading to two crashes, 346 casualties[2], and nearly 800 brand new planes banned from the sky for 20 months[3]. This sort of example shows the importance of model checking at design time for safety critical software development.

Model checking relies on three elements:

- a model – usually a state-transition system – to model the behavior of the targeted system;
- a temporal logic to specify relevant system properties (paths in the model);
- a model checker engine that can automatically verify a logical property and return a Boolean (property true or false) or a numerical value (e.g., probability to satisfy the property).

[1] https://en.wikipedia.org/wiki/Maneuvering_Characteristics_Augmenttion_System

[2] https://transportation.house.gov/imo/media/doc/2020.09.15%20FINAL%20737%20MAX%20Report%20for%20Public%20Release.pdf

[3] https://en.wikipedia.org/wiki/Boeing_737_MAX_groundings

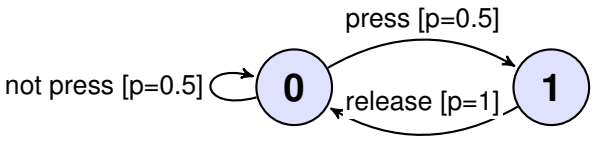


FIGURE 2.1: DTMC representing a simple game. Each edge is labeled with both an action and the corresponding probability.

Several probabilistic model checkers exist (such as UPPAAL [15], Storm [39], or PAT [115]). We decided to rely on both PRISM [78] and Storm, which have provided *probabilistic* model checkers since their beginning. PRISM is well established in the literature, and compatible with many supplementary tools such as parameter synthesis tools and also with other model checkers, such as Storm. 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 [110]. It supports several types of probabilistic models, discrete as well as continuous. On the other hand, Storm is more recent but proved to be at least as efficient as PRISM [57]. Moreover, it accepts PRISM models as input, thus it is a good alternative candidate to verify our models.

Modeling Formalisms

In this thesis we rely on discrete-time Markov chains (DTMCs), 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 for a simple two-state game is depicted in figure 2.1. In this game, the player has to press a button as many times as she wishes. The figure shows labels (press and release) over transitions. Even though they are not part of the definition of DTMCs, these labels are implemented by PRISM. This usual extension does not change the semantics of DTMCs, it allows to synchronize different modules in PRISM. Moreover, it makes the models easier to read and to understand.

To simulate and check models of activities, the simulation and checking times need not reflect the physical execution time of the activity itself. 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, restricting it to the instants when some significant events occur, hence reducing the checking time. This justifies the choice of Markov chains that do not impose to associate a real duration with each action. Moreover, as the games tackled in this thesis are "scoring" games and do not really lead to a win or loss of the player, it seemed reasonable not to use timed game automata such as UPPAAL [15] ones.

Note that the models presented in this thesis 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.

Temporal Logic

The dynamics of DTMCs can be specified thanks to the PCTL* (Probabilistic Computation Tree Logic) temporal logic [59]. PCTL* extends the CTL* logic (Computation Tree Logic) [34] 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 future state on the path, **G** (always in the future), which imposes that a property is true at every future state on the path. The until operator, **U**, is such that *p1* **U** *p2* means that property *p1* remains *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 saying that some property holds for all paths or for some paths, one says that a property holds for a certain fraction of the paths [59].

The most important operator in PCTL* is **P**, which allows to reason about the probability of event occurrences. A property `P bound [prop]` is true in a state s if the probability that the property *prop* holds in all the paths from s satisfies the bound *bound* (where a bound is a comparison operator followed by a probability value). 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, the **P** PRISM operator can be used as `P=? [prop]`; this computes the probability for *prop* to occur. For instance, the property `P =? [G (y = 0)]` expresses the probability that `y` always equals 0.

PRISM and Storm also support the notion of integer user-defined "rewards" which can be seen as counters. Their computation and value 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 in PRISM 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 [34] 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. In particular, it deals with the PCTL* fragments PCTL (Probabilistic Branching Time Logic) and PLTL (Probabilistic Linear Logic). On the other hand, Storm can only deal with PCTL properties.

PRISM and Storm propose different model checking engines. Three out of four PRISM engines (MTBDD, sparse, and hybrid) use data structures such as 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 of which only a fraction is actually reachable. Regarding Storm, we mainly use its *sparse* engine which relies on "explicit-state" data structures too.

In addition to 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". This feature allows users to obtain curves displaying the evaluation results of a property with respect to one or several variables. Besides, PRISM also proposes *statistical model-checking*, a way to test properties through several simulations.

2.2 Serious Game Modeling

Observing patients while they perform simple activities provides interesting clues on their cognitive status. To this aim, medical doctors have used "behavorial" tests 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. They offer a framework in which patients are expected to act a certain way to win: the expected behavior is well identified and it is possible to rely on different sensors (bio-metric and external) while playing the game. Thus, the set of human activities associated with these games is constrained compared to usual daily human activity. Therefore modeling and evaluation are feasible.

In the health domain, serious games can be used to incite patients to practice physical exercises [30], to train medical staff with engaging activities [27], or to help diagnose and treat patients [11, 50]. 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 set according to physicians' past experience. Properties can then be defined to extract relevant data and to compare them, first with experimental results in order to refine the model and ultimately with real patients results in order to assess the patient cognitive health.

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 to describe human behavior. We can mention two examples, though: in the context of pedestrian localization, [60] uses DTMCs to describe pedestrian trajectories; in computer vision, Hidden Markov Models (HMMs) are a popular approach for the description of activities [2, 69]. However, neither PRISM nor Storm (nor most of the other probabilistic model checkers) allow to check temporal logic properties over HMMs.

Methodology

Our methodology to study the behavior of patients playing serious games is depicted on figure 2.2. In step 1, together with the medical staff (psychiatrists, psychologists, nurses), we select a serious game that has a medical interest and we discuss with the staff to understand precisely the rules of the game. Step 2 consists in translating these rules into a conceptual model that facilitates their encoding into a PRISM model.

Together with the medical staff we choose *a priori* probabilities that we incorporate in the PRISM model (step 3) and that represent the likelihood of patients' actions. This phase also offers an opportunity to discuss 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*. 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.

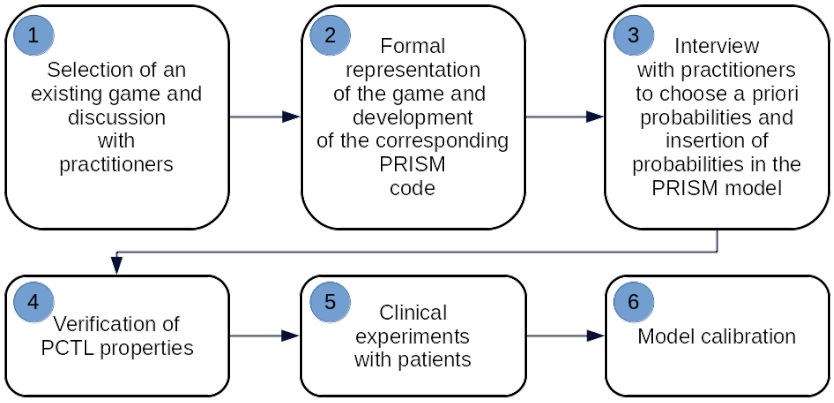


FIGURE 2.2: Workflow displaying the methodology adopted in the thesis.

For step 1, we discussed with Pr Philippe Robert (neuro-psychiatrist), Dr Valeria Manera, and Dr Jérémy Bourgeois (neuropsychologists) from Claude Pompidou institute in Nice. These discussions led to the selection of four games, each targeting a different cognitive function:

- the *Code game* [118] targets selective and sustained visual attention functions,
- the *Recognition game* [107] targets episodic memory,
- the *Inhibitory Control game* [119] targets the inhibitory control function the role of which is to inhibit some reflexes.
- the *Tapiscine game* targets the motivation functions.

All these games were developed to provide fun tools for either training or evaluating patients. The design of these games followed a gamification process of preexisting cognitive tasks or paper exercises. The first three games are training tools for patients with mild NeuroCognitive Disorders (mild NCD), whereas the fourth one evaluates motivation disorders. At the time of the game selection, only the three first games were already released, the fourth one was still under development.[4] Since these games are selected by practitioners 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. Such games may assist doctors during the diagnosis process. They could provide new non-invasive screening methods based on the observation of patient performance.

After modeling these games, we proposed a poll to psychiatrists, neuropsychologists, speech therapists, and nurses during a meeting of hospital practitioners in order to complete the game models. A dozen practitioners gave their feedback on the models (key actions, frequency of behaviors...) and they also provided us with insights concerning *a priori* probabilities on interesting actions with respect to their clinical experience, thus completing step 3.

[4]The first tests of this game started late 2019.

For step 4, as previously mentioned, we used both PRISM and Storm model checkers as PRISM modeling language can be used as input for both. An independent competition [57] compared them to other probabilistic model checkers: PRISM and Storm both showed to be the fastest in computation in this competition. In this work, we decided to use both of them and to compare their advantages and drawbacks.

For step 5, we collaborated with Alexandre Derreumaux (CoBTeK engineer) to facilitate the access to data of interest. For each game, the sequence of actions, as well as the timing of each action were saved. After analysis, these data were used to calibrate the models in step 6.

In the long run, we expect that this approach could become part of routine clinical practice. Indeed, it is an original application of model-checking that offers the possibility to compare the actual behavior of a player with different verified behavior models such as a healthy or a pathological behavior model. Considering the behavior of patients as a path in the state transition diagram of one of these models, we could automatically match each patient with a particular model. This model gives indications to the practitioners about the condition of the patient. The matching relies on the probability to follow a particular path. In the case of general practitioners, these indications could support them in their decision to send patients to a specialized consultation.

The remainder of this chapter concentrates on steps 2 and 4 for the first three games: Code game, Recognition game, and Inhibitory Control game. The fourth game was not modeled yet since, as already said, it was not fully available at the time.

2.2.1 Model Checking Usage

As seen in step 4 of figure 2.2, this work resorts to model checking techniques. However, it does not use these technique in the classical way.

The classical engineering purpose of model checking is to verify that the system design meets the system specification. In the medical domain, this approach has 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 [21] 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 [92] and [12] in which formal methods are used to detect inconsistencies of software components. In [92], such 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 [12], 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 [20] 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 [6] 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 sort of model checking aiming at design verification, our aim is to model and verify patient actions and behavior in order to help physicians refine their diagnosis about patient 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 [6], 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 behavior 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.

Hence, our use of model checking is related to human activity recognition. It is close to [55] where model checking is used to detect differences between actual medical actions and ideal ones as described in a formal model of clinical guidelines. The guidelines are represented as state-transition systems and the model checking of (CTL) temporal logic properties investigates the consistency of the formalized guidelines with the actual treatment. Similarly, in [88] the authors apply an extended temporal logic and model checking to support automated real-time recognition of daily living activities of elderly people. These approaches 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 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 was also used to debug PRISM *models* of activities, as in [99]. In our case, we use probabilities to explore paths associated with different behaviors.

2.2.2 PRISM Modeling Language

PRISM provides a state-based modeling language inspired from the reactive module formalism of [3]. 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:

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

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 2.1 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* [79], 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.

Algorithm 1 PRISM code for figure 2.1 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
```

We model the behavior of a patient playing serious games with discrete-time Markov chains (DTMCs).

As we started modeling with the PRISM modeling language, we had to cope with one of its limitations: in PRISM Markov chains, it is not possible to put a limit on the number of times the model can loop over a state. Even with a low probability on the loop transition, there is still a risk for a simulation to never quit the loop (fairness is not imposed). To avoid this issue, all the possible states must be explicitly represented in the model, leading to an "unwound" model. But this unwinding turns out to be an advantage to implement the effect of patients' fatigue; indeed as the game proceeds, patients are likely to become tired, thus the probabilities of some actions should vary accordingly. Thus, we included in all game models a simplified yet realistic fatigue profile based on *pre-computed* probabilities at each time step of the game in the unwound model.

The following sections describe the implementation of each behavior model and some properties of interest. As a first step, sections 2.3 to 2.6 show the results of the automatic verification of these properties with PRISM. As a second step, section 2.7.2 shows the comparison of the results and time performances of PRISM and Storm.

FIGURE 2.3: Display of the Code game.

2.3 Code Game Model

In the Code game [118] 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 2.3). 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.

2.3.1 Model

A simplified pseudo-code program describing this game is given in Listing 2.1.

```
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 2.1: 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 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.

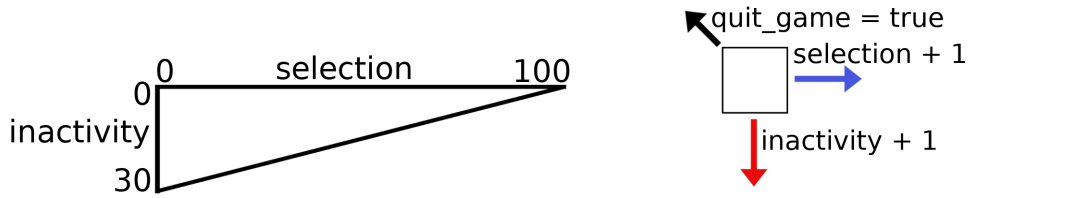
As stated in the previous section, we use a DTMC to model the behavior of a patient playing this game. Due to a limitation in PRISM, as explained in subsection 2.2.2, we must explicitly represent all the possible states in the model.

Since the game activity lasts at most five minutes (or 300 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).

Model Design

With the previous assumption, step 2 of figure 2.2 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 2.1). 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 2.4a). The edge of length 100 (representing the scenario of a succession of *selections*) and the edge of length 30 (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 2.4b. 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



(A) Combinations of events triangle.

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

FIGURE 2.4: Activity model for a player of the Code game.

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*.

PRISM Implementation

The model is composed of a single module called "Code_game"[5]. 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.

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):

$$inactivity = \left\lceil \left(-\frac{inactivity\_max}{selection\_max} \right) \times action + inactivity\_max \right\rceil \tag{2.1}$$

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*.

[5] PRISM code at https://gitlab.com/ThibLY/thibaud_l_yvonnet_phd_thesis_models_and_scripts

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_interac-tion* 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 "Code_game" module (enumerated below). 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 2.2, 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 2.2: Excerpt from the Code_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 NeuroCognitive Disorder (mild NCD). These probabilities are used in the following section.

2.3.2 Verification

For the model described in the previous section, we encoded and tested several properties in PCTL* (step 4 of figure 2.2).

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 impairment.

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

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 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 2.1, 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, there is the same structure checking the values of *selection* and *inactivity* together with another structure checking the value of the variable *location*. 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 expected behavior of a mild NCD 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 properties 4,5, and 6 should be observed quite rarely (respectively

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 2.1: Results from property 2 to 6.

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 mild NCD 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.

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 2.2a and 2.2b.

Property 7 returns a Boolean[6]. 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 *Inactivity_bool_reward*. According to its results, the average "score" for a cohort of patients matching this model parameters

[6] 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 2.2: Results for the properties concerning the quality of actions.

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 is 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).

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?

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

where *i* is the number of steps to perform before checking the reward and *C* is the "cumulative-reward" operator presented in section 2.1. This property is also applied to *Sad_smiley_reward*, *Inactivity_bool_reward*, *Gaming_time*, and *Leave_game_reward*.

In figure 2.5, 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.

In figure 2.6, 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 2.1 in section 2.3 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

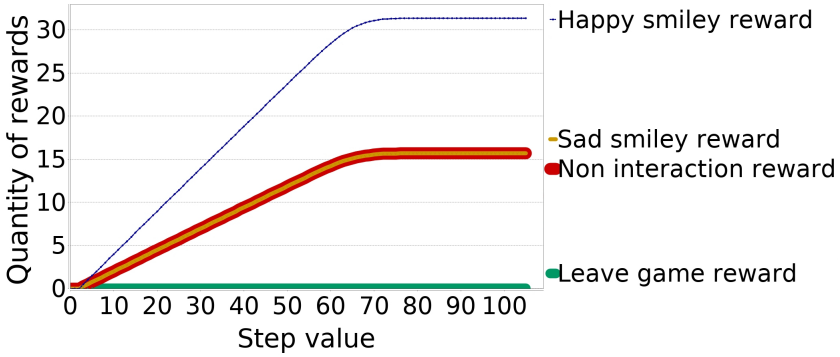


FIGURE 2.5: Average model checking results for rewards related to good answers, bad answers, non-interaction, and game leaving behavior.

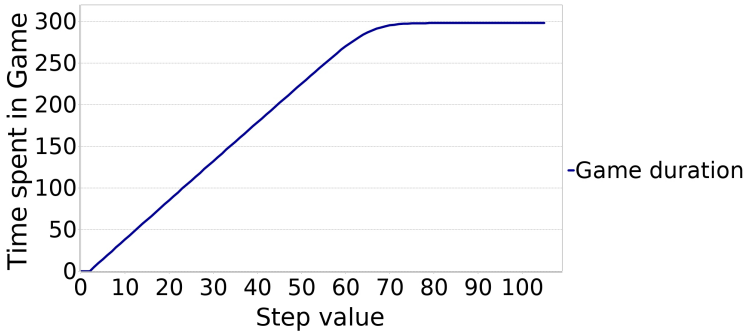
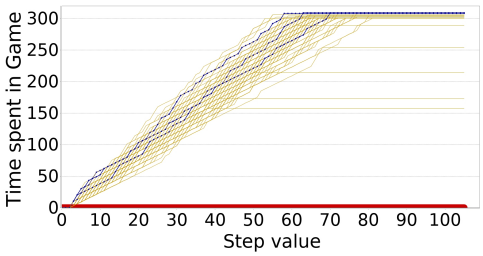
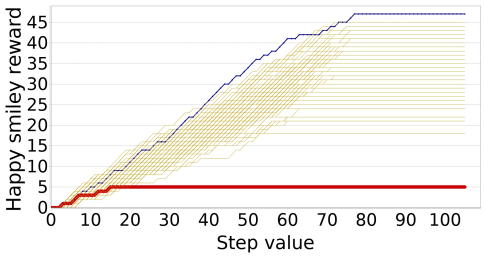


FIGURE 2.6: Average duration of the game obtained with model checking.



(A) Duration of the game over 100 runs.



(B) Accumulation of good answers over 100 runs.

FIGURE 2.7: Experiment results on the accumulation of rewards over 100 simulation runs.

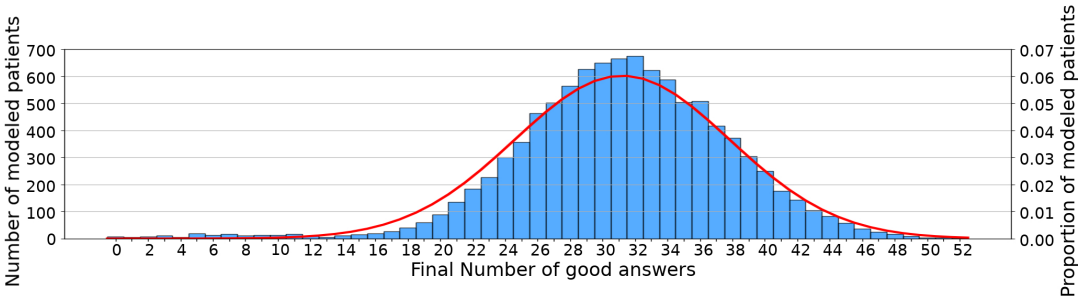


FIGURE 2.8: Frequency of good answers over 10,000 runs (in blue) and its fitting normal distribution with $\mu = 31.2131$ and $\sigma^2 = 43.9271$ (in red).

duration reaches the plateau around the seventy-fifth step. This is due to the fact that most of the paths go several times through non-interaction transitions. Should they not go through these transitions at all, the plateau might have been reached around the 100^{th} step.

In figure 2.7a, 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 2.1). Among these 100 simulations, some do not reach 300 seconds, one of them (in red in figure 2.7a) 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 2.7, 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 2.7 as well as in figure 2.5, 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 2.7, a shell and a Python scripts were written to retrieve raw data from simulations. These data are used in figure 2.8 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 Code 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.

2.4 Pre-computed Time Dependent Probabilities for Code 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 [122] and early Alzheimer's subjects [64]. 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 Code 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 2.3: Probability definition of the new Code_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 [122, 64] 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 2.3.2. The new results from property 2 to property 6 are shown in table 2.3.

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 2.3: 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 2.1). 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 2.1.

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

Discussion on reward based properties Table 2.4a 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.

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

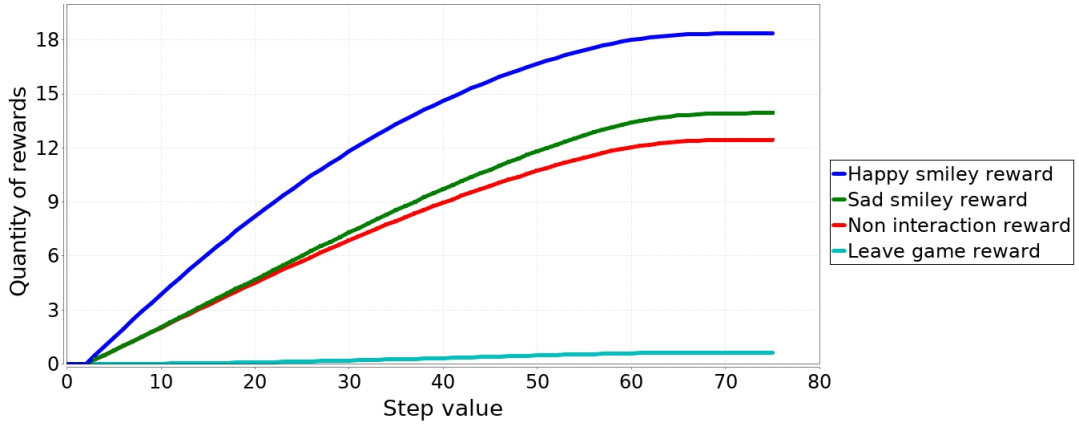


FIGURE 2.9: Average model checking results for rewards related to good answers, bad answers, non-interaction, and game leaving behaviors in the new version of the model.

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.

Discussion on figures 2.9 and 2.6 There are several changes in figure 2.9 compared to figure 2.6. The change of maximum reward accumulation is similar to the one already depicted in table 2.4a, but the shape of the curves changes. The curve of happy smiley reward is most noticeable as time goes, as 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.

This last observation explains figure 2.10. 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 Code 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 of fatigue.

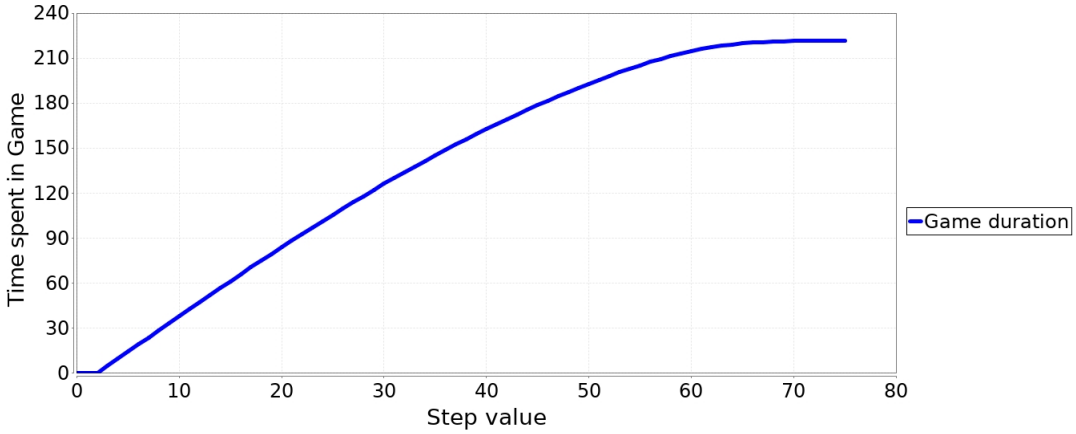


FIGURE 2.10: Average duration of the game obtained with model checking in the new version of the model.

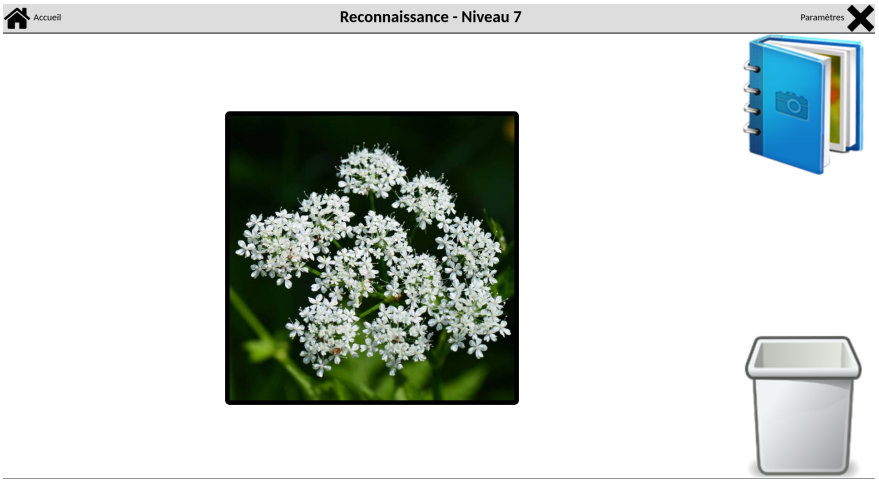


FIGURE 2.11: Display of the Recognition game, with a flower example of an intermediate complexity picture.

2.5 Recognition Game Modeling and Verification

The Recognition game is part of the MeMo Web platform [7] which provides several serious games to train different neuro-cognitive functions. This game trains the memory functions. A set of pictures is displayed one by one in front of the patient. 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 appears 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.

The difficulty increases with the number of unique, original, and duplicate pictures as well as 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] https://games.memory-motivation.org/?lang=en

(A) Easy picture. (B) Intermediate picture. (C) Difficult picture.

FIGURE 2.12: Example of pictures of various complexity. Picture 2.12c is categorized as difficult because it is displayed some time after 2.12b which is close in color and in flower shape.

2.5.1 Model Design

To simplify the modeling task on this game we designed, with the help of 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 to 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 subsection, we focus on the third level of difficulty, using flower pictures.

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 "duplicate". The second describes the complexity of the picture and can be "easy", "intermediate", or "difficult" (see figure 2.12).

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" (right 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 Code game model, the time dependent probabilities of the unwound model are pre-computed to take into account the patient's fatigue.

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 2.13.

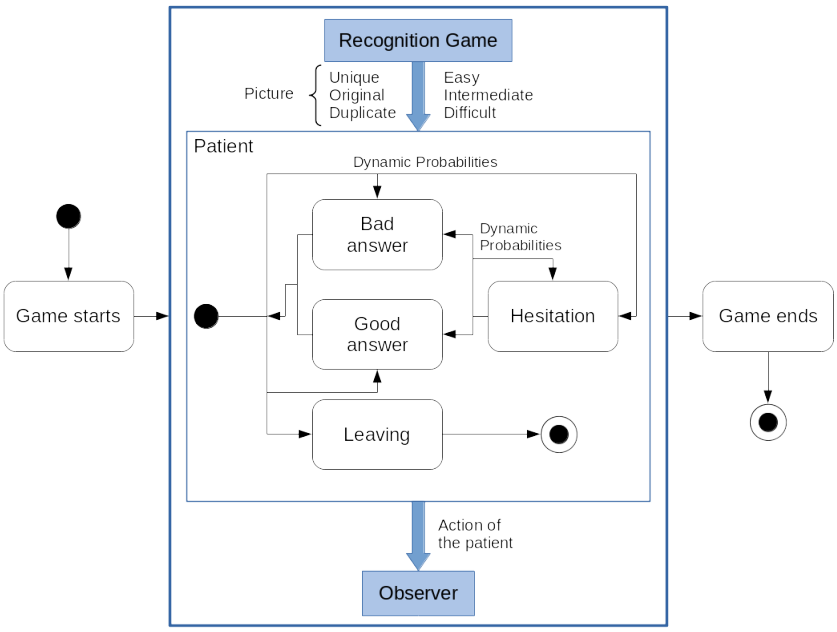


FIGURE 2.13: Recognition game module interactions.

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 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[8]; 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* 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 2.4 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 2.2).

```
// 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
```

[8]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 game sessions.

```
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 2.4: Probability calculation for an easy unique/original picture.

Listing 2.4 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).

2.5.2 Verification

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 2.5 summarizes the correspondence between the model variables and what they represent.

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 Code 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.

Medically Oriented Properties

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

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 2.5: Summary of Recognition game model variables.

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

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

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

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

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?

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

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

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

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

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

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 2.6: Results from property 1 to 5. Result of property 5 is given for $i = 85$.

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

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

Discussion The results for these properties are displayed in table 2.6, 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 of the patient hesitating 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

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 2.7: Results of property 6 for various rewards.

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 when 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.

Property 6 is reward-based and gives an idea of the overall performance of a patient during the game. It 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 2.7, together with their computing time. 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 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 mild NCD diagnosis. The reward-related properties results are consistent with these previous results as the modeled patient makes mistakes on nearly all duplicate pictures and even on unique and original pictures. Thus, the patient would need a deeper examination from medical practitioners.

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 2.14 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?

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

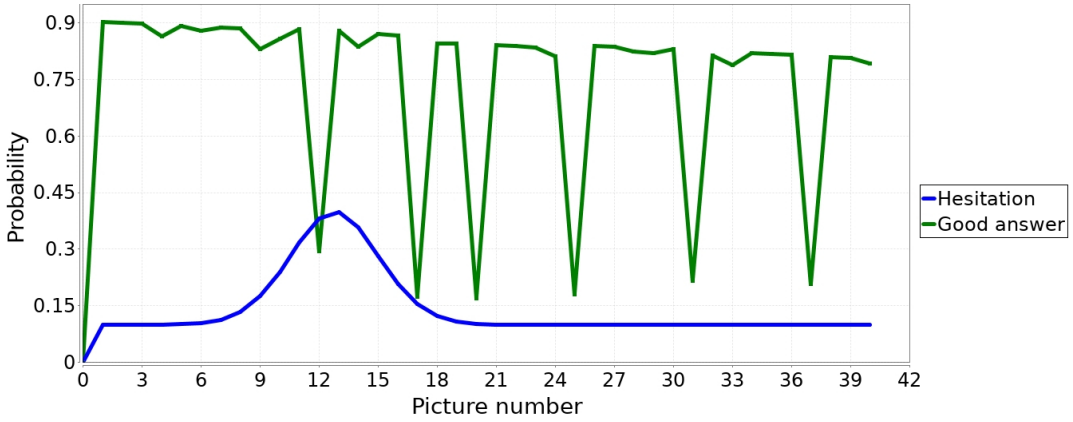


FIGURE 2.14: Probabilities to give a good answer and to hesitate per picture.

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

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

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 2.14 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.

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 2.15 shows the accumulated values of rewards (using the C operator defined in section 2.1) during a game session. For this purpose, we use the following property:

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

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

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.

Discussion Figure 2.15 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

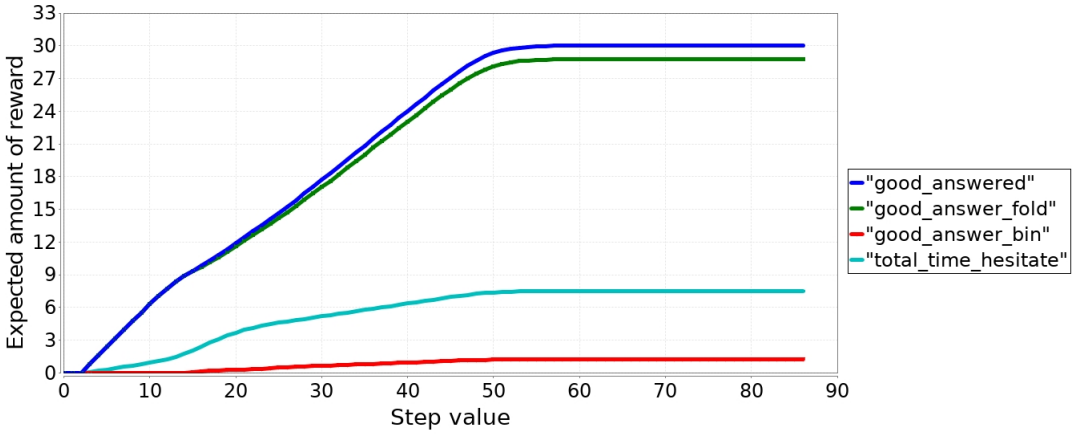


FIGURE 2.15: Average model checking results for rewards related to good answers and hesitation.

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 (2.15) 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 Code 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 Code 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.

2.6 Inhibitory Control Game Modeling and Verification

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 functions 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.

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 2.16 left). Knowing that human's visual reaction time for a healthy adult performing a "go/no-go" task is around 259 ms [46], 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

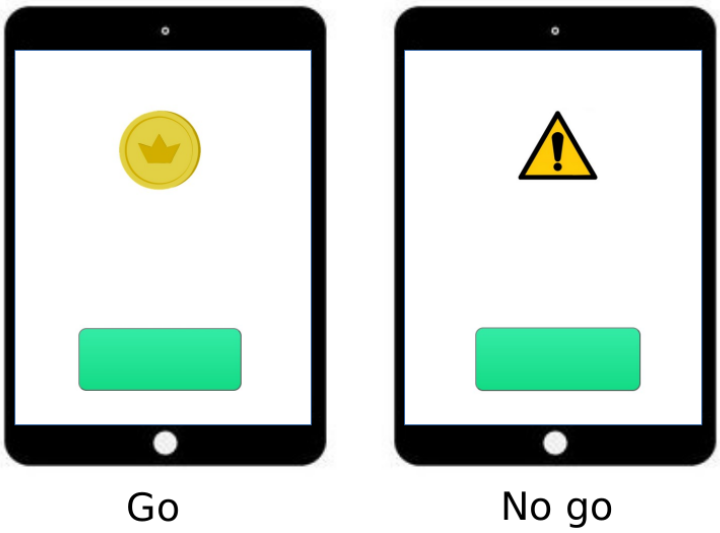


FIGURE 2.16: Display of the Inhibitory Control game, with both signals. The coin is the target and the warning sign is the decoy.

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 [46]. 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 2.16 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.

2.6.1 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.

PRISM Implementation

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

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 2.5 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 2.5: 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 other labels are associated with transitions leading either to a new signal or to the end of the game.

Figure 2.17 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, 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 2.17 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 2.6 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 2.6: 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 2.2). 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 2.2). All the parameters will be adapted with respect to the results of clinical experimentation.

2.6.2 Verification

To ease the comprehension of the following section, table 2.8 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 2.3.2), we will not describe it in this section. For this model, the model checker takes 1.225 s to check this property.

Medically Oriented properties

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

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 indicating that the end of the game is reached.
game_on	Boolean indicating if the game is on or off.

TABLE 2.8: Summary of Inhibitory Control game model variables.

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	2.1374×10^{-4}	0.53

TABLE 2.9: Results from property 1 to 6.

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?

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

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 The results for these properties are displayed in table 2.9 and table 2.10, together with their computing times.

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	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 2.10: Results of property 7 for various rewards.

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.

Property 7 is reward-based and gives an idea of the overall performance of a patient during the game. It 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 2.10, together with their computing times. 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.

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. The reward related results are consistent with these low chances According to medical practitioners, this behavior could be observed on patients suffering from early stages of dementia diseases such as Alzheimer or Parkinson.

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 2.18, 2.19, and 2.20 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 2.8.

Figure 2.18 shows the results for property 8 with *num_act* equal to *num_act_targ*, *prev_signal* equal to *prev_targ* and with *action* being 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 2.19 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 2.20 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 2.21 and 2.22 present the result for the following cumulative reward property:

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

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

We applied the same property to other rewards: *"good_on_target"*, *"good_on_decoy"*, *"good_on_signal"* (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"*.

The two figures 2.21 and 2.22 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 2.20. 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 2.20 gives a clear and precise idea of the probability distribution within a whole game session, figures 2.21 and 2.22 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 2.21 and 2.22 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 2.20 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 hundredths 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).

2.7 Experience Feedback

Translating the rules of the different games (as given by medical practitioners) into a formal model suitable for model checking is not straightforward and there is no unique modeling approach, let alone a translation tool. This process highly depends on the modeler's personal experience with the selected tool and with activity modeling. In the previous sections, we described our modeling approaches for three different serious games, each having its own requirements. These three models not only satisfy the requirements but also permit verification with model-checking. However, as mentioned before, we are aware that this approach is not the only possible one.

First, for a given game, more than one model may be necessary, at least possibly two of them: one for the patient's behavior and the other one for the program that implements the game. The patient's model should represent possible "deviant" behaviors with respect to the rules. The game model represents, on the one hand, the intrinsic game strategy and, on the other hand, possible reactions to patients' actions. These two models, although different, must communicate and exchange various kinds of information. In our case, PRISM provides a synchronization mechanism based on transition labels. This mechanism corresponds to a *rendez-vous* communication: the first module which reaches a transition waits for the other ones to reach the transition bearing the same label. Second, on the formal side, all the models must satisfy the applicability hypotheses of model checking in particular a discrete and bounded time and a finite (possibly huge) state space. In case these hypotheses do not hold, it is necessary to slightly modify the game while keeping its basic principles. For this purpose, serious games specialists should be involved once more to validate the required changes and to respect the game purposes. Examples of parameter tuning include reducing the bounding game duration or limiting the number of steps in the game. This feedback loop between game implementation and model design helps manage model complexity.

When modeling our case study games, we adopted and implemented different modeling strategies corresponding to the specific features of each game. Based on our modeling experience with these three serious games, we identified three different factors impacting the difficulty to model a serious game. These factors are: time, past event dependency, and action randomness. These factors, even more when they are combined, may prevent game modeling unless the game parameters are tweaked suitably.

2.7.1 Game Models

The Code game only requires that the patient touches pictures displayed onto the screen within five minutes. The quality of the answer for each picture is right or wrong and does not depend on the picture itself nor on past events. The only factor of difficulty in this game is its timed aspect which makes it rather easy to model. In this kind of games, only the patient's model is needed: it is a simple state machine which is the same for all the game steps. Of course time must be discretized.

In the Recognition game the patient's answer is based on previous game events. Thus, this game requires two models, one representing the game sequence of events, the other the patient's behavior. For the patient's model to take past game events into account, the states of the automaton for a game step reflect the event history that allowed to reach them. Thus the patient's model can adjust the probabilities associated with state transitions depending on the current state of the second model, the game one. In a previous version of this game, the sequence of events was entirely random. The combination of the past event dependency, the random aspect but also the size of the game made model-checking difficult. After discussion with specialists, the random aspect was removed. In its current version, the only difficulty that remains is to model past event dependency. Even though the physical time is not a parameter of such games, we include it in the patient's model when it is relevant to the patient's behavior analysis.

In the Inhibition Control game the patient has to perform an action, as fast as possible, based on the random display of a visual signal. To model this game, as for the previous game, we need two models. However, the game model is more complex than for the Recognition game: it must incorporate the random selections of both the signal to display and the delay to display it. This random behavior greatly increases the state space. Thus, this game combines the three factors of difficulty (time, past event dependency, and randomness) making it the most complicated game to model.

The solutions we presented can be seen as a first contribution to set up a methodology of serious game modeling. To precisely define this methodology, to refine and validate it, more games with similar and different features should be modeled.

2.7.2 PRISM and Storm Comparison

In this section we compare PRISM and Storm results on all properties described in the previous sections (from 2.3 to 2.6). For the sake of readability, all results are truncated to 3 decimals but they are generally equal (in PRISM and Storm) down to the ninth decimal.

In some rare cases, we noticed a difference in the results given by PRISM and Storm. With the help of Tim Quatmann of the Storm team, we figured out that the accuracy of PRISM and Storm are not the same. Indeed, when computing properties, PRISM algorithms often use iterative numerical methods. The iteration stops once the maximum difference between elements in the solution vectors drops below a threshold named "termination epsilon". This threshold is set by default to 10^{-6} but can be redefined by the user. In the cases when PRISM does not give the same result as Storm, the threshold is not low enough and changing it to 10^{-12} solves the issue.

Code game models

In the following tables[9], properties are numbered as in sections 2.3 to 2.6. When several properties are given the same main number, it indicates that they are all derived from a generic property with the same number in the previous sections to obtain real implementations of properties that can be tested. For example, property 8 from subsection 2.6.2 is written as follows in this subsection:

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

where *num_act*, *i*, *prev_signal* and *action* are generic variables that have to be replaced with actual variables such as in:

$$P =? [\, F\ (num\_act\_targ = 5\ \&\ prev\_targ\&(several\_click)\&!transiting)]$$

to be able to test it. Another case is for example property 8 from subsection 2.3.2 which was written using the reward associated with happy smileys:

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

but which can also be written with other rewards, for instance

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

The two first tables (2.11 and 2.12) display the computation times for the two Code game versions, with and without taking fatigue into account.

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
1- P=? [F location=location_max]	1.0	0.002	1.0	0.002
2- P=? [F (selection=0) & (inactivity=inactivity_max)]	8.544E-19	0.007	8.544E-19	0.001
3- P=? [F (selection=selection_max) & (inactivity=0)]	3.050E-13	0.019	3.050E-13	0.001
4- P=? [F (selection=43) &(inactivity=18)]	2.318E-2	0.017	2.318E-2	0.003
5a- P=? [F ((selection=0) & (inactivity=0)) U (location=location_max)]	4.997E-4	0.03	-	-
5b- P=? [((selection=0) & (inactivity=0)) U (location=location_max)]	4.997E-4	0.002	4.997E-4	0.000
6- P=? [F (quit_game)]	3.136E-2	0.028	3.136E-2	0.004
7- R"Happy_smiley_reward">=30 [F location=location_max]	true	0.016	true	0.005

TABLE 2.11: Code game without fatigue: results of properties 1 to 7 and computation times (in seconds).

Regarding property 5, it belongs to PLTL but not to PCTL. Indeed, finding a formula expressing "in the future" and "until" using only one operator is challenging. However, both Code game models (with and without fatigue) are particular cases. Indeed, the formula "What is the probability for patients to take a path in which, at

[9] For the sake of page layout, in the tables of this section, the notation '$\times 10^{xx}$' is replaced by 'Exx'

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
1- P=? [F location=location_max]	1.0	0.002	1.0	0.001
2- P=? [F (selection=0) & (inactivity=inactivity_max)]	2.173E-17	0.027	2.173E-17	0.001
3- P=? [F (selection=selection_max) & (inactivity=0)]	4.896E-16	0.029	4.896E-16	0.001
4- P=? [F (selection=43) & (inactivity=18)]	1.229E-2	0.162	1.229E-2	0.002
5a- P=? [F ((selection=0) & (inactivity=0)) U (location=location_max)]	4.997E-4	0.04	-	-
5b- P=? [((selection=0) & (inactivity=0)) U (location=location_max)]	4.997E-4	0.003	4.997E-4	0.000
6- P=? [F (quit_game=true)]	6.395E-1	0.236	6.395E-1	0.004
7- R"Happy_smiley_reward">=30 [F location=location_max]	false	0.228	false	0.005

TABLE 2.12: Code game with fatigue: results of properties 1 to 7 and computation times (in seconds).

some point in the future, their selection and activity score are equal to zero until they leave the game?" verifies the same paths as the formula "What is the probability for patients to keep their selection and activity score equal to zero until they leave the game?". As a result, both formulas 5a and 5b have the same probabilities.

Regarding speed performance, Storm is slightly faster than PRISM with a gain factor of the order of 10 for properties 3, 4, 6, and 7 of table 2.11 and for properties 2 and 3 and of table 2.12. Storm can be even faster on other properties with, for example, a gain factor of the order of 100 for properties 4, 6, and 7 of table 2.12. This difference between PRISM and Storm is due to the increased size of the model compared to the model without fatigue.

The next tables focus on the last properties of the two Code game versions. The PRISM formulation of property 9a is rather complex. Because of the the nesting of G operators, it is impossible to verify this formula with Storm. Indeed, this type of nesting (which characterizes PLTL formulas included in PCTL*) is not supported by Storm. Thus, following Tim Quatman's advice, we wrote another formula in pure PCTL (property 9b) that checks the same paths for Storm. This new property relies on probabilities to gather rewards, a feature not supported by PRISM. The probability to verify this property is exactly the same as in PRISM.

The different properties labeled as 11 show, for each different reward, the same particular case of property 11 used for figure 2.6 "Average duration of the game obtained with model checking" in subsection 2.3.2. We arbitrarily chose the case where $i = 50$ to compare results and computation times for both model checkers

As seen in the previous pair of tables, Storm can be faster in some cases, especially with the model that takes the fatigue into account. We can highlight property 9 from both table 2.13 and table 2.14 as they respectively take 1.92 and 3.717 seconds to be computed by PRISM whereas Storm computes them within 0.288 and 0.315 second.

The two Code game models were useful to compare PRISM and Storm as they started to show a difference in computation as well as some limits and features from both model checkers.

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
8a- R"Happy_smiley_reward"=? [F location=location_max]	31.364	0.016	31.364	0.005
8b- R"Sad_smiley_reward"=? [F location=location_max]	15.682	0.015	15.682	0.005
8c- R"Non_interaction_reward"=? [F location=location_max]	15.682	0.015	15.682	0.005
9a- P=? [(F happy_smiley) & (G((happy_smiley) => (X G !happy_smiley & !quit_game)))]	3.301E-12	1.92	-	-
9b- P=? [true U ^ {rew{"Happy_smiley_reward"}<=1, rew{"Happy_smiley_reward"}>=1, rew{"Leave_game_reward"}<=0} (location = 2)]	-	-	3.301E-12	0.288

TABLE 2.13: Code game without fatigue: results of properties 8 to 9 and computation times (in seconds).

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
8a- R"Happy_smiley_reward"=? [F location=location_max]	18.385	0.227	18.385	0.005
8b- R"Sad_smiley_reward"=? [F location=location_max]	13.942	0.222	13.942	0.005
8c- R"Non_interaction_reward"=? [F location=location_max]	12.461	0.222	12.461	0.005
9a- P=? [(F happy_smiley) & (G ((happy_smiley) =>(X G !happy_smiley & !quit_game)))]	1.647E-9	3.717	-	-
9b- P=? [true U ^ {rew{"Happy_smiley_reward"}<=1, rew{"Happy_smiley_reward"}>=1, rew{"Leave_game_reward"}<=0} (location = 2)]	-	-	1.647E-09	0.315

TABLE 2.14: Code game with fatigue: results of properties 8 to 9 and computation times (in seconds).

Recognition game model

The next table (2.17) presents the results for a big part of the Recognition game properties. As property 8 is quite long to express in PRISM, it will be presented separately.

The model checking of the Recognition game model shows fewer differences in term of computation time as none of the checked properties showed a speed gain factor of the order of 10. On the other hand, a slight difference in the results tends to show a difference in the algorithm's accuracy. Properties 2, 4, and 5a show really small differences as the results are of the same order of value. Property 5b shows a slightly bigger difference as PRISM gives 0 and Storm gives 1.815×10^{-13}, which is

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
10- P=? [F (selection=1 & happy_smiley)]	6.662E-1	0.005	6.662E-1	0.001
11a- R"Happy_smiley_reward"=? [C<=50]	23.708	0.01	23.708	0.002
11b- R"Sad_smiley_reward"=? [C<=50]	11.854	0.011	11.854	0.001
11c- R"Non_interaction_reward"=? [C<=50]	11.854	0.011	11.854	0.001
11d- R"Leave_game_reward"=? [C<=50]	2.370E-2	0.007	2.370E-2	0.001
11e- R"Gaming_time"=? [C<=50]	225.227	0.007	225.227	0.001

TABLE 2.15: Code game without fatigue: results of properties 10 to 11 and computation times (in seconds)

Property number and formula	PRISM		Storm	
	Result	Time	Result	Time
10- P=? [F (selection=1 & happy_smiley)]	6.633E-1	0.004	6.633E-1	0.001
11a- R"Happy_smiley_reward"=? [C<=50]	16.680	0.23	16.680	0.001
11b- R"Sad_smiley_reward"=? [C<=50]	11.817	0.215	11.817	0.001
11c- R"Non_interaction_reward"=? [C<=50]	10.733	0.216	10.733	0.001
11d- R"Leave_game_reward"=? [C<=50]	4.714E-1	0.216	4.714E-1	0.001
11e- R"Gaming_time"=? [C<=50]	192.831	0.221	192.831	0.001

TABLE 2.16: Code game with fatigue: results of properties 10 to 11 and computation times (in seconds)

however still close to 0. These differences are corrected when the termination epsilon is set to 10^{-12}.

Property 8 of subsection 2.5.2 can be divided into two properties. The first one checks for the probability to give a good answer on picture number i where $i = 30$:

$$P =? [F\ (nb\_pictures = 30\ \&\ ((choose\_folder\ \&\ (nb\_pictures <= 3$$
$$|\ nb\_pictures = 5\ |\ (nb\_pictures >= 7\ \&nb\_pictures <= 8)$$
$$|\ nb\_pictures = 11\ |\ nb\_pictures = 13\ |\ (nb\_pictures >= 15\ \&\ nb\_pictures <= 16)$$
$$|\ (nb\_pictures >= 26\ \&\ nb\_pictures <= 27)\ |\ nb\_pictures = 30$$
$$|\ (nb\_pictures >= 34\ \&\ nb\_pictures <= 36)\ |\ (nb\_pictures >= 38$$
$$\&\ nb\_pictures <= 39)\ |\ nb\_pictures = 6\ |\ nb\_pictures = 18\ |\ (nb\_pictures >= 21$$
$$\&\ nb\_pictures <= 23)\ |\ nb\_pictures = 28\ |\ nb\_pictures = 32\ |\ nb\_pictures = 40$$
$$|\ nb\_pictures = 10\ |\ nb\_pictures = 14\ |\ nb\_pictures = 19\ |\ nb\_pictures = 29$$
$$|\ nb\_pictures = 9\ |\ nb\_pictures = 33\ |\ nb\_pictures = 4\ |\ nb\_pictures = 24))$$
$$|\ (choose\_bin\ \&\ (nb\_pictures = 17\ |\ nb\_pictures = 20\ |\ nb\_pictures = 25$$
$$|\ nb\_pictures = 37\ |\ nb\_pictures = 12\ |\ nb\_pictures = 31))))]$$

Property	PRISM		Storm	
	Result	Time	Result	Time
0a- P=? [F finish_play & !game_on]	9.921E-1	0.047	9.921E-1	0.029
0b- P=? [F finish_play & !front_screen]	1.0	0.014	1.0	0.007
1- P=? [F (counter=40 & !front_screen & !quit_game)]	1.022E-3	0.006	1.022E-3	0.005
2- P=? [G (!game_on | (game_on & nb_pictures=0) | (nb_pictures>=1 & !choose_bin) & !quit_game)]	9.304E-4	0.031	9.295E-4	0.023
3- P=? [F (hesi_count=5)]	2.935E-2	0.029	2.935E-2	0.020
4- P=? [F (counter=240)]	0.0	0.005	1.676E-113	0.006
5a- P=? [F (counter=85)]	1.525E-13	0.02	4.919E-13	0.011
5b- P=? [F (counter=86)]	0.0	0.02	1.815E-13	0.011
5c- P=? [F (counter=30)]	9.952E-1	0.006	9.952E-1	0.002
6a- R"good_answered"=? [F finish_play & !front_screen]	30.002	0.105	30.002	0.027
6b- R"good_answer_fold"=? [F finish_play & !front_screen]	28.762	0.106	28.762	0.030
6c- R"good_answer_bin"=? [F finish_play & !front_screen]	1.240	0.092	1.240	0.027
6d- R"total_time_hesitate"=? [F finish_play & !front_screen]	7.523	0.092	7.523	0.026
7- P=? [F (nb_pictures=30 & hesitate)]	9.942E-2	0.016	9.942E-2	0.018s
9a- R"good_answered"=? [C<=30]	17.701	0.013	17.701	0.004
9b- R"good_answer_fold"=? [C<=30]	17.042	0.023	17.042	0.004
9c- R"good_answer_bin"=? [C<=30]	6.585E-1	0.015	6.585E-1	0.004
9d- R"total_time_hesitate"=? [C<=30]	5.194	0.01	5.194	0.004
9e- R"total_time_spent"=? [C<=30]	27.932	0.013	27.932	0.004
10- P=? [F (counter=30 & hesitate)]	1.170E-1	0.007	1.170E-1	0.008

TABLE 2.17: Recognition game results and computation times (in seconds) for properties 6 to 10

The second one checks for the probability to give a good answer on instant number i where $i = 30$:

$$P =? [F\ (counter = 30\ \&\ ((choose\_folder\ \&\ (nb\_pictures <= 3 \mid nb\_pictures = 5 \mid (nb\_pictures >= 7\ \&\ nb\_pictures <= 8) \mid nb\_pictures = 11 \mid nb\_pictures = 13 \mid (nb\_pictures >= 15\ \&\ nb\_pictures <= 16) \mid (nb\_pictures >= 26\ \&\ nb\_pictures <= 27) \mid nb\_pictures = 30 \mid (nb\_pictures >= 34\ \&\ nb\_pictures <= 36) \mid (nb\_pictures >= 38\ \&\ nb\_pictures <= 39) \mid nb\_pictures = 6 \mid nb\_pictures = 18 \mid (nb\_pictures >= 21\ \&\ nb\_pictures <= 23) \mid nb\_pictures = 28 \mid nb\_pictures = 32 \mid nb\_pictures = 40 \mid nb\_pictures = 10 \mid nb\_pictures = 14 \mid nb\_pictures = 19 \mid nb\_pictures = 29$$

$$| \ nb\_pictures = 9 \ | \ nb\_pictures = 33 \ | \ nb\_pictures = 4 \ | \ nb\_pictures = 24))$$
$$| \ (choose\_bin \ \& \ (nb\_pictures = 17 \ | \ nb\_pictures = 20 \ | \ nb\_pictures = 25$$
$$| \ nb\_pictures = 37 \ | \ nb\_pictures = 12 \ | \ nb\_pictures = 31))))]$$

PRISM and Storm give the same result on both properties (8.296×10^{-1} for the first version and 6.138×10^{-1} for the second version) for approximately the same computation time (respectively 0.03s and 0.02s for PRISM; 0.019s and 0.008s for Storm). The reason these properties have so long formula is that for each instant checked by the model checker, we want to verify which picture is associated with this instant to know in which category it shall be sent to ("first-seen" or "duplicate") and what is the action performed by the player.

This third model did not show any major difference in computation speed between PRISM and Storm but it did show some slight differences in results.

Inhibitory Control game model

The next following tables show the results and computation times of the Inhibitory Control game model checking.

Property	PRISM		Storm	
	Result	Time	Result	Time
0- P=? [F (!game_on & next_end)]	1.0	0.429	1.0	0.132
1- P=? [G (prev_targ & !transiting) => (click_fast | click_slow)]	4.773E-4	0.423	4.773E-4	0.464
2- P=? [F (prev_deco & !transiting & several_click)]	2.504E-1	0.488	2.504E-1	0.516
3- P=? [G (prev_deco & !transiting) => (several_click)]	5.471E-7	0.371	1.057E-08	0.436
4- P=? [G ((prev_deco | prev_targ) & !transiting) => (several_click)]	7.273E-08	0.241	1.110E-16	0.338

TABLE 2.18: Inhibitory Control game property results and computation times (in seconds) for properties 0 to 4.

The first table (2.18) does not show any major differences in PRISM and Storm computation times but it shows some differences in the results. Property 3 shows a difference (corrected by the termination epsilon calibration) of a factor of 10 and property 4 shows a difference of a factor of 10^8. The latter difference is reduced to 10^3 with a termination epsilon of 10^{-12}. It seems that the model checking algorithm is more accurate in Storm than in PRISM. To better understand this observation, we asked the Storm development team for an opinion. They noticed that this model is kind of acyclic. As Storm has specifically tailored algorithms for this type of models, this might be the reason why it is relatively easy to solve for it.

In the second table (2.19) there is a slight difference of computation speed for property 5.

In table 2.20, the properties labeled as 7 show differences in computation speed with approximately 1.5s for PRISM and 0.5s for Storm. Notably with property 7a for which PRISM takes 2.428 s whereas Storm only takes 0.534 s to compute it. This computation time can be explained because this property computes the average accumulation of rewards from the beginning of the game until its end. Thus, a large part of the model states are explored to compute this property. It seems that Storm

Property	PRISM		Storm	
5- P=? [(((prev_targ & !transiting) => (click_fast | click_slow)) & ((prev_deco & !transiting) => (not_click))) U (num_sign=13)]	1.190E-4	0.211	1.190E-4	0.076
6- P=? [(((num_sign>=13 & prev_targ & !transiting)) => (click_fast | click_slow)) & ((num_sign>=13) & prev_deco & !transiting) => (not_click)))) U (next_end & !game_on)]	2.137E-4	0.49	2.137E-4	0.377

TABLE 2.19: Inhibitory Control game property results and computation times (in seconds) for properties 5 to 6.

Property	PRISM		Storm	
7a- R"good_on_target"=? [F (!game_on & next_end)]	5.556	2.428	5.556	0.534
7b- R"good_on_decoy"=? [F (!game_on & next_end)]	8.673E-1	1.434	8.673E-1	0.526
7c- R"good_on_signal"=? [F (!game_on & next_end)]	6.423	1.407	6.423	0.577
7d- R"bad_on_target"=? [F (!game_on & next_end)]	4.443	1.536	4.443	0.544
7e- R"bad_on_decoy"=? [F (!game_on & next_end)]	4.132	1.51	4.132	0.532
7f- R"bad_on_signal"=? [F (!game_on & next_end)]	8.576	1.563	8.576	0.534
8a- P=? [F (num_act_targ=5 & prev_targ & (click_fast) & !transiting)]	2.690E-1	0.155	2.690E-1	0.195
8b- P=? [F (num_act_targ=5 & prev_targ & (click_slow) & !transiting)]	3.404E-1	0.145	3.404E-1	0.168
8c- P=? [F (num_act_targ=5 & prev_targ & (anticipate) & !transiting)]	1.245E-1	0.144	1.245E-1	0.169
8d- P=? [F (num_act_targ=5 & prev_targ & (not_click) & !transiting)]	4.896E-2	0.151	4.896E-2	0.169
8e- P=? [F (num_act_targ=5 & prev_targ & (double_click) & !transiting)]	9.792E-2	0.141	9.792E-2	0.167

TABLE 2.20: Inhibitory Control game property results and computation times (in seconds) for properties 7 to 8.

explicit engine is more suited for exploring large models. Its algorithms may better suit our models.

Table 2.21 shows a slight difference of a factor of 5 in the computation time of property 9a for which PRISM takes 0.551s and Storm takes 0.094s.

The next following tables show properties for the Inhibitory Control game that are not mentioned in subsection 2.6.2. These properties concern the model itself and were not discussed with practitioners.

Property	PRISM		Storm	
9a- R"good_on_target"=? [C<=25]	2.789	0.551	2.789	0.094
9b- R"good_on_decoy"=? [C<=25]	6.000E-1	0.146	6.000E-1	0.092
9c- R"good_on_signal"=? [C<=25]	3.389	0.165	3.389	0.093
9d- R"bad_on_target"=? [C<=25]	1.877	0.153	1.877	0.091
9e- R"bad_on_decoy"=? [C<=25]	1.733	0.148	1.733	0.094
9f- R"bad_on_signal"=? [C<=25]	3.610	0.175	3.610	0.092

TABLE 2.21: Inhibitory Control Game property results and computation times (in seconds) for properties 9

Property	PRISM		Storm	
10- P=? [F (num_act_targ>3) & num_act_targ=5 & prev_targ) & (click_fast | click_slow))]	8.420E-1	0.152	8.420E-1	0.163
11- P=? [F (num_act_targ=5 & prev_targ & (several_click) & !transiting)]	1.190E-1	0.672	1.190E-1	0.183
12- P=? [F (num_act_targ=5 & prev_targ (click_fast | click_slow | anticipate | not_click | double_click) | several_click) & !transiting)]	1.0	0.152	1.0	0.019
13- P=? [F (num_act_deco>3 & num_act_deco=5 & prev_deco & (click_fast | click_slow))]	8.450E-1	0.369	8.450E-1	0.430

TABLE 2.22: Inhibitory Control Game property results and computation times (in seconds) for properties 10 to 13

In table 2.22, the computation of property 12 is slightly faster with Storm (0.019s) than with PRISM (0.152s).

Property	PRISM		Storm	
14a- P=? [F (num_act_deco=5 & prev_deco & (click_fast) & !transiting)]	4.301E-1	0.378	4.301E-1	0.461
14b- P=? [F (num_act_deco=5 & prev_deco & (click_slow) & !transiting)]	2.150E-1	0.385	2.150E-1	0.455
14c- P=? [F (num_act_deco=5 & prev_deco & (anticipate) & !transiting)]	1.708E-1	0.380	1.708E-1	0.474
14d- P=? [F (num_act_deco=5 & prev_deco & (not_click) & !transiting)]	1.123E-1	0.382	1.123E-1	0.444
14e- P=? [F (num_act_deco=5 & prev_deco & (double_click) & !transiting)]	2.150E-2	0.375	2.150E-2	0.472
14f- P=? [F (num_act_deco=5 & prev_deco & (several_click) & !transiting)]	5.018E-2	0.371	5.018E-2	0.437

TABLE 2.23: Inhibitory Control Game property results and computation times (in seconds) for properties 14

Table 2.23 does not show any difference between both model checkers.

Property	PRISM		Storm	
15- P=? [F (num_act_deco=5 & prev_deco & (click_fast | click_slow | anticipate | not_click | double_click | several_click) & !transiting)]	1.0	0.194	1.0	0.099
16a- P=? [F (num_action=5 & ((prev_none & (not_click)) | (prev_targ & (click_fast | click_slow)) | (prev_deco & (not_click))) & !transiting)]	5.400E-1	0.473	5.400E-1	0.122
16b- P=? [F (num_action=5 & ((prev_none & (!not_click)) | (prev_targ & (!click_fast & !click_slow)) | (prev_deco & (!not_click))) & !transiting)]	4.599E-1	0.906	4.599E-1	0.122

TABLE 2.24: Inhibitory Control Game property results and computation times (in seconds) for properties 15 to 16

In table 2.24 shows that properties labeled as 16 have a slightly faster computation time with Storm compared to PRISM.

Conclusion of the comparison

The PRISM model checker engines benefit from several features such as the verification of PCTL* properties and the "run experiments" tool which is pretty convenient to visualize the chronology of the results under the form of diagrams. It is also easy to install and it proposes a visual interface and many tutorials. On the other hand, it can be rather slow to compute some properties with its "explicit" engine and it may even enter an infinite loop thus providing no results at all.

Storm is usually faster than PRISM and is specialized in explicit methods allowing a higher precision. Storm main engine is the sparse engine in the sense that it tends to have the most features. It takes the model description and directly builds a representation based on explicit data structures, mainly bit vectors and sparse matrices. Then, model checking is performed using these data structures. Since these permit easy access to single elements, they are standard representations for many tasks involved in the solution procedure (like solving linear equations). This enables the use of off-the-shelf libraries, for instance Eigen [56] or gmm++ [106], that implement sophisticated solution methods. Among these methods, some are more efficient to explore the state space as they are specifically tailored for acyclic models. It also allows users to write probabilistic properties that use rewards just as any other state variables. But Storm has also some drawbacks. It only recognizes PCTL formulas prohibiting the nesting of state quantifiers (PLTL formulas are not accepted). As for example, a property such as $P =?[XGF]$ is not accepted by Storm. Also, if they want to obtain a diagram out of Storm results, users must create their own scripts.

2.8 Conclusion

Complex human activity recognition remains a challenging research area [76], especially to build effective recognition systems. We propose a formal approach based

on discrete-time Markov chains to model such human activities. Important properties of these 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.

As a first step, together with clinicians, we selected four serious games for Alzheimer patients and we encoded three of them as DTMCs in PRISM. To this end, it was necessary to use different approaches as each game had its own specific rules and evaluation methods. Indeed, whereas speed and quality of response are key points for the Code game, it is the nature of the presented item as well as the patient associated response that are the key points of the Recognition game. The Inhibitory Control game mixes both features as it focuses on speed, on the presented item, and on the associated response. For the two last games, it was necessary to model both the game and the player behavior.

We tested a dozen of medically-oriented PCTL* properties per game, thanks to the PRISM model checker. These properties include the use of rewards to quantify the performances of patients. They gave an overview of the modeled patients performance in a fairly short amount of time on a standard computer configuration. PRISM was able to return the probability for the modeled patients to go through specific group of paths representing behaviors considered as pathological as well as mean of response and time scores. These three models and their verification allowed us to validate our approach and to test its scalability for rather different applications.

As a second step, we compared PRISM and Storm on the same set of properties in order to determine which of these well known tools is more suitable for our application. We chose Storm as its model checker accepts several modeling languages as input, including the PRISM one, and as it shows good performances [57]. In this comparison, we used different model-checker engines: the hybrid and explicit engines of PRISM (the hybrid engine for both versions of the Code game model) and the sparse engine of Storm to verify medically oriented properties. Generally speaking, Storm displays the best performance. Our intuition is that, at least for the Code game, since states and paths cannot be easily compacted, the decision diagrams used to build models in the (default) hybrid engine of PRISM cannot be fully exploited. The explicit data structures used to build models by the (default) sparse engine of Storm seem more suitable for our case studies. The other models cannot be explored with the hybrid engine of PRISM and require its explicit engine for the same reason. The computation times are usually less for Storm as its algorithms are tailored for the kind of models we implemented. Nonetheless, the features available in PRISM, such as the computation of PCTL* formulas or the "run experiment", make it hard to replace. In brief, these two probabilistic model checkers do not present the same flows or advantages. For our work, they are complementary. When computing PCTL formulas on rather big models, Storm is the most appropriate tool. In the case of larger PCTL* properties with nested operators, PRISM is the one to go for.

Finally, it is essential to test our models in a real clinical experimentation to adjust the models presented in this chapter (model calibration, Step 6 of Figure 2.2), especially to fix the probabilities and to obtain realistic models. To this end, we carried out a medical protocol that ran from Sept 2020 to Sept 2021. This protocol and its results are the purpose of the next chapter.

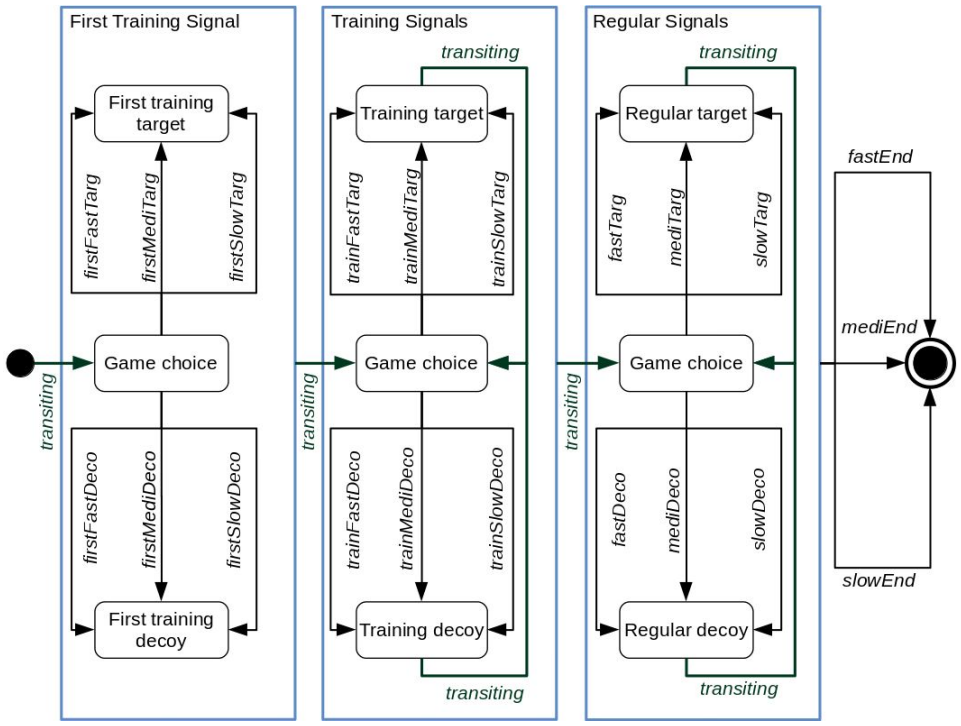


FIGURE 2.17: Simplified automaton of the Inhibitory Control game.

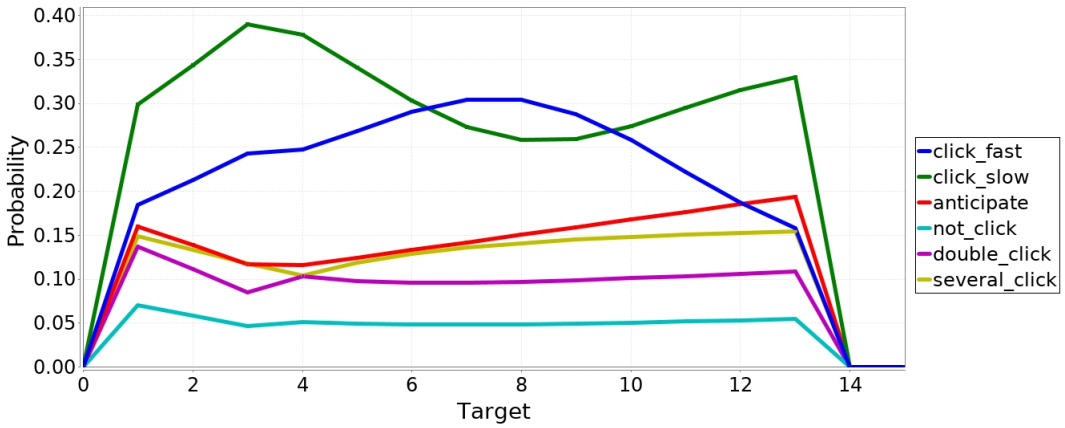


FIGURE 2.18: Probability to perform an action for a specific target.

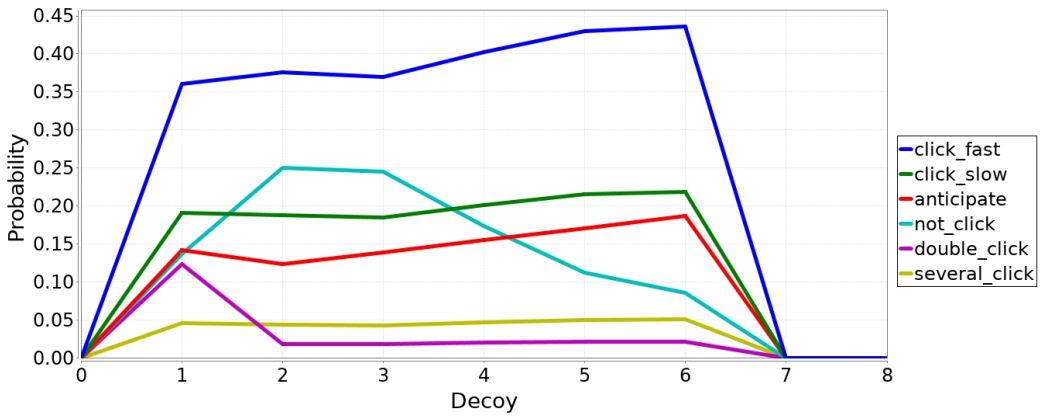


FIGURE 2.19: Probability to perform an action for a specific decoy.

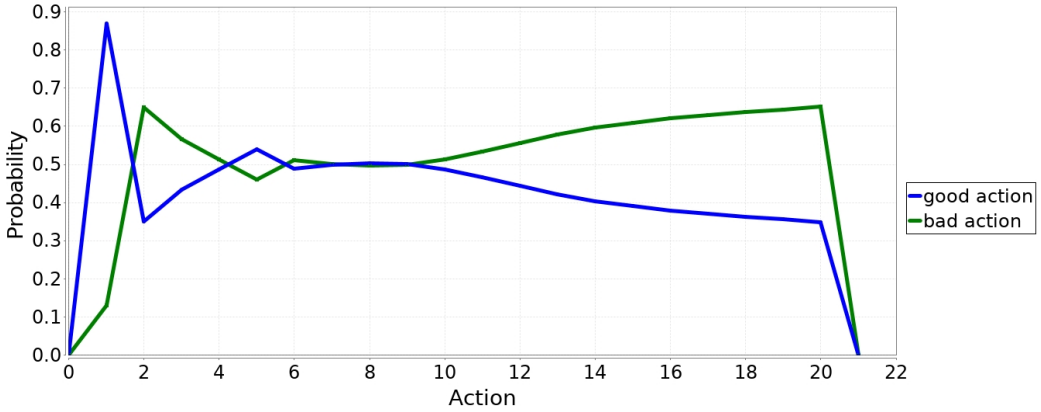


FIGURE 2.20: Probability to perform a good or a bad action for each instant when an action is expected from the patient. Here, the action number 1 on the horizontal axis is the one recorded before the occurrence of the first signal.

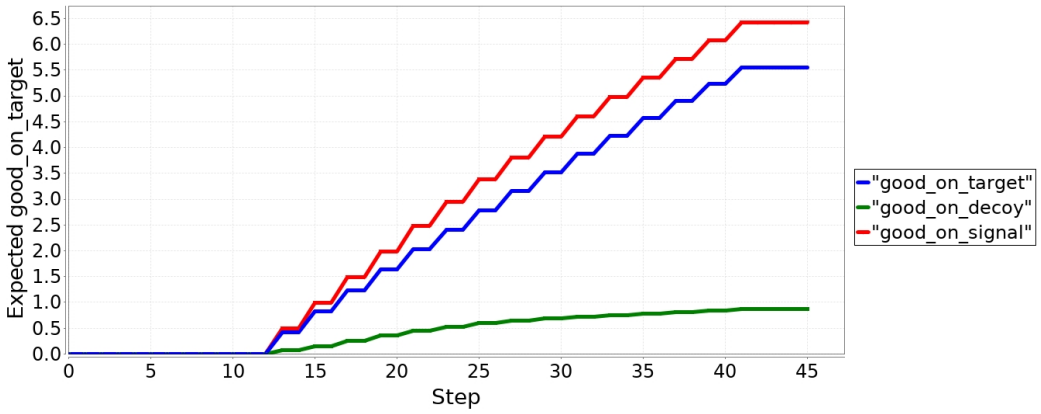


FIGURE 2.21: Average model checking results for rewards related to good actions.

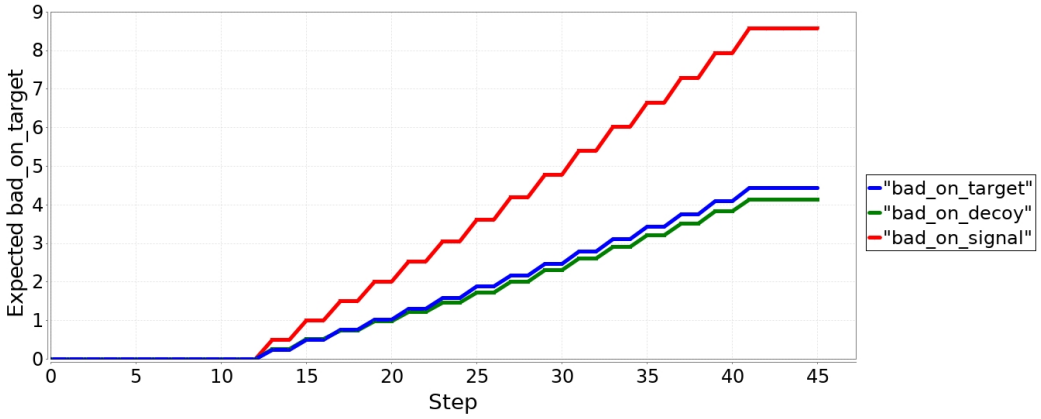


FIGURE 2.22: Average model checking results for rewards related to bad actions.

Chapter 3

Clinical Protocol and Experimental Results

In order to validate our models in a clinical experimentation, we proposed a clinical protocol involving patients coming to the Memory Centre (CMRR) of the Centre Hospitalier Universitaire (CHU) of Nice at the Institut Claude Pompidou (ICP). In this protocol, both mild neurocognitive disorder (mild NCD) patients and subjective cognitive decline (SCD) ones played the three games previously presented as well as a fourth one, the Tapiscine (targeting motivation disorders). This clinical protocol allowed us to validate these games as suitable tools to discriminate between mild NCD patients and SCD ones. This experiment led to calibrate two game models.

3.1 Clinical Experimentation Protocol

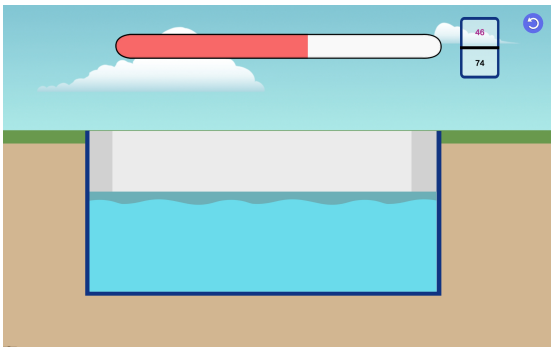
The objective of the protocol is to collect real data to validate the hypothesis that models can differentiate mild NCD patients apart from control subjects, based on their behavior and performance. To verify the acceptance of this hypothesis, the predictions of the models should be consistent with experimental results. The main criterion for assessing our approach is thus to verify the consistency of the observed behaviors (score, action time/delay, gestures, choices) with the diagnoses obtained by classical validated cognitive and behavioral tests.

Using serious games in such tests is relevant because it is known that observing people when they perform simple activities provide interesting clues on their cognitive status. For instance, a previous work in our laboratory showed that the observation of patient carrying out daily activities revealed significant differences according to their stage of severity of Alzheimer's disease [80]. Moreover, as already mentioned, each of the selected games assesses a different cognitive and behavioral dimension: selective and sustained attention, episodic memory capacity, inhibitory control, or sensitivity to reward. Thus, they constitute a quite extensive panel of tests.

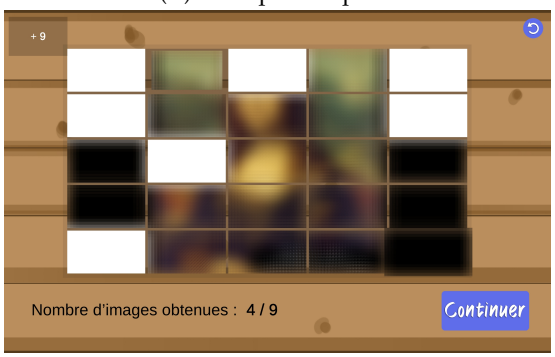
3.1.1 A Fourth Game: Tapiscine

At the beginning of the PhD project, the Tapiscine game was only a prototype and clinicians had no experience with it, thus it was not possible to create a model for it. Since it has been included in the protocol, we present its rules here.

This game (fig. 3.1) targets sensitivity to reward, involved in motivational processes and is installed on Android tablets. The goal is to "fill" different (virtual) pools (by tapping the screen of the tablet) in exchange for rewards. Each tap corresponds to a quantity of water poured into the pool.



(A) Example of a pool.



(B) Example of a reward jigsaw.

FIGURE 3.1: Display of the Tapiscine game.

The game session starts with a calibration phase to determine the maximum pool size for a given player. This maximum size corresponds to the number of times (N) that the player is able to tap on the screen in 10 seconds. Then, the game deduces four possible pool sizes: small corresponding to N/4 taps, medium-small to N/2, medium to ¾N, and large to the maximum N). At each round, the game proposes a pool size and a number of reward points to the player; these parameters are chosen randomly and independently. The size is drawn among the 4 previously defined ones and the number of associated rewards among 3, 6, or 9 points. There are thus 12 configurations and the system proposes each configuration exactly 3 times. For each configuration, the player can choose either to rest for ten seconds and win only 1 reward point or to try to fill the pool in 10 seconds (or less), to win the corresponding reward points. A patient who does not choose after 30 seconds or who decides to play but cannot fill the proposed pool does not win any reward.

On the psychological side, this game requires the patient to take a strategic decision: resting or filling the pool. For instance, it is more interesting to fill a small pool with a high reward than to struggle to fill a large pool associated with a minimal reward. The game ends when all the possible configurations of sizes and numbers of rewards have been proposed three times.

A reward point corresponds to one piece of a jigsaw representing, e.g., a well-known classical art painting. At the end of each round, the game displays a blurred view of the current jigsaw puzzle with the pieces won so far, as well as the number of jigsaws completed since the beginning of the session (fig. 3.1b). This partial display works as an incentive to motivate the patient to continue playing and to complete the picture. Each time the player gathers 25 pieces, the complete picture of the current jigsaw is unveiled and a new jigsaw is chosen. At the end of the session, the game congratulates the patient and displays all the jigsaw pictures gathered during the session.

3.1.2 Protocol Context

Unlike other approaches [120], we chose two-dimensional games in this protocol. Such games are easier to produce and distribute than 3D ones. Moreover, they do not require specialized hardware and they are available on touch pad or tablets. As shown in a previous experiment, using an apathy assessment tool in older adults with and without minor and major neurocognitive disorders [128], participants have no problems with such devices.

The rest of this subsection details the constraints that had to be taken into account in the design of the protocol.

Neurocognitive Tests used in the Protocol

As many cognitive and behavioral tests exist, it was necessary to select a set that mainly focuses on the cognitive functions previously selected with clinicians. The first constraint was to find cognitive tests used daily in the ICP institute to avoid going through additional tests during the game sessions. The second constraint was to find cognitive tests with a good sensitivity yet fast enough (as the protocol is not devoted to complete the patient clinical records). Thus the selected neurocognitive tests are the following.

- The "Mini Mental State Examination" (MMSE) which serves as a first screening for cognitive deficits [100]. This test groups several questions and problems related to the orientation in space and time, memorization of words, calculation, attention, comprehension, oral expression, and praxis. It is a rather general test but it suffers from a low sensitivity to mild neurocognitive disorder [24]. Clinicians can't rely on this test alone to make an appropriate diagnosis [7] but it is useful to assess the degree of the global cognitive impairment.

- The "Five Word Test" [43] and the "Digit Span" [74] from "Wechsler Adult Intelligence Scale-III" [109] target the working memory. The Five Word Test consists in four phases: (i) memorization of a list of verbal words; (ii) immediate restitution to check the encoding of the memory; (iii) interference with an other task to draw the patient's attention away; (iv) delayed restitution (with or without any hint) to evaluate the memory. The Digit Span consists in the memorization of a list of numbers given in a certain order by the practitioner and in its restitution in the same order (short term memory) and in the reversed order (working memory).

- The "Digit Symbol Substitution" (DSS) is among the oldest and most established neurocognitive tests [124]. It is known to measure visual processing speed [125] and attention [83]. The test is relatively simple to administer as it only requires the printed statement of the test (see figure 3.2), a pen, and a timer. The printed test displays a translation table (in which each digit is associated with a symbol) and a table of digits associated with empty cells. The goal for the patients is to fill in as many blank cells as they can with the corresponding symbols following an imposed order within two minutes. This test is simple and very sensitive to impairments but, as a standalone tool, it does not give any indication on the origin of the impairment [67].

- The "Frontal Assessment Battery at bedside" (FAB) is a "cognitive and behavioral battery to assess frontal lobe functions" [44]. It evaluates through different tasks several cognitive functions associated with the brain frontal lobes such

as inhibitory control or mental flexibility. We mainly focus on the inhibitory control evaluation through a "go/no go" task but other tasks focus on different functions such as the fine motor and sequencing motor skills for the FAB Luria's task (used in section 3.2.5).

- The "Apathy Inventory" is an interview of a patient by a practitioner. The goal is to evaluate three "clinical dimensions of apathy: emotional numbing, loss of initiative, and loss of interest" [25].

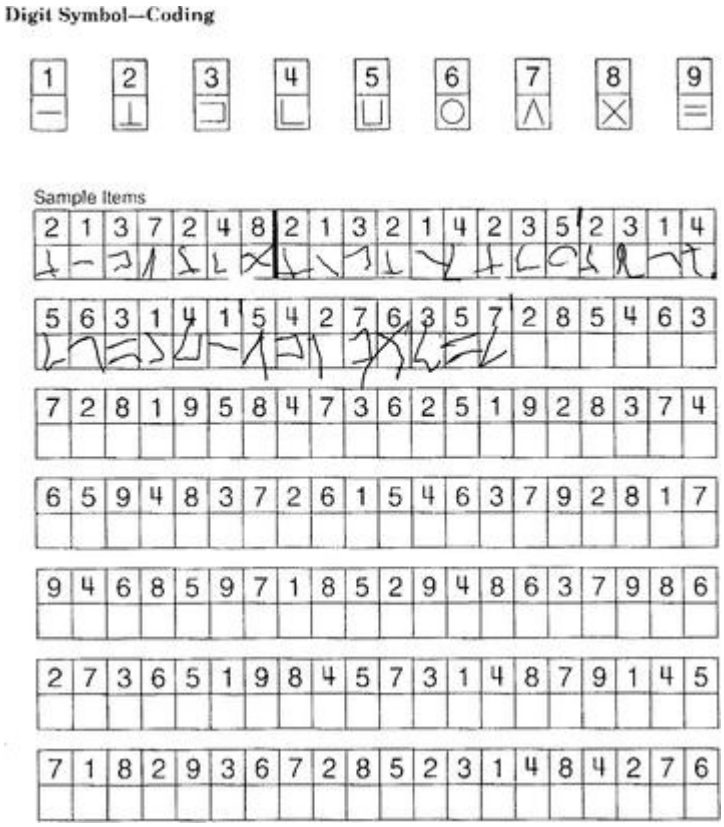


FIGURE 3.2: Example of Digit Symbol Substitution (DSS) paper test.

Participants

The protocol targets senior people aged 60 years and over [48] with diagnosed mild NCD (according to the DSM-5) and people with Subjective Cognitive Decline (SCD) [71]. This raised new constraints: indeed, as the protocol started in the middle of the PhD project, it was impossible to recruit enough volunteers for a large scale statistical study. Thus it has been decided that this protocol would take the form of a feasibility study. We decided to recruit 50 volunteers including 30 mild NCD subjects and 20 SCD ones. This choice was based on previous studies using the same type of materials, for example, in [117], 30 participants with a MMSE score more than or equal to 16 tested a game support system and 47 older subjects with major and minor cognitive impairment tested the Code game; similarly, for the pilot study of an exergame by [16], 18 subjects were selected.

3.1.3 Protocol Provisional Planning

We expected this protocol to last 13 months, with 10 months of inclusion of patients and 3 months of result analysis. The detailed program is shown in figure 3.3).

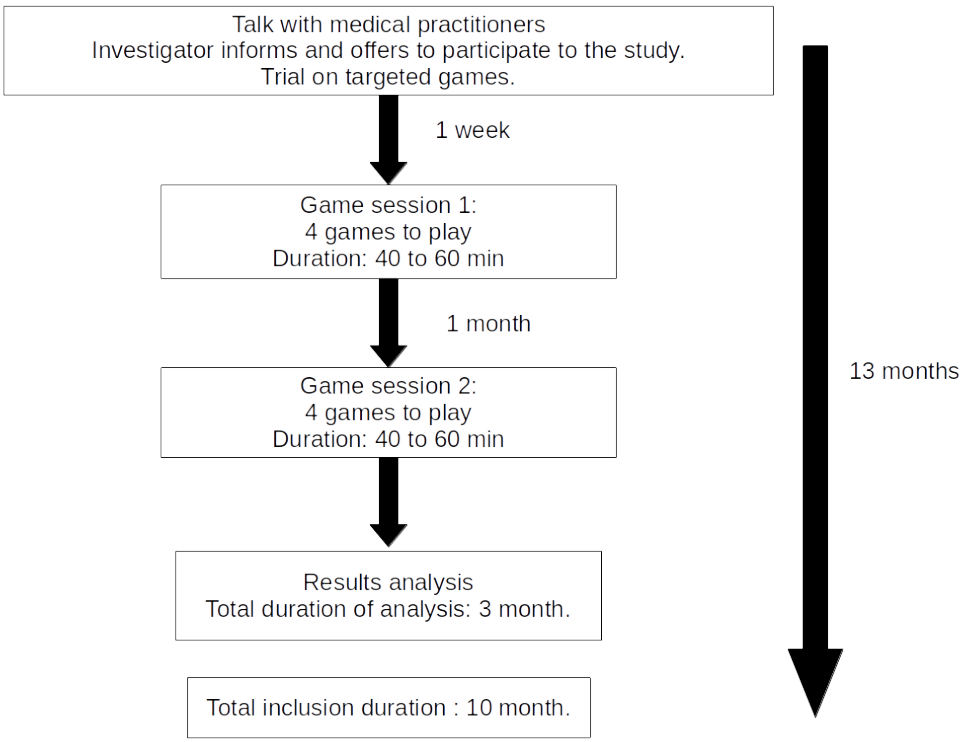


FIGURE 3.3: Experimental protocol flowchart.

Each participant has to be seen three times:

- First, during a regular clinical consultation at the Memory Center, in order to present the protocol and, if necessary, to complete the clinical records with the cognitive and behavioral tests previously described.
- Second and third, during two game sessions, one month apart, to play the four serious games. Each game session should last 30 to 45 minutes. The second session is similar to the first one, its objective is to minimize the impact of external events in the patient lives on their measured performance.

During the game sessions, an investigator is expected to stay with the volunteers to explain the rules of the games during a training phase, to assist the participants if necessary, and also to note particular events that may occur during the session (for example, if a participant asks again about the rules or leaves the game).

Game Session Planning

All volunteers had to play all games during each session. If a patient could not finish a session (for any reason, e.g., time constraints or tiredness) he/she could come back again to finish it. Each game proposes at least one training phase before playing the actual game. The order in which the games are played by the volunteers was randomized except for the Code game that is played three times, once after every other game. For example, the order of the games in a session could be: (i) Tapiscine;

(ii) Code game; (iii) Recognition game; (iv) Code game; (v) Inhibitory Control game; (vi) Code game. The Tapiscine game has an integrated step by step tutorial proposed at the beginning of the game. The Recognition game is composed of four levels, the first one being a tutorial with 10 pictures before the three other levels of 40 pictures with randomized themes. The Inhibitory Control game contains two levels, one tutorial of 10 targets and 5 decoys and the actual level of 30 targets and 10 decoys. The Code game also has two levels: one tutorial without timer but with visual and audible indications on the validity of the answers and one actual level with a two minutes timer and without visual indications. A volunteer who did not understand the game on the first try of a training level receives more explanations and is entitled to another try.

All the data (including scores, answers, and response times) as well as video recordings (focused only on the hand gestures above the screen, for privacy) are to be collected and anonymized.

As the necessary delay to obtain the ethical acceptance of a protocol is rather long, the protocol was designed in parallel with the models described in chapter 2.

Ethical Validation

Under French Law each medical protocol must undergo a preliminary ethical validation. The goal is to ensure that patient rights and safety are respected; it is all the more important as the proposed protocol targets senior populations. The protocol proposal was submitted to two committees. First, the CERNI (Comité d'Éthique sur les Recherches Non Interventionnelles) required some clarifications and redirected us to a second committee: the CPP (Committee for the protection of people). After a complete makeover and additional documents, the final version was submitted on April 29th, 2020, and accepted on May 14th, 2020. From this point on, we could carry out preliminary tests to ensure that the equipment was ready to perform the real protocol. Volunteer recruitment started on September 29, 2020 after receiving the CHU authorization.

3.1.4 Procedure

The inclusion phase was performed at the ICP with the help of on-site staff. For each session, there was one investigator (mainly this thesis candidate). For some sessions a clinician was also required to complete the patient clinical record. The investigator was present at ICP 2 to 4 days a week to meet potential volunteers. A typical day of inclusion was as follows:

- ICP staff provides a list of potential volunteers.
- Potential volunteers meet their practitioners for their clinical check up. Practitioners assess the state of the patients and their ability to join the protocol.
- If a patient validates all inclusion criteria, the practitioner presents a protocol overview to him/her.
- If the patient accepts to participate, the investigator meets her to give a detailed presentation of the protocol and to set up an appointment for the first game session.

This recruitment strategy resulted in many refusal (of the order of 40) and some rare patients feigned their interest but could not be contacted again. However, in the

end 51 volunteers were recruited within the planned schedule (10 months of inclusion). Among these volunteers, one abandoned after the first session due to relocation and another stopped in the middle of the first session because of fatigue. This second volunteer could not come back because of her degraded health condition.

The sanitary crisis of Covid-19 mostly impacted the previous phase of design and ethical and administrative validation as communications were more difficult during the first Covid lockdown. The recruitment phase got slightly slowed down at the end of 2020 but did not suffer any long interruption. In fact, the sanitary crisis has rather favored patient inclusion, since many volunteers complained about their canceled activities and were actually happy to participate to this "authorized" one.

The duration of each game session was not constant from a person to another or from a session to another. First, this protocol was grouped with another protocol focusing only on the Tapiscine game. For the sake of this second protocol, a set of surveys was provided to each participant at their first session. These surveys took approximately 15 minutes for the fastest participants. The corresponding data as well as the Tapiscine ones are out of the scope of this thesis and were independently analyzed by the experts who created the game. The slowest participants could take up to 45 minutes to complete their surveys. With this combination, the first session could last from 50 to 120 minutes and the second session could last from 30 to 90 minutes. This amount of time depended mostly on the possible "chatty" nature of the participants but also on the investigator capacity to explain the rules of each game, which has improved over the course of the sessions.

Particular Cases

In some rare cases patients added to the difficulty of the protocol. The smallest annoyance was the volunteers forgetting their appointments. A more troublesome issue was that some patients could have an ambiguous diagnosis (e.g., high scores in neurocognitive tests but diagnosed as mild NCD) that required an analysis of their record by the ICP chief psychiatrist. Another complicated case was patients denying their deficits. This denial was difficult to deal with. It was, for some patients, part of their health condition and is named anosognosia. In fact, one of such patients should not have been included in the protocol and her records have been removed. The most difficult case to handle was a patient who tried to push the investigator to assert that she had no disorders. Finally, the patient accepted to participate to the protocol without any counterpart.

3.2 Result Analysis

This section analyzes the data gathered during the protocol to assess the validity of the selected games in discriminating mild NCD and SCD patients. This analysis lasted approximately six weeks. The section first details the analysis of the recruited population before the analysis of each game separately. The final part gives an overview and an overall analysis. The gathered data have been formatted and filtered using Python3 scripts. All statistical tests were performed using SPSS version 27 [65]. In this manuscript, only the data of the three games modeled in chapter 2 are analyzed. The Tapiscine data have been processed by statisticians and psychologists of CoBTeK.

3.2.1 Statistical Hypothesis Tests

When performing in vivo or in vitro experiments, one uses statistical tools to analyze data and to determine if the observed differences result from the tested parameters or from randomness. As they are used to verify hypotheses, these tests are named "Statistical hypothesis tests". The hypotheses are of two natures:

- the *null hypothesis*, denoted *H0*, which varies according to the test;
- the *alternative hypothesis*, denoted *H1*.

The presented work aims to determine characteristics of two populations out of the data analysis of two samples. This process, named inferential statistics, defines H0 as the hypothesis that there are no differences between both samples. Thus in the following subsections:

- H0 writes "both samples are from a same population";
- H1 writes "both samples are from different populations".

As participants who are included in the research studies represent only a small part of the population of interest, the results of statistical tests allow to refuse H0 only with a certain probability of error. The probability of risks of errors are between 0 and 1 and are of different nature:

- α: the risk to reject H0 when in fact it should not be rejected;
- β: the risk not to reject H0 when in fact it should be rejected.

The value of risk α is set by the experimenter. In psychology as well as in biology, this value is usually set to 0.05 as the observation of behaviors or living matters implies much variability. Risk β depends on two factors: the chosen test and the size of the sample. Usually, experimenters try to find an appropriate sample such that β is less than 0.20.

To help in the decision to reject H0 or not, most tests give a probability named *p-value* (probability value). In an experiment, this value is interpreted as the probability to observe differences (even larger than experimentally observed) between both samples, even if the H0 hypothesis is not rejected. In psychology and biology, this value has to be inferior to a threshold set to 0.05 for H0 to be rejected. This criterion of rejection has been broadly used but is at the center of a debate [17]. Indeed, the p-value does not measure the probability for H0 to be true but only the probability to observe the tested data given a true H0. Hence, a p-value threshold of 0.05 does not make experiment results rare enough when H0 holds, leading to poorly reproducible results. A better solution (yet not the best) is to qualify results with a p-value under 0.005 as "statistically significant" and those between 0.005 and 0.05 as “suggestive”.

To determine this p-value, many tests compare the obtained statistic (i.e., the result of the test computation) to a model of distribution. This model of distribution often requires, as a parameter, the *degree of freedom* (df). The degree of freedom can be described as follows: given a data set of observed variables for which we know the means, the degree of freedom corresponds to "the number of observations minus the number of necessary relations among these observations"[1] [123]. For example, in

[1] Necessary number of observations in the sample so that the statistical law (applied to the chosen test) is relevant.

the context of the test of Student [54] on a data set of one variable observed 50 times, the formula would be $df = 50 - 1$. Once the model is chosen and adjusted with the necessary parameter, the p-value depends on the location of the test statistic in the modeled distribution: e.g., if the value of the test statistic is located in 5% of the distribution, then the p-value is $p = 0.05$.

The rest of this subsection introduces the various tests [54] used in our study.

Parametric Tests

Parametric tests are a class of powerful tests based on the assumption that the data distribution follows a given statistical model (e.g., a normal or a Poisson distribution). As the distribution is known, this category of tests uses the parameters of the matching model (e.g., the mean and standard deviation in the case of a normal distribution) to compute their results, it is why they are qualified as parametric. Nevertheless, it is required to first use non-parametric tests (detailed further) to determine the likelihood of the assumption on data distribution.

In this work, the data do not match such assumption on data distribution. But, as there are no non-parametric alternatives, the SPSS software "general linear model" feature was used to check the effect of several parameters upon different variables. Using this feature, we focus on two statistical tests in the context of a ANalysis Of VAriance (or ANOVA): the λ of Wilk and the F-test.

- An analysis of variance (ANOVA) is a mathematical method to compare variations in a given data set. One of the main principles of the ANOVA is the comparison of the two components of the total sum of squares, the "explained" sum of squares and the "residual" sum of squares. This total sum of squares is written as follows:

$$\sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n}(\bar{x}_i - \bar{x})^2 + \sum_{i=1}^{n}(x_i - \bar{x}_i)^2$$

  where n is the number of observations, $\bar{x}$ the total mean of variable x, and $\bar{x}_i$ the mean of variable x in the i^{th} sample. The first term of this equation is the total sum of squares, the second is the explained sum of squares (as it reveals how each sample differs from the total mean) and the final term is the residual sum of squares (as it shows how much observations differ from the mean of their sample). Once these values are known, one can evaluate the ratio of the explained sum of squares over the residual sum of squares using a F-test.

  The data analysis in this work uses an extension of the classical ANOVA which aims to analyze the effects of multiple parameters on a given variable. More precisely, the process is called "two-way mixed design measures ANOVA".

- The Mauchly's sphericity test is used to verify one of the ANOVA's prerequisites, i.e., the sphericity of data. Sphericity is met if the variance of differences between all pairs of related groups are equal. Hence, in Mauchly's test, the H0 hypothesis states that sphericity is respected. To verify this hypothesis, Mauchly's test computes a value denoted W that evaluates this similarity of the variances. Once W is known, a χ^2 test determines whether it shows a similarity between each variance. However, this test is highly sensitive to both the size and the distribution of data and it can accept the hypothesis H0 when in fact it should reject it. As the data in this work do not follow a normal distribution, we will assume that H0 should be rejected. To deal with data that

violate the sphericity assumption, it is necessary to apply a "correction" of the degree of freedom used in the ANOVA final F-test. This correction induces an increase of the p-value (compared to a F-test with a non-corrected degree of freedom). This increase makes the test "stricter" to compensate the sphericity assumption violation. The choice of the correction to apply relies on a value denoted ϵ that is computed by these correction methods. If $\epsilon > 0.75$, as it is the case in all ANOVA tests performed in this work, it is recommended to use the correction of Huynh-Feldt.

- The F-test actually includes many generic parametric tests that are based on the assumption that under the H0 hypothesis, the test statistic (result of the test) follows a Fisher–Snedecor distribution. The statistic F of this test is the ratio "variation between sample means" over "variation within the samples" which, in an ANOVA, corresponds to:

$$F = \frac{\frac{1}{df_1}\sum_{i=1}^{n}(\bar{x}_i - \bar{x})^2}{\frac{1}{df_2}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2}$$

  Where df_1 is the degree of freedom associated with the the explained sum of squares and df_2 the one associated with the residual sum of squares. The closer F is to 1 the more H0 tends to be acceptable. Then this value is confronted to the Fisher–Snedecor distribution with df_1 and df_2 as parameters to determine the p-value.

- The λ of Wilk is a multivariate test that extends the F-test. In the context of an ANOVA, it tests whether the different means of a repeated variable vary with the combinations of other dependent and independent variables. This test returns a value named λ that ranges from 0.0 to 1.0 and that estimates the strength of the relationship between independent and dependent variables. The closer the value is to 0.0, the more the independent variables may contribute to the observed variations of the dependent variables. To determine the p-value associated with a given λ, SPSS uses a F-test. For example, in the Code game study section 3.2.3, this test is used to verify if the means of the amount of answers of both groups vary with the number of times the subjects played the game and with this number of times combined with the type of the subject group.

Non-parametric Tests

Contrarily to parametric tests, non-parametric ones are not based on statistical distributions. Thus, they do not require the satisfaction of the same prerequisites. These tests are used in many situations and they are known to be more robust but they may lead to the inadequate acceptance of H0 and thus are less powerful. The non parametric tests used in this work are detailed below. Most of these tests use "critical" value tables precomputed by the authors of the tests.

- Normality tests verify whether a data set follows a normal distribution or not. In these protocol, we used the test of Shapiro-Wilk. In this test, H0 states that a given sample comes from a population that follows a normal distribution. This test is particularly powerful for small samples ($n < 50$) which is our case.

The test statistic, denoted W, is the squared correlation coefficient between the generated normal distribution and the actual data. The bigger W the more the data are compatible with a normal distribution. The p-value is determined by comparing W to W_{crit} found in the Shapiro-Wilk table knowing the chosen α and the size of the sample.

- The Wilcoxon signed ranks test checks for the differences between two paired samples using its own ranking method. For this test, H0 states that there are no differences between the paired data. The statistic returned by this test is denoted Z and results from the comparison of the rank mean of one of the paired sample with the overall rank mean. The closer it is to 0 the fewer the differences between both groups. To determine the significance of the test, Z must also be compared to the associated Z_{crit}. For example, in the demographic study section 3.2.2, this test is used to verify that repeating the game session did not impact the subject performance. In other words, this test checks that each subject (paired samples) had consistent results in both sessions.

- The Mann-Whitney U test that derives from the Wilcoxon one, compares two unpaired samples and determines if they come from the same population. To do so, it ranks the observed values of a variable. In this test, H0 states that both samples are from the same population. The test statistic is denoted U and is the smaller of U_1 and U_2 defined below:

$$U_i = n_i n_j + \frac{n_i(n_i+1)}{2} - R_i$$

where n_i and n_j are the sizes of samples i and j and R_i the sum of the ranks for sample i. Thus, U ranges from 0 to $n_1 \times n_2$ where 0 means that there is a big difference of ranks between both groups (rejection of H0). To determine whether H0 must be rejected, the obtained U must be confronted to the corresponding U_{crit} in the table of critical values of U. This table takes as entries the chosen α and the sample sizes to return U_{crit}. For example, in the demographic study section below, this test is used to verify that distribution of ages is similar in both groups (SCD and mild NCD). In other words, this test checks that both groups, made of different subjects (unpaired samples), are samples of the same population of age.

- The Friedman test is an extension of the Wilcoxon signed ranks test. It is an alternative to the repeated measures ANOVA for data that do not validate the normality assumptions. It is used to detect differences in the distributions of three (or more) paired groups. This method based on ranks leads to the computation of statistic noted Q. If the size of the data allows it ($numberofgroupsk > 4$ or $numberofrowsn > 15$), Q can be approximated by a χ^2 distribution. In this case, the p-value is given by $P(\chi^2_{k-1} \geq Q)$. For example, in the Code game result analysis, the Friedman test was used to determine whether the repetition of the game impacts the patients results. In this case the number of rows, n, corresponds to the number of patients and the number of groups, k, corresponds to the number of game iterations. In other words, this test checks that repeating the game does (or does not) improve (or decrease) the patients performance.

- The χ^2 of Pearson test checks for the independence of two variables. For this test, H0 states that the two variables are independent. The statistic returned by

this test is denoted $\chi^2_{computed}$ and displays the difference between the observed frequencies and the theoretical frequencies (or expected frequencies if H0 is true). The formula is the following:

$$\chi^2_{computed} = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_i j}$$

where O_{ij} is the observed frequency of event i in sample j and E_{ij} is the theoretical frequency for the same event. To deduce the p-value, the $\chi^2_{computed}$ is compared to the χ^2 distribution for a chosen α and the degree of freedom associated with $\chi^2_{computed}$. For example, in the demographic study section, this test is used to verify that the distribution of women and men is similar in both groups. In other words, this test checks that the variables *group* and *sex* are independent.

- The Kruskal-Wallis H test, also called one-way ANOVA on ranks, checks if samples come from the same distribution. This test ranks the observed values of a variable. In this test, H0 states that each sample has the same mean of ranks for a given variable. The statistic of this test is denoted H and is computed as follows:

$$H = [\frac{12}{n(n+1)} \sum_{j=1}^{c} \frac{T_j^2}{n_j}] - 3(n+1)$$

where n is the sum of all sample sizes, c the number of samples, T_j the sum of ranks for sample j, and n_j the size of sample j. If H0 is true, this score follows a χ^2 distribution and should be compared to the critical χ^2 determined with the degree of freedom and α. For example, in the demographic study section below, this test is used to verify that the order in which the games were played did not impact the results. In other words, the test checks whether the samples (the three different positions in a game session), for a given variable (one of the game scores), come from the same distribution.

- The ρ of Spearman test checks for the existence of a correlation between two variables. For this test, H0 states that there is no correlation between the variables. The statistic generated by this test is denoted ρ and it evaluates the strength of the association of two variables. It ranges from -1 to 1, with 0 meaning the absence of correlation in the given data. In the SPSS software, the p-value associated with the correlation coefficient ρ is computed using an approximation that utilizes a t-distribution. For example, in the Code game study section 3.2.3, this test is used to verify if any correlation exists between the results of the Code game and the results of neurocognitive tests such as the DSS.

3.2.2 Demographic Study

The data gathered on the patients were:

- age, sex, laterality, education level;
- diagnosis, as described in the French Alzheimer Bank or BNA [82] (stage, syndromic diagnosis and etiological diagnosis);

- neurocognitive test results, including MMSE, detailed FAB, Five Words, Digit Span, and DSS.

The group of mild NCD patients and the SCD patients one are respectively composed of 31 and 20 subjects which is consistent with the provisional plan.

Results

The first test used on all data (demographic and game related) was the Shapiro-Wilk normality test. This test revealed that a large majority of the data does not follow a normal distribution. Thus most of the tests used in this section are non-parametric.

To allow a comparison of the game results of the two diagnosis groups (SCD and mild NCD), the distribution of ages and sexes should be similar in both. Hence, the first focus of this study is to verify this assumption.

First, the Mann-Whitney U test was used to check that both groups are composed of seniors in the same range of age. The results of this test are displayed in table 3.1 as well as results of the Mann-Whitney U test on the distribution of MMSE and FAB scores. To better visualize their significant levels, p-values are qualified by different numbers of stars: $0.05 > p\text{-}value > 0.005$ by "*", $0.005 > p\text{-}value > 0.0005$ by **, and $0.0005 > p\text{-}value$ by ***.

	mild NCD (n=31)	SCD (n=20)	U value	p-value
Age (y), mean±Std Err	74.870 ± 1.201	71.850 ± 1.576	245.00	0.209
MMSE, mean±Std Err	26.580 ± 0.453	29.500 ± 0.246	66.50	$< 0.001^{***}$
FAB, mean±Std Err	15.677 ± 0.283	17.500 ± 0.184	98.00	$< 0.001^{***}$

TABLE 3.1: Mann-Whitney U test on the distribution of ages, MMSE and FAB results within SCD and mild NCD groups.

The p-value associated with the age variable is above the standard threshold of 0.05. Thus this result does not allow to reject the H0 hypothesis that both groups belong to the same population. This distribution of ages is illustrated in figure 3.4 showing the associated box plots for both groups. On the other hand, performing this test on the scores of MMSE and FAB allowed to reject the hypothesis H0 as the p-value is under the threshold of 0.005. This first table suggests that both groups have a similar age distribution but show differences in terms of neurocognitive test performance; these results were expected.

Second, the χ^2 test on a pivot table (table 3.2) compares the proportion of women in both groups. For this test, the null hypothesis suggests that the distribution of women is independent from the group.

Sex	SCD	Mild NCD
Woman	16	22
Man	4	9

	Value	df	p-value
χ^2	0.522	1	0.470

TABLE 3.2: Pivot table of the proportions of women and men in SCD and mild NCD groups and χ^2 test results.

Once again, the p-value is above the standard threshold of 0.05. Thus the null hypothesis cannot be rejected which allows one to think that the distribution of women is similar in both groups. This distribution is shown in figure 3.5.

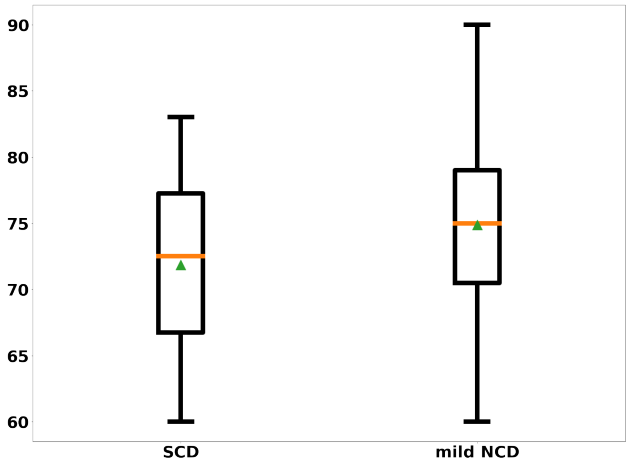


FIGURE 3.4: Boxplot of the distribution of ages within the SCD and the mild NCD groups.

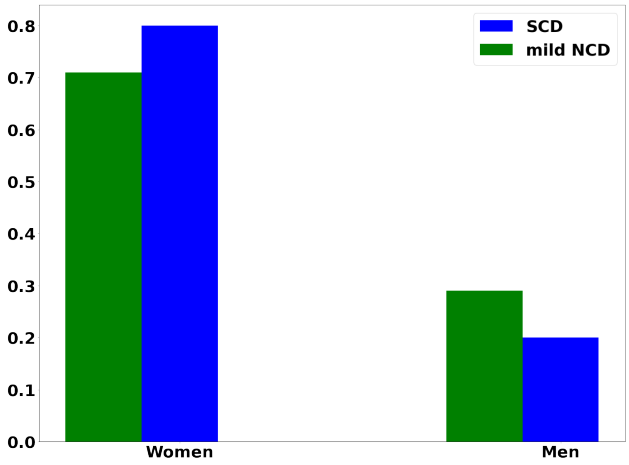


FIGURE 3.5: Women/Men distribution in SCD and mild NCD groups.

These parameters are important as they define the protocol groups. Indeed, a difference in these distributions could be considered as the reason of the neurocognitive test result differences. The selected groups do not show such discrepancy and allow a comparison of the game results.

Third, the study of the mild NCD group etiology can give hints on expected behavioral patterns. For instance, one expects an Alzheimer patient to suffer from memory losses and a vascular cognitive impaired patient from memory losses and/or praxis problems.

As shown in figure 3.6, three main groups can be distinguished:

- a group of patients diagnosed with confirmed or probable Alzheimer disease;
- a group of patients diagnosed with vascular cognitive impairment (caused by vascular lesions in the brain);
- a group of patients with psycho-affective problems such as anxiety or depression.

A fourth group, named "Other", mostly includes not fully diagnosed patients.

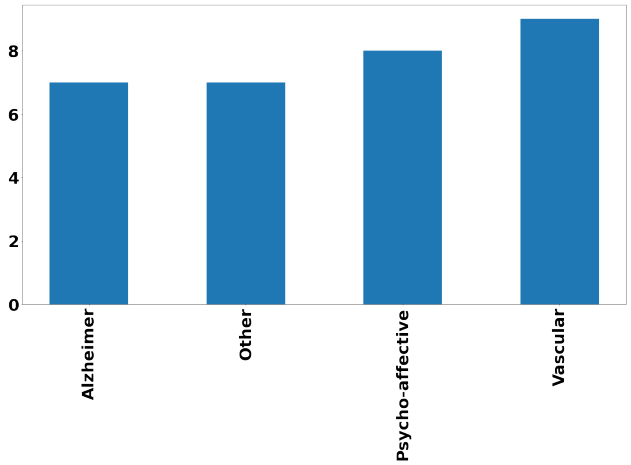


FIGURE 3.6: Major groups of etiologies in the mild NCD group.

Fourth, before analyzing each game individually, the Kruskal-Wallis test verifies whether the random order in which the game are played has an effect on the player performance. Overall, there are three possible orders. We want to check that this order does not influence the results of the players. To do so, game scores were grouped according to their position in the game session. The null hypothesis of the Kruskal-Wallis test is that there are no differences between the game results whatever the position is. An exert of the results are displayed in table 3.3.

	1^{st}session		2^{nd}session	
	Inhibitory Control game		Recognition game 3^{rd}level	
	Valid go	Invalid no-go	Correct folder	Correct trash
Kruskal-Wallis H	0.306	3.732	0.079	5.093
df	2			
p-value	0.858	0.155	0.961	0.078

TABLE 3.3: Examples of results obtained for the Kruskal-Wallis test.

Most of the results of this test indicate that H0 cannot be rejected and that the order in which the games were played did not seem to interfere with the subject performance. Rare cases of p-values under 0.05 have been observed for some Recognition game variables but they are not consistent from a session to another.

Fifth and finally, the Wilcoxon signed ranks test checked whether both sessions of a same patient have consistent results. Tables 3.4 show the results for the variables which had a p-value under 0.05.

As seen in these tables, only a few variables of each game showed a p-value under 0.05. Thus, for the majority of the data, H0 can't be rejected and one can assume that subjects had similar results in both sessions with the exceptions shown in table 3.4. However, to go further in the analysis, this test must also be executed separately on the mild NCD and SCD groups. Tables 3.5 and 3.6 show the results for the variables which had a p-value under 0.05.

Tables 3.5 and 3.6 show results similar to tables 3.4. Thus, the assumption that subjects had similar results in both sessions still hold with exception of the first and second iterations of the Code game.

	Inhibitory Control game	Recognition game
	Invalid no-go	Correct folder 1st level
Mean 1st session	1.95	31.714
Mean 2nd session	2.84	30.795
Std Error 1st session	0.298	0.341
Std Error 2nd session	0.373	0.558
Sample size	37	49
Z	-2.059	-2.075
p-value	0.039^{*}	0.038^{*}

	Code game					
	Answer 1st Iter.	Answer 2nd Iter.	Right 1st Iter.	Right 2nd Iter.	Time 1st Iter.	Time 2nd Iter.
Aver. 1st Sess.	102.270	108.92	101.104	108.00	1199.3569	1123.060
Aver. 2nd Sess.	106.48	112.96	105.71	111.88	1152.489	1083.043
Std Error 1st Sess.	2.350	2.345	2.390	2.384	32.886	30.069
Std Error 2nd Sess.	2.477	2.470	2.530	2.529	32.494	27.176
Sample size	48	49	48	49	48	49
Z	-3.373	-3.674	-3.525	-3.411	-3.539	-3.397
p-value	0.001^{**}	$< 0.001^{***}$	$< 0.001^{***}$	0.001^{**}	$< 0.001^{***}$	0.001^{**}

TABLE 3.4: Results of the Wilcoxon signed ranks test for selected variables of each game.

	Code game				
	Answer 1st Iter.	Answer 2nd Iter.	Right 1st Iter.	Right 2nd Iter.	Time 1st Iter.
Aver. 1st Sess.	96.566	103.000	95.566	101.800	1274.866
Aver. 2nd Sess.	100.275	106.724	99.344	105.482	1231.275
Std Error 1st Sess.	3.077	3.086	3.091	3.118	46.158
Std Error 2nd Sess.	3.475	3.438	3.550	5.528	47.385
Sample size	28	29	28	29	28
Z	-2.693	-2.859	-2.862	-2.769	-2.824
p-value	0.007^{*}	0.004^{**}	0.004^{**}	0.006^{*}	0.005^{*}

TABLE 3.5: Results of the Wilcoxon signed ranks test on the mild NCD group for selected variables of the only game showing significant results: the Code game.

Discussion

This first analysis leads to the following observations:

	Inhibitory Control game	Recognition game
	Invalid no-go reaction time standard deviation	Correct trash 1st level
Mean 1st session	33.42	4.15
Mean 2nd session	136.987	4.900
Std Error 1st session	5.273.	0.301
Std Error 2nd session	53.138	0.228
Sample size	9	20
Z	-2.521	-2.152
p-value	0.012*	0.031*

	Code game				
	Answer 1st Iter.	Answer 2nd Iter.	Right 1st Iter.	Right 2nd Iter.	Time 1st Iter.
Aver. 1st Sess.	111.100	118.450	109.750	117.950	1081.444
Aver. 2nd Sess.	116.800	122.000	116.250	121.150	1025.483
Std Error 1st Sess.	2.319	2.262	2.564	2.260	21.806
Std Error 2nd Sess.	1.980	2.287	1.981	2.307	17.875
Sample size	20				
Z	-2.130	-2.188	-2.256	-2.038	-2.091
p-value	0.033*	0.029*	0.024*	0.042*	0.037*

TABLE 3.6: Results of the Wilcoxon signed ranks test on the SCD group for selected variables of each game.

- both groups are similar in their age and sex distribution;
- the differences between both groups in their neurocognitive test results are significant;
- the game order in a session does not impact the results of the subjects;
- subjects had similar performances in both sessions.

These observations validate both groups and allow to analyze each game without caring about the game position in the session. The last item legitimates the use of the results of only one of the two sessions or, if need be, the merge of both session results.

3.2.3 Code Game

As a reminder, the Code game [118] asks the player 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 2.3). To comply with the DSS test, the game lasts at most two minutes (contrary to the hypothesis of the model in chapter 2).

The data gathered in this game were:

- series of icons displayed to the player, of answers of the player, of time taken before each answer;
- amount of answers, of right answers, of wrong answers, and time mean.

The following analysis mainly focuses on both sessions except for the correlation test that only focuses on the third iteration of the first session. According to tables 3.4 to 3.6, the third iteration allowed to obtain similar results in both sessions.

Results

First, the Mann-Whitney U test checked that the Code game scores show possible differences between the mild NCD and the SCD groups. As this game is played three times in a session, each iteration was checked. Results are displayed in tables 3.7 for the first session and tables 3.8 for the second session.

	mild NCD (n=30)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	96.5 ± 3.0	111.1 ± 2.3	133.5	0.001**
Right, mean±Std Err	95.5 ± 3.0	109.7 ± 2.5	139.5	0.001**
Wrong, mean±Std Err	1.0 ± 0.2	0.4 ± 1.9	290.5	0.838
Time (ms), mean±Std Err	1274.8 ± 46.1	1081.4 ± 21.8	134.0	0.001**

(A) First iteration of the Code game.

	mild NCD (n=30)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	103.0 ± 3.0	118.4 ± 2.2	129.0	0.001**
Right, mean±Std Err	101.8 ± 3.1	117.9 ± 2.2	125.0	0.001**
Wrong, mean±Std Err	1.2 ± 0.2	0.5 ± 0.1	198.5	0.030*
Time (ms), mean±Std Err	1191.9 ± 43.1	1012.4 ± 19.3	131.0	0.001**

(B) Second iteration of the Code game.

	mild NCD (n=30)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	109.9 ± 3.0	122.8 ± 2.3	159.5	0.005*
Right, mean±Std Err	108.5 ± 3.0	121.8 ± 2.4	163.0	0.007*
Wrong, mean±Std Err	1.4 ± 0.3	1.0 ± 0.2	293.0	0.883
Time (ms), mean±Std Err	1102.8 ± 30.9	977.7 ± 19.6	163.0	0.007*

(C) Third iteration of the Code game.

TABLE 3.7: Mann-Whitney U test results over the Code game scores of the first session.

The results of the test show that all iterations of the Code game display differences between the mild NCD and the SCD groups. Three out of four variables always show a difference between both groups with a p-value under 0.05 and even under 0.005: the number of answers, of right ones, and the mean time taken before each response. In fact these three variables are strongly inter-dependent. Indeed, as each subject has two minutes to get as many right answers as possible, the faster they are, the better. The distribution of right answers during the first iteration of the first session in both groups is shown in figure 3.7.

	mild NCD (n=29)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	100.2 ± 3.4	116.8 ± 1.9	128.0	0.001**
Right, mean±Std Err	99.3 ± 3.5	116.2 ± 1.9	127.5	0.001**
Wrong, mean±Std Err	0.9 ± 0.2	0.5 ± 0.1	241.0	0.270
Time (ms), mean±Std Err	1231.2 ± 47.3	1025.4 ± 17.8	130.0	0.001**

(A) First iteration of the Code game.

	mild NCD (n=29)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	106.7 ± 3.4	122.0 ± 2.2	148.0	0.004**
Right, mean±Std Err	105.4 ± 3.5	121.1 ± 2.3	152.0	0.005*
Wrong, mean±Std Err	1.2 ± 0.2	0.8 ± 0.2	250.5	0.380
Time (ms), mean±Std Err	1151.3 ± 39.4	983.9 ± 19.1	150.0	0.004**

(B) Second iteration of the Code game.

	mild NCD (n=29)	SCD (n=20)	U value	p-value
Answers, mean±Std Err	108.7 ± 3.5	124.7 ± 2.6	120.5	0.001**
Right, mean±Std Err	107.1 ± 3.7	124.2 ± 2.6	116.0	0.001**
Wrong, mean±Std Err	1.6 ± 0.3	0.5 ± 0.1	177.0	0.022*
Time (ms), mean±Std Err	1132.1 ± 40.9	964.3 ± 21.7	120.0	0.001**

(C) Third iteration of the Code game.

TABLE 3.8: Mann-Whitney U test results over the Code game scores of the second session.

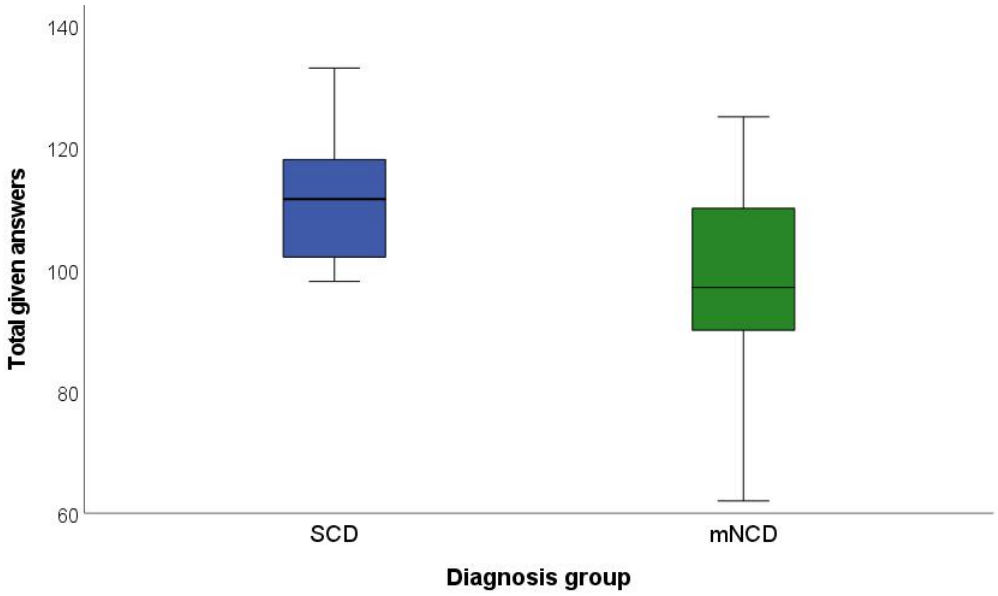


FIGURE 3.7: Box plot of the answers gathered by the mild NCD and the SCD groups on the first iteration.

This box plot highlights the differences in the amount of answers. Indeed, even if the mild NCD results seem to vary in a range that almost include the whole range of SCD results, one can notice that two entire quartiles of the mild NCD results are actually under the lowest score of the SCD group.

Second, in an exploratory attempt and as there are no non-parametric equivalent, a two-way mixed-design ANOVA checked for the causes of differences of scores between groups and iterations[2]. All results of the two-way mixed-design ANOVA are reported in tables 3.9 for the first session and tables 3.10 for the second one.

Effect	Value	F	df	p-value
Iteration	0.290	56.249	2	$< 0.001^{***}$
Iteration*diagnosis	0.963	0.883	2	0.421

(A) Wilks' Lambda.

	Mauchly's W	df	p-value	Hyunh-Feldt ϵ
Iteration	0.930	2	0.190	0.993

(B) Mauchly sphericity test.

Source	Type III Sum of ^{2}	df	Mean ^{2}	F	p-value
Iteration	3438.388	1.986	1731.635	69.158	$< 0.001^{***}$
Iteration*diagnosis	36.510	1.986	18.387	0.734	0.482
Error iteration	2336.755	93.325	25.039		

(C) "Test of within subjects effect" with Hyunh-Feldt correction.

TABLE 3.9: Two-way mixed-design ANOVA: effect on the amount of answers of the repetition of the game within the first session and of the diagnosis group of the subject.

Effect	Value	F	df	p-value
Iteration	0.671	11.011	2	0.000^{***}
Iteration*diagnosis	0.994	0.132	2	0.877

(A) Wilks' Lambda.

	Mauchly's W	df	p-value	Hyunh-Feldt ϵ
Iteration	0.752	2	0.002	0.844

(B) Mauchly sphericity test.

Source	Type III Sum of ^{2}	df	Mean ^{2}	F	p-value
Iteration	1729.962	1.687	1025.340	16.765	0.000^{***}
Iteration*diagnosis	11.240	1.687	6.662	0.109	0.865
Error iteration	4746.705	77.312	61.160		

(C) "Test of within subjects effect" with Hyunh-Feldt correction.

TABLE 3.10: Two-way mixed-design ANOVA: effect on the amount of answers of the repetition of the game within the second session and of the diagnosis group of the subject.

As introduced in section 3.2.1, to execute the "test of within subjects effect" of SPSS and to understand its results, it is necessary to perform a Mauchly sphericity test and to apply a correction of the degree of freedom with the ϵ of, e.g., Hyunh-Feldt. Indeed, as a reminder, the data obtained in this protocol are non normal and do not allow a Mauchly sphericity test to give robust results, thus we assume the

[2] As the "Box's test of equality of covariance" is really sensitive to non normal data such as the ones collected in this protocol, it was not included in the analysis.

non-sphericity of the data. As the Hyunh-Feldt ϵ falls between 0.75 and 1.00, the degree of freedom of the "test of within subjects effect" has to be corrected by Hyunh-Feldt method. As tables 3.9 and 3.10 show, both results of the Wilks' Lambda test and the "test of within subjects effect" show that only the repetition of the game permits to reject hypothesis H0. Thus, one can assume that only the repetition of the game had an effect on the progression of the subjects. This result suggests that both groups should have a similar progression over the game iterations. Such progression can be observed in figures 3.8 and 3.9. This result should be taken for what it is worth, as the necessary assumptions were not met.

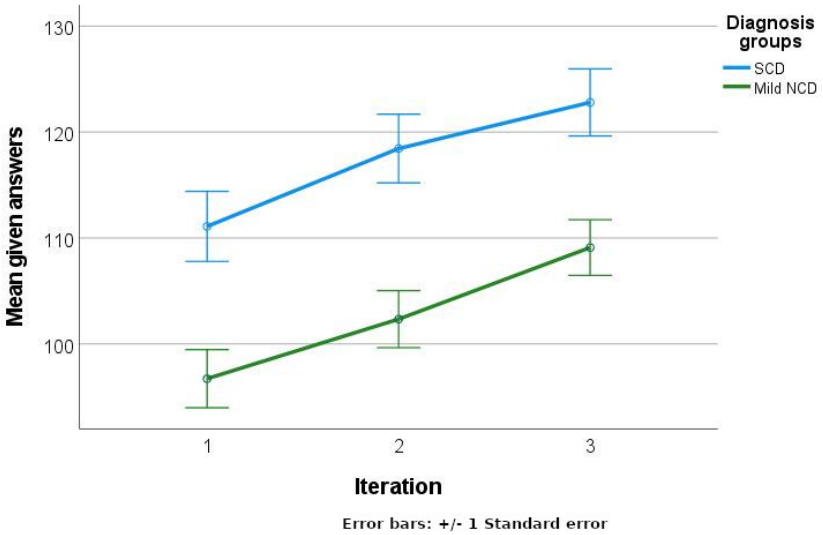


FIGURE 3.8: Variations of means of total amount of answers in the first session.

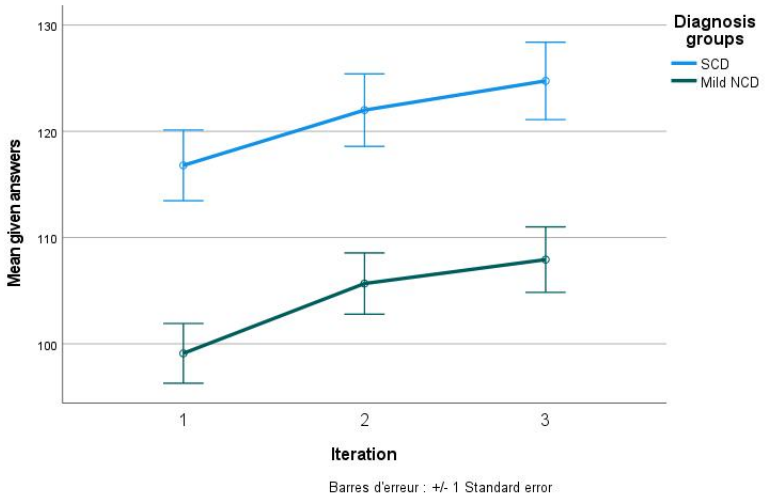


FIGURE 3.9: Variations of means of total amount of answers in the second session.

In order to validate the effect of the repetition of the game over the patients results, the Friedman statistical test has been performed on each group for both session. The results are displayed on tables 3.11 to 3.14.

N	28
Khi^2	17.643
df	2
p-value	0.000***

TABLE 3.11: Friedman Test result of the mild NCD group first session.

N	30
Khi^2	36.867
df	2
p-value	0.000***

TABLE 3.12: Friedman Test result of the mild NCD group second session.

N	20
Khi^2	32.076
df	2
p-value	0.000***

TABLE 3.13: Friedman Test result of the SCD group first session.

N	20
Khi^2	18.260
df	2
p-value	0.000***

TABLE 3.14: Friedman Test result of the SCD group second session.

According to tables 3.11 to 3.14, the iteration of the game has an impact on the patient results in both groups and in both sessions.

Third, a ρ of Spearman correlation test checked the correlation between the Code game scores and the neurocognitive test results. The results of this correlation test are summarized in table 3.15.

		Answers	Right answers	Wrong answers	Time
MMSE	coeff. corr	0.433	0.431	-0.002	0.275
	p-value	0.002**	0.002**	0.990	0.053
	Sample sizes	50			
DSS	coeff. corr	0.783	0.790	-0.214	0.166
	p-value	0.000***	$<$ 0.000***	0.136	$<$ 0.248
	Sample sizes	50			
FAB	coeff. corr	0.274	0.267	0.089	0.115
	p-value	0.054	0.061*	0.540	0.428*
	Sample sizes	50			
Digit Span reversed	coeff. corr	0.182	0.166	-0.027	0.350
	p-value	0.237	0.283	0.863	0.822
	Sample sizes	44			

TABLE 3.15: ρ of Spearman test on the correlation of the third iteration of the first session of the Code game scores and neurocognitive test results.

This table shows that the H0 hypothesis can be rejected for several relations (p-value under 0.05 and even 0.005). Hence, one can assume that some Code game scores are correlated to several neurocognitive tests. As expected, the scores have

a strong correlation coefficient with the DSS test (which inspired the game) and the MMSE one (which is a global neurocognitive test). The strongest correlation coefficients are the time mean and the total amount of answers. As expected, according to the first test of this subsection, the number of wrong answers is not correlated to any neurocognitive test. The correlation of the total amount of answers variable with the MMSE and the DSS results are displayed in figure 3.10

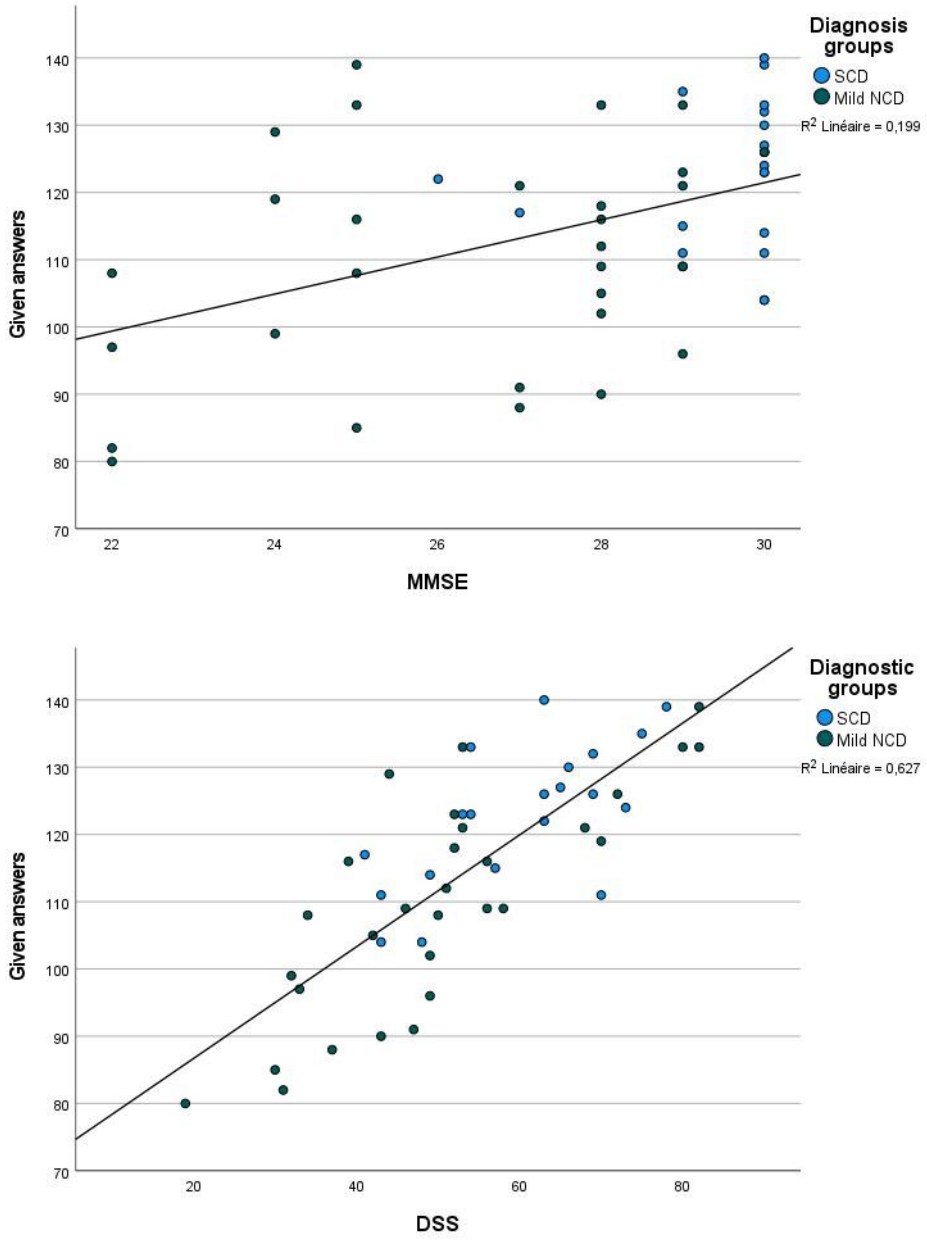


FIGURE 3.10: Correlation of total amount of answers in the third iteration of the first session with MMSE and DSS tests.

Discussion

The Code game score analysis showed that this game could differentiate participants with neurocognitive disorders from control ones. Indeed, the scores are significantly different between groups and this difference remains consistent from an iteration to

another. The scores are also strongly correlated to well established neurocognitive test results. The test from which it was inspired (the DSS one) is known to measure processing speed and attention. The game showed to be relevant in measuring processing speed but this experiment cannot measure any attention assessment. As the number of wrong answers is not recorded in the DSS results it is difficult to say for sure that the Code game cannot measure the efficiency of the player's attentional process.

Regarding a first comparison with the model described in the previous chapter, one can notice major differences. Indeed, even though subject are given less time (two minutes instead of five), mild NCD subjects get far more right answers (95.5 instead of 31) and far less wrong answers (1 instead of 15). Such difference led to a redesign of the model before an actual calibration.

3.2.4 Recognition Game

As a reminder this game displays pictures randomly to the patient. Some pictures have a duplicate that appears several pictures after its original. 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. This game has three levels of difficulty and three picture themes randomly attributed to each level for each patient. A fourth level with a fourth theme is used as training to help subjects understanding the rule of the game.

The data gathered in this game were:

- series of pictures displayed to the subject, of actions of the subjects, and time for each action;
- levels played, theme for each level, number of unique and original pictures put in the folder, number of duplicates put in the trash, and total amount of time taken to end each level.

The following analysis mainly focuses on the first session.

Results

First, a Kruskal-Wallis test checked whether the theme (nature, animals, or flowers) of each level had an impact on the subject results. As a reminder, the null hypothesis H0 for this test states that all samples come from the same population. Here, for each level, the samples are composed of both groups (mild NCD and SCD subjects) for each theme (3 samples for each level). The results are displayed on tables 3.16.

These tables show that, except for the time variable of the first level, the p-values are above 0.05. Thus the H0 hypothesis cannot be rejected and on can assume that the picture theme does not impact the subject performance. This result indicates that if there are some differences between the results of levels 1, 2, and 3, they are more likely to be caused by the difficulty of the level rather than by a difference of picture theme. Hence, it is reasonable to focus on the levels of difficulty only.

To go further, the Kruskal-Wallis test has also been performed on the same data but for each group. The results of these tests are displayed from tables 3.17 to tables 3.18.

The results shown in these tables are consistent with the more global results of table 3.16. There are only three variables that are exceptions:

	Correct folder	Correct trash	Time(ms)
Mean	31.588	4.1373	111944.254
Std Error	0.339	0.181	5000.792
Kruskal-Wallis H	4.488	1.047	7.835
df		2	
p-value	0.106	0.592	0.020*

(A) 1st level

Theme	Rank
Nature	28.670
Animals	25.250
Flowers	24.600

(B) 1st level ranks associated to time

	Correct folder	Correct trash	Time(ms)
Mean	31.1373	4.745	112225.235
Std Error	0.418	0.172	5159.651
Kruskal-Wallis H	0.566	5.554	2.232
df		2	
p-value	0.753	0.062	0.328

(C) 2nd level

	Correct folder	Correct trash	Time(ms)
Mean	30.137	5.019	117972.392
Std Error	0.432	0.117	5469.389
Kruskal-Wallis H	5.436	1.991	0.701
df		2	
p-value	0.066	0.964	0.704

(D) 3rd level

TABLE 3.16: Examples of results obtained for the Kruskal-Wallis test on the first session.

- the correct trash of the second level for the mild NCD group (the ranks suggest that the theme "flowers" leads to different results);
- the time to finish the first level for the SCD group (the ranks suggest that the theme "animals" leads to different results);
- the correct folder of the third level for the SCD group (the ranks suggest that the theme "animal" leads to different results);

These variables are highlighting some levels that could be improved for future development but tend to be the exception, rather than the rule as they are only three.

Second, a Mann-Whitney U test checked the differences of scores between both groups. The results are shown in table 3.19.

This table shows that only the variable *correct folder* reaches a p-value under 0.05 (under 0.005 for the first level). Thus the null hypothesis H0 stating that mild NCD and SCD subjects have similar scores can only be rejected for the *correct folder* variable (visualization in figure 3.11).

	Correct folder	Correct trash	Time(ms)
Mean	30.838	4.129	111169
Std Error	0.479	0.230	6587.981
Kruskal-Wallis H	1.803	2.922	2.494
df	2		
p-value	0.406	0.232	0.287

(A) 1st level

	Correct folder	Correct trash	Time(ms)
Mean	30.2253	4.580	114692
Std Error	0.573	0.248	7605.81
Kruskal-Wallis H	0.055	6.108	0.089
df	2		
p-value	0.973	0.047*	0.957

(B) 2nd level

Theme	Rank
Nature	13.500
Animals	13.180
Flowers	21.600

(C) 2nd level ranks associated to correct trash

	Correct folder	Correct trash	Time(ms)
Mean	29.225	5.000	119974
Std Error	0.559	0.160	8053.386
Kruskal-Wallis H	2.259	2.183	4.126
df	2		
p-value	0.323	0.336	0.127

(D) 3rd level

TABLE 3.17: Examples of results obtained for the Kruskal-Wallis test on the first session for the mild NCD group.

Third, as it was done for the Code game, in an exploratory attempt, a two-way mixed design ANOVA checked the differences of variation of scores for each level of difficulty. This analysis focuses on the *correct folder* and *correct trash* variables. All results are presented in table 3.20 for *correct folder* and table 3.21 for *correct trash*.

This table shows that only the tests regarding the effect of the difficulty level reach a p-value under 0.005 and leads to reject hypothesis H0. Thus, one can assume that only the level of difficulty of the game had an effect on the progression of the subjects. This result suggests that both groups should have a similar progression over the game levels, but it should be taken for what it is worth, as the necessary assumptions were not met.

This second ANOVA table shows once more that only the tests regarding the effect of the difficulty level reach a p-value under 0.005 leading to the same observation as for table 3.20. Figure 3.12 shows a summary of this ANOVA study. Once again, this result should be taken for what it is worth, as the necessary assumptions were not met.

Fourth and last, a ρ of Spearman correlation test checked if the results can be correlated to the subjects clinical data. The results of this test are shown in table 3.22

	Correct folder	Correct trash	Time(ms)
Mean	32.750	4.150	113146.500
Std Error	0.306	0.301	7840.390
Kruskal-Wallis H	0.752	0.583	7.010
df	2		
p-value	0.687	0.747	0.030*

(A) 1st level

Theme	Rank
Nature	12.330
Animals	3.500
Flowers	12.000

(B) 1st level ranks associated to time

	Correct folder	Correct trash	Time(ms)
Mean	32.550	5.000	108402.000
Std Error	0.444	0.205	5983.809
Kruskal-Wallis H	0.512	2.409	3.298
df	2		
p-value	0.774	0.300	0.192

(C) 2nd level

	Correct folder	Correct trash	Time(ms)
Mean	31.550	5.050	114869.000
Std Error	0.559	0.169	6416.96
Kruskal-Wallis H	8.563	0.689	1.911
df	2		
p-value	0.014*	0.709	0.385

(D) 3rd level

Theme	Rank
Nature	6.200
Animals	15.500
Flowers	8.810

(E) 3rd level ranks associated correct folder

TABLE 3.18: Examples of results obtained for the Kruskal-Wallis test on the first session for the SCD group.

This correlation test shows that the H0 hypothesis can be rejected for two relations, the relation between the *correct folder* variable and, on the one hand, with the memorization part of the "5 Words test" and, on the other hand, with the ordered part of the "Digit Span test" (p-value under 0.05). H0 can also be rejected for the relation between the *time* variable and both the DSS test and the memorization part of the "5 Words test" (p-value under 0.05). Thus one can assume that both the *correct folder* and the *time* variables can be correlated to some neurocognitive tests.

Discussion

The analysis of this game can be summarized as follows:

	mild NCD (n=31)	SCD (n=20)	U value	p-value
Correct folder, mean±Std Err	30.8 ± 0.4	32.7 ± 0.3	148.0	0.001**
Correct trash, mean±Std Err	4.1 ± 0.2	4.1 ± 0.3	308.5	0.976
Time (s), mean±Std Err	111.1 ± 6.5	113.1 ± 7.8	305.0	0.923

(A) 1st level

	mild NCD (n=31)	SCD (n=20)	U value	p-value
Correct folder, mean±Std Err	30.2 ± 0.5	32.5 ± 0.4	168.5	0.005*
Correct trash, mean±Std Err	4.5 ± 0.2	5.0 ± 0.2	268.0	0.396
Time (s), mean±Std Err	114.6 ± 7.6	108.4 ± 5.9	212.5	0.862

(B) 2nd level

	mild NCD (n=31)	SCD (n=20)	U value	p-value
Correct folder, mean±Std Err	29.2 ± 0.5	31.5 ± 0.5	172.5	0.008*
Correct trash, mean±Std Err	5.0 ± 0.1	5.0 ± 0.1	308.5	0.975
Time (s), mean±Std Err	119.9 ± 8.0	114.8 ± 6.4	298.0	0.817

(C) 3rd level

TABLE 3.19: Mann-Whitney U test on the distribution of scores within SCD and mild NCD groups of the first session.

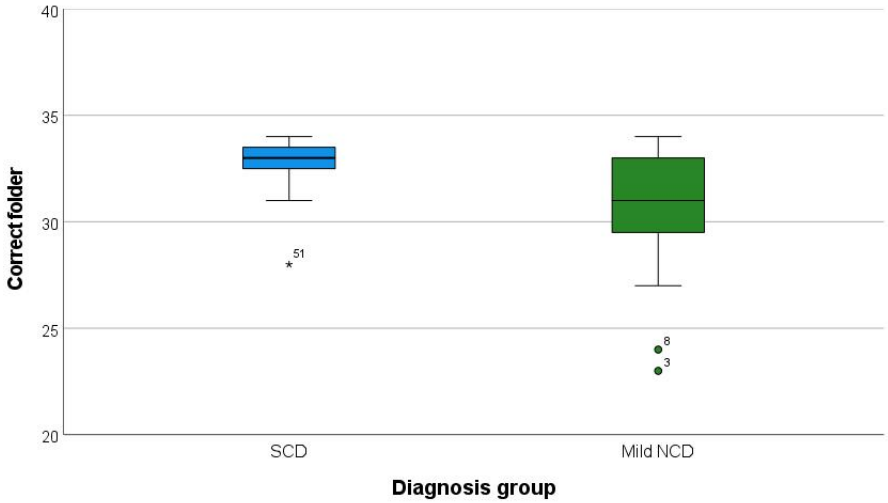


FIGURE 3.11: Box plot of correct folder actions of the mild NCD and SCD groups.

- the 3 themes did not impact the subject performance;
- a significant difference of performance can be observed between both groups;
- both groups show the same evolution of scores in the different levels;
- the results of this game are correlated to working-memory tests and more specifically to their memorization phase.

The similar evolution of performances within the game could be interpreted as a similar strategy chosen by both groups. Indeed, as difficulty increases, more subjects

Effect	Value	F	df	p-value
Level	0.758	7.678	2	0.001^{**}
Level*diagnosis	0.988	0.300	2	0.742

(A) Wilks' Lambda.

	Mauchly's W	df	p-value	Hyunh-Feldt ϵ
Level	0.867	2	0.033	0.932

(B) Mauchly sphericity test.

Source	Type III Sum of ^{2}	df	Mean ^{2}	F	p-value
Level	50.950	1.864	27.334	7.208	0.002^{**}
Level*diagnosis	1.382	1.864	0.741	0.195	0.808
Error level	346.370	91.335	3.792		

(C) Test of within subjects effect with Hyunh-Feldt correction.

TABLE 3.20: Two-way mixed design ANOVA: effect on the *correct folder* variable of the level of difficulty and of the diagnosis group.

Effect	Value	F	df	p-value
Level	0.0.736	8.597	2	0.001^{**}
Level*diagnosis	0.974	0.647	2	0.528

(A) Wilks' Lambda.

	Mauchly's W	df	p-value	Hyunh-Feldt ϵ
Level	0.979	2	0.601	1.000

(B) Mauchly sphericity test.

Source	Type III Sum of ^{2}	df	Mean ^{2}	F	p-value
Level	20.467	2.000	10.234	10.029	$< 0.001^{***}$
Level*diagnosis	1.199	2.000	0.600	0.588	0.558
Error level	100.003	98.000	1.020		

(C) Test of within subjects effect with Hyunh-Feldt correction.

TABLE 3.21: Two-way mixed design ANOVA: effect on the *correct trash* variable of the level of difficulty and of the diagnosis group.

tended to place pictures in the trash thus reducing the errors on pictures that should be placed there but increasing errors on those that should be placed in the folder.The main difference between the two groups seems to rely on a "initial memory capital" that is greater in SCD subjects over mild NCDs.

Regarding the comparison with the model described in the previous chapter, one can notice that the correct folder categorization are pretty close (29.2 observed for 28.7 estimated). However, the good trash categorization was underestimated (5.0 observed for 1.2 estimated) and the "hesitation time" would require a deeper analysis to be extracted. This model can be calibrated without any modification of its structure.

3.2.5 Inhibitory Control Game

This game required to filter the data as the tablet had issues in some cases to detect finger removal and the engineer could not find the cause of this problem. Thus the

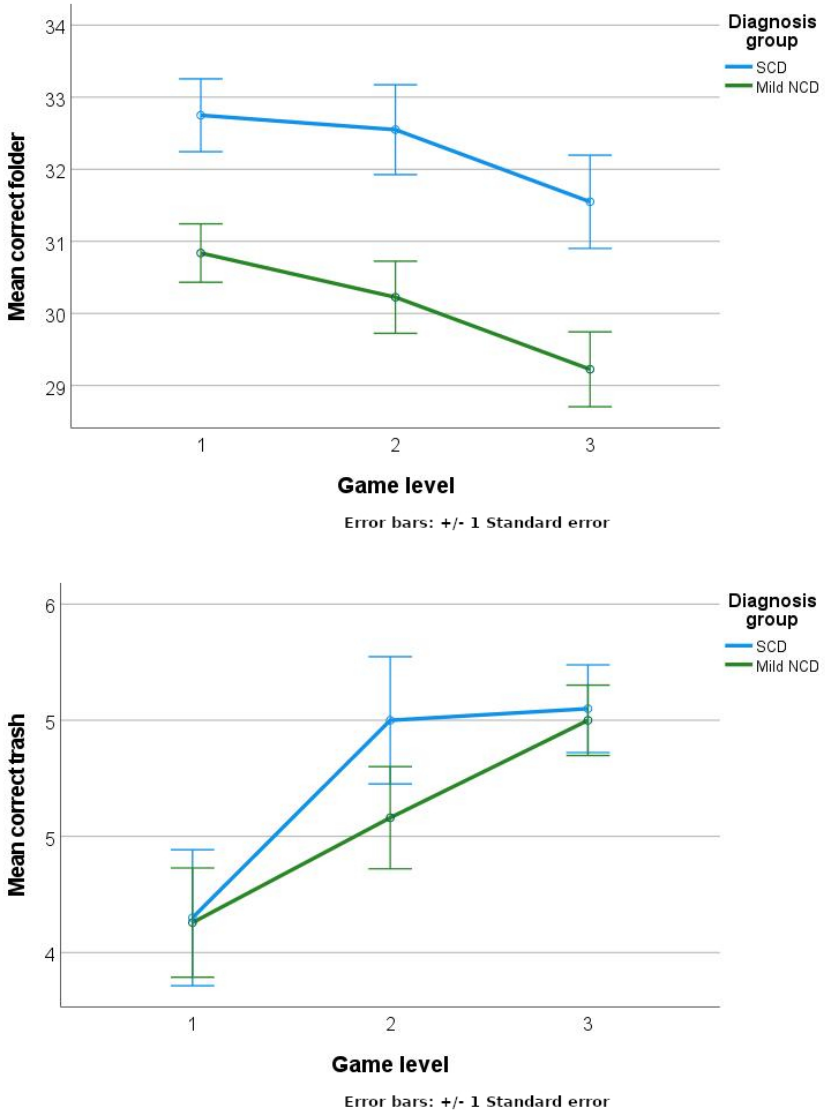


FIGURE 3.12: Variations of means of *correct folder* and *correct trash* according to difficulty levels.

following analysis focuses mainly on the second session which showed less problems. However, since both sessions are equivalent according to the Wilcoxon signed ranks test, the first session was kept when the second one was unusable. The subject results which showed issues on both sessions were removed from the analysis. At the end, 46 subjects remained (19 SCD and 27 mild NCD).

As a reminder, this game is inspired by the go/no-go task. It displays a target (or a decoy) to the player who has to react (or not) accordingly. Each move starts with the player touching the green button (see figure 2.16 of previous chapter). If a target is displayed on the screen, the player must release the button to touch the target as fast as possible and get back to the initial position. If a decoy is displayed, the player must not move. The measured reaction time is the interval when the button is released. Releasing the button when the target appears is a right action whereas releasing it when the decoy appears is a wrong action.

The data gathered in this game were:

		Correct folder	Correct trash	Time
MMSE	coeff. corr	0.258	0.008	-0.101
	p-value	0.068	0.957	0.482
	Sample sizes	51		
DSS	coeff. corr	0.179	0.079	-0.305
	p-value	0.210	0.584	0.030*
	Sample sizes	51		
5 Words memorization	coeff. corr	0.304	-0.203	-0.332
	p-value	0.034*	0.162	0.020*
	Sample sizes	49		
5 Words restitution	coeff. corr	0.204	-0.039	-0.126
	p-value	0.159	0.789	0.388
	Sample sizes	49		
Digit Span ordered	coeff. corr	0.300	0.020	0.034
	p-value	0.045*	0.897	0.825
	Sample sizes	45		
Digit Span reversed	coeff. corr	0.076	-0.102	0.191
	p-value	0.621	0.506	0.209
	Sample sizes	45		

TABLE 3.22: ρ of Spearman test on the correlation of the Code game scores and neuro-cognitive test results.

- the series of signals displayed to the subject, the actions of the subject, the reaction time for each action;
- the number of right and wrong actions, the time mean for the right actions and for the wrong ones.

Results

First, the Mann-Whitney U test checked if one of the recorded variables shows differences between both groups. The results are displayed in table 3.23

	mild NCD $(n^a = 27)(n^b = 23)$	SCD $(n^a = 19)(n^b = 16)$	U value	p-value
Valid go^a, mean±Std Err	29.3 ± 0.1	29.7 ± 0.1	173.0	0.029*
Invalid no-go^a, mean±Std Err	2.0 ± 0.3	1.8 ± 0.2	247.0	0.828
Time go^a (ms), mean±Std Err	498.0 ± 11.5	458.8 ± 9.8	162.0	0.035*
Time no-go^b (ms), mean±Std Err	409.5 ± 21.0	361.0 ± 9.5	110.0	0.035*

TABLE 3.23: Mann-Whitney U test on the distribution of scores within SCD and mild NCD groups.

This first table shows that, regarding the score means, only the valid go actions and the reaction times give p-values under 0.05. Thus the H0 hypothesis can be rejected and one can assume that these variables distinguish the two groups. This distribution of scores is illustrated in figure 3.13.

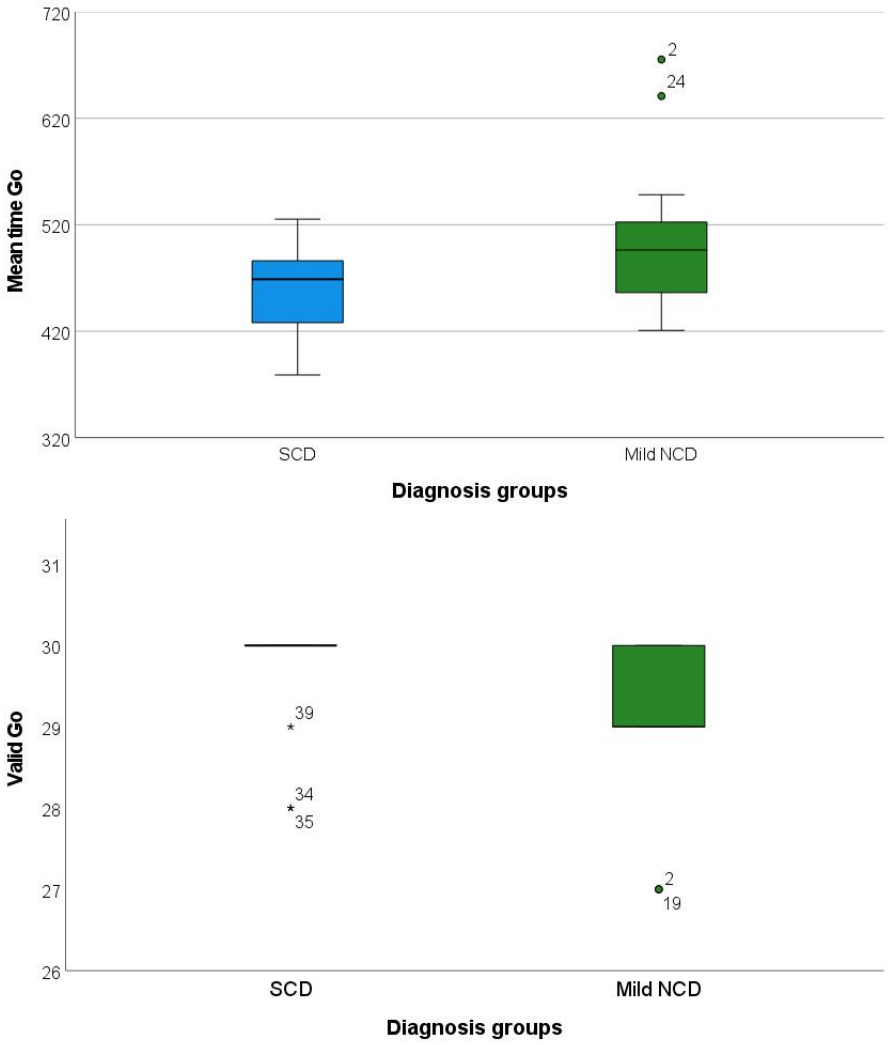


FIGURE 3.13: Box plot of the reaction time and valid go for mild NCD and SCD groups.

The box plot on valid go actions shows a difference as it is really visible that most subjects of the SCD group have a score of 30 while the mild NCD group is scattered between 30 and 29. The box plots display also a noticeable difference in the reaction time between groups as mild NCDs take more time to react.

Second, the ρ of Spearman test checked the correlation between the game variables and the subject clinical data. For the sake of clarity, only the most relevant data are displayed in table 3.24.

As expected, the MMSE shows a significant correlation with at least a part of the result variables: the number of valid go actions and the reaction time on invalid no-go actions. The DSS results are also strongly correlated with both the number of valid go actions and the reaction times on the target. Surprisingly, the FAB test shows no correlation at all. To go further, the different parts of the FAB have been checked as well. The go/no-go task, which is the typical measure method for the inhibitory control, is not correlated at all to the game results. On the other hand, the Luria's task (assessing both fine motor and sequencing motor skills) result is strongly correlated to the number of valid go actions and both reaction times. Figure 3.14 gives a better view on the interaction of the game scores with the go/no-go task, the Luria's task, and DSS.

		Valid go	Invalid no-go	Time go	Time no-go
MMSE	coeff. corr	0.323	-0.085	-0.1295	-0.390
	p-value	0.028*	0.577	0.393	0.014*
	Sample sizes	46			39
FAB	coeff. corr	0.286	0.149	-0.256	-0.237
	p-value	0.054	0.323	0.085	0.147
	Sample sizes	46			39
FAB go/no-go	coeff. corr	0.271	0.177	-0.117	-0.043
	p-value	0.068	0.240	0.438	0.794
	Sample sizes	46			39
FAB Luria's task	coeff. corr	0.317	-0.041	-0.432	-0.375
	p-value	0.032*	0.787	0.003**	0.019*
	Sample sizes	46			39
DSS	coeff. corr	0.400	-0.181	-0.408	-0.143
	p-value	0.006*	0.228	0.005*	0.385
	Sample sizes	46			39

TABLE 3.24: ρ of Spearman test on the correlation of game scores and neurocognitive test results.

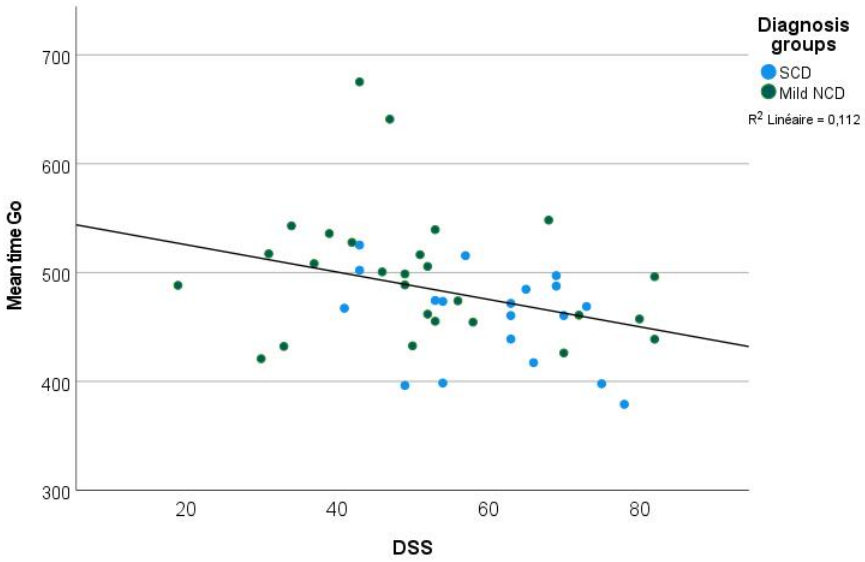


FIGURE 3.14: Correlation between the mean reaction times and DSS test.

Discussion

This analysis showed that the Inhibitory Control game was suitable to differentiate SCD and mild NCD subjects. However, it does not target the right cognitive function. Indeed, the number of right actions on targets and the reaction time on both target and decoy were significantly different in both groups. But surprisingly, the ρ of Spearman test did not show any correlation with the go/no-go task that inspired this game. On the other hand, the Luria's task showed way more correlation. This part of FAB requires the patient to reproduce movements to demonstrate fine motor skills and sequencing motor skills thus revealing possible executive dysfunction or damage.

Regarding the comparison with the model described in chapter 2, there are major

differences. Indeed, the final version of the game includes more signals and subjects succeed more than what had been estimated. For go signals (targets in the model), mild NCD subjects got 29.3/30 but only 5.55/10 had been estimated in the model. For no-go signals (decoys in the model), they got 8/10 but only 0.86/5 had been estimated. Moreover, some data were unfortunately not recorded (e.g., signal display times). Such conditions led to a redesign of the model before an actual calibration.

3.2.6 Result Overview

Discussion

The feasibility protocol shows that the games selected for this work seem to be suited to discriminate between mild NCDs and SCDs. It particularly shows that the Code game and the Recognition game scores seem to be consistent with the neurocognitive tests assessing the cognitive functions of these games. As all the games give a positive result for this study, they make it possible to calibrate their models detailed in section 3.3; this calibration is the next step of figure 2.2 of chapter 2.

Protocol Feedback

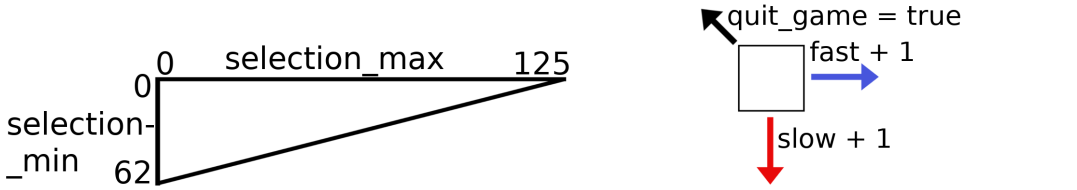
Several points could be raised to increase the efficiency of future protocols.

- Collect the number of mistakes made by subjects in their DSS test and also other interesting data such as signal display times.
- Select another frontal test to replace the FAB one. Indeed, as this test grades each assessed function with a rank from 0 to 3, it was difficult to get a robust correlation analysis. A more specialized test on the go/no-go task would be more efficient.
- Hide all game parameters to put subjects in a situation closer to actual neurocognitive tests and to avoid "cheating". For instance, it is not advisable that the subjects know in advance internal parameters of the game (such as the total amount of expected actions) and possibly base their behavior on this knowledge.
- Improve the training phases so that subjects can learn the games without being trained on the exact same critical parameters as the actual games (e.g., maximum time to react, arrangement of buttons ...).

3.3 Model Redesign and Calibration

The results obtained in the protocol experimentation differ from those of the model checking in chapter 2. This difference can be explained by the changes in the game parameters and because the initial profiles used in the models were not adapted. Indeed, these profiles represented major NCDs rather than mild NCDs and the SCD profile was not fully specified yet.

This section focuses on the redesign as well as the calibration of the Code game and Inhibitory Control game PRISM models in order to make them compliant with the observed experimental results. To do so, the statistics described in section 3.2 as well as observations on raw data were used to adjust the models parameters.



(A) Combinations of events triangle.

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

FIGURE 3.15: Activity model for a player of the Code game.

3.3.1 Code Game

The Code game was initially modeled with a five minutes duration. Subjects were supposed to select a picture only once every three seconds and to have some inactivity moments. With such parameters, subjects were expected to select 46 pictures for 31 right answers and 15 wrong answers. It was also expected that subjects had an overall risk of 3% to quit the game before the end of the duration. All these previsions are far from the protocol results. A redesign of the model was thus required.

Redesign

The main focus points in the redesign are the duration of a session and the frequency of actions. Indeed, the actual version of the game played during the protocol lasts two minutes and the experiment results show that the frequency of actions is the main criterion (if not the only one) to differentiate mild NCD from SCD subjects. In the new version of the model, the axis "selection" of figure 2.4a is replaced with an axis "*selection_max*" and axis "inactivity" is replaced with an axis "*selection_min*". The values given to these variables depend on the raw data. For example, for mild NCDs, the maximum number of answers in the raw data is 125 and the minimum is 62; thus "*selection_max*" is equal to 125 and "*selection_min*" is equal to 62. Hence, figure 2.4a becomes figure 3.15a.

As raw data did not show any inactivity within a Code game session, it seemed reasonable to change the *inactivity* variable in the model. Thus, in the new version of the Code game model, the modeled subject can only select an answer in a fast or a slow way which are represented with two ranged integers *fast* and *slow* (represented in figure 3.15b). On the other hand, we decided to keep the *quit_game* action but with a probability close to zero (10^{-54}). Indeed during the protocol, only one patient interrupted a game session and it was not in the middle of a game.

Looking at raw data information on mistakes, one can notice different types of errors. The first and most noticeable type of error is the "miss click" one, that is the error of a subject selecting the answer located directly beside the right answer in the selection bar at the bottom of the screen. The second major type of error is the "anticipation" one, that is the error of a subject selecting the answer corresponding to the next picture in the list at the center of the screen. Other potential types of error were identified but won't be described here as they represent a small amount of errors for both mild NCD and SCD subjects. The raw data show that both groups had almost the same amount of errors but with a different distribution of the two main types of errors (see table 3.25). This difference was observed but not statistically tested yet.

These types of errors have been included in the model as Boolean variables in the module representing the subject. For example the variable *miss_click* represents the "miss click" error type and *anticipate* represents the "anticipation" one.

Error type	mild NCD	SCD
Total amount	31	27
Miss click	15	6
Anticipation	1	9

TABLE 3.25: Error types in mild NCD and SCD groups.

Once the model has its new structure, the next step consists in its calibration to represent the behavior of a group. This section only focuses on the first play of the first session of the mild NCD subjects.

Calibration for Mild NCD Profile

As said earlier, the maximum number of answers for mild NCDs in the raw data is 125 and the minimum is 62. The *selection_max* and *selection_min* variables are thus set to 125 and 62. According to the data analysis, within two minutes, mild NCD subjects gathered a mean of 96.5 answers including a mean of 1 wrong answers. To simplify the model, we consider that the number of answers for patients of the mild NCD group follows a normal distribution centered on the mean 96.5. Looking at the raw data, subjects seem to show quite a constant speed all along the game session. Hence, the probabilities for the patients to select an answer fast or slow are given by a constant value *pfast* and a formula *pslow* with $pslow = 1 - pfast$. Setting *pfast* to 0.7 leads the model to compute a mean of 96.3 rewards; these rewards represent answer gathering.

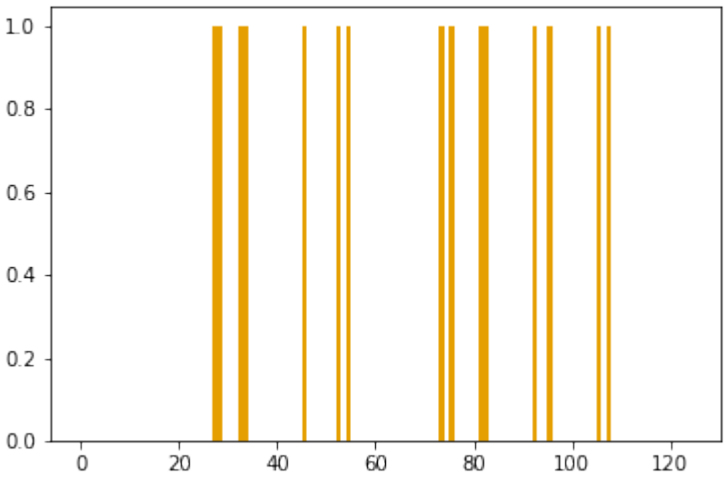
The probabilities to give a right or a wrong answer were calibrated in three steps. First, the easiest was to determine the constant probability values for right and wrong answers, regardless of the type of error, to compute a mean reward close to 95.5 for right answers and to 1.0 for wrong answers. Second, the wrong answer probability was redistributed to the probabilities of each type of error weighted by their observed ratio. Third and last, *ad hoc* functions were introduced to approximate the observed distribution of error types in a game session. For example, figure 3.16a shows that most errors of miss click are made between the 25^{th} and 55^{th} answers and between the 70^{th} and 110^{th} answers. To represent this distribution, we chose to define a density function consisting of two Gaussian-like peaks as shown in figure 3.16b. All constants of these Gaussian-like functions were manually chosen to get closer to the results observed during the experimentation.

Once the model calibrated, properties have to be adapted and checked to compare with the experimental results. The main changes as well as two of the new variables are summarized in table 3.26.

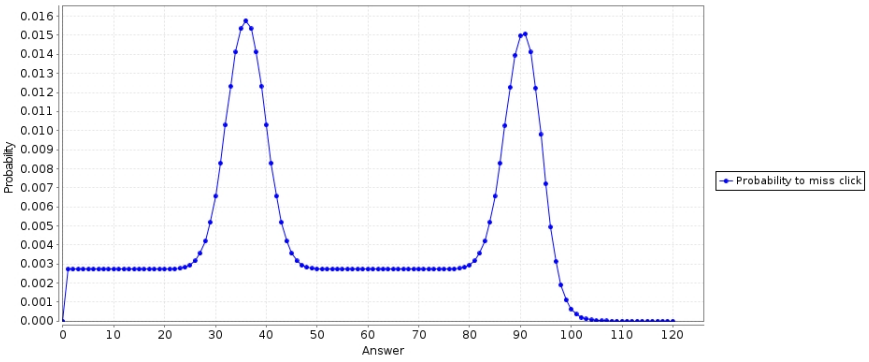
Verification

The properties found in this section are either adapted versions or the exact same properties as the ones found in section 2.3.2 of chapter 2. For this version, instead of the "hybrid" model checking engine, PRISM software required the "explicit" engine to compute the results with no errors.

Properties about speed. The following properties are extracted from properties 2, 3, 4 and 6 of chapter 2, section 2.3.2. In this chapter, we keep the same numbers as in chapter 2, section 2.3.2. They evaluate the probability for a path to go through i occurrences of *fast* actions and j occurrences of *slow* actions. The first three properties



(A) Overall miss click errors observed for mild NCD patients.



(B) Probability density for modeled mild NCD patients to miss click on a given answer.

FIGURE 3.16: Miss click errors, from observations to probabilities.

check the probability to end the game with $i = selection\_max$ or $j = selection\_min$, or i in between 0 and $selection\_max$ and j in between 0 and $inactivity\_max$ so that $i + j$ is equal to the mean number of actions observed in the protocol (96). The last one checks the probability for the patient to leave the game unexpectedly.

Property 2. What is the probability for a patient to be as slow as the slowest subject observed in experiment?

$$P =?[F\ (fast = 0)\ \&\ (slow = selection\_min)]$$

Property 3. What is the probability for a patient to be as fast as the fastest subject observed in experiment?

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

Property 4. What is the probability for a patient to start the game and to interact with it, e.g., 96 times independently of the speed (*fast* or *slow*)?

$$P =?[F\ (location = location\_max)\ \&\ (fast + slow = 96)]$$

Original variable	New variable	Motivation
inactivity	*slow*	No inactivity observed, associated probability is *pslow*
selection	*fast*	Compliant with previous change, associated probability is *pfast*
inactivity_max	*selection_min*	Compliant with previous changes
–	*miss_click*	Observed mistake, associated probabilities are *pmiss_click1* and *pmiss_click2*
–	*anticipate*	Observed mistake, associated probability is *panticipate*

TABLE 3.26: Summary of changes in the Code game model.

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 3.27, together with their computing times.

The probability obtained for properties 2 and 3 are so low that PRISM considers them as 0. As the probabilities for the actions *slow* and *fast* are simply defined with fixed values, this result was expected. Indeed, there is only one existing path for each property to be satisfied. Moreover, on one path, there are many states to go through (62 for the *slow* action and 125 for the *fast* one), where the patient can escape the path; thus the final probabilities get pretty low.

On the other hand, the computation of property 4 reaches a value of 11%. This is due to the formula that includes all values of *slow* and *fast* variables verifying $fast + slow = 96$ when the model reaches its final state. Moreover, the model parameters were set so that the distribution of the final number of actions looks like a Gaussian centered on 96.5. Thus this result is consistent with our expectation.

Finally, the probability to verify property 6 is, as for properties 2 and 3, a really small one. As intended, this probability is even smaller (of the order of 10^{-53}). Even though this event never occurred during a Code game session of the protocol, it was considered to be a valuable information, should it happen.

Property	PRISM		Storm	
	Result	Time(seconds)	Result	Time(seconds)
Property 2	0.00	0.066	3.8152×10^{-33}	0.461
Property 3	0.00	0.047	4.3376×10^{-20}	0.068
Property 4	1.1471×10^{-1}	0.136	1.1471×10^{-1}	0.140
Property 6	0.00	0.078	9.6331×10^{-53}	0.190

TABLE 3.27: Results from properties 2 to 6.

Based on the assumption that subject speed distribution follows a Gaussian law, these results can be interpreted as follows. If patients finish the game with the maximum or minimum number of actions, there are chances for them not to be part of the mild NCD profile, even more if they decide to leave the game before the end of the timer.

Properties about quality of actions. The following property is directly imported from section 2.3.2 of chapter 2 where it is indexed as property 8. It is relative to the quality of the actions (correct or not) that can be performed. This reward-based property provides an average "score" for the model.

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

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

Discussion The results of this property for the different rewards of interest are displayed in table 3.28, together with their computing time.

Property 8 can be written for *Happy_smiley_reward*, *Sad_smiley_reward*, *miss_click_rew*, and *anticipate_rew*. According to its results, the average "score" for a cohort of patients matching this model parameters should be 95.32 right answers against 1.00 wrong ones for 0.49 miss click and 0.03 anticipation errors.

Reward	PRISM		Storm	
	Result	Time(seconds)	Result	Time(seconds)
Happy_smiley_reward	95.32	0.686	95.32	0.220
Sad_smiley_reward	1.00	0.42	1.00	0.218
miss_click_rew	0.49	0.381	0.49	0.215
anticipate_rew	0.03	0.377	0.03	0.210

TABLE 3.28: Results of property 8.

These results are close to those observed in the experimentation. Indeed, the observed mean of right and wrong answers are respectively 95.5 and 1.0. Regarding the "miss click" and "anticipation" errors, their proportions among all observed errors is respected with $15/31 = 0.48$ for the "miss click" error and $1/31 = 0.03$ for the "anticipation" one. Thus the model is quite representative of the mean results of mild NCD subjects.

Cumulative Rewards and Simulations The last property of this section corresponds to property 11 of section 2.3.2 in chapter 2. This property uses 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 "run experiment" feature of PRISM and the creation of graphs of results.

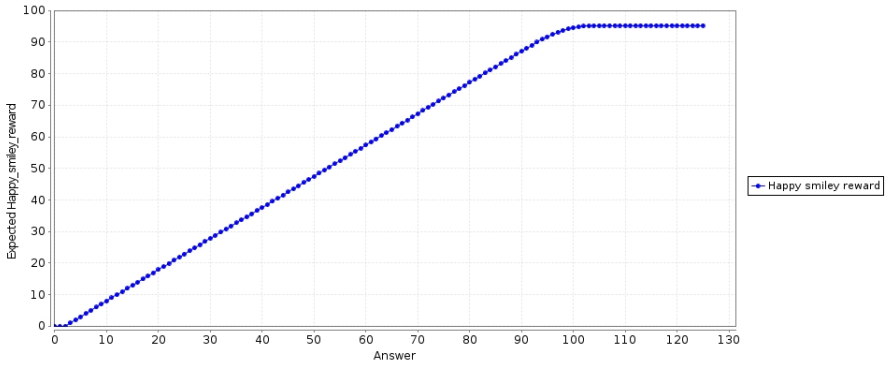
Property 11. What is the amount of happy smileys accumulated within i steps?

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

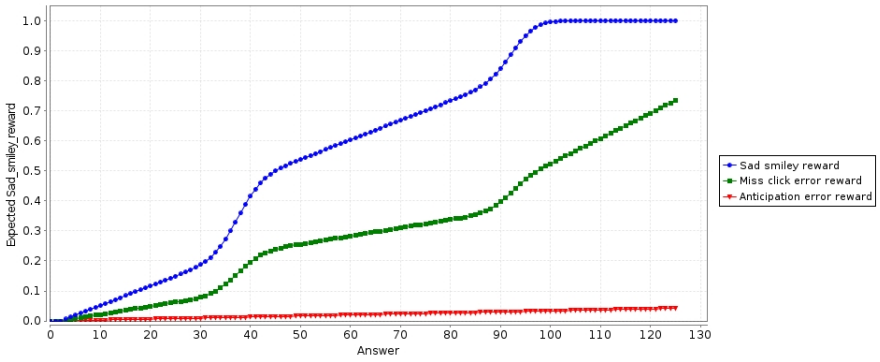
where i is the number of steps to perform before checking the reward and C is the "cumulative-reward" operator presented in section 2.1. This property is also applied to *Sad_smiley_reward*, *miss_click_rew*, *anticipate_rew*.

The diagrams displayed in figure 3.17 show the average accumulation of happy smiley rewards (subfigure 3.17a) and of sad smiley ones as well as rewards for the two main types of errors discussed above (subfigure 3.17b). Diagram 3.17a shows a relatively stable accumulation of happy smileys. Diagram 3.17b displays a more irregular accumulation of sad smileys which corresponds to the observed results in figure 3.16a.

https://fr.overleaf.com/project/616463c9ba65ec775aabf3d6



(A) Mean accumulation of rewards related to right answers.



(B) Mean accumulation of rewards related to wrong answers.

FIGURE 3.17: Average model checking results for rewards related to right and wrong answers.

3.3.2 Inhibitory Control Game

The Inhibitory Control game was initially modeled with 10 target (go signal) occurrences, 3 training occurrences of target, 5 decoy (no go) occurrences, and 1 training instance of decoy. The initial model also took into account the time elapsed between two signals. As depicted in the experimental protocol (see section 3.2.5), the actual number of targets is 30 and the actual number of decoys is 10. Moreover, the time elapsed between two signals and the action for the 3 training targets were not recorded in the game's log. Thus, model redesign was required.

Redesign

The whole redesign concerned the removal of the probabilistic time span between two signals. This removal implied a heavy deletion of sets of guards and updates as well as some variables. These deletions led to the modification of a large part of the remaining set of guard and updates.

Calibration for Mild NCD Profile

The easiest part of the calibration was to change the number of targets and decoys as the model was designed to ease these changes.

As said in the experimental results section, the protocol did not show differences in the number of mistakes on the decoy signals by the mild NCD and SCD groups. The main differences highlighted by the statistical analysis were on the number of errors on the target signals and on the reaction speed for both types of signals.

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_sfast *click_fast* *click_medi* *click_slow* *click_sslow*	Boolean indicating if the patient clicked in a fast or slow way for the previous signal. The fastest corresponds to *click_sfast* and the slowest to *click_sslow*.
not_click	Boolean indicating if the patient did not click at all for the previous signal.
action	Variable name used in properties to refer to one or more of the following variables: *click_sfast*, *click_fast*, *click_medi*, *click_slow*, *click_sslow*, or *not_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 indicating that the end of the game is reached.
game_on	Boolean indicating if the game is on or off.

TABLE 3.29: Summary of Inhibitory Control game model variables.

The data exploration showed only one pattern if two signals for which the mild NCD and SCD groups behavior differ. This exploration concerned both the temporal location of mistakes during a session as well as the verification of the signals and actions preceding a mistake. The exploration showed that 2/3 of the mistakes of type "no click" on target were just after the apparition of a decoy. The remaining calibration of the probabilistic parameters was made with the results displayed in the experimental protocol section.

To better represent the speed capacity of the patient, we categorized the reaction time in five classes instead of the two ones envisioned in the previous version of the model (*click_sfast*, *click_fast*, *click_medi*, *click_slow*, and *click_sslow*).

Verification

The properties found in this section are either adapted versions or the exact same properties as the ones found in section 2.3.2 of chapter 2. PRISM software required the "explicit" engine to compute the results with no errors as it was also the case for the initial version of this model.

To ease the comprehension of the following section, table 3.29 summarizes the correspondence between the model variables and what they represent.

Property	Result	Time(seconds)
Property 1	4.9369×10^{-1}	0.039
Property 5	4.6772×10^{-1}	0.011
Property 6	1.2272×10^{-1}	0.034

TABLE 3.30: Results from property 1 to 6.

Properties on patient actions The following properties are extracted from properties 1, 5, 6 and 7 of chapter 2, section 2.6.2. In this chapter, we keep the same numbers as in chapter 2, section 2.6.2. The first three properties evaluate the probability for a path to go through states representing good actions. The fourth one checks the amount of rewards accumulated for different actions.

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

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

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

$$P =? [(((prev\_targ\ \&\ !transiting) => (click\_sfast \mid click\_fast \mid click\_medi \mid click\_slow \mid click\_sslow))\ \& ((prev\_deco\ \&\ !transiting) => (not\_click)))\ U\ (num\_sign = 13)]$$

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

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

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 The results for these properties are displayed in tables 3.30 and 3.31, together with their computing times.

The probability obtained for property 1 is around 49%. This complies with the experimental results in which patients succeeding on a high number of targets were common observations.

Properties 5 and 6 check the probabilities for a 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 4.67×10^{-1}, 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 1.22×10^{-1}. The probabilities to perform a correct action before or after the thirteenth signal show a big difference between the two versions. Given that after this signal there are 27 signals left, the chances to give good answers after signal 13 are significantly lower than before.

Property	Result	Time(seconds)
"good_on_target"	29.30	0.045
"good_on_decoy"	7.99	0.058
"good_on_signal"	37.30	0.044
"bad_on_target"	0.69	0.057
"bad_on_decoy"	2.00	0.047
"bad_on_signal"	2.69	0.044

TABLE 3.31: Results of property 7 for various rewards.

Property 7 is reward-based and gives an idea of the overall performance of a patient during the game. It 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 3.31, together with their computing times. The model checker takes less than 0.1 second to give the result of these properties. The *"good_on_signal"* reward shows that the average amount of good answers for a patient represented by this model is 37.3 out of the 40 signals, complying with our experimental results.

Evolution of Probabilities and Cumulative Rewards The last properties of this section corresponds to property 8 and 9 of section 2.6.2 in chapter 2. This properties uses variables as well as the PRISM "cumulative reward" facilities to track how the model probabilities behave and how the model accumulates rewards over time.

The first three diagrams present the evolution of the probabilities during the game. Figures 3.18, 3.19, and 3.20 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 3.29.

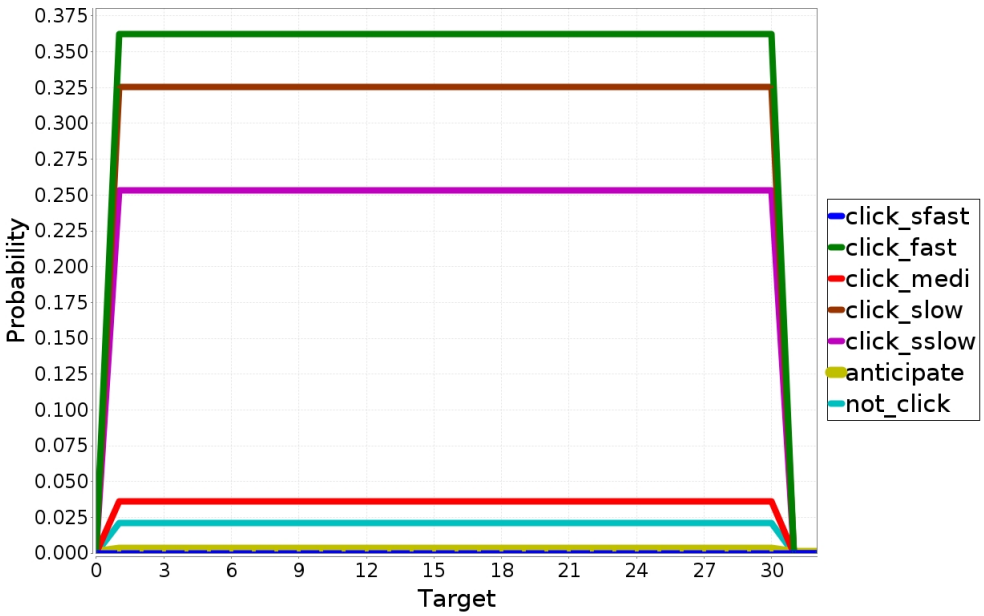


FIGURE 3.18: Probability to perform an action for a specific target.

Figure 3.18 shows the results for property 8 with *num_act* equal to *num_act_targ*, *prev_signal* equal to *prev_targ* and with *action* being a Boolean presented in the legend of the figure and in table 3.29. The probabilities displayed in this figure are a lot simpler than the probability shown in figure 2.18. This is intended as our data exploration did not show any obvious variations within a session. This remark can be extended to figure 3.19 and figure 3.20.

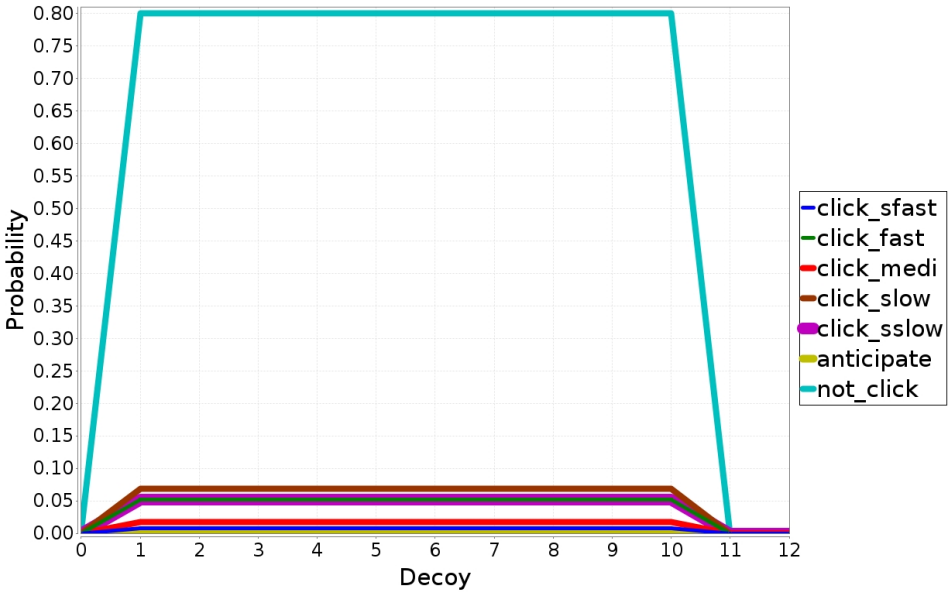


FIGURE 3.19: Probability to perform an action for a specific decoy.

Figure 3.19 shows the results for property 8 with *num_act* as *num_act_deco* and *prev_signal* as *prev_deco*.

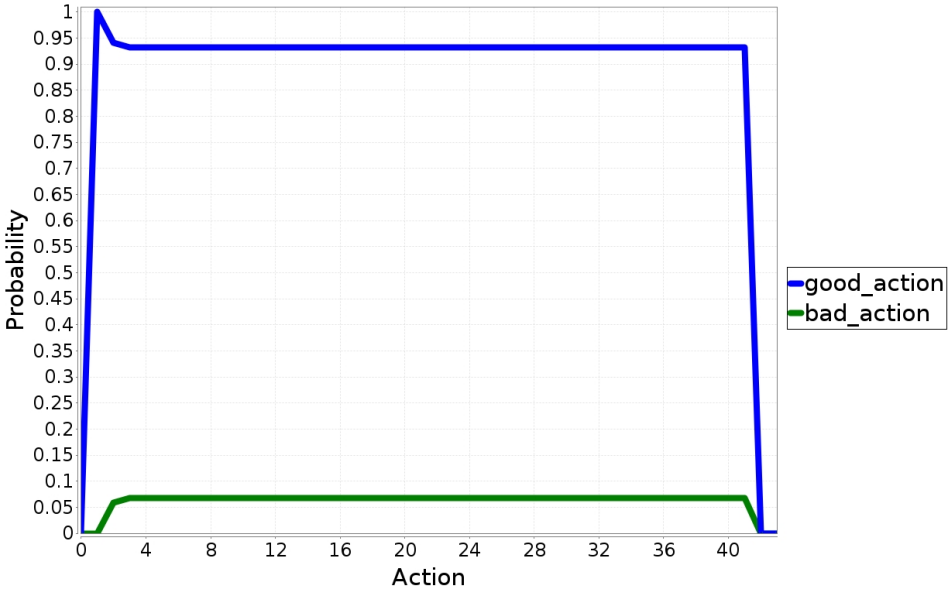


FIGURE 3.20: Probability to perform a good or a bad action for each instant when an action is expected from the patient. Here, the action number 1 on the horizontal axis is the one recorded before the occurrence of the first signal.

Figure 3.20 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. A small peak at the first action can be seen reaching 100%. This first action is the one made before the arrival of the first accounted game signal for which no patients had any error.

The next figures 3.21 and 3.22 present the results for the following cumulative reward property:

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

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

We applied the same property to other rewards: *"good_on_target"*, *"good_on_decoy"*, *"good_on_signal"* (the sum of the previous ones), *"bad_on_target"*, *"bad_on_decoy"*, *"bad_on_signal"* (the sum of the previous ones), *"bad_tar_no_dec"*, and *"bad_tar_aft_dec"* that are sub-parts of *bad_on_target*. The reward *"bad_tar_no_dec"* increments itself for bad actions on a target that was not directly preceded by any decoys and *"bad_tar_aft_dec"* increments itself for bad actions on a target that was directly preceded by a decoy.

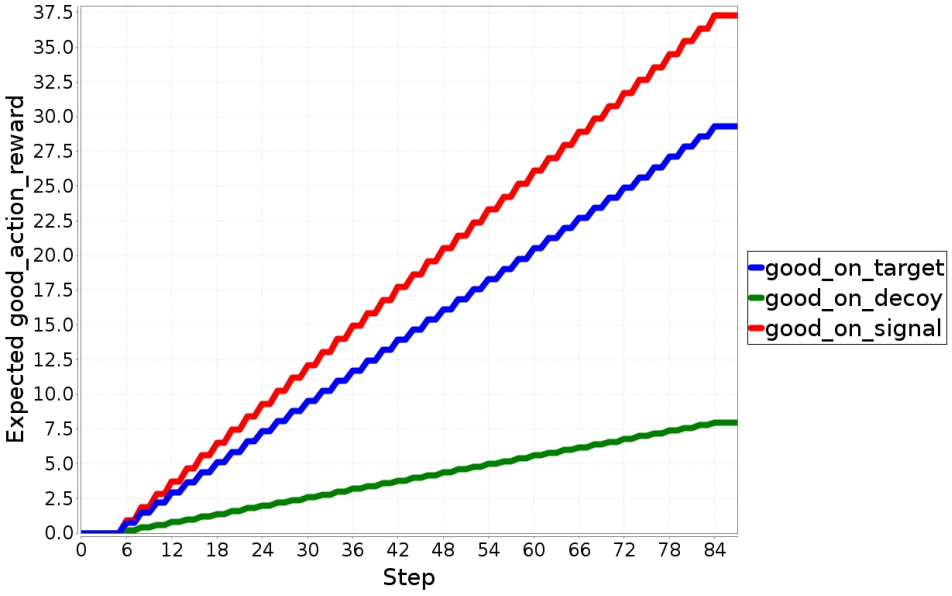


FIGURE 3.21: Average model checking results for rewards related to good actions.

The two figures 3.21 and 3.22 display "stair-like" diagrams for the same reasons stated for figures 2.21 and 3.22. The accumulation of reward are following a linear increase which was to be expected with the new set of probabilities. Figure 3.22 displays the accumulation of the rewards *"bad_tar_aft_dec"* and *"bad_tar_no_dec"*. This accumulation complies with the observation that two thirds of error on targets were associated with the display of a decoy directly before the target.

3.4 Conclusion

Conducting a clinical protocol from design to data analysis is a challenging experience. The main constraint was to go through several ethical validation processes. Even though the protocol described in this chapter is perfectible, its results showed

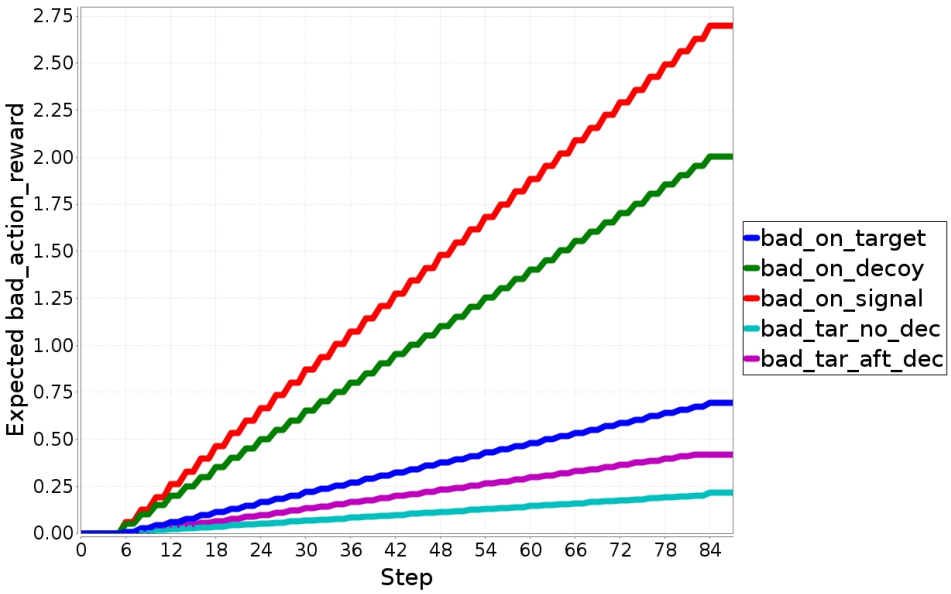


FIGURE 3.22: Average model checking results for rewards related to bad actions.

the potential of serious games in the screening of mild NCD patients. The selected games were able to display differences between mild NCD and SCD groups as well as correlations with classical neurocognitive tests. These results were used to redesign and manually calibrate the Code game and the Inhibitory Control game models. This first calibration experience is encouraging as it led to a better representation of the behavior variants and make the models effective. It can be noted that the calibration can result in models far less complex than originally expected as seen in the case of the Inhibitory Control game. This simplification encourages us to tackle games associated with more complex behavior to model. However, the Inhibitory Control game would definitely benefit from a second improved protocol to assess its capacity to evaluate the inhibitory control function of the player. Moreover, the calibration of the models could go further to perfectly match the observed data. To do so, we will consider the possibility to use parameter synthesis (e.g., PROPhESY [40] which accepts PRISM format as input) or other classical optimization methods [58].

One of the natural continuation of this work is to study the potential link between these activity models and a brain model focused on a given cognitive function. To this end, the next chapter targets the inhibitory control function through formal modeling in an attempt to explore the relations of the underlying biological inhibition mechanism and the observed behavior during the Inhibitory Control game.

Chapter 4

Model of the Inhibitory Control Circuit in the Brain

This chapter proposes a model of the neural network governing the inhibitory control function and studies some of its dynamic properties. The material of this chapter comes mostly from the internship work of Benjamin Lapijover [1]. The internship aimed to artificially represent the behavior of inhibitory control in humans through a probabilistic model. To do this we modeled the different structures implied in the inhibitory control loop thanks to a probabilistic discrete Markov chain model implementing *Leaky Integrate and Fire* artificial neurons. The objective is to compare this model with the activity model of the Inhibitory Control game and to explore their relationships.

4.1 Cognitive Functions and Brain Structures

Cognitive functions is a broad designation that covers brain processes necessary in the acquisition and the treatment of information and in reasoning. They include domains such as learning, awareness, and decision making [75]. The concept of cognitive functions is the base of many existing models which aim to deepen the current knowledge on the brain mechanisms. For instance, Hopfield neural networks represent the associative memory [62]. More recently, artificial neuron models have been used to understand the relations between the brain waves and cognitive functions such as working memory or executive control [112].

In the field of neuropsychology, a daily activity of practitionerss is to assess these cognitive functions with pen and paper tests. One of the main challenges of neurocognitive science is to build a better understanding of the cognitive functions and of their interactions. This knowledge may lead to improve both the clinical assessment methods and the therapies for a given disorder.

In this work, we focus on the inhibitory control function. This choice is motivated by three reasons:

- this function has been studied for a long time and several models already exist [121];
- this cognitive function can be evaluated with well established tests (e.g., go/no -go task) for patients with age related diseases;
- the inhibitory control function is managed by a fairly restricted amount of brain structures, the basal ganglia, which makes it easier to model, compared

[1] supervised by Dr. Elisabetta De Maria and co-tutored by the author

to attention (as in the Code game) or memory (as in Recognition game) functions;

- other well studied and modeled brain processes such as the "Incentive Salience" [18, 129] are not the most suited for age related diseases except in some particular cases.

The following sections detail the brain structures and the diseases associated with inhibitory control.

4.1.1 Biological Neuron

Biological neurons (see figure 4.1) allow inter cellular communication via electrical signals. They can have thousands of ramifications called dendrites that receive nerve (sensory and motor) information, named *afferent signals*. On the other hand the axon is usually unique and sends information named *efferent signals* to other cells.

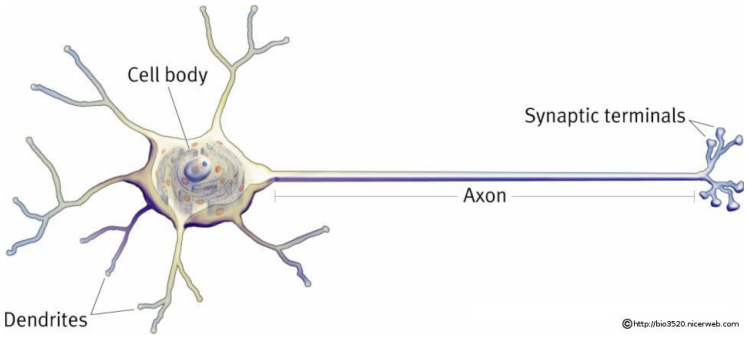


FIGURE 4.1: Biological neuron.

Neurons are specialized in the conduction of information and their size varies. For instance, brain inter neurons that are central nodes of neural circuits (but not direct motor neurons, nor sensory ones) are a few millimeters long. On the other hand, central nervous system projection neurons that carry information from a part of the brain to another, or from the spinal cord to other organs, may be up to one meter long. The *action potential* that runs through the axon is a nerve impulse caused by the difference in ion concentration between the inside and the outside of the neuron. When a neuron does not receive any signal, the membrane maintains a resting potential generated by the ionic current at rest. The resting potential is caused by the leaking channels which constantly balance the concentration of the potassium ions on both sides of the membrane. Nerve impulses modulate the membrane potential and if this potential exceeds the threshold of excitability, it triggers in turn an *action potential* or *spike*. This action potential is defined as a quick rise and fall of the membrane potential. When triggered, the spike travels along the neuron axon to reach the synaptic terminals [104]. Depending on the neuron, the spike either directly travels to the dendrites of the connected neuron or triggers the release of molecules, named neurotransmitters. These neurotransmitters reach receptors on the dendrite side of the synaptic connection. In both cases, the receiving neuron follows the same cycle as the neuron that sent a spike.

4.1.2 Inhibitory Control Circuit

According to the literature [8, 9, 68], the brain regions involved in the inhibitory control cognitive function are the cortex, the basal ganglia, and the thalamus. Together,

these anatomical structures make it possible to temporarily stop the action of the motor cortex and therefore to inhibit an action initially planned. At the attentional level the role of the inhibitory control circuit is to ignore irrelevant stimuli. A deficit in this circuit can cause cognitive impairment [23] and there exist neuropsychological tests to assess it.

The basal ganglia (see figure 4.2) are considered as motor structures that allow movements to start and stop, although they also have a role in non motor functions. Some researches showed that a dysfunction of the inhibitory control circuit might be involved in disorders other than the inhibitory control ones such as obsessive-compulsive disorders [51, 41]. In this thesis, we only focus on the basal ganglia and their role in the inhibitory control.

Basal ganglia are essentially a part of the *motor loop* and thus of the inhibitory control circuit. They contain several anatomical structures like the *striatum* (STr) which is itself composed of the putamen and caudate nucleus. These two latter structures are often distinct in the circuit, but in our formal model we only represent the STr. This simplification is advisable to limit the size and complexity of the model and acceptable because the striatum can be considered as a whole functional zone by itself [72].

Basal ganglia also contain:

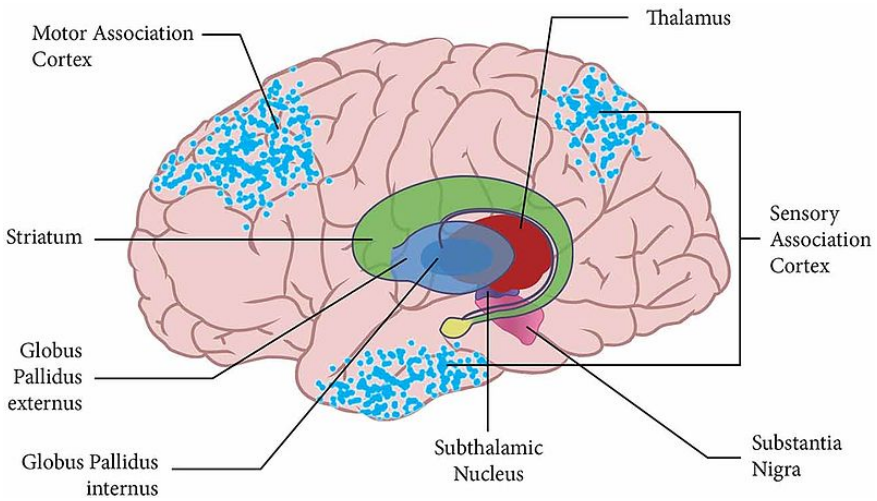
- the *globus pallidus* which is separated into two structures:
  - the *outer segment* (Gpe),
  - and the *inner segment* (Gpi) which is related to the *substantia nigra pars reticulata* (SNpr) (in our model we will represent them together);
- the *subthalamic nucleus* (STN);
- the *substantia nigra pars compacta* (SNpc).

The *cortex* (Cx) and the *thalamus* (Th) are also part of the inhibitory control loop but not part of the basal ganglia [104].

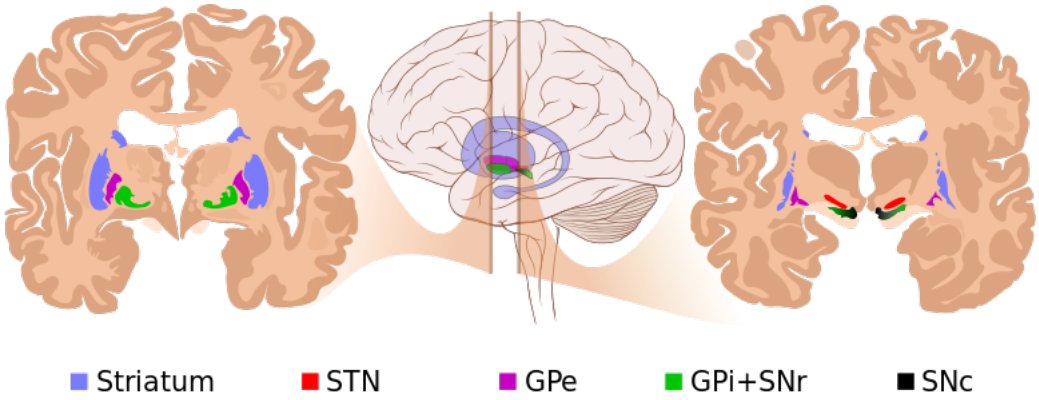
There are two distinct pathways in inhibitory control. First, the direct path starts from the STr which receives input signals (named afferent signals at the level of neurons [85]) from several components of the cortex. When excited, STr inhibits the Gpi/SNpr complex, leading to a disinhibition of the thalamus (Th) that communicates with the motor cortex of the frontal lobe. The effect of this pathway is the "release" [53] of the desired physical movement.

Second, the indirect path allows suppressing unwanted movements. It starts from both STr and STN. In this path, STr inhibits GPe leading to the disinhibition of STN which establishes connections onto the GPi/SNpr complex via excitatory fibers. When receiving this excitation signal, the GPi/SNpr complex inhibits Th [53]. At the same time, STN receives powerful afferent signals from the cortex making it possible to modulate the release activity of the direct pathway [104]. More precisely:

- In the direct pathway, corresponding to the release of the programmed motor action, specialized STr neurons target specific neurons of the Gpi/SNpr complex neurons with "slow", "focused", and "convergent" inhibitory inputs [95]. These inputs trigger the disinhibition of the Th zone controlling the expression of the desired motor program.
- Whereas in the indirect pathway, STN specialized neurons target the Gpi/SNpr complex with "fast" and "divergent" excitatory inputs [95]. These inputs



(A) Localization of basal ganglia in the brain. Illustration from B. Colder work [35].



(B) Coronal section of the brain highlighting basal ganglia. Illustration from A. Gillies, M. Häggström and P. J. Lynch work.

FIGURE 4.2: Basal ganglia location and its components.

trigger the inhibition of Th in a way that removes all the unwanted motor programs and permits the movement initiated by the disinhibition done by the direct pathway.

The decline of inhibitory control efficiency in aging subjects is due to anatomical and functional changes in prefrontal/frontal regions [63]. However, there are differences of loss between healthy and pathological aging. It is one of our goals to differentiate these two conditions. It has been shown [36] that in neurodegenerative diseases, such as Parkinson's disease and Alzheimer's disease with Parkinsonian syndrome, there is a degeneration of the neurons of the substantia nigra pars compacta (SNpc). The dopaminergic influx originating from the SNpc and targeting STr is considerably reduced. STr is then less inhibited which consequently reduces the inhibitory emission of the basal ganglia and removes the inhibition of the thalamus and therefore the motor inhibition.

4.1.3 Alzheimer's and Parkinson's Diseases

As said in the introduction 1, the World Health Organization categorizes the Alzheimer's disease as the major cause of dementia (60-70% of the cases). The Parkinson's disease is another frequent dementia related disease that affects nearly 2% of seniors

over the age of 65 according to the Alzheimer's Association. These neurodegenerative diseases impact both memory and behavioral inhibition. The patients are no longer able to suppress a non-dominant motor response and the executive function of the patients is affected at an early stage of the disease. These diseases modify patients functional and cognitive behavior and may alter their social behavior in the long term.

The Alzheimer's disease has some physiopathological specificities [31]:

- the presence of tau protein and beta amyloid aggregates are characteristic signs of Alzheimer's disease;
  - the cytoskeleton of neuron cells becomes unstable;
  - in the extracellular environment, the beta amyloid peptides aggregate abnormally into senile plaques;
- the porosity and loss of structure of the neurons progressively lead to a loss of function and ultimately to the death of the neurons;
- the brain hippocampus is the first anatomical structure affected by Alzheimer's disease.

The Parkinson's disease presents characteristic motor disorders. It is defined by symptoms such as tremors, hypertonia (stiffness of the muscles), and akinesia. Akinesia is the reduction of voluntary and automatic motor skills which decreases the initiation of movement [41]. Neuronal degeneration is characterized by Lewy bodies which are neuronal inclusions made up of aggregates of α-synuclein and ubiquitin proteins. Motor symptoms occur rather late in the disease, the main cause being the dysregulation of the dopaminergic system. The disease actually begins with the degeneration of the vagus nerve which subsequently causes the death of dopaminergic neurons in the basal ganglia [23]. Parkinson's disease is often at the center of studies on the inhibitory control such as the work of [96] in which deep brain stimulation are used to explore the role of STN in this cognitive function.

The death of nerve cells for both diseases begins at the periphery of the basal ganglia which are at the center of inhibitory control. The inhibition process is generally slower with age, but in patients with an early stage of cognitive impairment (such as the onset of undiagnosed Alzheimer's or Parkinson's diseases) the process is even slower.

One type of task that can measure cognitive motor skills in patients with such diseases is the *go/no-go task* which consists in suppressing an initially programmed intentional movement. This task has not yet been able to differentiate an Alzheimer population from a healthy one [51], but at the motor level, the *antisaccade task*, which is close to the go/no-go one, made it possible to find alterations between the two populations. In this task, it is sufficient to look at a particular point without eye jerking in the presence of distractors [36]. In our case, the Inhibitory Control game described in chapter 2 is inspired by such tasks and thus targets inhibitory control capacities. This game has been tested by people with cognitive complaints or with minor cognitive impairment (as shown in chapter 3 section 3.2.5).

4.2 Inhibitory Control Computational Modeling

4.2.1 Inhibitory Control Models

Several explicative models of the inhibitory control use the go/no-go task to study and describe its underlying mechanisms. As a matter of example, the pioneer horse race model [84] inspired several models such as the interactive horse race model [22]. These models are reviewed in [121]; they describe the inhibitory control function as a competition between a "go" and a "stop" process in the brain. This modeling approach gave many insights on the mechanisms of the inhibitory control [111]. With the rise of computational simulation techniques, the past decades have seen the development of several models based on neural networks for the inhibitory control loop with respect to neuroanatomy [114]. Among these models, some focus only on the inhibitory control and the basal ganglia (e.g., [14]) while others attempt to include this loop in bigger brain circuits (e.g., [97]). In both previous examples, authors tried to model healthy, pathological, and under medications behaviors. Note that modeling the treatment is not part of the scope of this PhD thesis.

The model presented in this thesis finds its inspiration in the work of [126] the authors of which attempted to model the inhibitory control function with a set of (heavy) biological "Leaky Integrate and Fire" neuron networks representing each brain structure implied in this function. This type of neuron model appears as the most efficient for mathematical analysis (see next section).

4.2.2 Artificial Neural Network Models

The aim of formal biological neuron modeling is to obtain models that can be both analyzed mathematically and sufficiently realistic to represent the essentials of neural processing.

The modeling of neural networks is often functionally classified into three generations [86]. The first generation is based on the "formal neuron" of McCulloch and Pitts [93]. It has several binary inputs associated with weights and the sum of these weights is compared to a threshold value determining activation or non-activation of the neuron output [102].

The second generation of models of which the most representative example is the *multi-layer perceptron* of Cybenko [37], exploits real valued activation functions. These real values represent the frequency at which a neuron emits spikes (firing rate). This kind of artificial neurons is still widely used in supervised and unsupervised learning algorithms.

The third generation of neural network models, also called *spiking neural networks* [101], are characterized by the relevance of time aspects. Precise spike firing times are taken into account. Moreover, past input spikes are also considered in the computation of the membrane potential (temporal summation).

Among the different models of the third generation, we chose the *Leaky Integrate and Fire* model [26], which is a good compromise between biological fidelity and computational efficiency for mathematical analysis [66].

Leaky Integrate and Fire Discrete Model

According to the Leaky Integrate and Fire (LI&F) model, the membrane of a neuron can be represented as an electronic circuit as proposed in the pioneering work of Lapicque [81, 26].

In this model the intensity of the action potential is neglected, but the instant of its occurrence plays an important role [38, 101]. We considered a discrete version of the model where the membrane potential u at time t is defined by the following equation:

$$u(t) = \begin{cases} \sum_{i=1}^{n} = w_i \cdot x_i(t) + r \cdot u(t-1) \; if \; u(t-1) < Tau \\ \sum_{i=1}^{n} = w_i \cdot x_i(t) \; otherwise \end{cases}$$

In the equation:

- $x_i(t) \in \{0,1\}$ is the signal received at time t by the neuron through its i^{th} input synapse;
- w_i is the weight associated with the i^{th} input synapse to assign a strength to the input;
- r is the leak factor caused by ion exchanges through the membrane;
- Tau is the excitability threshold beyond which the neuron emits an action potential (a spike).

The threshold for the emission of an action potential (spike) is compared with the membrane potential at time t. If the membrane potential exceeds the threshold, an action potential is emitted and the membrane potential is reset to zero. The neuron output function $s(t) \in \{0,1\}$ is therefore defined by

$$s(t) = \begin{cases} 1 \; if \; u(t) \geq Tau \\ 0 \; otherwise \end{cases}$$

Generalization to Neuron Boxes

In this thesis, the goal is to model the interactions of several structures of the brain (which are made of thousands of neurons) while keeping the model tractable for model checking. As modeling each and every neurons would make the model very difficult to check, we introduce a generalization of the "LI&F" neuron to neuron boxes. This means that we modified the equation of the LI&F neuron so that it may represent several neurons, each one producing its own output at a given instant.

The generalization consists in representing each of the anatomical structures of the inhibitory control circuit by a box containing ten neurons of LI&F type. This number is not proportional to the actual number of neurons found in the brain structures nor does it follow the proportions found in the model of Wei and Wang [126]. This simplification is a trade off between biological accuracy and computation capacity. Indeed, a network of ten neurons cannot mimic the behavior of complex and heavy networks with hundreds or thousands of neurons. However, it allows the observation of a "firing ratio" (number of firing neurons out of the ten ones) for each time step. Moreover, the generalization allows to model these ten neurons with only one entity called box that directly computes the firing ratio at each time step. Thus the choice of boxes makes it possible to have a behavior relatively close to a small network of neurons, without requiring a lot of computing power.

To take the difference of sizes between biological structures into account, the weight of each connection of the model was first set to a value proportional to

- the weight used for the same connection in the model of Wei and Wang;

- the number of neurons connecting two structures in the model of Wei and Wang.

The formula defining the dynamic of each box is the following:

$$U(t) = \sum_{i=1}^{n} w_i \cdot X_i(t) + r \cdot U(t-1) \cdot \left(\frac{N - S(t-1)}{N}\right)$$

This formula is close to the usual LI&F neuron dynamic formula [66] except that here, the Boolean x_i is replaced by the integer X_i ranging from 0 to 10 to mimic the possible 10 neuron inputs from another box. Indeed, another simplification of this model is that when a biological connection exists between two boxes A and B, all neurons of box A are considered to be connected to all neurons of box B. Thus, the new formula does not take Boolean inputs anymore, but input firing ratio. This simplification led to the insertion of a new variable: N. This variable represents the number of neurons in the whole box ($N = 10$ in the presented model). As neurons are far less sensitive to stimuli after emitting a spike due to the "Refractory Periods" [104], the last term of the formula was introduced. In this term, $S(t-1)$ represents the number of neurons which discharged at the previous time step. More precisely, $s(t)$ is defined with the following formula:

$$S(t) = \frac{U(t)}{Tau}, \text{ where } 0 \leq S(t) \leq N$$

Thus, to track the activity of a box in the model, one should check the values of $S(t)$ of this box in the time window of interest.

Biological inputs from the cortex are irregular, thus, to get as close as possible to this irregularity, the input spikes are modeled by a Poisson law [61]. In their work, Wei and Wang [126] modeled these inputs as Poisson spike trains. Following this line, the computation of the activation probabilities of the cortex and SNpc neurons in our model follows the Poisson function with parameter lambda below:

$$P(k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

where k is the number of cortex and SNpc neurons that send a spike.

4.3 Model of the Basal Ganglia and Validation

The first task was to provide a formal model of the main interactions between the different basal ganglia nuclei. As in chapter 2, we rely on a probabilistic modeling approach based on the use of discrete-time Markov chains. We developed a PRISM model to artificially mimic the behavior of human inhibitory control through a probabilistic model. The purpose of this model is to be coupled to the activity model associated with the game evaluating this function in order to explore modifications in the neural network that can generate a patient behavior characteristic of neurocognitive disorders.

Second, we automatically tested probabilistic temporal properties of this model thanks to model-checking (in PRISM and Storm) to explore potential sources of patho -logical behavior in the inhibitory control circuit.

4.3.1 Model Overview

As said in section 4.2.1, neuropsychologists and neurobiologists have theorized several models of the functioning of the basal ganglia which have already been integrated into software systems. The model implemented in this thesis is inspired by the basal ganglia model of [126]. In this paper the authors developed a "physiologically based network model of spiking neurons with feedback mechanisms" to study inhibitory control in particular. They proposed a neural network model of LI&F neurons for the basal ganglia following the architecture shown in figure 4.3.

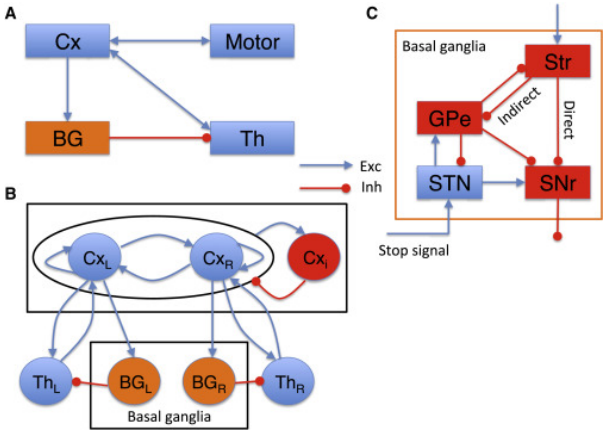


FIGURE 4.3: Graph describing the neural network of [126].

In the work of Wei and Wang, each structure of the basal ganglia in figure 4.3 was modeled using a different amount of neurons (e.g., 2500 for GPe and 250 for Str) with either all to all connections or sparse connections (e.g., STr neurons are all to all connected, whereas GPe has sparse connections with STN and itself). The thalamus (Th) has also its own neuron network. The cortex signals being the inputs of this model, they were modeled using a Poisson law that computes the input spike trains. The authors represented the basal ganglia on both sides of the brain. With this model, the authors obtained diagrams of activity for each structure to visualize concepts such as the importance of some specific connections in inhibitory control.

Our approach is different as we do not rely on simulation only and we represent basal ganglia on one side of the brain only but we kept the same division into different boxes. As previously introduced, our work relies principally on model checking techniques. The advantage of model checking is that it allows the automated exploration of each possible state of a model to ensure the validation or the rejection of a given property. However, this automated computation constrains experimenters to implement models with lower complexity compared to simulation methods. Thus, our model follows the architecture shown in figure 4.3 using LI&F neuron boxes and other adaptations, resulting in the graph of figure 4.4.

The implementation followed several steps:

1. build a model for a single box of neurons (with formula 4.2.2 from section 4.2.2);
2. add conditions to limit the output firing ratio of boxes to values between 0 and 10 as each box represents 10 neurons;
3. build the model with all the biological structures represented as blue rectangles in the basal ganglia square of figure 4.4 as well as the thalamus and introduce excitatory and inhibitory connections between them as symbolized in figure 4.4;

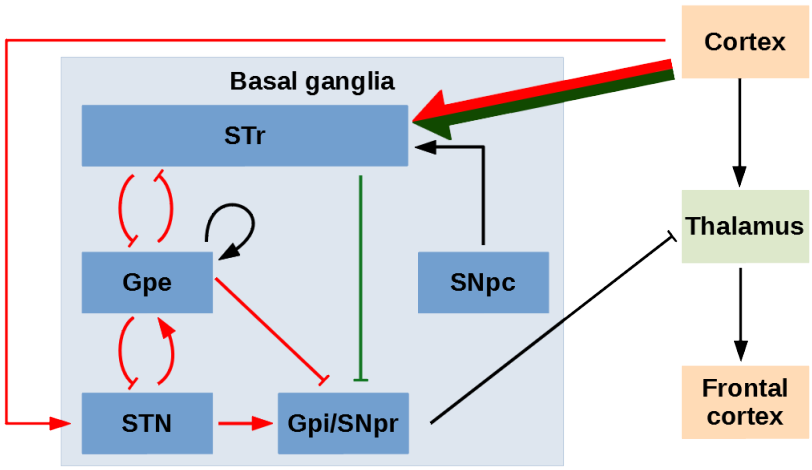


FIGURE 4.4: Inhibitory control circuit diagram inspired by the work of Wei et al. In green the direct pathway, in red the indirect one. A classic arrow corresponds to an excitation, a flat-tipped arrow to an inhibition.

4. add connections from the cortex as inputs; these inputs follow a Poisson law determining the number of spikes sent to the STr and Th boxes (section 4.2.2);

5. implement an additional *Delay* box between the STN box and the SNpr one allowing a discrete functioning of the loop;

6. integrate a SNpc box to differentiate a healthy brain from a brain with a degenerated SNpc. For this we defined 2 different formulas (see PRISM code in section 4.3.2):

   - the first takes into account both the inputs of the SNpc and those of the cortex;
   - the second takes into account only the inputs of the cortex.

We considered that a pathological brain has a 30% probability to follow the first formula and a 70% probability to take into account the second. These numbers were chosen in accordance with Cheng's paper [29] which shows that the death of dopaminergic neurons reaches 70% in the later phases of Parkinson's disease. Dopaminergic neurons are not considered to be dead in a healthy brain and therefore it will always follow the first formula (see figure4.5).

For each of the intermediate steps of the development, it was necessary to understand the activity of the neural network. To do so PCTL properties on the activation and inhibition of the boxes were tested with both PRISM and Storm. To check these properties, PRISM required the use of the "explicit" engine and Storm handled the properties computation with its default engine (the "sparse" one).

4.3.2 PRISM Model Implementation

This section focuses on the implementation of the model with the PRISM language and its verification with PRISM and Storm frameworks.

Each neuron box is implemented by a single PRISM module. All the neuron box modules have a common structure with two variables (for global membrane potential and firing ratio) and one set of *guards* and *updates*.

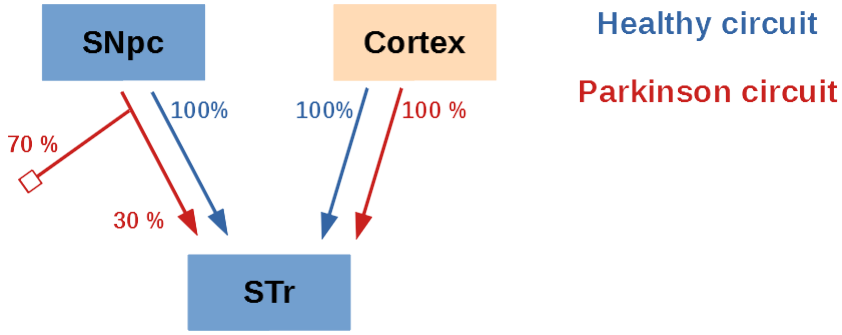


FIGURE 4.5: Model differences between a healthy brain and a brain with degenerated SNpc. The healthy brain always takes all inputs from the SNpc and the cortex. The pathological circuit promotes the inputs from the cortex and considers the SNpc inputs only 30% of the time.

Example of a Simple Model

To illustrate this implementation, the simple case of two boxes, each with 10 neurons, connected together is best suited. Figure 4.6 below presents the PRISM code of these two boxes.

Between lines 13 and 20 are the constants used to compute the state of a given neuron box. The *tau* constant corresponds to the spike activation threshold of a classical LI&F neuron and has a value of 80[2]. The *r* constant is the leak factor which decreases the impact of the past inputs with time. Finally the constants between line 17 and 20 are the weights associated with each connection between the boxes.

Between lines 28 and 44, one can find the two modules representing the neuron boxes. In these modules the variables are:

- $pb1$ and $pb2$ the global membrane potential of B1 and B2 respectively;
- $nb1$ and $nb2$ the number of output spikes (or "firing ratio") of B1 and B2 respectively.

The values of $pb1$ and $pb2$ are computed with formulas $b1$ and $b2$ at lines 23 and 26. In the update command of $B1$ (resp. $B2$) box module, the value of $nb1$ (resp. $nb2$) is computed from the value of $b1$ (resp. $b2$). This computation aims at determining the firing ratio, thus, the formula divides the global membrane potential $b1$ (resp. $b2$) by the threshold tau. For example, if $b1 = 400$, $nb1$ is set to $400/tau = 5$.

Lines 3 to 11 define a third module named *Entry*. This module is the input module of the model and represents the cortex inputs of the neural networks. It has one variable, $t1$, that represents the number of input spikes that are sent to the neuron boxes and ranges from 1 to 10. The only set of guards and updates of this module was generated thanks to a Python script that implements a Poisson law for the value of $t1$. For example, the probability for the *Entry* module to send one spike ($t1 = 1$) is 0.000072991952613. This module gives the inputs to $B1$ as it can be seen in formula $b1$.

In this model, module $B1$ receives entries from both the *Entry* and the $B2$ modules whereas $B2$ receives entries only from $B1$.

[2]Since the resting membrane potential of a neuron is −10mV and the biological threshold is 70mV, we chose a threshold value of 80 because the resting potential in our model is 0.

```
dtmc

module Entry
//Input spike that follow a Poissonian law with 7 as parameter
//Prism does not support more than 15 digit after the dot for this command
t1:[0..10];
[to] true -> 0.000072991952613:(t1'=1) + 0.003437086558390:(t1'=2)
+ 0.021604031452484:(t1'=3) + 0.059540362609726:(t1'=4) + 0.104444862957054:(t1'=5)
+ 0.137676978041126:(t1'=6) + 0.149002779674338:(t1'=7) + 0.139586531950597:(t1'=8)
+ 0.117116124452909:(t1'=9) + 0.26751825035076304:(t1'=10);
endmodule

//tau => Spike activition threshold
const tau=80;
//r => Leak factor
const double r=0.5;
//Input weight of each box
const entry=80;
const b2b1=-50;
const b1b2=90;

//Global membrane potential formula for B1
formula b1=floor((entry*t1)+(b2b1*nb2)+(r*pb1*(1-(floor(pb1/tau)/10))));

//Global membrane potential formula for B2
formula b2=floor((b1b2*nb1)+(r*pb2*(1-(floor(pb2/tau)/10))));

module B1
pb1:[0..800];//Global membrane potential of B1 at step t, maximum value is 800
nb1:[0..10]; //Total number of output spikes of B1 at step t, maximum value is 10
//If pb1 goes under 0, both nb1 and pb1 are resetted to 0
//Or if pb1>tau*10, nb1 is set to 10 and pb1 to 800
//Thus, the Prism set of guard/update is the following:
[to] true-> 1:(nb1'=b1<0?0:b1>800?10:floor(b1/tau)) & (pb1'=b1<0?0:b1>800?800:b1);
endmodule

module B2
pb2:[0..800];//Global membrane potential of B2 at step t, maximum value is 800
nb2:[0..10]; //Total number of output spikes of B2 at step t, maximum value is 10
//If pb2 goes under 0, both nb2 and pb2 are resetted to 0
//Or if pb2>tau*10, nb2 is set to 10 and pb2 to 800
//Thus, the Prism set of guard/update is the following:
[to] true-> 1:(nb2'=b2<0?0:b2>800?10:floor(b2/tau)) & (pb2'=b2<0?0:b2>800?800:b2);
endmodule

```

FIGURE 4.6: Example of a PRISM code with two boxes.

Finally, one can see that the three modules are synchronized by the PRISM synchronization mechanism using a common transition label: here all the modules share the same label ($[to]$).

Healthy Inhibitory Control Model

The implementation in PRISM of a healthy inhibitory control follows the architecture described above (see figure 4.6), but with six modules, one for each of *Entry* (partial representation of the cortex), STr, GPe, STN, SNpr (representing the Gpi/SNpr complex which can be considered as a single functional structure), and Th. These boxes have an associated formula for their global membrane potential and constants for their connection weights except module *Entry* that has the behavior described in section 4.3.2. These connections are organized following the architecture of the basal ganglia shown in figure 4.4. However, we add two modules: *Delay* and *Inhibitor*. The *Delay* module has a behavior close to a neuron box but is far simpler as it sends the same amount of spikes that it receives. This supplementary module is necessary as our model is a discrete synchronous system. Indeed, in [126], the continuous

modeling allows the SNpr to receive signals from both GPe and STN simultaneously. To imitate this behavior, the *Delay* module delays the signal from STN and makes it reach SNpr at the same instant as the one from GPe. The *Inhibitor* module is used as a generator of no-go signals at regular intervals to simulate external inhibitory events. In the model, it sends "stop" signals to the *STN* module. For this purpose, the *STN* module has one more set of guards and updates to to take this "stop" signal into account. The *Inhibitor* module sends arbitrarily a "stop" signal every ten counts. To observe a successful inhibition, one has to check the output of module *Th*. As shown in figure 4.4, this module is the one receiving the final inhibition signal; hence, if its output firing ratio is low, it indicates a successful inhibition.

Parkinsonian Inhibitory Control Model

In an attempt to model the behavior observed in Parkinson's disease, we defined an alternative formula for the global membrane potential of STr (see figure 4.7).

```
//Global membrane potential formula for STr
formula STr_healthy=floor((-wsnpcstr*t1)+(wcxstr*t1)+(-wgpestr*n_GPe)
+(r*potentiel_STr*(1-(floor(potentiel_STr/tau)/10))));
formula STr_patho=floor((wcxstr*t1)+(-wgpestr*n_GPe)
+(r*potentiel_STr*(1-(floor(potentiel_STr/tau)/10))));
module STr
potential_STr:[0..800];
n_STr:[0..10];
[to] n_STr>=0 -> 0.3:(n_STr'=STr_healthy<0?0:STr_healthy>800?10:floor(STr_healthy/tau))
& (potential_STr'=STr_healthy<0?0:STr_healthy>800?800:STr_healthy)
+ 0.7:(n_STr'=STr_patho<0?0:STr_patho>800?10:floor(STr_patho/tau))
& (potential_STr'=STr_patho<0?0:STr_patho>800?800:STr_patho);
endmodule
```

FIGURE 4.7: Code excerpt for a brain with a early Parkinson syndrome.

The STr neuron box module is thus written with one more update in its set of guards and updates. It now has a probability of 0.9 to update its membrane potential with the "healthy" formula and a probability of 0.1 to update it with the "pathological" formula. These formulas and probabilities were extrapolated from Wei and Wang model.

4.3.3 Properties of Individual Boxes

The first step was to validate the different boxes individually. Each neuron box must respect the specifications stated in section 4.2.2. In particular, at the box level, the model must verify the conditions concerning the global membrane potential and the firing ratio: the maximum and minimum global membrane potential must not be exceeded; the maximum and minimum number of spikes must be respected; as long as the global membrane potential is not greater than or equal to a certain threshold, there should be no spike.

Thus, we verified that the following PCTL* properties which are invariants of the boxes, hold for all boxes. In these properties, X is a variable denoting the current box under test, n_X is the number of spikes emitted by box X, and $potential\_X$ is X global membrane potential. As a reminder, $P =?$ is the PCTL operator to compute a probability and G is the operator to verify that a PCTL formula is true for every state of a model.

Property 1. What is the probability for the number of spikes to always be greater than or equal to zero and less than or equal to the maximum value (10)?

$$P =?[G\ (n\_X \leq\ 10\ \&\ n\_X \geq 0)]$$

Property 2. What is the probability for the global membrane potential to always be greater than zero and less than $tau \times 10 = 800$?

$$P =?[G\ (potential\_X \leq 800\ \&\ potential\_X \geq 0)]$$

Property 3. What is the probability for the number of spikes to always be zero until the potential is equal to or exceeds $tau = 80$?

$$P =?[n\_X = 0\ U\ potential\_X \geq 80]$$

Both model checkers give a positive answer (P=1) for all three properties and all boxes. Results for box *STr* are shown in table 4.1. PRISM and Storm have comparable computation time except for property 3 for which Storm is a lot faster.

Property	PRISM		Storm	
	Result	Time (s)	Result	Time (s)
Property 1	1	0.357	1	0.489
Property 2	1	0.626	1	0.506
Property 3	1	1.028	1	0.000

TABLE 4.1: Results from property 1 to 3 for box *STr*

4.3.4 Properties about Box Synchronization

The next step is to validate the synchronization of the boxes. To best represent the inhibitory control circuit, the boxes must be connected together following figure 4.4 so that the model may respect the known properties about the connections of the corresponding biological structures in this circuit.

The first property checks that the boxes cannot all be off (i.e., that at least one box has firing neurons).

Property 4. What is the probability for all boxes to not have a single output spike after the first time step?

$$P =?[(X\ (G\ n\_STN = 0\& n\_GPe = 0\ \&\ n\_STr = 0\ \&\ n\_Delay = 0$$

$$\&\ n\_SNpr = 0\ \&\ n\_Th = 0))]$$

Here the expected right answer is negative (P=0).

The following properties verify that the boxes are working in cascade for the STN-Gpe-STr and STN-Delay-SNpr pathways as described in [126].

Property 5. What is the probability for STN to be activated (stop signal arrival), and, in the next instant, GPe is activated, and again in the next instant, the STr is inhibited?

$$P =?[(F\ n\_STN > 5\ \&\ (X\ n\_GPe > 5\ \&\ (X\ n\_STr < 5)))]$$

Property 6. What is the probability for STN to be activated (stop signal arrival), and at the next instant, the Delay box is also activated, and one more instant later, SNpr is activated at its turn?

$$P =?[(F\ n\_STN > 5\ \&\ (X\ n\_Delay > 5\ \&\ (X\ n\_SNpr > 5)))]$$

The next properties check the behavior of both the direct and indirect pathways from the instant the stop signal is sent; we arbitrarily fixed this instant to be the tenth one.

Property 7. What is the probability for the stop signal to rise at the tenth instant?

$$P =?[F = 10\ n\_STN > 3]$$

Property 8. What is the probability for GPe and the Delay box to be activated at the eleventh instant?

$$P =?[F = 11\ \ n\_GPe > 3\ \&\ n\_Delay > 3]$$

Property 9. What is the probability for STr to be inhibited and SNpr to be activated at the twelfth instant?

$$P =?[F = 12\ n\_STr < 5\ \&\ n\_SNpr > 3]$$

Finally, the last property checks that the GPe and Delay boxes do not activate until a stop signal occurs.

Property 10. What is the probability for Gpe and Delay boxes to not be activated before the stop signal arrives (at the tenth instant)?

$$P =?[G < 11\ n\_GPe = 0\ \&\ n\_Delay = 0]$$

PRISM and Storm give the expected valid answers, that is P=1 for all properties (except P=0 for 4). All results are displayed in table 4.2. Storm could not handle all properties; properties with nested PCTL operators and property 10 could not be verified. One can see that for the three properties verified with both PRISM and Storm, PRISM was ten times faster.

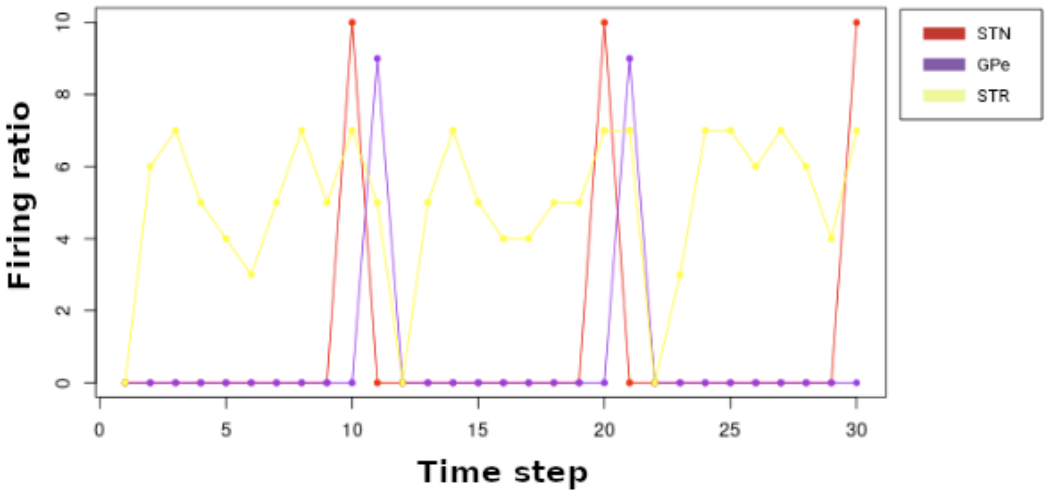
This verification shows that the model corresponds to our expectations. Indeed, the model shows the disinhibition and activation of Gpi/SNpr nuclei that trigger the inhibition of the thalamus and thus, the inhibition of an action. Moreover, the simulation graph in figure 4.8 (obtained with the "run experiment" tool of PRISM) showing spiking activity also confirms the inhibition of STr and Th following a stop signal. Though they are not at the same scale level at all, these activities can approximate the simulation results displayed in [126].

4.3.5 Property related to Thalamus Inhibition

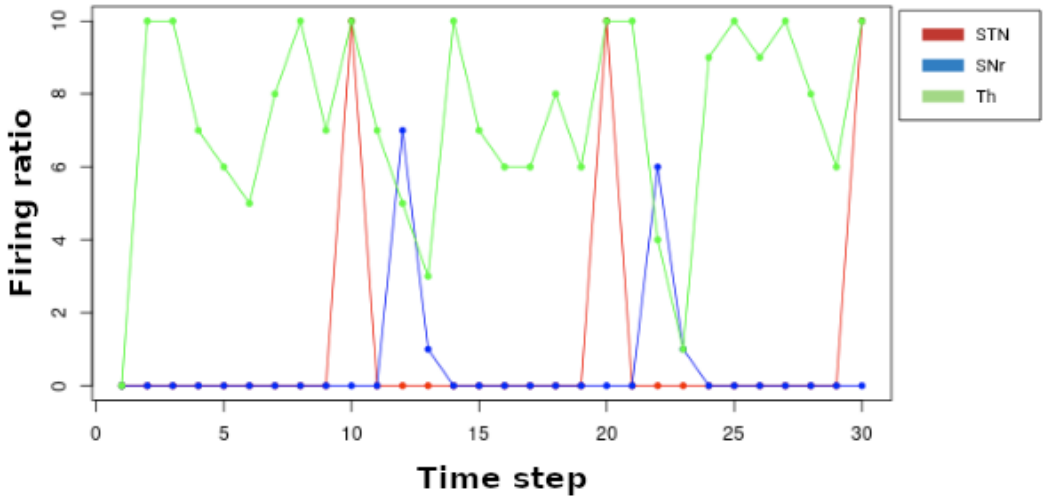
To deduce the probability that a movement is actually inhibited, we defined a property evaluating the probability to observe a low number of firing neurons in the thalamus box. We arbitrarily consider that this latter box is inhibited when it releases less than 4 spikes (less than 40% of the maximal "firing ratio" which is 10 for the implemented boxes). In PCTL* this property is written:

Property	PRISM		Storm	
	Result	Time (s)	Result	Time (s)
Property 4	0	5.189	−−	−−
Property 5	1	10.169	−−	−−
Property 6	1	9.667	−−	−−
Property 7	1	0.240	1	2.094
Property 8	1	0.289	1	2.160
Property 9	1	0.308	1	2.160
Property 10	1	0.298	−−	−−

TABLE 4.2: Results from property 4 to 10



(A) STN-GPe-STr path. The stop signal arrives at the STN, which permits the activation of GPe and then the inhibition of STr.



(B) STN-SNpr-Th path. The stop signal reaches the STN, then enables the activation of the SNpr (thanks to the inhibition of the STr at the same time ($t = 13$)), and finally the inhibition of the thalamus (and therefore of the behavioral response).

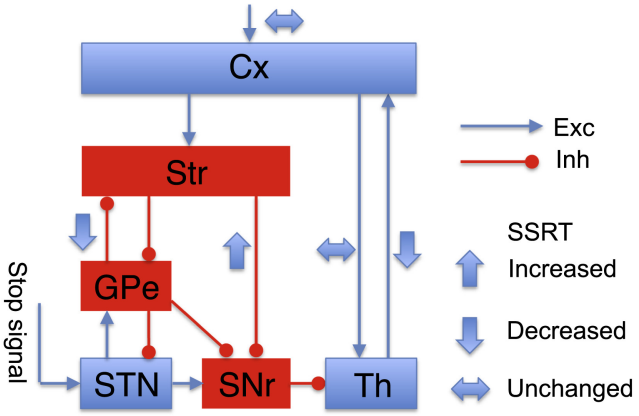
FIGURE 4.8: Paths allowing thalamus inhibition. Thanks to the different paths the arrival of a stop signal at $t = 10$ leads to the thalamus inhibition and therefore to the inhibition of the behavioral response at $t = 13$.

Property 11. What is the long run probability for Th to emit less than 3 spikes at once?

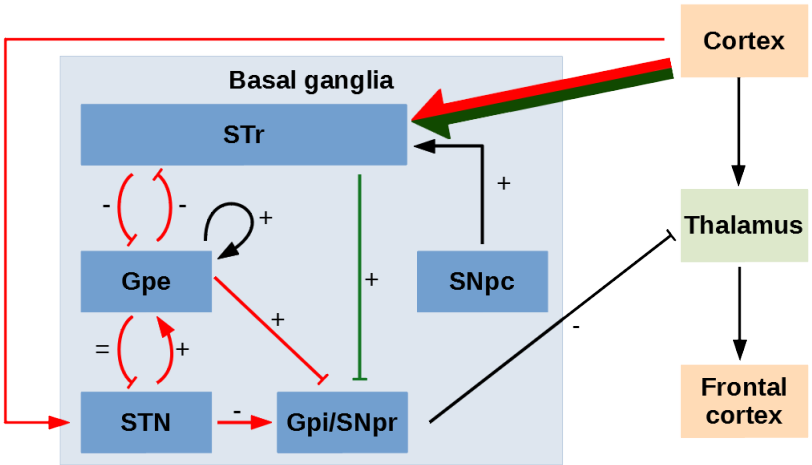
$$S =?[n\_Th < 4]$$

We compared the model checking results of our model with the simulation results from the article by Wei and Wang. As said in subsection 4.3.4, the behavior of our formal inhibitory control model is close to the results presented in this article. This behavior is still described as a race in which the stop signal information has to transit from STN to SNpr before the STr inputs inhibit SNpr [113].

Wei and Wang conducted an experiment on their model by modulating the network weights. The goal of this experiment was to determine if the inhibitory control behavior is more sensitive to the modulation of some connections than to others. We reproduced this experiment on our model. To this end, we doubled the weight of all the connections one at a time and we computed the probability to reach a state where the number of spikes emitted by the thalamus is less than 4. The results are similar to those of [126] and go slightly beyond. For the GPe-STr and STr-SNpr connections, the results found are similar to those of Wei and Wang: the strength of some connections can modulate the *Stop Signal Reaction Time* (SSRT) of the inhibitory control (see table 4.3 and figure 4.9).



(A) Results of Wei and Wang paper.



(B) Results obtained with our formal method. Each weight was doubled. A + indicates that increasing the weight of this connection increases SSRT (lower probability for Th inhibition). A − indicates that increasing this weight decreases SSRT (higher probability for Th inhibition).

FIGURE 4.9: Inhibitory control circuit with modification of connection weights.

As seen in figure 4.9b we found that thalamus inhibition was less likely to occur when increasing the weights of the following connections:

- GPe-GPe
- STN-GPe
- Gpe-SNpr
- STr-SNpr

The latter connection (STr-SNpr) modification results are also observed in the work of Wei and Wang (see figure 4.9a).

On the contrary, inhibition is more likely when increasing the weights of the following connections (see figure 4.9b):

- STr-GPe
- STN-SNpr
- SNpr-Th
- GPe-STr

The latter connection (GPe-STr) modification results are also observed in figure 4.9a.

Table 4.3 shows the detailed results and computation times for the verification of property 11 for the model without modification and for each connection modification. For this experiment, Storm was usually more than hundred times faster than PRISM to return a result. The two model-checkers do not display the exact same results but the variations between the original results and the modified connection results are consistent. Some variations are really small (e.g., of the order of 10^{-11} for STN-GPe connection) and others can be more important (e.g., of the order of 10^{-1} for the Gpe-SNpr connection).

The decrease of the number of firing neurons in the STr-GPe, STN-SNpr, GPe-STr, and SNpr-Th connections causes a decrease in the probability of inhibition and consequently a decrease in behavioral inhibition. A decrease in the inhibition probability in the model means a longer SSRT leading to the fulfillment of an unwanted action. As explored in the analysis of the model of Wei and Wang, the impact of the modification of the weight of the GPe-STr connection is a good representation of experimental results on specific neurons, named arkypallidal cells, that connect the GPe and STr [89]. More bibliographical research on the other connections will allow to evaluate the whole relevance of these results.

4.3.6 Comparison of Impaired versus Healthy Brain Models

To differentiate healthy and Parkinsonian brains we considered the two models of figure 4.5.

- The first represents a healthy brain where all neurons from the SNpc are present.
- The second represents an unhealthy brain. As the number of neurons depletes in SNpc in Parkinson's disease [29], with an estimation of 30% of neurons remaining in SNpc of advanced Parkinson's, this circuit only has 30% probability of taking SNpc neurons into account.

Connection	PRISM		Storm	
	Result	Time (s)	Result	Time (s)
Original	0.10323869094174158	375.410	0.103238690941621	2.440
Gpe-STN	0.10323869094174158	374.411	0.103238690941621	2.428
GPe-GPe	0.10117554696965828	866.167	0.1011755469658018	5.202
STN-GPe	0.10323869092606311	381.648	0.1032386909260895	2.282
Gpe-SNpr	0.06961314994688503	399.140	0.06961314994682553	2.609
STr-SNpr	0.06596933727644758	396.324	0.06596933727649107	2.375
STr-GPe	0.1108495612659358	413.993	0.1108495612660819	2.059
STN-SNpr	0.11535715582011276	325.601	0.1153571558199706	2.039
SNpr-Th	0.12755757824032712	314.628	0.1275575782403877	1.923
GPe-STr	0.10323869099184278	391.233	0.1032386909917619	2.513

TABLE 4.3: Results from property 11 for the model without modifications (Original) and after weight modification of specific connections.

Still with property 11 we found that there is a greater chance of getting a thalamus inhibition in the case of a healthy brain. Table 4.4 shows this result and displays that Storm is once again more than a hundred times faster than PRISM. The second formula depicted in figure 4.7 and the probability of 30% associated with it was enough to lead to smaller chances to see an action inhibition (of the order of 10^{-1}). This is compliant with results of [29].

Connection	PRISM		Storm	
	Result	Time (s)	Result	Time (s)
Healthy	0.10323869094174158	375.410	0.103238690941621	2.440
Parkinson	0.09344162726849187	2552.475	0.0934416272696533	19.300

TABLE 4.4: Results of property 11 for the model without modifications (Healthy) and the Parkinsonian model with weight modifications.

4.4 Brain and Game Performance Coupled Model

The next step of this work is to implement a relationship between this brain model and the calibrated Inhibitory Control game model. To do so, a new model is created to merge the calibrated game model and the basal ganglia one. In this new model, the game module is the entity deciding whether or not a stop signal is displayed. For this, instead of a *Inhibitor* module sending stop signals, the *STN* module receives its stop signal inputs directly from the Inhibitory Control game model each time a decoy transition occurs. The basal ganglia set of modules then decides on the reaction to send to the game model based on the firing ratio of the thalamus. In this combined model, the patient performance of the initial model of the Inhibitory Control game model as calibrated in chapter 3 has to be split in two parts (see figure 4.10). The first part corresponds to all the patient's action that have no interest for the inhibitory control in the brain, i.e., anticipation, reaction speed.... The second part concerns actions that send signals usable by the inhibitory control model. The brain model presented in this chapter was unchanged. The complete model is a large one and the model-checking is difficult....

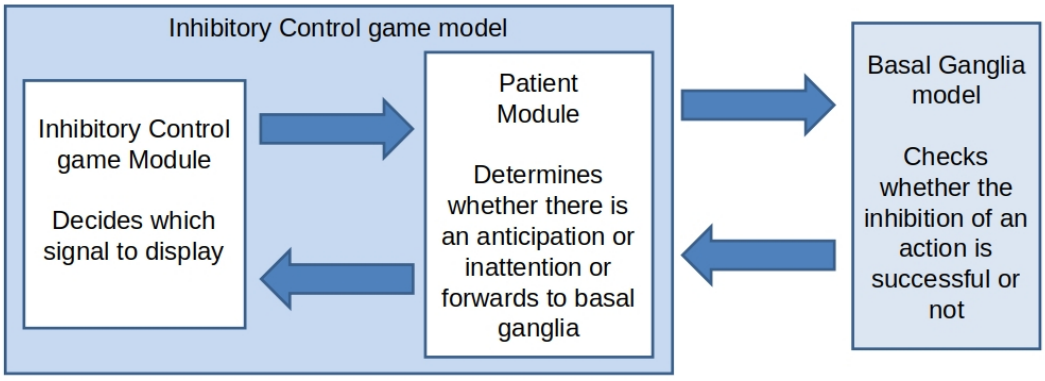


FIGURE 4.10: Modules of the complete model.

4.5 Conclusion and Perspectives

A better understanding of the mechanisms of inhibitory control could allow targeted treatments for different classes of patients with dementia. This chapter proposed a discrete probabilistic model of the inhibitory control circuit of the brain and its formal validation. This model allowed us to reproduce known biological behaviors from the literature such as the pathway race [113]. Our model also faithfully represent the importance of some connections in the pathways (e.g., GPe and STr connection [89]). These behaviors were translated in PCTL properties to check the adequacy of the model. Moreover, running an adapted modulation experiment inspired from [126] allowed to explore the sensitivity of inhibitory control to the modulation of some connections that were not shown in [126]. Some modifications allowed us to model a behavior compliant with Parkinson disease. Other modifications to represent other pathologies such as Alzheimer disease could not be done yet but are scheduled for future work.

This merge is underway and should meet its end before March 2022. This modeling project is a new opening in the cognitive function modeling. The use of probabilistic formal models allows to preserve a wide variety of behaviors while enabling model checking. For this model to be checked with off the shelf tools, it was necessary to brought up a new generalization of the LI&F classical model to represent small networks behavior with a single module. The implementation of the relationship between the basal ganglia model and the calibrated game model is an attempt to link the behavior in a game with a pathology through formal computational models.

This work opens new avenues for the formal modeling of cognitive functions and their relations with diseases via rather simple tools like serious games. Moreover, it has proven the feasibility of such model exploration using only off the shelf laptops.

Chapter 5

Conclusion

This thesis aimed to provide a new approach based on formal methods to support the early detection of neurodegenerative diseases. Monitoring and diagnosing early mild NeuroCognitive Disorders (mild NCD) are known to be difficult. We follow a current research direction which is to use serious games to evaluate the cognitive functions of patients. The overall goal is to propose a computer aided system that helps medical practitioners in monitoring patients with early cognitive deficiencies. Another more biologically oriented goal is to provide a computational cognitive model of related structures of the brain.

More precisely, we have four sub-goals:

1. formal modeling of human activities on serious games: we modeled patient behaviors during the games as Discrete Time Markov Chains (DTMCs);

2. validation of relevant activities to differentiate mild NCD and Subjective Cognitive Decline (SCD) patients: we designed and conducted a clinical protocol on these two populations;

3. formal modeling of the underlying biological mechanisms of one of the cognitive functions of interest;

4. exploring the implementation of a common model representing both the biological mechanisms and the behavior of patients during a serious game.

First, we proposed a formal approach based on DTMCs to model human activities of different populations playing clinical serious games. Important properties of these 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. This proposal is part of the evaluation of a new non-invasive screening methods based on the observation of patient performance that can assist clinicians in the early diagnosis of patients. Together with clinicians, we selected four serious games for Alzheimer patients and we modeled three of them as DTMCs in PRISM. These models are parameterized with *a priori* probabilities provided by clinicians. It was necessary to use different modeling approaches as each game has its own specific rules and evaluation methods. We tested a dozen of patient behavioral properties per game, thanks to the PRISM and Storm model checkers. These properties gave an overview of the modeled patients performance in a fairly short amount of computation time on a regular laptop. PRISM and Storm return the probabilities for the modeled patients to go through specific paths representing various pathological behaviors. They also compute the means of answers and time scores. The models and their verification allowed us to validate our approach and to test its scalability for rather different applications. Regarding the

comparison of PRISM and Storm, we observed that Storm is usually faster in its computation time as its algorithms are tailored for the kind of models we implemented. Nonetheless, some features available in PRISM, such as the computation of PCTL* formulas or the "run experiment", make it hard to replace.

Second, we designed a clinical experimentation in association with clinicians of the Institut Claude Pompidou of Nice to validate the activities of the selected serious games. The main constraint was to go through several ethical validation processes. The inclusion of subjects into the protocol lasted from Sept 2020 to June 2021 and the whole experiment ended late Sept 2021. The results showed the potential of the selected serious games in the screening of mild NCD patients. These games were able to display differences between mild NCD and SCD groups as well as correlations with classical neurocognitive tests. Indeed, this protocol represents an interesting feasibility study; yet it is perfectible. The protocol results were used to redesign and manually calibrate (i.e., adjust DTMC probabilities) two game models. This first calibration experience led to a better representation of the behavior variants and made the model more effective. However, the calibration could go further to better match the observed data.

Third, we chose to model the inhibitory control cognitive function. This choice is mainly motivated by the fact that it is a routinely assessed function in elder patients and a big part of its mechanisms and potential source of dysfunctions is located in a rather small region of the brain named basal ganglia. We proposed a discrete probabilistic model of the inhibitory control circuit of the brain and its formal validation. This model allowed us to reproduce known biological behaviors from the literature such as the pathway race and to state the importance of some connections in the pathways. These behaviors were translated into PCTL properties to check the adequacy of the biological model. Further bibliographical investigations has to be done to validate all results obtained in the experiment.

Fourth, the work on the common model is still in progress. Indeed, we have to redesign and calibrate the Inhibitory Control game model similarly to the Code game model to better match the observed results. Once this calibration done, the representation of the relationships between the basal ganglia model and the game model will be possible. To do so, a new model has to be created to merge the calibrated game model and the basal ganglia one. In this new model, the game module will be the entity deciding whether or not to send a no-go signal.

Contributions

The contributions of this PhD projects can be synthesized as follows.

1. Proposal of a probabilistic formal approach adapted to model human activity during serious games: these models can reproduce behaviors observed in clinical conditions.

2. Identification of temporal activities as possible key points to differentiate mild NCD and SCD profiles.

3. New generalization of the LI&F neuron, named neuron box, allowing to observe some rather complex behaviors.

4. New model of the inhibitory control function using these neuron boxes.

The two first contributions are clinically oriented and may serve as a basis for a computer aided system for early diagnosis of dementia in elder patients. The last

two ones are more biologically oriented. They provide a new reliable tool to represent and understand neurobiological knowledge and its relation with clinical observations. In the scope of cognitive function evaluation for elder patients, the game duration may rarely exceed a duration of a few minutes. Moreover, these games are intended to be easy to understand for users who are not necessarily familiar with tablets or computers. These two factors are convenient because they limit the complexity of the our models.

Perspectives

This work also opens new goals and questions.

- A larger scale clinical experiment with some protocol corrections would lead to higher quality data.
- What are the limits of probabilistic model checking regarding serious game modeling in terms of computation time, complexity/size of the model, and fidelity or adequacy of the representation?
- How many relevant sensors (such as cameras, microphones, or bio-metric sensors), hence new type of events, can be added to increase the accuracy of the human activity formal models?
- What are the limits of the neuron box model?
  - Is increasing the range of the firing ratio a good solution to represent more complex behaviors?
  - Can the proposed neuron box model represent more complex behaviors than the inhibitory control?

Each question opens new challenging research directions in both clinical and biological domains.

Bibliography

[1] Clark C Abt. *Serious games*. University press of America, 1987.

[2] A. S. Ahouandjinou, C. Motamed, and E. C. Ezin. "A temporal belief-based hidden Markov model for human action recognition in medical videos". In: *Pattern Recognition and Image Analysis* (2015).

[3] Rajeev Alur and Thomas Henzinger. "Reactive Modules". In: *Formal Methods in System Design* (1999).

[4] Julian Alvarez, Jean-Pierre Jessel, and Gilles Méthel. "PBL and Serious Game". In: *7th ALE International Workshop, G. Moore–A. Hernandez, Toulouse*. 2007.

[5] Joaquin A Anguera et al. "Video game training enhances cognitive control in older adults". In: *Nature* 501 (2013).

[6] Sven-Thomas Antoni et al. "Online model checking for monitoring surrogate-based respiratory motion tracking in radiation therapy". In: *International journal of computer assisted radiology and surgery* 11.11 (2016), pp. 2085–2096.

[7] I Arevalo-Rodriguez et al. "Mini-Mental State Examination (MMSE) for the detection of Alzheimer's disease and other dementias in people with mild cognitive impairment (MCI)". In: *Cochrane Database of Systematic Reviews* 3 (2015).

[8] Adam R Aron and Russell A Poldrack. "Cortical and subcortical contributions to stop signal response inhibition: role of the subthalamic nucleus". In: *Journal of Neuroscience* 26.9 (2006), pp. 2424–2433.

[9] Adam R. Aron et al. "Converging Evidence for a Fronto-Basal-Ganglia Network for Inhibitory Control of Action and Cognition". In: *Journal of Neuroscience* 27.44 (2007), pp. 11860–11864.

[10] Teeba Athar, K Al Balushi, and Shah Alam Khan. "Recent advances on drug development and emerging therapeutic agents for Alzheimer's disease". In: *Molecular biology reports* 48.7 (2021), pp. 5629–5645.

[11] S. D. Atkinson and V. L. Narasimhan. "Design of an introductory medical gaming environment for diagnosis and management of Parkinson's disease". In: *Trendz in Information Sciences Computing(TISC)*. 2010.

[12] Muhammad Saad Ayub and Osman Hasan. "Formal Probabilistic Analysis of a Virtual Fixture Control Algorithm for a Surgical Robot". In: *Verification and Evaluation of Computer and Communication Systems*. Ed. by Kamel Barkaoui et al. Cham: Springer International Publishing, 2017, pp. 1–16. ISBN: 978-3-319-66176-6.

[13] A Bahar-Fuchs, L Clare, and B Woods. "Cognitive training and cognitive rehabilitation for mild to moderate Alzheimer's disease and vascular dementia". In: *Cochrane Database of Systematic Reviews* (2013).

[14] Chiara Baston and Mauro Ursino. "A Biologically Inspired Computational Model of Basal Ganglia in Action Selection". In: *Intell. Neuroscience* 2015 (Jan. 2015).

[15] Gerd Behrmann, Alexandre David, and Kim 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*. 2004, pp. 200–236.

[16] Grégory Ben-Sadoun et al. "Physical and cognitive stimulation using an exergame in subjects with normal aging, mild and moderate cognitive impairment". In: *Journal of Alzheimer's Disease* 53 (2016).

[17] Daniel J Benjamin et al. "Redefine statistical significance". In: *Nature human behaviour* 2.1 (2018), pp. 6–10.

[18] Kent C. Berridge and Terry E. Robinson. "Control versus causation of addiction". In: *Behavioral and Brain Sciences* 19.4 (1996), 576–577.

[19] GITA BĒRZIŅA. "Σπουδαιογέλοιον in Xenophon's Symposium". In: *DE RISU–Representations and evaluations of laughter in Greek and Latin literature* (2021), p. 81.

[20] Matthew L. Bolton, Ellen J. Bass, and Radu I. Siminiceanu. "Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking". In: *International Journal of Human-Computer Studies* 70.11 (2012), pp. 888 –906. ISSN: 1071-5819.

[21] Silvia Bonfanti, Angelo Gargantini, and Atif Mashkoor. "A systematic literature review of the use of formal methods in medical software systems". In: *Journal of Software: Evolution and Process* 30.5 (2018), e1943.

[22] Leanne Boucher et al. "Inhibitory control in mind and brain: an interactive race model of countermanding saccades." In: *Psychological review* 114.2 (2007), p. 376.

[23] Heiko Braak et al. "Staging of the intracerebral inclusion body pathology associated with idiopathic Parkinson's disease (preclinical and clinical stages)". In: *Journal of Neurology* 249.0 (Oct. 2002), pp. 1–1.

[24] Alexandre Breton, Daniel Casey, and Nikitas A. Arnaoutoglou. "Cognitive tests for the detection of mild cognitive impairment (MCI), the prodromal stage of dementia: Meta-analysis of diagnostic accuracy studies". In: *International Journal of Geriatric Psychiatry* 34.2 (2019), pp. 233–242.

[25] Patrice Brocker et al. "Inventaire Apathie: Evaluation de l'apathie chez des sujets presentant une maladie d'Alzheimer ou un trouble cognitif leger". In: *Revue de gériatrie* 28 (2003), pp. 473–480.

[26] Nicolas Brunel and Mark CW Van Rossum. "Lapicque's 1907 paper: from frogs to integrate-and-fire". In: *Biological cybernetics* 97.5 (2007), pp. 337–339.

[27] Fabio Buttussi et al. "Evaluation of a 3D serious game for advanced life support retraining". In: *Int. Journal of Medical Informatics* (2013). ISSN: 1386-5056.

[28] Muffy Calder et al. "Stronger computational modelling of signalling pathways using both continuous and discrete-state methods". In: *International Conference on Computational Methods in Systems Biology*. Springer. 2006, pp. 63–77.

[29] H. C. Cheng, C. M. Ulane, and R. E. Burke. "Clinical progression in Parkinson disease and the neurobiology of axons". In: *Annals of Neurology* 67.6 (June 2010), pp. 715–725.

[30] Luca Chittaro and Riccardo Sioni. "Turning the Classic Snake Mobile Game into a Location–Based Exergame that Encourages Walking". In: *Persuasive Technology. Design for Health and Safety*. 2012. ISBN: 978-3-642-31037-9.

[31] Fong Ping Chong et al. "Tau proteins and tauopathies in Alzheimer's disease". In: *Cellular and molecular neurobiology* 38.5 (2018), pp. 965–980.

[32] Allan Clark et al. "Verification and testing of biological models". In: *Proceedings of the 2010 Winter Simulation Conference*. IEEE. 2010, pp. 620–630.

[33] Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. "Automatic verification of finite-state concurrent systems using temporal logic specifications". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* (1986).

[34] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999. ISBN: 0-262-03270-8.

[35] Brian Colder. "The basal ganglia select the expected sensory input used for predictive coding". In: *Frontiers in Computational Neuroscience* 9 (2015), p. 119.

[36] Trevor J. Crawford et al. "Inhibitory control of saccadic eye movements and cognitive impairment in Alzheimer's disease". In: *Biological Psychiatry* 57.9 (2005), pp. 1052–1060.

[37] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[38] Elisabetta De Maria et al. "Modelling and Formal Verification of Neuronal Archetypes Coupling ". In: *CSBio 2017 - 8th International Conference on Computational Systems-Biology and Bioinformatics*. Vol. 17. CSBio '17 Proceedings of the 8th International Conference on Computational Systems-Biology and Bioinformatics. Nha Trang, Vietnam: ACM, Dec. 2017, pp. 3–10.

[39] Christian Dehnert et al. "A storm is coming: A modern probabilistic model checker". In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 592–600.

[40] Christian Dehnert et al. "PROPhESY: A PRObabilistic ParamEter SYnthesis Tool". In: *Computer Aided Verification*. Ed. by Daniel Kroening and Corina S. Păsăreanu. Cham: Springer International Publishing, 2015, pp. 214–231.

[41] Georg Dirnberger and Marjan Jahanshahi. "Executive dysfunction in Parkinson's disease: A review". In: *Journal of Neuropsychology* 7.2 (2013), pp. 193–224.

[42] Damien Djaouti et al. "Origins of serious games". In: *Serious games and edutainment applications*. Springer, 2011, pp. 25–43.

[43] B Dubois et al. ""The 5 words": a simple and sensitive test for the diagnosis of Alzheimer's disease". In: *Presse Medicale (Paris, France: 1983)* 31.36 (2002), pp. 1696–1699.

[44] B Dubois et al. "The FAB: a frontal assessment battery at bedside". In: *Neurology* 55.11 (2000), pp. 1621–1626.

[45] Flora H Duits et al. "The cerebrospinal fluid "Alzheimer profile": easily said, but what does it mean?" In: *Alzheimer's & Dementia* 10.6 (2014), pp. 713–723.

[46] James T Eckner et al. "A novel clinical test of recognition reaction time in healthy adults." In: *Psychological assessment* 24.1 (2012), p. 249.

[47] Fifth Edition et al. "Diagnostic and statistical manual of mental disorders". In: *Am Psychiatric Assoc* 21 (2013).

[48] The Editors of Encyclopaedia Britannica. *Old age*. In: *Encyclopaedia Britannica*. 2021.

[49] Gwenith G. Fisher, Marisol Chacon, and Dorey S. Chaffee. "Chapter 2 - Theories of Cognitive Aging and Work". In: *Work Across the Lifespan*. Ed. by Boris B. Baltes, Cort W. Rudolph, and Hannes Zacher. Academic Press, 2019.

[50] Theresa M. Fleming et al. "Serious Games and Gamification for Mental Health: Current Status and Promising Directions". In: *Frontiers in Psychiatry* (2017). ISSN: 1664-0640.

[51] Nathalie Fournet, Chrystèle Mosca, and Olivier Moreaud. "Deficits in inhibitory processes in normal aging and patients with Alzheimer's disease: a review". In: *Psychologie & NeuroPsychiatrie du vieillissement* 5.4 (Dec. 2007), pp. 281–294.

[52] Marian Gheorghe, Savas Konur, and Florentin Ipate. "Kernel P systems and stochastic P systems for modelling and formal verification of genetic logic gates". In: *Advances in unconventional computing*. Springer, 2017, pp. 661–675.

[53] Ann M Graybiel. "The basal ganglia". In: *Current biology* 10.14 (2000), R509–R511.

[54] Judith Greene and Manuela d'Oliveira. *Learning to use statistical tests in psychology*. McGraw-Hill Education (UK), 2005.

[55] P. Groot et al. "Using Model Checking for Critiquing based on Clinical Guidelines". In: *AI in Medicine* 46.1 (2008), pp. 19–36.

[56] Gaël Guennebaud, Benoit Jacob, et al. "Eigen". In: *URl: http://eigen. tuxfamily. org* 3 (2010).

[57] Ernst Moritz Hahn et al. "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.

[58] Nikolaus 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.

[59] Hans Hansson and Bengt Jonsson. "A logic for reasoning about time and reliability". In: *Formal aspects of computing* (1994).

[60] M. Hassan. "A performance model of pedestrian dead reckoning with activity-based location updates". In: *2012 18th IEEE Int. Conf. on Networks (ICON)*. 2012.

[61] David Heeger. "Poisson model of spike generation". In: *Handout, University of Standford* 5.1-13 (2000), p. 76.

[62] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[63] Sien Hu et al. "Changes in cerebral morphometry and amplitude of low-frequency fluctuations of BOLD signals during healthy aging: correlation with inhibitory control". In: *Brain Structure and Function* 219.3 (2014), pp. 983–994.

[64] Jonathan D Huntley et al. "The importance of sustained attention in early Alzheimer's disease". In: *International journal of geriatric psychiatry* 32.8 (2017), pp. 860–867.

[65] IBM Corp. *IBM SPSS Statistics for Windows*. Version 27.0. Armonk, NY: IBM Corp, 2020.

[66] Eugene M Izhikevich. "Which model to use for cortical spiking neurons?" In: *IEEE transactions on neural networks* 15.5 (2004), pp. 1063–1070.

[67] Judith Jaeger. "Digit symbol substitution test: the case for sensitivity over specificity in neuropsychological testing". In: *Journal of clinical psychopharmacology* 38.5 (2018), p. 513.

[68] Marjan Jahanshahi et al. "A fronto–striato–subthalamic–pallidal network for goal-directed and habitual inhibition". In: *Nature Reviews Neuroscience* 16.12 (2015), pp. 719–732.

[69] Ahmad Jalal, Shaharyar Kamal, and Daijin Kim. "A Depth Video-based Human Detection and Activity Recognition using Multi-features and Embedded Hidden Markov Models for Health Care Monitoring Systems." In: *Int. Journal of Interactive Multimedia & Artificial Intelligence* (2017).

[70] M. Kathryn Jedrziewski et al. "The Impact of Exercise, Cognitive Activities, and Socialization on Cognitive Function: Results From the National Long-Term Care Survey". In: *American Journal of Alzheimer's Disease & Other Dementias®* 29 (2014).

[71] Frank Jessen et al. "The characterisation of subjective cognitive decline". In: *The Lancet Neurology* 19.3 (2020), pp. 271–278.

[72] Paul Johns. "Chapter 3 - Functional neuroanatomy". In: *Clinical Neuroscience*. Ed. by Paul Johns. Churchill Livingstone, 2014, pp. 27–47.

[73] Pamela M. Kato and Sebastiaan de Klerk. "Serious Games for Assessment: Welcome to the Jungle". In: *Journal of Applied Testing Technology* 18 (2017).

[74] Roy PC Kessels, Pieter W Molleman, and Joukje M Oosterman. "Assessment of working-memory deficits in patients with mild cognitive impairment and Alzheimer's dementia using Wechsler's Working Memory Index". In: *Aging clinical and experimental research* 23.5 (2011), pp. 487–490.

[75] Kim M. Kiely. "Cognitive Function". In: *Encyclopedia of Quality of Life and Well-Being Research*. Ed. by Alex C. Michalos. Dordrecht: Springer Netherlands, 2014, pp. 974–978.

[76] Eunju Kim, Sumi Helal, and Diane Cook. "Human activity recognition and pattern discovery". In: *IEEE pervasive computing* (2009).

[77] David S Knopman, David T Jones, and Michael D Greicius. "Failure to demonstrate efficacy of aducanumab: An analysis of the EMERGE and ENGAGE trials as reported by Biogen, December 2019". In: *Alzheimer's & Dementia* 17.4 (2021), pp. 696–701.

[78] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd Int. Conf. on Computer Aided Verification (CAV'11)*. 2011.

[79] Marta Kwiatkowska, Gethin Norman, and David Parker. "Stochastic model checking". In: *Int. School on Formal Methods for the Design of Computer, Communication and Software Systems*. 2007.

[80] Alexandra König et al. "Validation of an automatic video monitoring system for the detection of instrumental activities of daily living in dementia patients". In: *Journal of Alzheimer's disease : JAD* 44 (2015).

[81] Louis Lapicque. "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation." In: *Société de Biologie* (1907).

[82] Franck Le Duff et al. "The 2008–2012 French Alzheimer plan: description of the national Alzheimer information system". In: *Journal of Alzheimer's Disease* 29.4 (2012), pp. 891–902.

[83] Muriel Deutsch Lezak et al. *Neuropsychological assessment*. Oxford University Press, USA, 2004.

[84] Gordon D Logan and William B Cowan. "On the ability to inhibit thought and action: A theory of an act of control." In: *Psychological review* 91.3 (1984), p. 295.

[85] Parker E Ludwig, Vamsi Reddy, and Matthew Varacallo. "Neuroanatomy, Neurons". In: *StatPearls [Internet]* (2020).

[86] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". In: *Neural Networks* 10.9 (1997), pp. 1659–1671.

[87] T. Magherini et al. "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.

[88] T. Magherini et al. "Using Temporal Logic and Model Checking in Automated Recognition of Human Activities for Ambient-Assisted Living". In: *IEEE Transactions on Human-Machine Systems* (2013).

[89] Nicolas Mallet et al. "Arkypallidal cells send a stop signal to striatum". In: *Neuron* 89.2 (2016), pp. 308–316.

[90] Valeria Manera et al. "'Kitchen and cooking,' a serious game for mild cognitive impairment and Alzheimer's disease: a pilot study". In: *Frontiers in Aging Neuroscience* 7 (2015).

[91] John Manning. *The emblem*. Reaktion Books, 2004.

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

[93] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[94] Anne-Sophie Melenhorst, Wendy A Rogers, and Don G Bouwhuis. "Older adults' motivated choice for technological innovation: Evidence for benefit-driven selectivity." In: *Psychology and aging* 21 (2006).

[95] Jonathan W Mink. "The Basal Ganglia: Focused Selection and Inhibition of Competing Motor Programs". In: *Progress in Neurobiology* 50.4 (1996), pp. 381–425.

[96] G Mirabella et al. "Deep brain stimulation of subthalamic nuclei affects arm response inhibition in Parkinson's patients". In: *Cerebral cortex* 22.5 (2012), pp. 1124–1132.

[97] Ahmed A. Moustafa and Mark A. Gluck. "Computational cognitive models of prefrontal-striatal-hippocampal interactions in Parkinson's disease and schizophrenia". In: *Neural Networks* 24.6 (2011). Special Issue: Neurocomputational Models of Brain Disorders, pp. 575–591.

[98] Rui Nouchi et al. "Brain Training Game Improves Executive Functions and Processing Speed in the Elderly: A Randomized Controlled Trial". In: *PLOS ONE* 7.1 (2012).

[99] Martin Nyolt, Kristina Yordanova, and Thomas Kirste. "Checking Models for Activity Recognition". In: *ICAART*. 2015.

[100] Verna C. Pangman, Jeff Sloan, and Lorna Guse. "An examination of psychometric properties of the Mini-Mental State Examination and the Standardized Mini-Mental State Examination: Implications for clinical practice". In: *Applied Nursing Research* 13.4 (2000), pp. 209–213.

[101] Hélene Paugam-Moisy and Sander M Bohte. "Computing with spiking neuron networks." In: *Handbook of natural computing* 1 (2012), pp. 1–47.

[102] Gualtiero Piccinini. "The First Computational Theory of Mind and Brain: A Close Look at Mcculloch and Pitts's "Logical Calculus of Ideas Immanent in Nervous Activity"". In: *Synthese* 141 (Aug. 2004).

[103] Martin Prince et al. "The global prevalence of dementia: A systematic review and metaanalysis". In: *Alzheimer's & Dementia* 9 (2013).

[104] Dale Purves et al. *Neurosciences*. Broché, 2019.

[105] J. P. Queille and J. Sifakis. "Specification and verification of concurrent systems in CESAR". In: *International Symposium on Programming*. Ed. by Mariangiola Dezani-Ciancaglini and Ugo Montanari. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 337–351. ISBN: 978-3-540-39184-5.

[106] Yves Renard and Julien Pommier. "Gmm++: a generic template matrix C++ library". In: *Short User Documentation, MIP, INSAT, Complexe scientifique de Rangueil* 31077 (2005).

[107] Philippe Robert et al. "Efficacy of a Web App for Cognitive Training (MeMo) Regarding Cognitive and Behavioral Performance in People With Neurocognitive Disorders: Randomized Controlled Trial". In: *Journal of medical Internet research* 22.3 (2020), e17167.

[108] Stewart Robinson. "Conceptual modelling for simulation Part I: definition and requirements". In: *Journal of the operational research society* 59.3 (2008), pp. 278–290.

[109] Joseph J Ryan and Shane J Lopez. "Wechsler adult intelligence scale-III". In: *Understanding Psychological Assessment*. Springer, 2001, pp. 19–42.

[110] Dorsa Sadigh et al. "Data-Driven Probabilistic Modeling and Verification of Human Driver Behavior". In: *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium (FVHMS)*. 2014.

[111] Jeffrey D Schall and David C Godlove. "Current advances and pressing problems in studies of stopping". In: *Current Opinion in Neurobiology* 22.6 (2012). Decision making, pp. 1012 –1021.

[112] Robert Schmidt et al. "Beta Oscillations in Working Memory, Executive Control of Movement and Thought, and Sensorimotor Function". In: *Journal of Neuroscience* 39.42 (2019), pp. 8231–8238.

[113] Robert Schmidt et al. "Canceling actions involves a race between basal ganglia pathways". In: *Nature neuroscience* 16.8 (2013), pp. 1118–1124.

[114] Henning Schroll and Fred Hamker. "Computational models of basal-ganglia pathway functions: focus on functional neuroanatomy". In: *Frontiers in Systems Neuroscience* 7 (2013), p. 122. ISSN: 1662-5137.

[115] Jun Sun et al. "PAT: Towards Flexible Verification under Fairness". In: *International Conference on Computer Aided Verification*. 2009.

[116] Tiffany Tong et al. "A serious game for clinical assessment of cognitive status: validation study". In: *JMIR serious games* 4 (2016).

[117] Minh Khue Phan Tran. "Maintaining the engagement of older adults with dementia while interacting with serious game". PhD thesis. 2017.

[118] Minh Khue Phan Tran, François Brémond, and Philippe Robert. "Assistance for Older Adults in Serious Game Using an Interactive System". In: *4th Int. Conf. on Games and Learning Alliance (GALA)*. 2015.

[119] *Tâches Attentionnelles Exécutives*. https://cmrr-nice.fr/lab/tae/. CoBTeK lab, Université Côte d'Azur. 2018.

[120] Vanessa Vallejo et al. "Evaluation of a novel Serious Game based assessment tool for patients with Alzheimer's disease". In: *PLOS ONE* 12 (2017).

[121] Frederick Verbruggen and Gordon D. Logan. "Models of response inhibition in the stop-signal and stop-change paradigms". In: *Neuroscience & Biobehavioral Reviews* 33.5 (2009). Translational Aspects of Stopping and Response Control, pp. 647 –661.

[122] Kristen L Votruba, Carol Persad, and Bruno Giordani. "Cognitive deficits in healthy elderly population with "normal" scores on the Mini-Mental State Examination". In: *Journal of Geriatric Psychiatry and Neurology* 29.3 (2016), pp. 126–132.

[123] Helen M Walker. "Degrees of freedom." In: *Journal of Educational Psychology* 31.4 (1940), p. 253.

[124] David Wechsler. "The measurement and appraisal of adult intelligence". In: (1958).

[125] David Wechsler and Psychological Corporation. Psychological Corporation, 1997.

[126] Wei Wei and Xiao-Jing Wang. "Inhibitory Control in the Cortico-Basal Ganglia-Thalamocortical Loop: Complex Regulation and Interplay with Memory and Decision Processes". In: *Neuron* 92 (2016).

[127] Jeannette M Wing. "A specifier's introduction to formal methods". In: *Computer* 23.9 (1990), pp. 8–22.

[128] Radia Zeghari et al. "The "Interest Game": A Ludic Application to Improve Apathy Assessment in Patients with Neurocognitive Disorders". In: *Journal of Alzheimer's Disease* Preprint (2020), pp. 1–9.

[129] Jun Zhang et al. "A Neural Computational Model of Incentive Salience". In: *PLOS Computational Biology* 5.7 (July 2009), pp. 1–14.

[130] M. Zyda. "From visual simulation to virtual reality to games". In: *Computer* 38 (2005).