Foundations of a Computational Science of Game Design: Abstractions and Tradeoffs

Rogelio E. Cardona-Rivera
Laboratory for Quantitative Experience Design
School of Computing and the Entertainment Arts and Engineering Program
University of Utah, Salt Lake City, UT, USA
rogelio@eae.utah.edu

Abstract

This paper presents the foundations of a computational science of game design—a model of abstraction: the outcome of a situated design process that conceptualizes *something* in terms of *something else* for a specific purpose. The outcome of abstraction is modeled via an *abstraction scheme*, a step toward modeling the process of game design abstraction from a human-centered perspective. A scheme's purpose is couched relative to a *scheme's properties*, also defined herein; collectively, the properties represent the space of tradeoffs designers must navigate for engineering virtual worlds. This model's analytical traction is evidenced by applying it to the design of pathfinding, a core behavior in artificial intelligence for games. More broadly, it is a foundation for shared progress because it affords directly comparing *particular* abstractions of concepts and phenomena across the gamut of research on intelligent systems in entertainment.

Introduction

Game design problems remain *wicked* (Conklin 2005). They are not fully understood until solved (Washburn Jr et al. 2016), have no a priori stopping criteria (Kultima and Alha 2010), and are unique one-shot operations (Murphy-Hill, Zimmermann, and Nagappan 2014). Further, their solutions are neither easily identifiable, nor strictly *right-or-wrong* but rather *better-or-worse* (Davis, Steury, and Pagulayan 2005).

Because virtual worlds are computer-mediated, artificial intelligence (AI) systems already assist building and/or independently build these environments (Liapis et al. 2019). AI-powered tools that simulate vast state-trajectories of gameplay (e.g. Robertson, Jhala, and Young 2019, Cook and Raad 2019) or construct games wholecloth (e.g. Summerville et al. 2017, Liapis et al. 2018) promise to help manage the unclear tradeoffs that underlie game design, presently dealt with by "tens, if not hundreds, of person-years of a coordinated effort among artists, programmers, designers, and authors" (Roberts 2011).

Work in this area has thus far focused on understanding aspects of game *artifacts* that enable their procedural generation. Relatively, much less attention has been paid to understanding the *design process* that gives rise to the artifact. Game design is a *situated* practice (Kuittinen and Holopainen 2009): designers, artifacts, and their environments are so intertwined, that they mutually inform and constrain each other. Failing to look at design theory and practice is thus a critical gap in our body of knowledge. The design sciences afford framing questions around AI-powered design from a more human-centered perspective (Cross 2001a). This is in contrast to other excellent efforts around AI-powered design, which seek to have the AI recognized as a creative entity (Cook 2017) or to design games through non-human ways (Zook and Riedl 2014). I begin addressing this gap via a model of game design *abstraction*, posited as a foundational part of game design.

The model is part of a *science of game design*, an application of the design sciences to the study of games. This paper sets forth a basis for the computational simulation of game design abstraction that affords comparing between *different* abstractions of a common phenomenon; I model the *outcome* of abstraction as a step toward modeling it as a *process*. I evidence the model's utility by analyzing a world abstraction for *pathfinding*. Finally, I notionally argue for our model's relevance as a foundation for shared progress. Succinctly, different strands of game AI research are *particular* abstractions over game design concepts and phenomena, which – through our framework – might now be directly comparable in a meaningful way.

Related Work

I propose to use *computational science*—an inter-discipline concerned with the design, computer-implementation, and use of mathematical models to analyze problems writ large (Rüde et al. 2018). A computational science of game design is a form of (conceptual) *theory*-building about game design via the analysis of mathematical models about design phenomena that are implemented on computers. Of the many ways to achieve computational game design, I pursue one rooted in AI. For me, "the value of asking the question 'Can a machine design?' is that it begs the corollary question, 'How do people design?'" (Cross 2001a p.50). Given my mode of inquiry, there are 3 "levels" of related work: conceptual, computational, and mathematical.

Conceptual

Abstraction is an elusive concept to define and computationally model. It is both (1) a *conceptual process* of analyzing and describing *something* in terms of *something else* believed to represent it at a "more fundamental" level, and (2) the *outcome* of said process.

Despite its elusivity, it is thought of as a key process *and* outcome of design (Lee, Pries-Heje, and Baskerville 2011). In game design, abstraction is primarily *ontological*: what is abstracted is the nature of a concrete "real" world, represented by some abstract "virtual" world that serves as the game's context (Fernández-Vara 2011). There are at least 2 types of game design-centric ontology abstractions: state-based and action-based. Abstractions of state target narrative-theoretic features (e.g. Damiano, Lombardo, and Pizzo 2019): examples include space, time, and entities (characters and objects). Abstractions of action target ludic-theoretic features: examples include mechanics, goals, and randomness (e.g. Debus 2019). These two abstraction types are intertwined (Cardona-Rivera, Zagal, and Debus 2020) and foundationally affect gameplay (Juul 2007): players negotiate varying levels of abstraction that help them act toward goals in virtual worlds. I aim to be computationally-precise about what abstraction means (as a product first and then as a process) in game design, which to my knowledge has not been explored by current research. Our model is agnostic to the type of abstraction, but our key example is an abstraction of *space* as relevant to the mechanic of *directed movement*. I expect my model to bear relevance to studying the effect of *particular* abstractions on gameplay.

Abstraction is so ready-to-hand (Heidegger 1962) that we use it as a matter of course in research on AI and interactive digital entertainment (AIIDE). Evidence of abstraction is recoverable through our use of metaphor in language, and appears across a great deal of AIIDE-relevant work; e.g., *Narratives as Plans* (Young et al. 2013), *Games as Conversation* (Cardona-Rivera and Young 2014), *Experience Management as Factored Markov Decision Processes* (Thue and Bulitko 2018), *Content Generation as Quality-Diversity Search* (Gravina et al. 2019), and *Game Interfaces as Programming Languages* (Martens and Hammer 2017).

Each metaphor gives a *particular* understanding of what is abstracted: it *highlights* features that facilitate computational modeling or generation while *hiding* features not thought of as interesting, relevant, or meaningful (Lakoff and Johnson 1980). The mentioned (and countless other) bodies of work represent *one* abstraction and do not reflect on the process that gave rise to the abstraction in the first place. This risks conflating automated task accomplishment with the simulation of human processes used for task accomplishment (Newell, Shaw, and Simon 1960; Searle 1980). The latter is what I seek to model.

Computational

Recent work *on* abstraction includes that by (1) Osborn, Lambrigger, and Mateas (2017) on *HyPED*, which switches between levels of spatial abstraction of platformer/action-adventure games to lower the dimensionality of the search space for planning, (2) Sturtevant et al. (2019) on a mechanism to abstract a virtual space as a directed graph for pathfinding that accounts for dynamic terrain with concordantly dynamic terrain costs, and (3) Diamanti and Thue (2019) on a mechanism that abstracts more-detailed and resource-heavy world states to logically-consistent but less-detailed and cheaper-to-maintain world states (and vice-versa), relative to the player's perception of the world.

In contrast, I aim to show how abstraction is relevant not *just* for planning, pathfinding, or world-simulation by introducing a computationally-realizable mathematical framework that is agnostic to an underlying representation of state or action. Pathfinding is an (obvious) application of our work, and level-of-detail/spatial abstraction is a *particular* form of abstraction can model via our work.

Mathematical

To model game design abstraction, I use the theory of *abstract intepretation* (Cousot and Cousot 1992); this theory has already been mechanized within the field of programming languages, paving the way to a computational implementation of these ideas. Abstract interpretation applies category theory (e.g. Kan 1958) to the field of programming languages, and is concerned with approximating the semantics of computer programs. A *sound* approximation of a target program must preserve the target's concrete semantics. The reason for developing said approximation is to know something about the target that is too expensive to compute (or even undecidable) *via* said program.

For example, imagine you want to verify the sign of a sequence of multiplication operations; whether the result of the sequence is positive, negative, or zero. If the sequence is very long, unrolling the full multiplication might be too expensive in terms of memory or speed. If, instead of using the values, you *only* looked at the signs – i.e. swap every value for $+1$, -1, or 0 as appropriate – you can determine the resulting sign without incurring a great memory cost.

Ultimately, "looking only at the signs" is an abstraction of the concrete and *more quantitative* semantics of the program (the sequence of multiplications of actual values) to an abstract *more qualitative* (sound, approximate) semantics of the program, which is more efficient to compute over (Kuipers 1994). I extend this idea to model game design abstraction *outcomes*.

Theoretical Backdrop: Game Design Science

Using AI-rooted computational science is a commitment on *how* we should study game design abstraction. This commitment is situated alongside 2 other implicit ones: *what* we study and *why* we pursue it.

Aim: Prediction via Research-*into*-(game)design

This work is part of a game *design science* discipline, in which *game design* is the object of study. I target *prediction* as the research end-goal: I seek causal structure that helps anticipate the effects of designed games on the people that will play them.

This work is *research*-into-*design* (Frayling 1994), which centers on creating *Standard Models* to capture community consensus over design activity; cf. the Standard Model of the Mind (Laird, Lebiere, and Rosenbloom 2017).

Research-into-design stands in contrast to other potential goals. While worthwhile, it is neither *research*-by-*design* – research via the act of designing; i.e. knowledge synthesis through "*designernly* ways" (Cross 2001b) – nor *research*-for-*design*, which aims to articulate knowledge directly relevant to (game design) practice (cf. O'Donnell 2014).

Concepts and Phenomena: FBS and Situatedness

I adopt the situated Function–Behavior–Structure (sFBS) model of the design process (Gero and Kannengiesser 2004). The sFBS model is a knowledge representation (KR, Davis, Shrobe, and Szolovits 1993) of design activity: it defines design as a *situated* (Dewey 1896) process an agent undertakes by reasoning about the function, behavior and structure of an artifact during its construction.

sFBS combines two KRs: FBS (Figure 1) is artifact-centric, and *situatedness* (Figure 2) is person-centric.

FBS This KR extensionally identifies *what* the designer processes and *how* (Gero 1990). It distinguishes 3 kinds of information, termed "variables." Each variable models a different bit of information about a designed artifact: (1) the **F**unction variables describe what the artifact is *for*, (2) the **B**ehavior variables describe what the artifact *does*; both what is expected (**Be**) and what follows from form (**Bs**), and (3) the **S**tructure variables describe what the artifact *is*; the constituent components and relationships that define its form, used to **D**escribe its construction or manufacture. Variables are manipulated by 8 design "sub-processes:"

(1) Formulation transforms design requirements expressed in **F** into **Be**havior expected to enable said function.
(2) Synthesis transforms expected **Be**havior into a **S**tructure intended to exhibit it.
(3) Analysis derives behavior (**Bs**) that follows from **S**.
(4) Evaluation compares actual **Bs** to desired **Be**, to check if **S** is acceptable.
(5) Documentation produces a **D**escription to construct or manufacture the artifact.

The last 3 processes change variables or values for them in response to **S** being deemed unacceptable by sub-process 4:

(6) Exploration (*Reformulation 1*, sFBS) changes **S**. So-named because **Be**'s conceptual space does not change, reflecting *exploratory creativity* (Boden 2003).
(7) Transformation (*Reformulation 2*, sFBS) changes **Be**. So-named because the conceptual space changes, reflecting *transformational creativity* (Boden 2003).
(8) Re-framing (*Reformulation 3*, sFBS) changes **F**. So-named because it re-conceptualizes the goals, as in *Framing* from reflection-in-action (Schon 1984).

Situatedness This KR models the worlds the agent is situated in; these are so intertwined that the agent's activity is in part determined by said worlds. It distinguishes 3 kinds of worlds that qualify design knowledge and reasoning.

The *external world* $\mathbf{X_e}$ contains the designer, composed of information external to them. The *interpreted world* $\mathbf{X^i}$ is mentally built by the designer via sensation, perception, and acquired knowledge; designers mentally manipulate information within *this* world. The *expected world* $\mathbf{X^i_e}$ is imagined by the designer as the result of their design. Each world modally qualifies each of FBS's variables; i.e. the situated FBS model contains $\mathbf{F_e}$, $\mathbf{F^i}$, $\mathbf{F^i_e}$, $\mathbf{Be_e}$, $\mathbf{Be^i}$, etc.

The worlds are recursively linked via 3 processes that manipulate information from one world's form into another. *Interpretation* goes from the external world to the interpreted world, or from the interpreted world to itself: it is a fusion of sensation, perception, and thought to produce an internal representation of information. *Abstraction* focuses on some aspect of the interpreted world to set goals for design activity within the expected world; this design activity can be *concretized* to simulate expected effects within the interpreted world, or *enacted* in the external world. Our work targets understanding the situated design process of *abstraction*: how game designers focus on certain aspects of their intepretation of some external world to form an expected world, in which they imagine activity that is then effected in the all-encompassing external world.

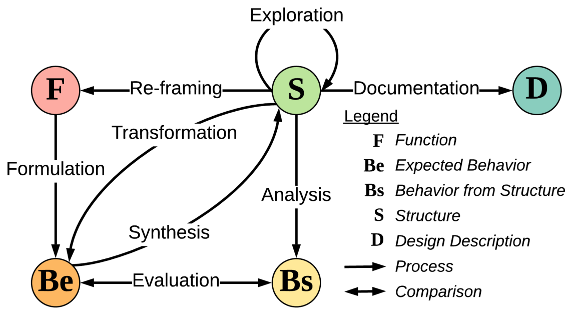

Figure 1: Illustration of FBS, a model of *design activity* as a system of 5 variables (nodes) and 8 processes (edges). Singly-directed edges represent processing of source variables *into* sink variables; doubly-directed ones represent comparing linked variables.

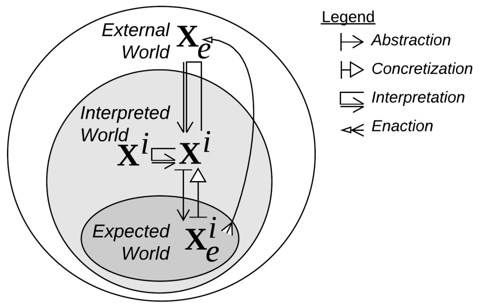
External World $\mathbf{X}_e$
Interpreted World $\mathbf{X}^i$ $\mathbf{X}^i$
Expected World $\mathbf{X}^i_e$
Legend
Abstraction
Concretization
Interpretation
Enaction

Figure 2: Illustration of *situatedness*: 3 worlds that qualify FBS's 5 variables (Figure 1). Designers *interpret* the external world $\mathbf{X_e}$ to mentally enact a model of it $\mathbf{X^i}$, which is subject to (re-)interpretation. The interpreted world is *abstracted* for theorizing solutions the designer wishes to effect in the expected world $\mathbf{X^i_e}$. Solutions are *concretized* for deliberation in $\mathbf{X^i}$ or *enacted* in $\mathbf{X_e}$. Arrows indicate flow of information, always connecting two worlds.

Modeling Design Abstraction via Schemes

Abstraction involves manipulating information from some interpreted world $\mathbf{X^i}$ to some expected world $\mathbf{X^i_e}$; an expected world is theorized *through* abstraction of the interpreted world. We model the *outcome* of this abstraction.

For game design, the interpreted world is scrutinizable: it is the information that defines a virtual world. This world encompasses *geometrical* and *physical* information; the former to define consistent & transformable primitives in Euclidian space, and the latter to define model behavior with respect to light, sound, and motion. A virtual world is an interpretation of the external world, codified via a *physics engine* (Gregory 2017) as an assignment of values for the independent parameters needed to uniquely determine all the properties of every geometric primitive in the virtual world, including position and orientation (with relevant higher-order derivatives), at a time index t. An assignment of values for all such parameters is the world's *state* (at t).

Let C_{VW} be a *partially-ordered set*[1] of these independent parameters; C_{VW} represents a *Concrete* Virtual World. This world contains the more-quantitative reasoning we care to abstract: a function f_C that maps elements from C_{VW} to itself—$f_C\colon C_{\mathrm{VW}} \to C_{\mathrm{VW}}$.[2] In other words, f_C captures the *concrete meaning* one is interested in abstracting. Taken together, the tuple $\langle C_{\mathrm{VW}}, f_C\rangle$ is a **concrete engine**; C_{VW} is its **representation** and f_C is its **reasoning**.

I assume that (for whatever reason) f_C is not *efficiently computable* for the Concrete Virtual World; i.e. either f_C is not computable or f_C's underlying algorithm fails to satisfy a design constraint on the resources it consumes during computation. This motivates introducing an efficiently computable way of reasoning, that operates over an alternate representation of C_{VW} and preserves the semantics of f_C for the reasoning we *want* to perform via f_C. This alternate representation and reasoning is an *abstraction* of a concrete engine and is called an **abstract engine**. The abstraction is *proper* just when $|C_{\mathrm{VW}}|>|A_{\mathrm{VW}}|$.

An abstract engine is defined as a tuple $\langle A_{\mathrm{VW}}, f_A\rangle$, where A_{VW} is another partially-ordered set representing an *Abstract* Virtual World and f_A is a function that maps elements from A_{VW} to itself—$f_A\colon A_{\mathrm{VW}} \to A_{\mathrm{VW}}$. In effect, f_A *is* the abstract reasoning one wants to perform.

An abstract engine is defined relative to a concrete engine via an *abstraction scheme*. Conceptually, the scheme is a method for partitioning the set C_{VW} into subsets of parameters that can be reasoned over, where each subset has some meaning beyond what its elements *intrinsically* represent; this meaning emerges from the structural grouping of the elements in the set. The set of all the subsets in the partition is the *abstracted* representation of C_{VW}, the A_{VW}. For example, one might define a *triangle* as a geometric primitive within C_{VW} as a triple of points $\triangle 123 = \langle p_1, p_2, p_3\rangle$ relative to some Cartesian system in the $\mathbb{R}^3$ Euclidian space; i.e. $p_i \in \mathbb{R}^3$. A common abstraction is to

[1]Thus, certain set elements cannot be relatively ordered; e.g. a mesh's *position* & *orientation* in a C_{VW} are both numerical, but neither precedes the other because they are different concepts.

[2]We omit dimensionality, but all functions can be multivariate.

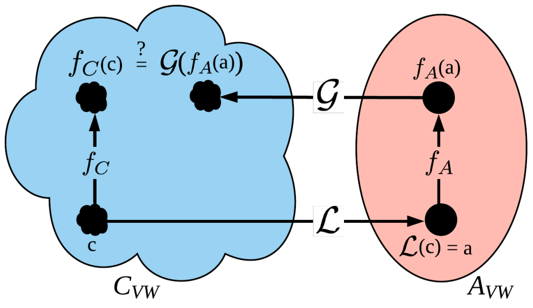

Figure 3: A conceptual diagram of an abstraction scheme. It is defined to carry out more-qualitative reasoning via an efficiently computable function f_A over A_{VW} as a proxy for more-quantitative reasoning via a non-efficiently computable function f_C over C_{VW}. An open question for any scheme is how close the qualitative reasoning matches the quantitative reasoning it aims to abstract; that is, if we let $a \leftarrow \mathcal{L}(c)$, then $\forall c \in C\colon f_C(c) \overset{?}{=} (\mathcal{G} \circ f_A)(a)$.

group 2 or more triangles into a *mesh*; here, the elements of C_{VW} are triangles, whereas those of A_{VW} are meshes.

An abstraction scheme is formalized via two functions. One is a **lifting function** $\mathcal{L}$ which takes elements from C_{VW} and maps them onto A_{VW}—$\mathcal{L}\colon C_{\mathrm{VW}} \to A_{\mathrm{VW}}$. It abstracts element c in C_{VW} by providing its qualitative meaning in terms of the set it is a member of within A_{VW}.

The other is the lifting function's inverse: it is a **grounding function** $\mathcal{G}\colon A_{\mathrm{VW}} \to C_{\mathrm{VW}}$. It concretizes element a in A_{VW} by providing the semantic meaning of a in terms of a potentially-partial assignment of values to elements in C_{VW}.

An **abstraction scheme**, illustrated conceptually in Figure 3, groups the aforementioned elements; i.e. it is a tuple $\Upsilon = \langle C_{\mathrm{VW}}, f_C, A_{\mathrm{VW}}, f_A, \mathcal{L}, \mathcal{G}\rangle$.

An Example Abstraction Scheme: Pathfinding This model affords stating the *result* of abstraction in game design. To evidence its utility, I apply it to pathfinding—the calculation of a suitable route through a virtual world from a start position to an end position, which "is everywhere in game AI" (Millington 2019 p.195).

Pathfinding algorithms typically do not operate over the geometry of the virtual world. Instead, they rely on a *representation* of it, often – not always (cf. Goodwin, Menon, and Price 2006) – a directed weighted graph: vertices represent world areas and weighted edges represent ease of access between connected ones. Abstraction schemes have been called "division schemes" (Millington 2019 p.237) with "quantization" & "localization" functions. I formalize and generalize all three notions.

While one might deem pathfinding irrelevant to game design, I argue it can certainly be a key part. Consider games that involve predicting what enemies will do to survive; e.g. *Firefight* in *Halo: Reach* (Bungie 2010). The path enemies take (f_A) to get closer to the player is critical; characters might find different paths based on health or rank (e.g. *Grunts v. Elites*). Thus, the example is both accessible and directly relevant to game design.

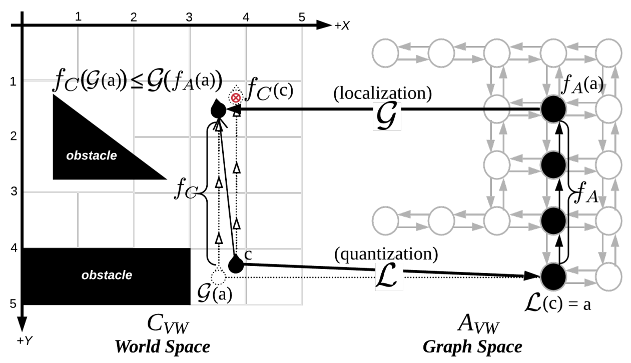

Figure 4: A commonly-used abstraction for pathfinding: a **tile-graph** (Millington 2019). The illustrated abstraction is **valid**, because $\forall a \in A_{\text{VW}}\colon (f_C \circ \mathcal{G})(a) \leq (\mathcal{G} \circ f_A)(a)$.

When pathfinding (pf) in Figure 4's virtual world, the Concrete Virtual World we care about is the World Space. The concrete reasoning to perform is: move from some position $c = (x, y) \in \mathbb{R}^2 = C_{\text{VW}}$ to a target position $c' \in \mathbb{R}^2$ (marked $\otimes$ in Figure 4) while never moving outside the *configuration space* (Lozano-Pérez 1990): agent-reachable areas. I denote the movement to execute as $f_{\text{pf}}(c, c', \mathbb{R}^2)$.

To perform this reasoning in a domain-independent manner, we *must* specify an abstraction scheme, because f_{pf} is non-efficiently computable. There is a vast range of values between any two points in $\mathbb{R}^2$'s configuration space and searching through them is infeasible. The scheme illustrated in Figure 4 is referred to as a **tile-graph** (Millington 2019 p. 239): C_{VW} is effectively partitioned into a grid made up of square tiles of size $s \in \mathbb{R}$. The tiles implicitly define A_{VW} as a directed graph representing Graph Space. Assuming that A_{VW} is represented as a 5×5 matrix $\mathbf{S}$ where cells represent nodes and cardinally-adjacent cells have directed edges between them, we can define the lifting and grounding functions relative to the matrix rows (r) and columns (c) as:

$$\begin{aligned} \mathcal{L}_{\text{tile}}(x, y, s) &= (r, c) & \mathcal{G}_{\text{tile}}(r, c, s) &= (x, y) \quad \text{where,} \\ r &= \lfloor y/s \rfloor & x &= (c \times s) + s/2 \\ c &= \lfloor x/s \rfloor & y &= (r \times s) + s/2 \end{aligned}$$

I adopt A* for qualitative reasoning, "the canonical algorithm for pathfinding in games" (Sturtevant et al. 2019 p.80). It finds a sequence of edges from node $a = (r, c) \in \mathbf{S} = A_{\text{VW}}$ to a target node $a' \in \mathbf{S}$. I denote the path to follow as $f_{\text{A}^*}\big(a, a', \mathbf{S}\big)$. The tile-graph scheme is thus:

$$\Upsilon^{\text{pf}}_{\text{tile}} = \big\langle \mathbb{R}^2, f_{\text{pf}}, \mathbf{S}, f_{A^*}, \mathcal{L}_{\text{tile}}, \mathcal{G}_{\text{tile}} \big\rangle$$

Modeling Tradeoffs via Scheme Properties

An abstraction scheme does not *require* any guarantees between the more-quantitative reasoning we want to perform via f_C and the more-qualitative proxy reasoning that f_A purports to represent. If f_C is not efficiently computable, and we rely on f_A to approximate it, it becomes important to know the **properties** of the scheme that relates them.

Further, abstraction schemes do not prescribe a particular choice of lifting or grounding functions. As defined, they allow designers to compare and contrast *different* abstractions of virtual worlds. The question then becomes: when is a particular abstraction scheme *useful*? This question may guide the choice of *particular* lifting or grounding functions based on the resulting properties of the defined scheme.

Thus, abstraction schemes afford a computationally-precise language to reason about *tradeoffs* for qualitative reasoning: their properties define the space of potential features to balance when considering choosing one scheme over another. I propose that evaluating the utility of an abstraction scheme is tantamount to evaluating the utility of a KR. As first discussed by McCarthy and Hayes (1981), later refined by Wilensky (1984) and Chittaro et al. (1993), a KR has several roles that underlie its utility. These roles have been inconsistently interpreted over time. I operationalize these polysemic roles as properties for abstraction schemes via mathematical formalisms that represent them.

An important drawback of these mathematical definitions is that they are not necessarily directly computable. The equations are presented to precisely capture the intuition of what they represent; in general, they may be computable via an alternative formulation or clever insight to the problem.

Validity Whether f_A is a *valid abstraction* of f_C. A function is valid when defined for every element of its domain. However, a *scheme* is valid if f_A *preserves the monotonicity* (i.e. order) of f_C.

$$\text{valid}(\Upsilon) = \begin{cases} \text{true} & \forall a \in A_{\text{VW}}\colon (f_C \circ \mathcal{G})(a) \leq (\mathcal{G} \circ f_A)(a) \\ \text{false} & \text{otherwise} \end{cases} \tag{1}$$

In essence, validity depends on how the abstract engine is grounded. The tile-graph in Figure 4 is valid: it depicts the intuition that elements in the A_{VW} can either be (a) first grounded and then quantitatively reasoned over (the left-hand side of Equation 1), or (b) first qualitatively reasoned over and then grounded (the right-hand side of Equation 1). This is because method (b) preserves the order of results of method (a). For this scheme, both Equation 1 sides are equal.

Distortion How *close* the grounded result of f_A is to the application of f_C for all potential applications of those functions. For a given scheme Υ: if we let $a \leftarrow \mathcal{L}(c)$, we care to know $\forall c \in C\colon f_C(c) \overset{?}{=} (\mathcal{G} \circ f_A)(a)$.

Distortion is particularly important in proper abstract engines: because $|C_{\text{VW}}| > |A_{\text{VW}}|$, there is *loss of information* when going from C_{VW} to A_{VW}. That loss manifests when grounding information from A_{VW} back to C_{VW}.

Distortion provides an *expected* value for the loss of precision, i.e. the error during abstract reasoning. For $c \in C_{\text{VW}}$, the scheme's error is $\big|f_C(c) - \big(\mathcal{G} \circ (f_A \circ \mathcal{L})\big)(c)\big|$. Whereas error is element-wise, distortion is the expected value of error for all elements. Canonically per rate-distortion theory (Shannon 1959), it is the mean-squared error (MSE):

$$\text{distortion}(\Upsilon) = \frac{1}{\big|C_{\text{VW}}\big|} \sum_{c \in C_{\text{VW}}} \Big(f_C(c) - \big(\mathcal{G} \circ (f_A \circ \mathcal{L})\big)(c)\Big)^2 \tag{2}$$

For pathfinding, this is not directly computable: $|\mathbb{R}^2|$ is the cardinality of the continuum, denoted $\mathfrak{c}$; further, $1/\mathfrak{c} = 0$, rendering Equation 2 useless.

Distortion *is*, however, approximable: because C_{VW} is a virtual world, the range of $\mathbb{R}^2$ is given by the floating-point precision that represents it. Thus, we might estimate the distortion by sampling in the space of floating-point values. We do not compute it here, but note: this approach treats the MSE as one of estimation with $\mathbf{E}$xpected value, such that:

$$\text{distortion}(\Upsilon) = \mathbf{E}\left[\Big(f_C(c) - \big(\mathcal{G} \circ (f_A \circ \mathcal{L})\big)(c)\Big)^2\right]$$

Adequacy The degree to which the abstraction scheme permits expressing the *state* of the C_{VW} that is important for reasoning within the A_{VW}, for all such states one *needs* to consider. For instance, a pathfinding (pf) abstraction scheme Υ_1^{pf} would be less adequate than another such scheme Υ_2^{pf} if there was a position in world space our character could be in that *could not* be mapped onto a node in the graph space via Υ_1^{pf} but *could* be mapped via Υ_2^{pf}. We quantify this using Iverson's bracket notation as:

$$\text{adequacy}(\Upsilon) = \frac{\sum_{c \in C_{\text{VW}}} \big[\mathcal{L}(c) = \varnothing\big]}{\big|C_{\text{VW}}\big|} \tag{3}$$

The scheme in Figure 4 is not fully adequate: there is a portion of the triangle obstacle that occupies part of a tile that is represented in the Graph Space. The rest of the C_{VW} is adequately represented via $\Upsilon_{\text{tile}}^{\text{pf}}$.

Flexibility The degree to which the scheme's A_{VW} supports multiple forms of *meaningful* qualitative reasoning over the C_{VW}. In essence, this affords the possibility of leveraging an abstraction scheme for a large variety of tasks.

Plainly, flexibility is the size of the *function space* over A_{VW} whose results *can* be mapped onto C_{VW}. The function space – i.e. set of all functions – over A_{VW} is denoted as the exponential object $A_{\text{VW}}^{A_{\text{VW}}}$. In set-builder notation:

$$\text{flexibility}(\Upsilon) = \left|\{f \mid f \in A_{\text{VW}}^{A_{\text{VW}}} \wedge \exists a \in A_{\text{VW}} \colon (\mathcal{G} \circ f)(a) \neq \varnothing\}\right| \tag{4}$$

Flexibility of $\Upsilon_{\text{tile}}^{\text{pf}}$ is assessed via constructive proof: it requires *evidencing* every *alternate* qualitative reasoning f_A that has a meaningful interpretation in the C_{VW}. Minimally, $\Upsilon_{\text{tile}}^{\text{pf}}$ is flexible in that it affords reasoning with *other* pathfinding algorithms that operate on the basis of $\mathbf{G}$.

Efficiency The resource cost that underlies the scheme's lifting, qualitative reasoning, and grounding; i.e. the combined algorithmic efficiency of $\mathcal{L}$, f_A, and $\mathcal{G}$. There are many ways to characterize efficiency, and it is important to qualify it with respect to particular methods and dimensions.

Efficiency can be assessed rationally via complexity analysis and empirically via performance benchmarking. It can focus on *time*, *space*, *electrical energy usage*, etc. For example, we might characterize reasoning efficiency with respect to the scheme's Big O time complexity as:

$$\text{efficiency}_{\text{time}}^{\text{complexity}}(\Upsilon) = \mathcal{O}(\mathcal{L}) + \mathcal{O}(f_A) + \mathcal{O}(\mathcal{G}) \tag{5}$$

Applying Equation 5 to $\Upsilon_{\text{tile}}^{\text{pf}}$ means that our tile graph abstraction's efficiency is tantamount to the efficiency of A*; both $\mathcal{L}_{\text{tile}}$ and $\mathcal{G}_{\text{tile}}$ have constant time complexity. The time complexity of A* depends on its heuristic function $h(x)$; assuming that the search proceeds on $\mathbf{G}$ as a tree yields performance on the order of $\mathcal{O}\big(\log h^*(x)\big)$, where $h^*(x)$ is the *optimal* heuristic estimate from node x to the goal.

Cognitive Coupling The degree to which the abstraction scheme's qualitative reasoning corresponds to how it is cognized by relevant stakeholders. While an abstraction scheme is designed for the purpose of carrying out more-quantitative reasoning in a more-qualitative fashion, schemes may have consequences that are displayed in virtual worlds (e.g. Halo: Reach's *Firefight*). Thus, it may be crucial to define the scheme such that it reflects how relevant stakeholders conceptualize what is being represented. This includes *other* designers who will manipulate what is abstracted and players who will be interpreting the resultant abstractions in order to play (Juul 2007). Succinctly, it "requires that our representations are *canonical with respect to understanding*." (Wilensky 1984 p.4, emphasis in original).

Cognitive coupling is difficult to assess because it is a perceptual measure. If we take it to mean *perceptual quality* – the character of the qualitative reasoning as judged relative to someone's conception of the reasoning "in nature" – we face the problem that "distortion and perceptual quality are at odds with each other" (Blau and Michaeli 2018 p.6228). One way to assess coupling based on perception is via the mean opinion score of human subjects in an experiment judgement task: the participants must judge whether some stimuli is "natural" or the output of some algorithm.

In this evaluation, two probability distributions are being compared: (1) $p_{\Upsilon}^{\text{human}}$, the distribution of behaviors deemed natural by people for the phenomenon that is abstracted by Υ and (2) $p_{\Upsilon}^{\text{perceived}}$, the distribution of perceivable behaviors intended to be captured by f_A as grounded and rendered in C_{VW}. The purpose of the experiment is to determine what is the probability of judgement success, which is given by:

$$p_{\text{success}}(\Upsilon) = \frac{1}{2}\delta_{\text{TV}}(p_{\Upsilon}^{\text{human}}, p_{\Upsilon}^{\text{perceived}}) + \frac{1}{2}$$

(Moorthy and Bovik 2011)

where $\delta_{\text{TV}}(\cdot,\cdot)$ is the Total-Variation (distribution) distance. The probability $p_{\text{success}}(\Upsilon)$ decreases as δ_{TV} decreases. In the best case $p_{\Upsilon}^{\text{human}} = p_{\Upsilon}^{\text{perceived}}$, such that $p_{\text{success}} = 1/2$ (no better than a coin flip). Based on this, a parsimonious way to define cognitive coupling is:

$$\text{cog-coupling}(\Upsilon) = \frac{\hat{p}_{\text{success}}(\Upsilon)}{\frac{1}{2}} = 2 \times \hat{p}_{\text{success}}(\Upsilon) \tag{6}$$

where $\hat{p}_{\text{success}}$ is the probability of judgment success, as assessed experimentally. While I have not conducted such an evaluation for $\Upsilon_{\text{tile}}^{\text{pf}}$, I expect it would succeed: A* will result in observably field-expedient behavior that would match a *rational agent's* "natural" movement through space.

Conclusion

I have, for the first time, articulated a human-centered computational science-based model of abstraction: abstraction schemes are a model of the *outcome* of human processes that conceptualize more-quantitative phenomena in a concrete virtual world in terms of a more-qualitative representation in an abstract virtual world. My model is *primarily* theoretical and is the basis for my future work of interest: the simulation of the human *process* of abstraction. I expect that process to be guided by the space of potential tradeoffs to make in converging upon one abstraction scheme over another. To that end, I have also mathematically defined 6 abstraction scheme properties – guided by foundational work in KR within AI – which I posit are key dimensions of that tradeoff space. My intent is to use these analytic metrics in a generative capacity, and future work will discover how.

Abstraction and its formalization are applicable across a wide-array of areas in theoretical and applied computer science, including programming languages, artificial intelligence, and human-computer interaction. Further, the running example illustrates the analytic utility of abstraction schemes and tradeoffs for pathfinding, a core technique within game AI. Pathfinding is foundationally a search problem; that diverse types of computational problems (e.g. decision problems, optimization problems) can be framed as search problems speaks to the potential of abstraction schemes and their properties to be useful for a wide array of approaches to computational game design.

Finally, I revisit an earlier point: abstraction is pervasively implicit within AIIDE-relevant research. Thus, this work can serve as a common framework to support systematicity within our community. Abstraction schemes and properties afford computationally-precise comparison of diverse approaches to qualitative reasoning about a common phenomenon. While we cannot do so here due to space, promising comparisons include: procedural narrative generation via linear logic (Martens 2015) v. planning (Young et al. 2013) and experience management via game-tree search (Robertson and Young 2018) v. Markov decision processes (Thue and Bulitko 2012). In the long term, we hope our formal theory of design abstraction evolves into a formal theory of game design that affords systematically understanding the space of AIIDE-relevant research.

Acknowledgments

I gratefully thank the kind reviewers who provided constructive feedback and productively engaged with me during the rebuttal phase of this paper's peer-review. The paper has been revised in good faith and I hope it reflects the feedback adequately. I also thank Chris Martens and the Laboratory for Quantitative Experience Design – especially Nancy Blackburn, Jennifer Nelson, Vanessa Cox, Dylan Stout, Alexander Bisberg, and Matthew Christensen – for all of the discussions that have greatly informed this work.

References

Blau, Y., and Michaeli, T. 2018. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6228–6237.

Boden, M. A. 2003. *The Creative Mind: Myths and Mechanisms*. Routledge, 2nd edition.

Bungie. 2010. *Halo: Reach*. Microsoft Game Studios.

Cardona-Rivera, R. E., and Young, R. M. 2014. Games as Conversation. In *Proceedings of the 3rd Games and NLP Workshop at the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Cardona-Rivera, R. E.; Zagal, J.; and Debus, M. S. 2020. Narrative goals in games: A novel nexus of story and gameplay. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*.

Chittaro, L.; Guida, G.; Tasso, C.; and Toppano, E. 1993. Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems: A case study in diagnosis. *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics* 23(6):1718–1751.

Conklin, J. 2005. Wicked problems & social complexity. In *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. Wiley.

Cook, M., and Raad, A. 2019. Hyperstate space graphs for automated game analysis. In *Proceedings of the 1st IEEE Conference on Games*.

Cook, M. 2017. A vision for continuous automated game design. In *Proceedings of the Experimental AI in Games Workshop at the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Cousot, P., and Cousot, R. 1992. Abstract interpretation frameworks. *Journal of Logic and Computation* 2(4):511–547.

Cross, N. 2001a. Can a machine design? *Design Issues* 17(4):44–50.

Cross, N. 2001b. Designerly ways of knowing: Design discipline versus design science. *Design Issues* 17(3):49–55.

Damiano, R.; Lombardo, V.; and Pizzo, A. 2019. The ontology of drama. *Applied Ontology* 14(1):79–118.

Davis, R.; Shrobe, H.; and Szolovits, P. 1993. What is a knowledge representation? *AI Magazine* 14(1):17–33.

Davis, J. P.; Steury, K.; and Pagulayan, R. 2005. A survey method for assessing perceptions of a game: The consumer playtest in game design. *Game Studies* 5(1):1–13.

Debus, M. 2019. *Unifying Game Ontology: A Faceted Classification of Game Elements*. Ph.D. Dissertation, ITU-Copenhagen.

Dewey, J. 1896. The reflex arc concept in psychology. *Psychology review* 3(4):357.

Diamanti, M., and Thue, D. 2019. Automatic abstraction and refinement for simulations with adaptive level of detail. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 17–23.

Fernández-Vara, C. 2011. From "open mailbox" to context mechanics: Shifting levels of abstraction in adventure games. In *Proceedings of the 6th International Conference on the Foundations of Digital Games*, 131–138.

Frayling, C. 1994. Research in art and design. *Royal College of Art Research Papers* 1(1):1–5.

Gero, J. S., and Kannengiesser, U. 2004. The situated function–behaviour–structure framework. *Design studies* 25(4):373–391.

Gero, J. S. 1990. Design prototypes: A knowledge representation schema for design. *AI Magazine* 11(4):26–36.

Goodwin, S. D.; Menon, S.; and Price, R. G. 2006. Pathfinding in open terrain. In *Proceedings of International Academic Conference on the Future of Game Design and Technology*, 8.

Gravina, D.; Khalifa, A.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2019. Procedural content generation through quality diversity. In *Proceedings of the 1st IEEE Conference on Games*, 1–8.

Gregory, J. 2017. *Game Engine Architecture*. CRC Press.

Heidegger, M. 1962. *Being and Time*. Harper & Row. Transl. by J. Macquarrie and E. Robinson from *Sein und Zeit* (1927).

Juul, J. 2007. A certain level of abstraction. In *Proceedings of the DiGRA Conference: Situated Play*.

Kan, D. M. 1958. Adjoint functors. *Transactions of the American Mathematical Society* 87(2):294–329.

Kuipers, B. 1994. Reasoning with qualitative models. *Artificial Intelligence* 59:125–132.

Kuittinen, J., and Holopainen, J. 2009. Some notes on the nature of game design. In *Proceedings of the DiGRA Conference: Breaking New Ground: Innovation in Games, Play, Practice and Theory*.

Kultima, A., and Alha, K. 2010. "Hopefully Everything I'm Doing has to do with Innovation": Games Industry Professionals on Innovation in 2009. In *Proceedings of the 2nd IEEE Consumer Electronic Society's Games Innovation Conference*, 1–8.

Laird, J. E.; Lebiere, C.; and Rosenbloom, P. S. 2017. A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *AI Magazine* 38(4):13–26.

Lakoff, G., and Johnson, M. 1980. *Metaphors We Live By*. University of Chicago Press.

Lee, J. S.; Pries-Heje, J.; and Baskerville, R. 2011. Theorizing in design science research. In *Proceedings of the 6th International Conference on Service-Oriented Perspectives in Design Science Research*, 1–16.

Liapis, A.; Yannakakis, G. N.; Nelson, M. J.; Preuss, M.; and Bidarra, R. 2018. Orchestrating Game Generation. *IEEE Transactions on Games* 11(1):48–68.

Liapis, A.; Yannakakis, G. N.; Cook, M.; and Colton, S. 2019. Guest Editorial Special Issue on AI-Based and AI-Assisted Game Design. *IEEE Transactions on Games* 11(1):1–4.

Lozano-Pérez, T. 1990. Spatial planning: A configuration space approach. In *Autonomous Robot Vehicles*. Springer. 259–271.

Martens, C., and Hammer, M. 2017. Languages of play: Towards semantic foundations for game interfaces. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 1–10.

Martens, C. 2015. Ceptre: A language for modeling generative interactive systems. In *Proceedings of the 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

McCarthy, J., and Hayes, P. J. 1981. Some philosophical problems from the standpoint of artificial intelligence. In Webber, B. L., and Nilsson, N. J., eds., *Readings in AI*. Elsevier. 431–450.

Millington, I. 2019. *AI for Games*. Taylor & Francis.

Moorthy, A. K., and Bovik, A. C. 2011. Blind image quality assessment: From natural scene statistics to perceptual quality. *IEEE Transactions on Image Processing* 20(12):3350–3364.

Murphy-Hill, E. R.; Zimmermann, T.; and Nagappan, N. 2014. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *Proceedings of the 36th International Conference on Software Engineering*.

Newell, A.; Shaw, J. C.; and Simon. 1960. Report on a general problem-solving program for a computer. In *Proceedings of the International Conference on Information Processing*, 256–264.

O'Donnell, C. 2014. *Developer's Dilemma: The Secret World of Videogame Creators*. MIT press.

Osborn, J. C.; Lambrigger, B.; and Mateas, M. 2017. Hyped: Modeling and analyzing action games as hybrid systems. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Roberts, D. L. 2011. Seven design challenges for fully-realized experience management. In *Proceedings of the 4th Intelligent Narrative Technologies Workshop at the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Robertson, J., and Young, R. M. 2018. Perceptual experience management. *IEEE Transactions on Games* 11(1):15–24.

Robertson, J.; Jhala, A.; and Young, R. M. 2019. Efficient choice enumeration for narrative world design. In *Proceeding of the 14th International Conference on the Foundations of Digital Games*.

Rüde, U.; Willcox, K.; McInnes, L. C.; and Sterck, H. D. 2018. Research and education in computational science and engineering. *SIAM Review* 60(3):707–754.

Schon, D. A. 1984. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books.

Searle, J. R. 1980. Minds, brains, and programs. *Behavioral and brain sciences* 3(3):417–424.

Shannon, C. E. 1959. Coding theorems for a discrete source with a fidelity criterion. *Institute of Radio Engineers, International Convention Record* 7(325-350).

Sturtevant, N. R.; Sigurdson, D.; Taylor, B.; and Gibson, T. 2019. Pathfinding and abstraction with dynamic terrain costs. In *Proceedings of the 15th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 80–86.

Summerville, A.; Martens, C.; Harmon, S.; Mateas, M.; Osborn, J.; Wardrip-Fruin, N.; and Jhala, A. 2017. From Mechanics to Meaning. *IEEE Transaction on Games* 11(1):69–78.

Thue, D., and Bulitko, V. 2012. Procedural Game Adaptation: Framing Experience Management as Changing an MDP. In *Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Thue, D., and Bulitko, V. 2018. Toward a unified understanding of experience management. In *Proceedings of the 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Washburn Jr, M.; Sathiyanarayanan, P.; Nagappan, M.; Zimmermann, T.; and Bird, C. 2016. "What Went Right and What Went Wrong": An Analysis of 155 Postmortems from Game Development. In *Proceedings of 38th International Conference on Software Engineering*, 280–289.

Wilensky, R. 1984. Some problems and proposals for knowledge representation. Technical Report UCB/CSF 87/351, Dept. of EECS, UC Berkeley.

Young, R. M.; Ware, S. G.; Cassell, B. A.; and Robertson, J. 2013. Plans and planning in narrative generation: A review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative* 37(1-2):41–64.

Zook, A., and Riedl, M. O. 2014. Automatic game design via mechanic generation. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*.