<https://helda.helsinki.fi>

Generalised Player Modelling : Why Artificial Intelligence in Games Should Incorporate Meaning, with a Formalism for so Doing

Cowley, Benjamin Ultan

Springer

2020-07-10

Cowley , B U 2020 , Generalised Player Modelling : Why Artificial Intelligence in Games Should Incorporate Meaning, with a Formalism for so Doing . in X Fang (ed.) , HCI in Games : Second International Conference, HCI-Games 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19-24, 2020, Proceedings . Lecture Notes in Computer Science , vol. 12211 , Springer , Cham , pp. 3-22 , International Conference, C & C 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19-24, 2020 , Copenhagen , Denmark , 19 /

<http://hdl.handle.net/10138/319281>

https://doi.org/10.1007/978-3-030-50164-8_1

publishedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

Generalised Player Modelling: why Artificial Intelligence in games should incorporate meaning, with a Formalism for so doing

Benjamin Ultan Cowley^{1,2}[0000–0001–8828–2994]

¹ Data Science, Faculty of Educational Sciences, University of Helsinki

² Cognitive Science, Faculty of Arts, University of Helsinki

ben.cowley@helsinki.fi

<https://blogs.helsinki.fi/bcowley/>

Abstract. General game-playing artificial intelligence (AI) has recently seen important advances due to the various techniques known as ‘deep learning’. However, in terms of human-computer interaction, the advances conceal a major limitation: these algorithms do not incorporate any sense of what human players find *meaningful* in games.

I argue that adaptive game AI will be enhanced by a generalised player model, because games are inherently human artefacts which require some encoding of the human perspective in order to respond naturally to individual players. The player model provides *constraints* on the adaptive AI, which allow it to encode aspects of what human players find meaningful. I propose that a general player model requires parameters for the subjective experience of play, including: player psychology, game structure, and actions of play. I argue that such a player model would enhance efficiency of per-game solutions, and also support study of game-playing by allowing (within-player) comparison between games, or (within-game) comparison between players (human and AI).

Here we detail requirements for functional adaptive AI, arguing from first-principles drawn from games research literature, and propose a formal specification for a generalised player model based on our ‘Behavlets’ method for psychologically-derived player modelling.

Keywords: Artificial Intelligence · game AI · general player model · player personality · Behavlets · formal models · category theory.

1 Introduction

Computer games have the potential to adapt themselves through changes to their difficulty, appearance, story, or even rules. Computer games thus offer a unique opportunity for play that is completely tailored to the individual player by adaptive artificial intelligence (AI) - instantiated as opponent agent AI or ‘game-management’ AI. Any non-trivial adaptation requires a player model, to encode relevant aspects of player individuality [36]. The concept of a generalised player model extends this, to describe the subjective experience of play in terms

of validated constructs, which could include psychology profiles, game design patterns, action patterns, and more. The aim is not to provide a one-size-fits-all model, but to provide a system for expressing the supra-specific elements that apply to all players, especially cognition, emotion, and personality. The specific implementation then depends on the game.

In this paper, I suggest that adaptive game AI will be optimised by a generalised player model. This is because games are inherently human artefacts which, therefore, require some encoding of the human perspective in order to effectively autonomously respond to the individual player. This argument is built on the idea that the player model will impose *constraints* which guide the AI to optimal performance.

In brief, game-playing AI must typically be constrained to function (see section 2.2). For example, in their ground-breaking Go and Atari-playing agents, Silver *et al*, and Mnih *et al* [29, 25] imposed well-chosen constraints on the problem domain to enable their solutions. Such constraints can be, e.g. dimensionality reduction, or *simulation* of the original system according to some simplification. Here it is proposed that a generalised player model gives a partial solution, built on two requirements: i) to capture information about player psychology (cognition, emotion and personality) and activity; ii) to represent that information in the context of the game. Requirement i) *constrains* the model of player behaviour to well-understood theoretical constructs; ii) presents the model as input to a learning algorithm.

A generalised player model requires a foundation of parameters that describe the subjective experience of play. The foundation will draw on established modelling tools, including at least: a) psychology of behaviour; b) general game design; and c) actions in the context of a given game. This foundation should also be integrated with the computational intelligence that drives the model. The ultimate aim is to *improve efficacy and viability of the artificial intelligence required to power games which adapt to their players*.

A generalised player model can bring added benefits. It can be more efficient than creating a novel model for every game, and can even improve algorithmic performance of a real-time player model [14]. It also allows comparison between games which makes it a useful tool for studying play in general.

In Part A, I discuss why adaptive game AI benefits from a general model of player psychology. First, the concept and meaning of an adaptive game is described, stating the aspects of player psychology to which games can adapt. I then make my case from first principles, drawing on the literature of games studies.

In Part B, I discuss possible but speculative solutions, and develop one as a proposal: a formal category theoretic basis to a ‘Behavlets’-driven generalised player model. The Behavlets method is designed to build facets a) to c) above into composite features of game-play defined over entire action sequences [10], and thus model players for, e.g. personality type classification [9]. This paper is built on several prior preprint papers exploring research questions related to Behavlets [12, 11].

2 Part A: Adaptivity in games

2.1 Background

Games can roughly be split into two categories: single player and multiplayer. Adaptive algorithms may be used in a multiplayer game for a range of reasons; in the main, the challenge in these games is due to other human players. In an online multiplayer game, with an interface via network but without camera or microphone, the players do not see each other nor do they have access to traditional social cues for understanding their opponents, e.g. body language. Such a game environment is *stigmergic*, as players do not interact directly but through the shared environment of the game space. They will read the signs left by their opponent, build a ‘theory of mind’ model around the complete set of actions observed, and classify the other player based on both what they know of the types of player of that game, and natural social recognition skills. This process contributes to the decision making process for how to play the game and is very different from a single player game.

In certain games, the inclusion of good quality, artificially-intelligent non-player characters (NPCs) can be central to gameplay design and is important for the player experience. Games can be created to be adaptive to the player through changes to NPC behaviour, or by altering other parameters of the game that affect the gameplay [36]. In both cases, a player learns to be more effective at playing a game by learning the rules of the game, including how NPCs behave. Unlike real players, NPC behaviour is usually more predictable and typically it is easier to develop strategies to be successful in competitive gameplay. Players expect NPCs to behave consistently. Nevertheless, one of the reasons that people like to play against other real players is that it is often more satisfying. People behave differently from each other, have complex capability profiles, and are motivated by different ways of playing. Players understand that other players are less predictable than NPCs and accept this.

Dynamic Difficulty Adjustment (DDA) is a popular approach to implement adaptive AI, e.g. [6]. It can work, for example, by altering the number of power-ups in a game or by making non-player characters more or less co-operative or competitive. Some of the earliest games to implement DDA systems were *Max Payne* (3D Realms, 2001) and *Prey* (3D Realms, 2006). However, the AI can do much more than control an opposing force. Forms of adaptive AI demonstrated in commercial games include adjusting player character attributes *MarioKart* (Nintendo, 1992), appearance *Fable* (Lionhead Studios, 2004), story *Facade* (Mateus & Stern, 2005), character learning *Black and White* (Lionhead Studios, 2001), and reactive squad tactics *Fear* (Monolith Productions, 2005). Related research (from disciplines including game AI, computational intelligence, and machine learning), has been reviewed [35] and categorised into three main areas: Player Experience Modelling (PEM), Procedural Content Generation (PCM), and Massive-Scale Game Data Mining (MDM).

I consider a game AI system holistically (with NPC behaviour as a core aspect), in which the AI has two coordinated systems: α) a user model to cap-

ture some aspect of player psychology, which then supplies parameters to β) controller(s) for adjusting some relevant game system(s). The user model α is intended to model relevant data about some area of the player's state, as discussed further in section 2.3. The issue of what the AI can control is somewhat out of scope, and thus is only briefly discussed here. In many ways the ultimate adaptive system would be based around a human or team of humans who dynamically adjust the gameplay experience based on player choice. Consider a Dungeon Master in table top role play games or the alleged human guidance of the Big Blue Chess playing algorithm. In both cases human guidance provides additional nuances, e.g. flexibility or experience, to the dynamic adjustment of a game playing experience. A game AI based on these principles could be more effective in tailoring fun experiences for a greater range of players. This is part of the motivation for the approach underlying our research.

Modus operandi Adaptive AI should be built around creating a *more engaging* game for the individual player. Obviously, a separate class of game-playing AI can be created with the sole purpose (utility function) of beating the opposing player, e.g. as Chess or Go playing programs. In this class of AI, player psychology is more or less irrelevant because maximising the utility of winning does not necessarily require opponent modelling.

A standard game is constrained by its ruleset, but an adaptive game has the potential to exceed known constraints, or the known/explored state space. Thus I contend that *unconstrained* adaptive AI can violate certain principles of good game design, such as logical consistency and a coherent Magic Circle [19], by exhibiting emergent behaviour. It follows that adaptive AI must be explicitly constrained to adhere to the prior assumptions of the player, which can be encoded as a player model.

Further, undesirable and unpredictable game play bugs can emerge from adaptive components, which could be difficult to test exhaustively. Thus adaptive AI requires a certain level of *formal understanding* of the game that is being designed, in order to give designers the tools to build meta-constraints in the abstract level of the game's possibility space. The use of formal modelling can also benefit the creation of a generalised player model, as described in Section 3 below (and earlier [11]).

Interaction modes It is useful to briefly describe the varying forms of player-game interaction, as this can affect how adaptive AI could be deployed. Salen & Zimmerman [27] illustrated four different modes of game interactivity:

1. Cognitive: psychological, emotional and intellectual interaction with the game.
2. Functional: essentially, the interaction with the game interface and the primary means for accessing the game mechanics.
3. Explicit: the interaction with the underlying game mechanics – this is the core of the game, the mechanics and the formal rules.
4. Cultural: occurs outside the bounds of the game in the form of fan sites, creation of and use of cheats, eSports events, etc.

Here we are interested in the first three modes of interactivity and how these relate to game adaptation. This interactivity is illustrated by the relational schema shown in figure 1. This is a combination of ideas following on from LeBlanc’s Mechanics Dynamics Aesthetics (MDA) method [20] and the USE model of user interaction with an automated system [8].

Fig. 1. A model of player interaction with a computer game, for two or more players. Adaptive AI involves leveraging information from the ‘User(s)’ to dynamically alter the elements of the central ‘System’ module, thus regulating activity within the right-hand ‘Experience’ module, which feeds back to the ‘User(s)’.

At the highest level the model illustrates how a player’s experience of the game arises from a player’s participation in a game in several modes of interaction. A player’s interaction with the underlying formal game system gives rise to a unique set of game play dynamics and a player receives either negative or positive feedback on performance etc. Thus one description of a game is as cybernetic system [27], i.e. a system with a control loop.

Adaptation as a part of this feedback loop can potentially provide improved control of the game system and thus a more tailored experience for individual players. This then gives rise to a more complex game dynamic and potential emergent game behaviour.

2.2 AI under constraint

One key constraint for adaptive AI is that the player’s original conception of the game rules and elements, including *a priori* knowledge³, must not change.

For instance, if the game is a simulation of competition or *agôn* [5] the player’s opponents usually appear to have abilities similar to the player — an example would be a fighting game like *Street Fighter II* (Capcom 1991). Adaptive AI should not suddenly change those abilities in an obvious way in mid-play, to adjust a game mechanic such as difficulty. Early examples of DDA in racing games caused dissonance among players by doing this: a terrible opponent who suddenly becomes lightning fast on the last lap would hurt players’ immersion.

In the DDA system for *Max Payne* (3D Realms, 2001) was designed with this in mind, trying to make it invisible to players. The aim was to not be obvious when the game is self-adjusting its difficulty level, to maintain the game’s immersion.

Yet hiding the rules in this way is a form of “black-box mechanics”. Some game designers think this is bad practice, ergo the player should know about the adaptive elements, but I argue that there is always a complexity limit on players’ knowledge of game mechanics. Thus they can be made aware of adaptive AI, if and only if they can be sufficiently informed of the logic under which the adaptive system works, so that they have some idea of why the game performs its actions. For instance, if the adaptive system is non-linear and/or composed of complex rules or predicates, it may be excessively difficult to explain to the player.

Logical consistency The great thing in game design is to create a game with no *capricious logic*. Capricious logic occurs when the game mechanics are not internally consistent, and this can occur for many reasons. A major cause is the fact that players *observe* a game logic whose rules are often bent or broken for expediency or speed within the game engine.

That players demand a self-consistent logic from their games can be seen in Steinkuehler and Duncan’s [30] study of *World of Warcraft* (Vivendi Universal 2004) players. They examined the cultural activities surrounding the game, such as online discussion forums. Here they discovered that players had been analysing game elements in an attempt to uncover hard information that would be useful in ‘beating’ the game. They claim that these are players of a young age with no scientific training, applying the scientific method to a game world because *they trust that the internal logic of that world will be self-consistent*, so that applying logical analysis will bear fruit. The rule-based structure of games demand logic even if their playfulness should allow logic to be sometimes set aside.

³ *A priori* knowledge includes knowledge of ‘realistic’ or ‘natural’ elements. This can help when adapting, as some changes need not be explicitly explained, such as the trivial example of player opponents that increase in toughness as they increase in size. *A priori* can also refer to game design patterns, existing conventions which somewhat binds developers to the forms of previous work in their chosen genre.

Logical self-consistency sufficient to cope with this type of meta-gaming analysis could be considered a benchmark objective for in-game adaptive AI, since it would be proof (through ‘natural use’) that the autonomic component of the game engine can withstand scrutiny. Pre-facing development of autonomous computer systems by modelling through formal methods should help to ensure their operational stability [32].

Coherent ‘Magic Circle’ The second reason adaptive AI must be constrained is that players are temporarily redefining themselves and their world in terms of a new set of rules, defined by the game: a ‘game world’. Huizinga [19] describes play as a free and meaningful activity, carried out for its own sake, spatially and temporally segregated from the requirements of practical life, *and bound by a self-contained system of rules that holds absolutely*. If these rules change players will be forced to ‘step out’ of the game to re-evaluate their perceived definition of the game world. This would destroy the sense of immersion in a game world which is important to the player’s enjoyment of the ‘fantasy’ element of play.

This concept of a game world is known as the Magic Circle [19] to games researchers. The Magic Circle pertains to the attitudinal psychology that is a prerequisite of play, as individuals must take on the role of players in order to play. Huizinga held that the ‘cheater’ is less deleterious to other players’ enjoyment than the ‘spoil-sport’, because the latter is denying the validity of the Magic Circle while the former is only trying to exploit it [31]. The above example of *World of Warcraft* players applying the scientific method can be thought of in the same way, since they are rejecting the attitude of playfulness in favour of production, sometimes known as ‘the grind’.

[23, 24] also identified fantasy as an important part of enjoyment in gaming. Drawing on the psychology of intrinsic motivation, he was among the first to experiment on the relation between fantasy and game-play in educational games. The fact that fantasy is hugely motivating in game play is quite well-established now, and it is more interesting *how* that fantasy is structured. It is necessary to provide some comprehensible metaphor within the fantasy, so that players can easily digest the information content of the fantasy and go directly to dealing with the game mechanic. Thus, here again adaptive AI requires constraint.

Constraint summary So there is a potential conflict of interest between adaptive AI which can alter game mechanics, and preservation of logical consistency and the Magic Circle. For instance, one way to introduce novice players to a complicated control scheme is to begin with a restricted subset of the full scheme. But if this is how the player initially understands the game, they will question the introduction of new control dimensions unless they are explained within the narrative - i.e. in the acquisition of new equipment, skills, companions etc. Meeting the player’s expectations for the logic and fantasy of their game is the key to creating effective adaptive components. These constraints on adaptive AI mean that great care must be taken when adapting in-game elements in real time - the player must either be forewarned that adaptation may take place (making

it part of the game’s rule set, which may conflict with realism), or must not be able to notice it at all.

2.3 Aspects of players to adapt to

Ability Ability within a game is influenced by a player’s position within the *learning curve* of the game. The learning curve is the sine qua non of game design: as argued by Koster [22], learning is the key ingredient that makes games fun.

Player ability is also influenced by their knowledge of the game’s design patterns [3]. A player who is fluent with a particular design pattern, such as the ‘Aim and Shoot’ pattern, can have a higher skill level when beginning to play than a player who is not so familiar.

Learning and ability are related in information processing terms. In [8] it was pointed out that players process information from the game world, trying to balance the complexity of this environment with internal cognitive complexity. The complexity of the game comes from its control scheme, narrative, objectives, opponents and other such elements; while cognitive complexity refers to the player’s ability to take all that in and react, enabled through prior experience of the form or innate ability. So a balance between the two is desired. If the player cannot comprehend everything being thrown at him, he will be overwhelmed and unable to function in an ideal way. If the game does not provide sufficient challenge or interest, the player will be left unengaged and will lose motivation.

Thus, negative imbalance leads to confusion and anxiety, and positive imbalance leads to boredom and apathy [26]. There is an echo of Ashby’s Law of Requisite Variety in this formulation, since the variety needed to support learning and thus optimal game play must be present in both sides: player and game. In this sense, an adaptive single-player system would resemble a pair of linked homeostats [16]. On one side, the player learns the game system and/or narrative, attempting to ‘beat’ it by application of experience-based skill. On the other side, the game maintains its novelty by adapting to the player’s current ability level. Therefore if one homeostat is a human player, the first requirement of an automated adaptive AI system is homeostasis of the player’s experience. This means being able to keep up with the player’s inevitable learning of the game system. Once that requirement is met, then the system can be tweaked to provide different levels of difficulty, types of experience, etc.

Learning Learning is a key aspect of game play, and the fact of learning implies the necessity of some form of teaching. At the least, one can say that there is a didactic process inherent in the way game content is structured so that the player can learn it without being over- or underwhelmed. In a standard game, designing how this structure is revealed during play is the job of the game developer. In an adaptive game, the adaptive AI is forced to deal with player learning, perhaps by constraint to a given possibility space.

Controlling the pace of learning of players is integral to a game’s design, as the quality of the play experience depends heavily upon it. Some games demand

mastery with a levelling or ‘power-up’ structure in a very discrete, ‘building block’ way. Another class of games have learning built into the basic structure of game play. For instance Tetris or Chess have relatively easy-to-learn mechanics but a great depth of emergent complexity, and the pace of learning follows the player’s own ability to uncover this complexity, enabled through practice.

DDA attempts to address these issues, but it must deal with a major hurdle: players vary in how much challenge they want to face, and DDA smoothes out the challenge. In other words, some players want to be challenged beyond their current abilities, and grow in skill to meet the challenge by replaying sections of the game over and over until they conquer the game. At the other end of the scale, some players want simply to wander, enjoy the game world and never be overly challenged, as discussed next.

Personality & interaction style The act of playing requires an attitude to the game being played that constitutes a personality, even in the case of AI agents where ‘attitude’ would only be attributable on observation by humans. The act of play requires commitment to a course of action that ends with an invested outcome, winning or losing being the most common type of outcome. The *perseverance* of a player, and their particular *style* in undertaking the play actions, contribute to their play personality. Perseverance in play refers to continuing in the face of setbacks, e.g. replaying until the point of mastery of a level, strategy, or skill. Style of play is used to encapsulate the differences between players in their approach to play tasks. As with any form of personality, a play personality is not to be thought of as static but quite contextualised and relative. It thus requires constant monitoring, with a dynamic player model.

Adapting play based on a fixed metric of player performance ignores the opportunity to refine adaptive AI based on types of players. This is a major problem with DDA, because adjusting difficulty adjusts the challenge of the game, and one difference between types (in many of the existing player typologies) is their preferred level of challenge. For instance, in the DGD typology [1], the pure *Conqueror* type requires very high challenge, while the pure *Wanderer* type requires stress-free play, i.e. little or no challenge. These are mutually exclusive and yet core requirements (of each type), so a game that ignores this in favour of adapting only to the player’s evinced skill risks alienating both types. The subtle indicators of a player’s type are in their *approach* to play, not their skill in playing. Thus adapting play based on an evaluation of the player’s type involves shifting the focus of play *overall*, encompassing cohesive changes to difficulty, reward structure, aesthetics and automated assistance.

A player approaches a game from the unique perspective of her own play history and personality, as discussed above; in addition, players vary fundamentally in information processing styles. This has been addressed in the study of temperament theory, which is regarded as biologically based, while personality is culturally based. I, and others, have previously covered this topic in detail [10, 13, 2]. Thus it is sufficient to reiterate that there is a close link between the interaction styles that characterise people, and the patterns which reoccur

throughout game design – and this is no accident because games are human artefacts designed for human minds.

Behavlets As stated in the Introduction, a general model should provide insight into different facets of player behaviour, for example the cognitive information processing ‘style’ of a player. It thus requires a foundation of parameters that describe the subjective experience of play. The foundation will draw on established modelling tools, including at least: i) psychology of behaviour; ii) general game design; and iii) actions in the context of a given game.

I previously proposed the *Behavlets* method [10] to build facets i) to iii) above into composite features of game-play defined over entire action sequences. The aim is to create player-modelling features linked to valid psychological theory. The Behavlet process integrates descriptive models for temperament theory, game design patterns, and patterns of player actions. The core concept is to capture behaviours with certain known bias of personality; e.g. aggression, caution; and thus observe the players’ self-expression. Behavlets have been used to model players for, e.g. personality type classification [9] and move prediction [14]. Thus I use the Behavlets method to fulfil requirement i (from the Introduction): to capture information about player psychology (cognition, emotion, personality).

How to fulfil requirement ii, and represent the Behavlet model in the context of a game AI? Below I propose an abstract mathematical formalisation, to provide a foundation for more specific solutions in future.

3 Part B: A General Player Model Proposal

This section aims to provide a notation to represent Behavlets as action sequences in a formally defined simulation of a game system, by extending [32]. The motivation is to generate a representation of possible player actions, and the archetypal behaviour traits that can shape those actions, such that the representation can be used as input for a learning system. Ultimately, the goal is to learn from real human behaviour.

3.1 Background

Formalism To represent Behavlets (or any other psychological model) ‘in the game’, i.e. in a manner both machine- and human-comprehensible, is (in principle) best done by *simulation*. As stated in the inspirational work of [18], simulation “models’ main purposes are to leave out certain aspect of complex systems to facilitate study of those systems.”

Note that games can be neatly modelled as a mathematical system because they rely on rule-based interactions defined on a possibility space, and the mechanics of play are essentially functions over that space.

Restricting the games under consideration to those with strictly bounded rules, observe that a state at time t is determined by the game state at time $t - 1$.

Thus the game can usually be represented by a finite-state Markov process⁴. A state-based model is often used for game representation, and Markov methods are often used for computational intelligence in games [35].

However, observe that play involves spaces and control systems; these can be either discrete, or approximately continuous with minimum lower bound, sometimes defined by the frame rate of e.g. 60fps or 16.67ms per frame.

For purpose of player modelling, the difference between approximate and truly continuous is not as important as the player’s *understanding* of the nature of the play space. A general player model must capture the player’s understanding, and deal with ‘approximately continuous’ data.

If the model must capture every frame of the game, it is hardly an efficient simulation. Far more parsimonious to use a modelling framework that can handle continuous entities.

For example, consider Go played with clocks. Players make a single discrete move while their clock elapses continuous time. The elapsed time value can be captured with a simple integer, but the elapsed psychological experience cannot.

Fortunately, the required tools are already in [32]’s category theory framework to model interactive control systems. The framework in [32] models both discrete and continuous control systems, in hybrid form and as abstraction simulations. I will draw on the definition of hybrid control systems (HCS), following [18] and building on [32].

[18] is an excellent complement for the reader; it works lucidly through the foundational technical aspects of applying this formalism to games. It also concludes at about the point where I aim to depart: the composition of micro-games (e.g. Behavlets) to form complete games (e.g. player models). The approach is more applied than [18], but as in that paper the aim here is to produce a simulation model with reduced complexity compared to the original game.

[18] described the *how* of game specification using HCS methods, but he himself questioned *why* one would wish to do it. I am interested in providing this motivating vision.

Prior Work A general player model has the difficult task to account for the variation between players, variability in their behaviour over time, and the reciprocal relationship of players to the game. For example, such a model should account not only for player learning, but also player emotions’ impact on play. There are many relevant fields of study in that problem, and I have previously reviewed literature contributing to generalised player modelling [13]. Below, literature on formal models is briefly reviewed.

Various *descriptive* models of game play have tried to include aspects of player psychology, such as emotions. For example, I proposed the User-System-Experience (USE) model [7, 8], to describe the intrinsic motivation of games

⁴ With a non-rational learning human player at the core of gameplay (who may display high *choice variance*, i.e. infer different predicates based on the same observations), game processes are usually strictly non-Markovian; however they can still be given Markovian representations as a simplifying assumption.

in terms of the cognitive neuroscience of information processing and learning. However the specification of games themselves was lacking in detail. Järvinen [21, pp.99-247] built a player experience model on top of a game decomposition theory. The model has two concepts: game experiences are composed of sequences of emotions; and game elements embody conditions that elicit emotions. [17] define a formalisation of ‘synthetic’ emotions using Decision Theory, to be used for player modelling or for communication of AI agent states to the player. Methods which codify game mechanics allow a model to capture player-game interactions. [28] attempts this, using the object oriented programming paradigm to define game mechanics as “methods invoked by agents”. [4] developed a formal modelling toolset to analyse player behaviour by action sequence mining. The method finds all action sequences and their frequency in a game log, representing common sequences as features, which are selected by ranking according to their mutual information with the class variable.

Formal specification of the play space can support the integration of game and psychological models. [33] defined game theory, which gives useful tools to analyse player behaviour: assuming that players are rational agents with definable utilities for action. Such assumptions do not serve our purpose to learn from real human behaviour. More generally, formal methods such as category theory [34], enable specification and verification of the objects and actions of the play space, and thus support rigorous testing of system coherence. Category theory was applied to game specification in [18], which leveraged [32]’s system of notation for abstractions. In [18]’s abstract specification, a game “consists of objects which change their state during the play, where the evolution of their state is governed by rules and influenced by the players or other objects”. [18] defined a game as a *triple* $(S, \mathcal{M}, F)$, where S is a set of game states; $\mathcal{M}$ is a monoid describing the inputs to the system; and F is an action of the monoid on the set, i.e. the rules. [18] also showed how the operation of composition defined in [32] could be used to create novel games; this was a useful abstract discussion.

This approach is flexible, but the complexity of the domain poses a large problem for this method. [18] agrees: “describing a game with this formalism seems to be a cumbersome task”. The task is cumbersome because the approach relies too much on one system; any such system will be either unwieldy or insufficiently descriptive. In a multi-step modelling approach, methods for action-tracking [4], design pattern analysis [3], and player psychology profiling [10] can first describe the game; i.e. requirement i) above. These descriptions can then be associated with a coding formalism for rigour, i.e. requirement ii) above.

Next section describes such an approach – note, some background knowledge about HCS literature and algebraic definitions is assumed.

3.2 Formal Model

Here, the full HCS defined by Tabuada et al [32] is extended. A game is modelled as a HCS; some rectifying operations are also defined, to force the HCS to behave as games do.

Model Foundation

Definition A game $G = (\mathcal{X}, \mathcal{M}, \Phi)$, consisting of:

- the state space $\mathcal{X} = \{\mathcal{X}_q\}_{q \in Q}$
- a monoid $\mathcal{M} = \prod_{n \in \mathbb{N}} (U^* \cup \Sigma^*)^n$
- a partial action Φ of $\mathcal{M}$ on $\mathcal{X}$, such that there exist invariants $Inv(q) \subseteq \mathcal{X}_q$

Note Here, $\mathcal{X}$ is a set of smooth manifolds parametrised by discrete states $q \in Q$; this allows modelling of any simulated spaces with entities, such as a game’s 3D environment with typical player-controlled unit(s) and opponent(s).

Note Monoid $\mathcal{M}$ is defined as the product union of the sets U^* , the set of smooth manifold inputs, and Σ^* , the set of discrete inputs⁵; which allows modelling of combined analogue and digital inputs, such as a joystick and buttons. Individual inputs are denoted by m , a map in $\mathbb{N}_0^+$, defined as a composition of finite $u^{t1\dots i}$ with finite $\sigma_{1\dots j}$. In this system, $u^{t'}$ indexes time, with ‘embedded’ discrete inputs from Σ^* , if modelling game time is required.

Note The partial action Φ implies a ruleset that can be defined over a subset of the state space; this allows modelling of rules such as power-ups, which alter some core function in a restricted area of state-space, i.e. after a power-up item has been consumed, and perhaps within limited time/space.

This general-form model may be revised to obtain the core framework for specific games. For example perfect-information purely-discrete games, such as Chess and Go, can be obtained when $\mathcal{X}_q$ is a singleton and $U = \emptyset$.

Example Let us model the game of *Noughts & Crosses* (TicTacToe in American) as a demonstration.

- $X_{\mathbf{x}\mathbf{o}} = (pos_\psi)_{\{\psi \in 1\dots 9\}}$, the set of 3×3 board positions, uniquely ordered by the magic square $n = 3$ ⁶.
- $\mathcal{M}_{\mathbf{x}\mathbf{o}} = \sigma_{\mathbf{x} \in \psi} \cup \sigma_{\mathbf{o} \in \psi}$, the act of placing an $\mathbf{x}$ or an $\mathbf{o}$.
- $\Phi_{\mathbf{x}\mathbf{o}} = \phi : \{1, 2, 3\}$, a map to three ‘rules’,
 1. $\sigma_{\mathbf{x}} \times \sigma_{\mathbf{o}} \longrightarrow {}_4P_\psi$, paired player turns involve sampling without replacement from the magic square $n = 3$, up to four times,
 2. $\mathbf{x} \cap \mathbf{o} = \emptyset$, choices are disjoint,
 3. $win \iff \sum \Sigma^* = 15$, winning condition such that player wins if and only if 3 choices sum to 15.

⁵ Although continuous systems are constrained to have finite duration of input times, they may have infinite number of inputs defined as vector field maps from an input manifold. This permits a model consistent with the player’s point of view, which is an important part of creating psychologically relevant models.

⁶ This is a rare occasion when a magic square becomes a Magic Circle (in the sense of Huizinga, not Yang Hui)!

Although *Noughts & Crosses* is a trivial child's game, it is a simple matter to adapt this specification to model *Gomoku*, which is also an m, n, k -game. From there, it is straightforward to model Go, at least for $(\mathcal{X}, \mathcal{M})$. To define Φ for the core Go rules, which for brevity are not stated here, would require significant effort but tractable complexity because the rules are all simply derived from the board and input definitions $\mathcal{X}, \mathcal{M}$.

In order to more flexibly create games, it helps to exploit modularity; for this one can use the operation *composition of monoids*. Composition implies that, given two monoids $\mathcal{M}_1$ and $\mathcal{M}_2$, one can form the composition $\mathcal{M}' = \mathcal{M}_1 \otimes \mathcal{M}_2$, which is also a monoid. $\mathcal{M}'$ has all possible evolutions of the composed monoids and no interaction between their parts.

For such a model $(\mathcal{X}, \mathcal{M}, \Phi)$ as described, a common shorthand notation is $\Phi_{\mathcal{X}}$, denoting $\Phi_{\mathcal{X}} : \mathcal{X} \times \mathcal{M}_x \rightarrow \mathcal{X}$. With this notation, and composition, one can thus describe a basic game, Φ_0 , and compatible game-parts $\Phi_{\mathcal{X}a}$ and $\Phi_{\mathcal{X}b}$, and obtain a complete game by composition, $\Phi_{\mathcal{X}b} \times \Phi_{\mathcal{X}a} \times \Phi_{\mathcal{X}0} \rightarrow \Phi_{\mathcal{X}ab}$. The goal is that such game-parts are used to represent Behavlets, as described below, section 3.3.

However as [18] pointed out, with such a framework it is not yet possible to build any reasonably interesting game, in the sense of a system which produces meaningful decisions and outcomes [27]. This is because the composition operator does not impose any interaction on the composed parts, leaving the resultant system causally heterogeneous and un-gamelike. Composition should also impose constraints on the composed monoids, such that the inputs of each are influenced by the other. And, tracking activity patterns allows us to see more clearly how the defined influences work in practice. Thus, two more concepts complete our core toolset: **composition with restriction**, and **orbits**.

Definition **Composition with restriction**, denoted $\otimes$, from [32], imposes a restriction of $\Phi_{\mathcal{X}}$ to a subset of $\mathcal{X} \otimes \mathcal{M}'_x$, such that the composed monoids are forced to synchronise by the restriction.

Definition An **orbit** is a set O_x containing all points visited on an evolution starting at x and controlled by some input $m \in \mathcal{M}$. Formally, $O_x = \{x' \in \mathcal{X} : x' = \Phi_{\mathcal{X}}(x, m') \text{ for some prefix } m' \text{ of } m\}$. [34] defines an orbit as the behaviour of an imperative program f , i.e. the effect of a series of inputs a on an initial state x , such that $f_{ai}(x_{i-1}) = x_i$.

For our purposes, an orbit of monoid $\Phi_{\mathcal{X}}$ will represent instances when the game-play activity pattern defined by $\Phi_{\mathcal{X}}$ is played. For example, in Noughts & Crosses there are well-known tactics, which when played according to the correct selection criteria will generate a perfect game. These include *Play Center*, *Block*, *Fork* and others defined in [15]. They can be modelled with a triplet, defining: player's move, state of the board, and a test for the type of tactic played.

In the game of Go there are also various well-known patterns of play, such as *atari*, *gote* vs. *sente*, *joseki*, *ko* fighting. These concepts may be captured by orbits, when the analysis of the pattern characteristics is algorithmic and tractable (as in Go, some expert judgement is often involved in assessing patterns).

To make a well-formed map from games to models, an orbit for modelling behaviour is under two constraints. It cannot be a cycle, as cycles are not game-like: consider the *ko* rule in Go. It cannot consist only of a stable state, as this cannot evolve (by definition [34]), and is therefore uninteresting from a player modelling point of view.

Complete Model Based on this framework, a modelling scheme for game behaviour is proposed, which is descriptive rather than generative; i.e. the aim is to simulate the game components that relate to player actions, rather than deriving a simulation of game play from a model of the engine mechanics.

Definition A game G' is a composition of HCSs $\Phi_{\mathcal{X}_i}$, $1 < i \leq k$, where each $\Phi_{\mathcal{X}_i}$ is used to model a distinct game play pattern, and the composition (by the properties of composition of monoids) is also a HCS.

The ‘base’ monoid $\Phi_{\mathcal{X}_0}$ represents the core game framework, with no orbit restrictions. A single game play pattern is represented by a monoid, $\Phi_{\mathcal{X}_i} : (\mathcal{X}_i, \mathcal{M}_i, \Phi_i)$, instantiated by an orbit $O_{\mathcal{X}_i}$, with a starting condition $(q, x_0)_i$, an initial condition from $\mathcal{X}_{q_i}$ which corresponds to the opening state of the game pattern. Such monoids are constrained from having initial or terminal objects (as defined by [34]), because they would then be allowed to define only a single function, violating the principle that games should be uncertain.

Modelling of the complete game is achieved by composition with restriction, where three restriction operators are defined.

1. $\mathcal{X}_{q_i} \subseteq \mathcal{X}_{q_{i-1}}$, i.e. state space is reduced every time by composition. This models the progression of games, i.e. the fact that a game can generally be modelled as a tree traversal, such that every move will reduce the remaining possible states.
2. $m_i \otimes m_{i-1}$ iff m_{i-1} is a prefix of m_i , such that e.g. in a time-indexed system $m_{t^1} \geq m - 1_{t^1}$, i.e. the start time of the orbit for the next monoid to be composed must be greater than or equal to the prior monoid, such that game progression is modelled.
3. $O_{x_i} \neq O_{x_{i-1}}$ where $(q, x_0)_i \cap (q, x_0)_{i-1} \neq \emptyset$ without any other restriction on $x \in O_{\mathcal{X}_i}$. I.e. monoids composed such that their orbits have overlapping time sequences, shall not be isomorphic.

Thus, based on this approach, a game G' is a basis monoid $\Phi_{\mathcal{X}_0} : (\mathcal{X}_0, \mathcal{M}_0, \Phi_0)$, which provides the complete state space and time registration, without inputs. The basis monoid is composed with $1..k - 1$ additional game pattern monoids, to describe those activities in the game that reduce the possibility space until game end. Each pattern monoid Φ_i is restricted to join the base monoid in a time-ordered fashion, without overlap of isomorphs, and without expanding the game tree with nodes excluded by previously composed monoids $\Phi_{1..i-1}$.

3.3 Behavlets-based formal model

As mentioned, the intention is to formalise the ‘Behavlets’ player modelling method. Per the formal model definition, this can be done by using orbits to represent the play patterns arising in-game, as elaborated here.

A Behavlet is essentially a game play pattern associated with a temperament trait. Thus, a well-chosen monoid representation of a pattern, $\Phi_{\mathcal{X}i}$, can also represent a Behavlet, and thereby be associated with a temperament trait. To select the right monoid for the Behavlet is quite straightforward. The instantiation orbit $O_{\mathcal{X}i}$ is equivalent to the Behavlet logic, defined in [10]. Further, the orbit starting condition $(q, x_0)_i$ is equivalent to the Behavlet concept of a *constraint harness*, defined in [10].

An example will illustrate the approach, for which are used already peer-reviewed [10] and empirically tested [14] Behavlets, derived for the game *Pac-Man* (Namco, 1980). The formal framework for the Behavlets model of a given game Γ is obtained and used with the following five step process:

1. define basis monoid for Γ
2. define compositional Γ play pattern monoids, each with temperament trait
3. define model *instance* as label for play personality in Γ
4. model reduction by the operation of *simulation*, giving representations of behaviour patterns in Γ -like games which can be compared
5. obtain generalised player model by iterating this process

I will apply this process to the Pac-Man Behavlets using the Pac-Man specification from my previous work (see e.g. appendix D to [9]). This specification was totally state-based, discretising the smooth movement of original Pac-Man. Thus, as with Chess or Go, this Pac-Man does not need U^* , and $\mathcal{X}$ is singleton. For brevity, further simplifications are made which do not relate to the example Behavlet. Only a single level is modelled, to avoid extra complications surrounding *tests* that would be needed to model end-of-level or loss of lives (for a description of test function construction, see e.g. [34, pp.46]). There are no extra state variables to model bonus items, points scored, or Pac-Man’s lives (these values are referenced but not defined). There is also no model of the driver of Ghost behaviour, which in the prior specification is simply a probabilistic map to adjacent positions, weighted toward Pac-Man in normal play and away from Pac-Man when a power pill is in effect. These features can be trivially added.

First, the basis monoid for a Pac-Man game $\mathcal{P}$:

Definition $\Phi_{\mathcal{P}} = (\mathcal{X}_{\mathcal{P}0}, \mathcal{M}_{\mathcal{P}0}, \Phi_{\mathcal{P}0})$,

- $\mathcal{X}_{\mathcal{P}0} = \{mat, xy_p\}$, where $mat = (hpos_x) \cup (vpos_y)$, $\{x, y \in \mathbb{N}, 1 \dots 20\}$ the Pac-Man ‘map’ matrix with values drawn from $\{\emptyset, wall, pill, powerpill\}$; and xy_p is a set of current position values for Pac-Man and the Ghosts $\{xy_p \mid p \in PM, G1..4\}$,

- $\mathcal{M}_{\mathcal{P}_0} = m : \{\leftarrow, \uparrow, \downarrow, \rightarrow\} \times \mathcal{X}_{xy_{PM}} \xrightarrow{d=1} \mathcal{X}$, a map from the four directions of movement to the matrix position adjacent to Pac-Man's current position,
- $\Phi_{\mathcal{P}_0} = \phi : \{1, 2, 3\}$, a map to three 'rules',
 1. $m \times pill \rightarrow \{+5_{points}, \mathcal{X}_{xy_{PM}} = \emptyset\}$, Pac-Man passes through a matrix position with a pill: increase points by +5, position becomes empty,
 2. $m \times powerpill \rightarrow \{+10_{points}, \mathcal{X}_{xy_{PM}} = \emptyset, \phi = \phi' : \{1, 2, 3'\}, t : (1..n)\}$, similar effects as *pill*; also transition to the map ϕ' , where vulnerability of Pac-Man to the Ghosts is inverted for a limited time n ,
 3. $m \times xy_{G1..4} \rightarrow -1_{life}$, Pac-Man and a Ghost enter the same matrix position: Pac-Man loses a life,
 - 3' $m \times xy_{G1..4} \rightarrow \{+50_{points} \times i, i \in (1..4), xy_{G_i} = xy_{0G_i}\}$, Pac-Man and a Ghost enter the same matrix position: +50 points (multiplied by consecutive Ghost order), Ghost returns to starting position

Second, the Behavlets themselves are modelled. To illustrate, I select a Behavlet listed in [10, pp.293], *A1-Hunt Close To Ghost House*. Behavlet *A1* (for short) tracks how often a player follows the Ghosts right up to their house while attacking them in *powerpill* mode.

Definition $\Phi_{A1} = (x', m', O_{\mathcal{P}A1})$,

- $x' = \forall 1..4, \text{dist}(xy_{G_i}, xy_{0G_i}) \leq 3$, the manhattan distance of each Ghost to its own starting position is three or less,
- $m' \subseteq \mathcal{M}_{\mathcal{P}_0} \forall t_{(1..n)}$, the orbit elapses for all inputs until the end of the *powerpill* timer,
- $O_{\mathcal{P}A1} = \{x' \in \mathcal{X}_{\mathcal{P}_0} : x' = \Phi_{\mathcal{P}_0}(x, m')\}$, an orbit defined on the Pac-Man basis monoid

Third, the composited game is produced, $\mathcal{P} = \Phi_{A1} \otimes \Phi_{\mathcal{P}} = \Phi_{\mathcal{P}A1}$. Game instances where this Behavlet monoid appears can be labelled as examples of *cautious play*, with a quantification scheme as described above.

Fourth, model reduction by simulation creates a simpler representation without reference to the specifics of the game. Thus one does not need to define, for example, the dimensions of game space states $\mathcal{X}_q$, only to define the *type* of spaces as they appear to the player. In this way, one can make an equivalence between models for Pac-Man, Go, even Noughts & Crosses if wished. Given the game produced by composition with restrictions $\Phi_{\mathcal{P}A1}$, two simulations of the composed parts are defined: β_{A1} and $\beta_{\mathcal{P}}$. The map φ which defines each β is an abstraction of the part of the model *which is not relevant to comparison with another simulated game*. β s are composed by restriction to produce a more general version of the model, $\beta_{\mathcal{P}A1}$.

$$\begin{array}{ccccc}
 \beta_{A1} & \otimes & \beta_{\mathcal{P}} & \longrightarrow & \beta_{\mathcal{P}A1} \\
 \varphi \uparrow & & \varphi \uparrow & & \uparrow \\
 \Phi_{A1} & \otimes & \Phi_{\mathcal{P}} & \longrightarrow & \Phi_{\mathcal{P}A1}
 \end{array} \tag{1}$$

The complete approach to simulation is detailed in [32], and is also discussed in [18]. Here it is enough to note that, given proven models of games such as those described above, the reduction can be pursued and the simulations are then ‘safe’ to study without reference to the messy details.

Fifth and finally, obtaining the generalised player model from the given framework is perhaps possible for a class of games between which simulation is well-defined. Proving this is clearly a matter for future work.

4 Discussion

It is clear that adaptive AI can be enhanced using in-depth knowledge of the player in real-time. It is our primary argument in this paper that a generalised player model is the most efficient way to capture player individuality which is required for good function of adaptive AI. Properly constraining adaptive AI should therefore be a function of a player modelling approach.

The approach described in Section 3 for a generalised player model draws on the Behavlet method to create psychologically-based features of game play, and redefines them as parts of a category theoretic formal model. The value of this approach is that, under a formal framework, Behavlet models of particular games can be further generalised by the operation of simulation.

The primary use case for the described method is to capture player variation. Consider that, if Behavlets are modelled as game parts $\Phi_{\mathcal{X}}$, then by the Behavlets method [10], each $\Phi_{\mathcal{X}_i}$ will have an associated behaviour trait. Thus, a game instance with a specific Φ composition will reflect the ‘character’ of a particular player’s play style. Potentially, characteristics of human play can be learned through enough such instances.

Also consider that the method allows abstraction and specificity: one can build a canonical game model and also simulate game instances quite easily from the same definitions. This allows exploration of the space of possible games.

This is a work at the concept stage, and like any concept there are many details lacking. The state of the method presented is probably sub-optimal, and this may frustrate the more engineering-minded reader; but the aim is initiate a conversation. It is to be hoped that the concept will provide fertile soil to grow more detailed methods.

In future work, we will study a single game under a finite set of adaptive AI conditions. We will use Behavlets [10] to provide the generalised player model, and the formal method (described above) to provide a rigorous framework for comparison. We will iteratively evaluate progressively more complex adaptations of this testbed game. The testbed game, such as a first-person shooter, will be chosen to provide a well-understood experience (in the sense of being well-studied), and also a rich activity (in the sense of allowing players to express varying behaviours). [10] has guidelines on choosing a viable testbed game.

Thus, in the first instance the formally defined player model will be built for the testbed game with no adaptation conditions. The game will then be altered in a simple way, in order to demonstrate abstract game comparison-by-simulation

[11], comparing the original and altered versions. Thereafter, adaptive AI will be added to the modelled game, in multiple conditions with increasing complexity. Based on the generalised player model, each adaptation will be designed to respond to some aspect of the player profile, for example a player's tendency towards cautious play, creating multiple versions of the game. These versions will be evaluated by play testing to establish whether the generalised player model does in fact support adaptive AI. To further demonstrate that the system can facilitate adaptive AI, the versions with more complex adaptations will be compared with the simpler versions, using the simulation features of the formal model. The threshold for complexity will be set where it is no longer possible to manually test all possible adaptive outcomes. This will show that issues of logical consistency can be dealt with more readily when the profile of the player is known through a formal generalised player model.

References

1. Bateman, C., Boon, R.: 21st century game design, vol. 1. Charles River Media, London (2005)
2. Bateman, C., Lowenhaupt, R., Nacke, L.: Player Typology in Theory and Practice. In: Digital Games Research Association conference. Utrecht, Netherlands (2011)
3. Björk, S., Holopainen, J.: Patterns in game design. Charles River Media, Hingham, Massachusetts (2005)
4. Breining, S., Kriegel, H.P., Schubert, M., Zuffe, A.: Action Sequence Mining. In: Croonenborghs, T., Driessens, K., Missura, O. (eds.) Second International Workshop on Machine Learning and Data Mining in Games, at European Conference on Machine Learning. Athens, Greece (2011)
5. Caillouis, R.: Man, play, and games. Free Press of Glencoe, New York (1961)
6. Chanel, G., Rebetez, C., Bétrancourt, M., Pun, T.: Emotion Assessment From Physiological Signals for Adaptation of Game Difficulty. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* **PP**(99) (2011)
7. Cowley, B., Charles, D., Black, M., Hickey, R.: User-System-Experience Model for User Centered Design in Computer Games. In: Adaptive Hypermedia and Adaptive Web-Based Systems. vol. 4018, pp. 419–424. LNCS, Dublin (2006)
8. Cowley, B., Charles, D., Black, M., Hickey, R.: Toward an understanding of flow in video games. *Comput. Entertain.* **6**(2), 1–27 (2008)
9. Cowley, B., Charles, D., Black, M., Hickey, R.: Real-time rule-based classification of player types in computer games. *User Modeling and User-Adapted Interaction* **23**(5), 489–526 (Aug 2013)
10. Cowley, B., Charles, D.: Behavlets: a Method for Practical Player Modelling using Psychology-Based Player Traits and Domain Specific Features. *User Modeling and User-Adapted Interaction* **26**(2), 257–306 (Feb 2016)
11. Cowley, B.U.: How to advance general game playing artificial intelligence by player modelling. *arXiv* **1606.00401** (Jun 2016)
12. Cowley, B.U., Charles, D.: Adaptive Artificial Intelligence in Games: Issues, Requirements, and a Solution through Behavlets-based General Player Modelling. *arXiv* **1607.05028** (Jul 2016)
13. Cowley, B.U., Charles, D.: Short Literature Review for a General Player Model Based on Behavlets. *arXiv* **1603.06996**, 7 (Mar 2016)

14. Cowley, B.U., Charles, D.: Utility of a Behavlets approach to a Decision theoretic predictive player model. arXiv **1603.08973** (Mar 2016)
15. Crowley, K.: Flexible strategy use in young children's tic-tac-toe. *Cognitive Science* **17**(4), 531–561 (dec 1993)
16. Espejo, R., Harnden, R.: *The Viable system model : interpretations and applications of Stafford Beer's vsm.* J. Wiley, New York (1989)
17. Gmytrasiewicz, P.J., Lisetti, C.L.: *Modeling users' emotions during interactive entertainment sessions* (2000)
18. Grünvogel, S.: Formal Models and Game Design. *Games Studies* **5**(1) (2005)
19. Huizinga, J.: *Homo ludens: a study of the play-element of culture.* Routledge, London (1949)
20. Hunicke, R., LeBlanc, M., Zubek, R.: MDA: A Formal Approach to Game Design and Game Research. In: *Proceedings of the Challenges in Game AI Workshop.* AAAI Press, Stanford University in Palo Alto, Ca (2004)
21. Järvinen, A.: *Games Without Frontiers: Methods for Game Studies and Design.* VDM Verlag, Copenhagen (2009)
22. Koster, R.: *A theory of fun for game design.* Paraglyph Press, Scottsdale, AZ (2005)
23. Malone, T.W.: What makes things fun to learn? heuristics for designing instructional computer games. In: *SIGSMALL '80: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems.* pp. 162–169. ACM Press, Palo Alto, California, United States (1980)
24. Malone, T.W.: What makes computer games fun? [for education]. *BYTE* **6**(12), 258–277 (1981)
25. Mnih, V., Kavukcuoglu, K., Silver, D., ..., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (Feb 2015)
26. Rauterberg, M.: About a framework for information and information processing of learning systems. In: Falkenberg, E.D., Hesse, W., Olivé, A. (eds.) *Information System Concepts: Towards a consolidation of views,* pp. 54–69. IFIP Advances in Information and Communication Technology, Springer US, Boston, MA (1995)
27. Salen, K., Zimmerman, E.: *Rules of play : game design fundamentals,* vol. 1st. MIT, London (2004)
28. Sicart, M.: Defining Game Mechanics. *Games Studies* **8**(2), Online (2008)
29. Silver, D., Huang, A., Maddison, C.J., ..., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (Jan 2016)
30. Steinkuehler, C., Duncan, S.: Scientific Habits of Mind in Virtual Worlds. *Journal of Science Education and Technology* **17**(6), 530–543 (2008)
31. Swalwell, M., Wilson, J., Kücklich, J.: Forbidden pleasures - cheating in computer games. In: *The pleasures of computer gaming : essays on cultural history, theory and aesthetics.* McFarland & Co., Jefferson, N.C. (2008)
32. Tabuada, P., Pappas, G.J., Lima, P.: Compositional Abstractions of Hybrid Control Systems. *Discrete Event Dynamic Systems* **14**(2), 203–238 (apr 2004)
33. Von Neuman, J., Morgenstern, O.: *Theory of games and economic behavior.* J. Wiley, New York (1944)
34. Walters, R.: *Categories and computer science.* Cambridge University Press, Cambridge (1991)
35. Yannakakis, G.N.: Game ai revisited. In: *Proceedings of the 9th Conference on Computing Frontiers.* pp. 285–292. CF '12, ACM, New York, NY, USA (2012)
36. Yannakakis, G.N., Togelius, J.: A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games* **7**(4), 317–335 (Dec 2015). <https://doi.org/10.1109/TCIAIG.2014.2339221>