From Mechanics to Meaning

Adam Summerville, Chris Martens, Sarah Harmon, Michael Mateas, Joseph Osborn, Noah Wardrip-Fruin, and Arnav Jhala

***Abstract*—While generative approaches to game design offer great promise, systems can only reliably generate what they can "understand," which is often represented in a limited, implicit form in hand-crafted evaluation functions or constructive rules. Proceduralist readings, a semiformal approach for interpreting the meaning of a game based on its underlying processes and interactions in conjunction with aesthetic and cultural cues, offer a novel, systematic approach to game understanding. We formalize proceduralist argumentation as a logic program that performs static reasoning over game specifications to derive higher level meanings, as part of *Gemini,* a bidirectional game analysis and generation system.**

***Index Terms*—Answer set programming (ASP), automated game analysis, proceduralist readings.**

I. Introduction

PROCEDURALLY generating components of games has the ability to create rich play environments that adapt in real time to a player's choices and behavior, paving the way for more personalized, and thus more impactful, conveyance of story and theme. However, focusing solely on visual stimuli and surface characteristics of games as the target for generativity has limitations. Perhaps the most potent messages in games are communicated through mechanics, i.e., the rules and control mechanisms that mediate interaction between the player's actions and the game state.

Consider as an example Lucas Pope's critically acclaimed game *Papers, Please* [1] in which the player serves as an immigration officer who must decide who can cross the border to a fictional country. The player must balance success at the job with moral questions around civil disobedience toward an increasingly menacing government. The game communicates deep political themes by asking the player to perform the mechanics of border control within a context of reward and punishment reflecting those themes.

Similarly, Paolo Pedercini's *Unmanned* [2] portrays a sequence of short playable vignettes in the life of a soldier engaged in drone warfare wrestling with his masculinity. Each vignette contains some textual narrative delivery paired with a mouse-controlled mini-game. For example, one scene (see Fig. 1) contains a branching text-based monologue of the player character's thoughts while shaving, together with a mini-game in which the player's control of a razor determines how many cuts he gets on his face (which persist throughout the game). In this case, the mechanics of the mini-game serve as an enactment of the character's grooming routine and support formation of an emotional connection (whether frustrated or reflective) between player and fictional character.

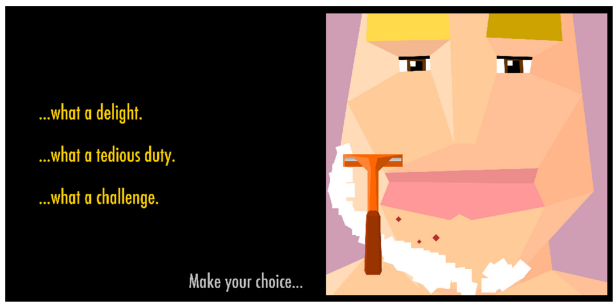

Fig. 1. Screenshot from *Unmanned*.

Games studies scholars have examined these and other games, arguing for the power of mechanics to create meaning: Bogost [3] demonstrates how persuasion may be realized through mechanics (through so-called procedural rhetoric), Delwiche [4] argues that mechanics can shape player attitude and behavior, and Zagal [5] illustrates how they can encourage players to reflect on ethical issues. This power motivates us to investigate, from a computational perspective, how mechanics operate to communicate meaning. An answer to this question would be a step along the path of generating meaningful games, which has the potential to shape an interactive experience in unprecedented ways.

The problem of full video game generation has attracted wide interest in the last decade [6]–[8] for a variety of reasons, one being simply the challenge of autonomously designing in a creative space. However, due to the infancy and difficulty of the game generation problem, most of these have operated on a limited subset of the larger problem, e.g., 2-D movement-based games with a finite set of possible behaviors upon collision or user input. Most of these approaches (save *Game-o-Matic*) have ignored the meaning of the games, as might be inferred by a player or game critic.

To be able to generate games with this kind of meaning, we need a way of establishing the relationship between

Manuscript received March 20, 2017; revised September 10, 2017; accepted September 30, 2017. Date of publication October 23, 2017; date of current version March 15, 2019. This work was supported by the National Science Foundation under Grant IIS-1409992. *(Corresponding author: Adam Summerville.)*

A. Summerville, S. Harmon, M. Mateas, J. Osborn, and N. Wardrip-Fruin are with the Department of Computational Media, University of California, Santa Cruz, Santa Cruz, CA 95064 USA (e-mail: asummerv@ucsc.edu; smharmon@ucsc.edu; michaelm@soe.ucsc.edu; jcosborn@ucsc.edu; nwf@soe.ucsc.edu).

C. Martens and A. Jhala are with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (e-mail: martens@csc.ncsu.edu; ahjhala@ncsu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2017.2765599

meaning and mechanics. The mechanics, dynamics, and aesthetics (MDA) framework of Hunicke *et al.* [9] provides a starting point, defining mechanics as "the various actions, behaviors and control mechanisms afforded to the player within a game context," dynamics as "the run-time behavior of the mechanics acting on player inputs and each others' outputs over time," and aesthetics as "the desirable emotional responses evoked in the player, when she interacts with the game system." They posit that these components of games form a hierarchy, with mechanics serving as the "axioms" at the bottom specifying the authored interaction rules of a game, dynamics emerging from mechanics through interaction with the player, and aesthetics emerging from dynamics as well as meaning inferred by the player on the basis of audio–visual stimuli.

To understand the meaning of a game requires accounting for each of these mechanisms and where they interact. Treanor *et al.* proposed proceduralist readings, a process of deduction that allows one to discuss dynamics, aesthetics, and higher level meanings on the basis of a game's mechanics and cultural knowledge about the significance of sensory cues, such as colors and icons [10]. They refer to the result of this process as a meaning derivation, a logical argument for any given interpretation of what a game communicates.

In this paper, we present the analysis half of *Gemini*, a bidirectional game analysis/generation system. The analysis half is an operationalization of proceduralist readings that enables a computational understanding of the relationship between mechanics and meaning. We have encoded the style of reasoning introduced by proceduralist readings as a logic program, which can autonomously derive higher level knowledge from lower level knowledge. As a starting point, we investigate deriving dynamics as emergent consequences of mechanics; then, by incorporating cultural and phenomenological knowledge, we derive aesthetics from the inferred dynamics. This information is computed using answer set programming (ASP) to statically derive all of these properties without need for simulation. In this paper, we do not demonstrate the generative portion of *Gemini* as the analysis portion requires significant detail to discuss sufficiently.

The result of this paper is a framework for deriving game meaning from mechanics and information about audiovisual representation, so as to support the generative capabilities of *Gemini*. The resulting analyses demonstrate a level of knowledge and reasoning beyond systems with prescribed meaning [11].

A. Related Work

The most closely related work is *Game-o-Matic* [12], which similarly combines cultural knowledge with mechanics to generate playable games from meaning descriptions. It makes use of "microrhetorics" to represent primitive mechanics paired with player interpretations of game entity verbs (e.g., support, eat, destroy, etc.). This paper generalizes from *Game-o-Matic* in a number of ways. First, *Game-o-Matic* only represents and reasons about game rules related to movement, collision detection, and a minimal resource representation (objects can grow and shrink), while this paper captures a broader range of potential mechanics (notably, we develop a theory of resource management mechanics much more deeply). Second, *Game-o-Matic*'s notion of game meaning is restricted primarily to single-layered mappings between mechanics and represented game verbs, with no explicit reasoning related to game dynamics. In contrast, this paper operationalizes meaning derivations to capture both broader and more abstract notions of dynamics and game meaning. Finally, *Game-o-Matic* is a pipelined architecture that can move from a game specification in the form of game entities with desired verb relationships to a game, while this paper uses a constraint satisfaction framework to be able to move from desired dynamics and meanings to games, or games to inferred dynamics and meanings, in a single framework. This paper focuses on the latter, which can be considered a generalized form of "reverse *Game-o-Matic*."

There is other work in game generation which also deals with inferring relationships between formalized games and meanings or constraints (the inputs to the generator). Nelson and Mateas [7] incorporate meaning through WordNet and ConceptNet and characterizes games in terms of patterns found in their generation target, *WarioWare* games, such as avoid and acquire. In contrast, our system can express a broader, finer grained space of mechanics, and express meaning in more systematic terms. The Variations Forever [13] system similarly uses ASP to search a fine-grained space of possible mechanics but does not make any attempt to relate generation to high-level conceptual meaning, nor is the generative space broad enough to express resource management or other mechanics not based on movement and collision of entities. Finally, the Angelina system has generated games based on theme in the context of the Ludum Dare game jam [14]. Angelina uses a data-driven approach based on word association databases and does not systematically relate mechanics to meaning.

Zook and Riedl used ASP to generate mechanics in a form similar to the approach proposed in this paper [15]. That work defined a mechanic as a set of preconditions and effects, but was limited to "avatar-centric" mechanics, and no effort was chosen to find the underlying meaning/semantics of the mechanics, simply playability constraints. Cook and Colton [16] used a process known as Google milking to find semantically meaningful pairings to map onto existing mechanical constructs (e.g., to find an item to heal, they searched for "Why do kids eat" and took the top result "boogers" to find an item to heal the player "kid" character), but this work focused on mapping semantics to preexisting mechanics. Cook and Colton also presented work on possible ways of linking code and meaning [17], but this was never extended to a working system. A distinction between our approach and their proposed one is that ours does not rely on semantically meaningful variable names while theirs does (e.g., for them a boolean variable named "alive" is important, while for us it is the act of entity deletion that is important no matter the names of the variables involved).

Another topic of research for general game playing (GGP) systems is reasoning over the structures of a game. The earliest work [18] examines the syntactic structures of the first-order logic program that defines a GGP game to learn underlying

game structures. This system could answer very basic questions, e.g., "Is this a successor function?" or "Is there a board?" Later Schiffel and Thielscher [19] extended this work to be able to determine if something represents a quantity (such as number of points). Due to the fact that the players are given no type information and very limited semantic information for the logic programs that make up a GGP game, even these basic tasks are arduous—determining if there are numbers or a board has to be teased out from the syntax. Our formalism lends itself to discovering higher level structures due to the fully specified semantics.

Game researchers are also interested in developing models to describe how players learn. One such model is a skill chain: A directed graph that identifies initial and gained skills over the course of a game, and the paths that lead players to develop (or reject) these skills [20]. Through the construction of a skill chain diagram, designers can better understand how players navigate a game experience based on what they have learned at each step in the path. Furthermore, they can identify potential mechanisms for player affectual impacts, such as pleasure and burnout. The pursuit of a formal procedure for game meaning derivation may lead to the automation of these types of analyses.

II. Approach

Treanor *et al.*, in introducing "proceduralist readings," demonstrate through example the process of constructing meaning derivations, or bodies of knowledge about a game that are built up in a proof-tree-like structure from axiomatic facts, such as mechanics and thematic elements [10]. Proceduralist readings provide a solid foundation for the human activity of game understanding, in which a reader thinks critically about which pieces of the game offer which interpretive affordances. (See Fig. 2 for a depiction of how meaning is built.) We build on this work by formalizing this reasoning as a computer program such that we can autonomously derive high-level knowledge, including culturally informed critical readings, from an input game description.

To do this, we define a formal language for describing mechanics (e.g., "the cursor exerts a force on particles"), atomic pieces of cultural knowledge (e.g., "the color green represents life"), and communication strategies (e.g., "a meter display communicates the state of a resource"). From these specified knowledge components, the task of game understanding is to derive higher order knowledge like "the ball moves perpetually back and forth unless the player intervenes," an example of deriving a dynamic from mechanics in the language of the MDA framework.

We note that the cultural knowledge assumes a specific cultural viewpoint (e.g., green may represent life and nature in the U.S., independence in Mexico, and infidelity in China) and may vary between players (e.g., some United States players may more strongly associate green with envy). Due to the modular nature of our approach, one can specify many different cultural contexts and swap them to see how different cultures and players might perceive the same game. For the purposes of this paper, we codified a U.S., westernized cultural viewpoint representing the assumptions we perceived in our chosen examples.

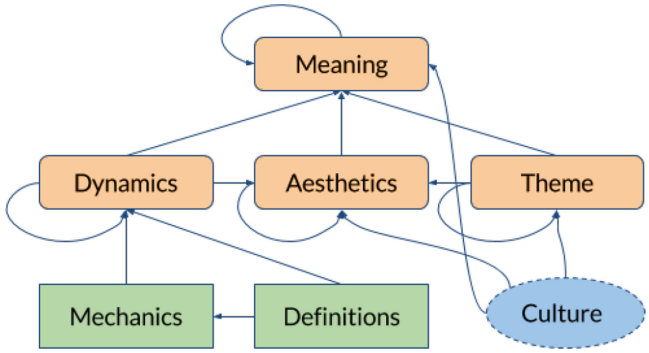

Fig. 2. Green nodes represent definitions (entities, resources, and other variables) and mechanics in the game. Blue nodes are the cultural context. Orange nodes are the derived dynamics, aesthetics, and meaning. Arrows represent the directions information can flow in the process of a proceduralist reading (e.g., Mechanics and Definitions inform Dynamics which can then inform the Aesthetics or Meaning of a game).

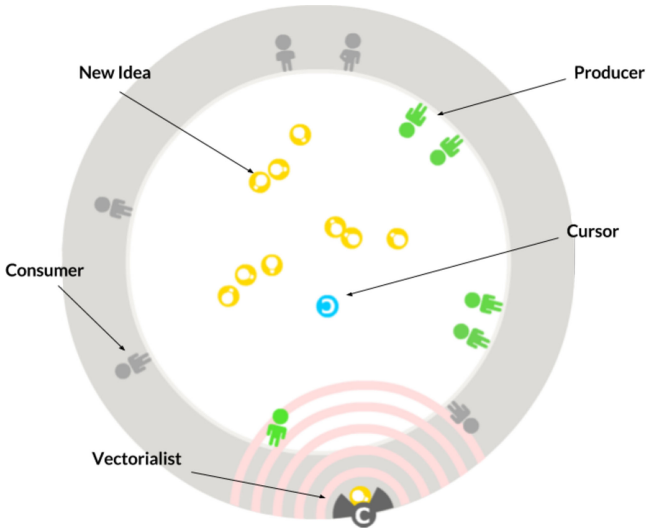

Fig. 3. Screenshot of the *Free Culture Game*.

A. Explanatory Example

One example carefully studied by Treanor *et al.* is *The Free Culture Game*, in which "new ideas" are represented as floating particles that must be herded toward producers in the creative commons to keep them inspired (creating new ideas) and away from the vectorialist, who takes ideas out of the creative commons to commodify them for consumers. The player exerts an indirect force via the mouse cursor on new ideas. Several proceduralist readings are extrapolated from these mechanics together with the game's interpretive affordances, such as the colors selected for producers versus consumers (green versus grey) and the robotic and malicious audio–visual character of the vectorialist. An annotated screenshot can be seen in Fig. 3.

For example, they read the following meanings from the game.

1) The player must navigate the cursor between the vectorialist and new ideas to prevent commodification.
2) The vectorialist is an evil adversary who does not care about the happiness of people.

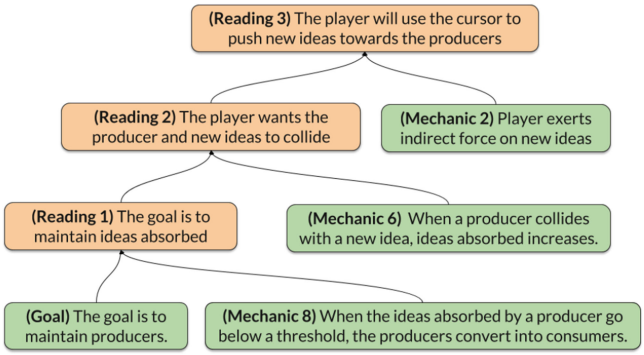

Fig. 4. Color-coded example meaning derivation of an aspect of the aesthetics of the *Free Culture Game*. Green nodes represent concrete pieces of game **M**echanics and stated **G**oals that are used as the building blocks for inference chains. Orange nodes represent **R**eadings, i.e., dynamics such as **(R1)** are derived from these mechanics and in turn used to derive (also orange) aesthetics, such as **(R2)** and **(R3)**.

They derive these meanings from a number of implicit rules, which they call dynamics. We read these as equivalent to the constitutive mechanics of [21], though we will use the term dynamics here to avoid confusion. For example, the first inference shown on the path to deriving the first reading is the following.

Because producers need new ideas to collide with them in order not to turn into consumers, the player's goal is to maintain as many producers as possible, and the player can exert a force on the new ideas, the player will push new ideas towards producers. (**R**eading 1).

This derivation can be made more explicit by identifying the base assumptions that can be directly observed about the game (e.g., **G**oals and **M**echanics), then building the argument in a tree structure:

Base assumptions:

(G) The goal is to maintain producers.
(M8) When the ideas absorbed by a producer go below a threshold, the producers convert into consumers.
(M6) When a producer collides with a new idea, the ideas absorbed increases.
(M2) The cursor pushes new ideas.

Meaning derivation:

By	
(G), (M8)	**(R1)**. The goal is to maintain ideas absorbed.
(M6), **(R1)**	**(R2)**. The player wants the producer and new ideas to collide.
(M2), **(R2)**	**(R3)**. The player will use the cursor to push new ideas toward the producers.

Characterizing game analysis as a process of constructing these reasoning structures constitutes an initial step toward a computable formalism. To operationalize this process, we need to understand the implicit inference rules that govern *why* each proof step is valid. For example, to state that **(R1)** follows from **(G)** and **(M8)** makes use of the reasoning that we do as humans about the relationship between goals and circumstances that defeat those goals (see Fig. 4).

Some other game reasoning principles that we must codify to fully formalize the *Free Culture Game* meaning derivation include the following.

1) If the player's goal is G and action A accomplishes G, the player will do action A.
2) Pushing an entity E with an entity P causes E to move away from P.
3) An entity moving away from something might move toward another entity.
4) When E moves toward X, E and X might collide.

This kind of reasoning is powerfully general in that we can expect to apply it to many games without having to encode game-specific interpretation knowledge. It includes knowledge about player goals and control, causal relationships between game events, and knowledge about state change phenomena, such as spatial and physical relationships, increase and decrease of resources, the passage of time, and win and lose conditions. In other words, this kind of game literacy must be made explicit in order to perform proceduralist readings.

Our proposed system for mechanizing proceduralist arguments thus consists of two pieces: 1) for each individual game, a specification of the game's mechanics and communicative affordances (hereafter "the specification"), and 2) a collection of literacy rules like the above for reasoning over specifications (hereafter "the reasoning principles"). We encode both of these knowledge sources as an ASP system, i.e., as a logic program interpretable by an answer set solver.

This paper describes the reasoning principles, a body of rules that enable deriving higher level meanings from game specifications, together with several example game specifications written in Cygnus, our domain-specific language for game specifications. We describe these efforts by example in Sections III and IV.

III. Game Specification

We describe game mechanics as a collection of named outcomes that have preconditions and results, similar to linear logic based game specification in Ceptre [22] or the sensors and actors of Kodu [23]. To standardize our game descriptions in such a way as to describe a broad range of genres, but apply the same reasoning principles to all specifications, we developed the Cygnus game specification language, which includes the aforementioned outcomes as well as notions of entity, resource, timer, and player controls. Using this formalism, we can describe the mechanics of *The Free Culture Game* as well as other classic arcade-style games.

A nonexhaustive list of possible preconditions includes the following:

1) comparisons of resources: e.g., $R_1 >= 0$;
2) Collision detection: $overlaps(E_1, E_2)$;
3) geometric proximity: $near(E_1, E_2)$;
4) timers elapsing: $timer\_elapsed(T_1)$;
5) player input: $button\_press(mouse, held)$.

A nonexhaustive list of possible results includes the following:

1) resource modification: $R_1 += 2$;

2) entity movement: e.g., $move(E_1, north)$; $move\_toward(E_1, E_2)$;
3) entity creation/deletion: e.g., $delete(E_1)$;
4) game mode changes: $mode\_change(game\_loss)$.

To carry out automated inquiry about a game, the author creates a specification of the game's mechanics in Cygnus, then runs the answer set solver on it in conjunction with the reasoning principles, resulting in a stable model (collection of logical facts) representing derived knowledge about the game. Here, we give one example in depth to illustrate the approach, and we later summarize further efforts for breadth.[1]

A. Free Culture Game

First, we describe our encoding of the *Free Culture Game*'s mechanics in Cygnus (which we directly embed as ASP predicates). One such mechanic is *The vectorialist pulls in new ideas*, which we describe as an outcome with a single precondition, *the vectorialist is near a new idea*, and a single effect, *the new idea moves toward the vectorialist*. In ASP predicate notation, we assign this outcome the name `pull_idea` with the following syntax:

```
precondition(
      near(vectorialist, new_idea),
      pull_idea).
result(pull_idea,
      move_toward(new_idea, vectorialist)).
```

The `pull_idea` token is simply an identifier to connect the precondition to the result, while the `near(-)` and `move_toward(-,-)` predicate designate specific meanings as preconditions and results in the Cygnus language.

We encode the rest of the game in a similar fashion; we will describe this in line with informal descriptions of preconditions and results. The interested reader may refer to Fig. 5 for the encoding of these rules as ASP predicates. For example, the following three mechanics establish the relationships between producers, new ideas, and the player.

1) Mechanic 1: "Producers make new ideas." Precondition: a slow repeating timer goes off. Result: a new idea is added to the game in a random location.
2) Mechanic 2: "The cursor exerts force on ideas." Precondition: the cursor is near a new idea. Result: the new idea moves away from the cursor.
3) Mechanic 6: "Collision between a new idea and a producer increases ideas absorbed for that producer." Precondition: a new idea and a producer overlap. Result: the producer's "ideas absorbed" resource increases, which is reflected by the producer shifting in color from grey to green.

In addition to specifying mechanics, we may also supply facts about the initial state of the game and the game's goal, such as *The player's goal is to prevent producers from converting into consumers.* See Fig. 6 for these auxiliary clauses.

[1]All code is available at: https://github.com/LudoNarrative/ClimateChange/tree/master/GameGenerator/Justifications

IV. Reasoning Principles

Next, we need to introduce rules for deriving consequences of game specifications, i.e., ways of deriving dynamics, aesthetics, and meanings from mechanics and cultural assumptions. We refer to this body of knowledge as the reasoning principles: Generic knowledge about games that can be composed through logical inference. While the principles of first-order logic allow us to instantiate generalizations with specifics and apply implications to known premises, exactly the content of those generalizations and implications remains a contingent body of knowledge to be authored. It requires logical modeling of game phenomena and their relationship in terms of the causal phenomena we wish to model.

For example, Reading 3—"The player will use the cursor to push new ideas toward the producers"—will need to be derivable as a formal logical statement of the form "the player will create condition C to cause some outcome O to occur." To derive this reading, we informally reasoned that it was because "the player wants the producer and new ideas to collide" and "the player exerts force on new ideas." More generically, these facts can be considered instances of "the player wants outcome O to occur" and "the player has control over condition C, which causes O." By generalizing the inference rule in this way, we arrive at our first reasoning principle: If the player can create condition C, condition C enables outcome O, and the outcome O is favorable, then the player will create condition C. In logic program notation, we write this rule as follows.

```
playerWillDo(Cond, Outcome)
      :- playerCreatesCondition(Cond),
       conditionEnables(Cond, Outcome),
       outcomeFavorable(Outcome).
```

The first line is the head of the rule, or the conclusion of the implication, where the following comma-separated lines are the conjoined premises. Each token beginning with a capital letter is a logic variable, implicitly universally quantified on the outside of the rule. In the rest of this section, we will explain the rules in prose, but each rule has a corresponding formal encoding in the logic program.

This rule is an example of one kind of game reasoning, namely player modeling—it is a prediction about the player's actions on the basis of the game's goals and affordances provided to the player. We identify player modeling as one of several kinds of reasoning principles employed by human experts, such as Treanor *et al.*'s Free Culture Game analysis [10]; we consider player modeling part of a more general category of agency ascription. In addition to agency ascription, we also group rules under the categories of operational semantics—the emergent causal behavior of mechanics—and meaning ascription—the assignment of properties like positive and negative valence to game objects.

```
%% (Mechanic 1): Producers make new ideas
precondition(slow_timeout, gen_idea).
result(gen_idea, add(new_idea)).

%% (Mechanic 2): The cursor exerts force on ideas.
precondition(near(cursor, new_idea), push_idea(cursor)).
result(push_idea(Entity), move_away(new_idea, Entity))
  :- physicsLogic(Entity, pushing).

%% (Mechanic 3): The vectorialist moves toward groups of new ideas.
precondition(far(vectorialist, new_idea), scan).
result(scan, move_toward(vectorialist, new_idea)).

%% (Mechanic 4): The vectorialist pulls in new ideas.
precondition(near(vectorialist, new_idea), pull_idea(vectorialist)).
result(pull_idea(Entity), move_toward(new_idea, Entity))
  :- physicsLogic(Entity, pulling).

%% (Mechanic 5): Collision between new ideas and vectorialist
%%                kills new idea & increases old ideas.
precondition(collide(new_idea, vectorialist), commodify).
result(commodify, delete(new_idea)).
result(commodify, increase(old_ideas, med)).

%% (Mechanic 6): Collision between new idea & producer increases
%%               ideasAbsorbed.
precondition(collide(new_idea, producer), learn).
result(learn, increase(ideasAbsorbed, mid)).
result(learn, set_color(producer,green)).

%% (Mechanic 7): ideasAbsorbed decreases with time.
precondition(tick, forget).
result(tick, decrease(ideasAbsorbed, low)).
result(tick, set_color(producer,gray)).

%% (Mechanic 8): If ideasAbsorbed goes to 0, producer turns into consumer.
precondition(le(ideasAbsorbed, 0), convert_producer).
result(convert_producer, delete(producer)).
result(convert_producer, add(consumer)).

%% (Mechanic 9) (only referred to later than 1st example):
%%  If ideasConsumed goes to 0, consumer turns into producer.
precondition(le(ideasConsumed, 0), convert_consumer).
result(convert_consumer, delete(consumer)).
result(convert_consumer, add(producer)).

%% (Mechanic 10) (only referred to later than 1st example):
%% Vectorialist provides consumers with old_ideas
precondition(gt(old_ideas, 0), feed_consumer).
precondition(near(vectorialist, consumer), feed_consumer).
result(feed_consumer, decrease(old_ideas, low)).
result(feed_consumer, increase(ideasConsumed, low)).
precondition(tick, forget_old).
result(forget_old, decrease(ideasConsumed, low)).
```

Fig. 5. Formal specification of the mechanics of the *Free Culture Game* in Cygnus.

A. Operational Semantics

We establish the semantics of our game representation language by relating effects to how they may influence conditions. For example, two entities moving toward each other may cause the "collide" condition between them, and increasing a resource may eventually cause it to satisfy the condition of being over a certain threshold. We formalize this relationship between results enabling conditions as a series of rules, including the following.

1) Entities moving toward each other enables their collision.
2) Entities moving away from one another enables their collision with other entities.
3) Increasing (decreasing) a resource enables a high (low) threshold to be met.
4) If an outcome O adds an entity E and outcome O' requires E, then O enables O'.
5) If an outcome O removes an entity E needed by O', then O hinders O'.

In this regard, the game's specific rules and the general reasoning principles form two halves of a game's meaning: the game specification connects conditions to results (through explicit mechanics), and the reasoning principles connect results to conditions (through a semantic, approximated interpretation of each result).

```
%% Initializations
initialize(set_to(ideasAbsorbed,10)).
initialize(set_sprite(producer, person)).
initialize(set_color(producer,green)).
initialize(set_sprite(consumer, person)).
initialize(set_color(consumer,gray)).
initialize(set_sprite(vectorialist,
  evil_robot)).
initialize(set_color(vectorialist,black)).

%% (Goal) The goal is to prevent producers
%% turning into consumers.
goal(prevent(convert_producer)).
```

Fig. 6. Auxiliary clauses for the *Free Culture Game*.

We also create a transitive closure of outcome enablement by including the following recursive rule.

1) If O_1 enables O_2 enables O_3, then O_1 enables O_3.

This rule allows us to reason about not just immediate cause-and-effect between mechanics but also indirect causality.

B. Meaning Ascription

To derive high-level notions like *an outcome is favorable* from low-level models of game world domain knowledge, we introduce a number of rules for meaning ascription. For example, we assign positive and negative valence (i.e., "good" or "bad") to things like resources, results, and outcomes on the basis of player goals and operational inference about how various mechanics, resources, and entities affect (enable or hinder) those goals.

As an example, the meaning derivation in Section II found a derived goal, maintaining "ideas absorbed," from the stated goal (prevent producers from converting) and one of the mechanics. We use a reasoning principle that states; if the goal is to prevent an outcome O that has a a low threshold on resource R as a precondition, then maintaining R is a goal. With O instantiated as the rule for producers converting, R maps to the "ideas absorbed" resource and we deduce that maintaining that resource is a goal.

Some instances are assumed to be axiomatic (ending the game in loss is axiomatically bad), while other valences are derived from this static analysis information. These rules include things like the following.

1) Winning (losing) is a good (bad) result.
2) If a resource R is good (bad), increasing R is a good (bad) result.

We also include rules that define certain phenomena in terms of emergent game dynamics, such as notions of positive feedback loops, bootstrapping, and investment as properties of resource management games. These rules are used to reason about games, such as *Cookie Clicker*, discussed in the next section.

C. Agency Ascription

To produce readings, such as *the vectorialist is an antagonist* for the *Free Culture Game*, we need to codify the kind of anthropomorphization commonly understood to be carried out by humans observing autonomous behavior. Thus, we introduce a notion of influence over an outcome by human players as well as autonomous agent (formally, entities). For instance:

1) if an entity E being near (or overlapping or colliding with) another entity E' is a precondition for outcome O, O is not controlled by the player, and some outcome not controlled by the player results in E's movement, then we say that E influences O;
2) if a timer's elapse associated with E is a precondition for O, then we also say that E influences O;
3) if a control event is a precondition for O, then the player influences O. If there are no other preconditions, then the player controls O;
4) if an entity influences an outcome O with a favorable (unfavorable) result, then the entity is an ally (antagonist).

V. Results

We pass our reasoning principles and game specification to an answer set solver, specifically Clingo [24]. The solver determines a set of facts (an answer set) that is consistent and complete with respect to all axioms and rules provided. This will include all facts derivable from the reasoning principles about the specified game mechanics. We note that, as in the original work by Treanor *et al.*, *Gemini* supports multiple readings for a single game.

The answer set generated when given the game specification for the Free Culture Game includes the following facts.

The goal is to maintain `ideasAbsorbed`. The outcome `forget` affects `ideasAbsorbed` negatively, and the outcome `learn` affects it positively, meaning that `learn` is a favorable outcome. The player has agency over the cursor and uses the cursor to influence the outcome `push_idea(cursor)`, which in turn enables the outcome `learn` by pushing ideas toward the producers. As such, `learn` is a favorable outcome that the player has indirect control over via pushing ideas with the cursor, so the player will place the cursor near `new_ideas` to try to enact the `learn` outcome. A graphical representation of these inferred properties along with the rules that were required to derive these can be seen in Fig. 7. Furthermore, collision with the `vectorialist` deletes `new_ideas` (which we label `destroys`) that are required for a player to influence the outcome `push_idea(cursor)`. An entity not under the control of the player that destroys an entity required for the player's progress is an `antagonist` (although not the only possible form of antagonism representable), thus the `vectorialist` is an antagonist. Cultural assumptions about colors (that green represents life and grey represents a lack of life) lead to the interpretation that the outcome labeled as `forget` leads to a worse outcome for the producer than either its initial state or its state as a result of the outcome `learn`.[2]

[2]Of course we do not stipulate that there is only one body of objectively correct cultural knowledge—we could just as easily represent other cultural assumptions and observe differences in interpretation, just as two human critics with different cultural backgrounds may perceive a game's meaning differently.

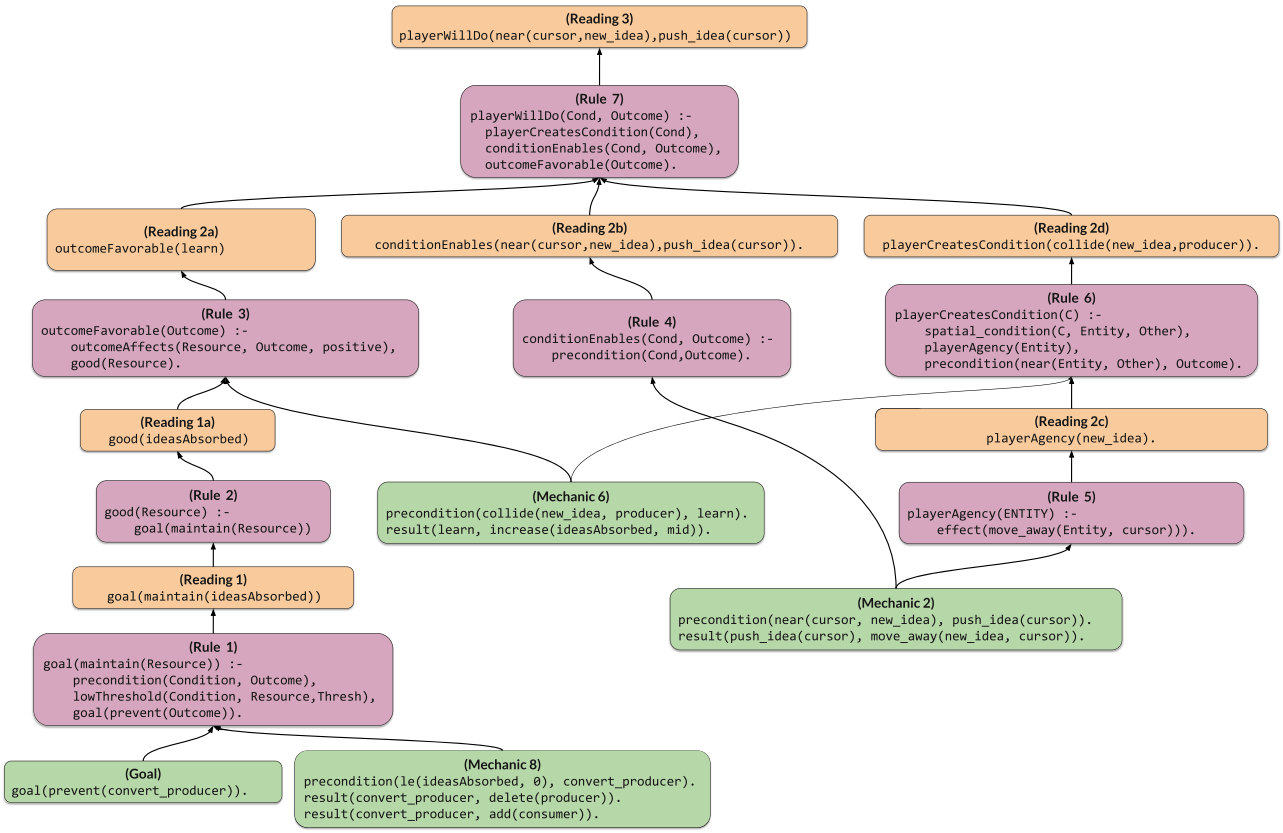

Fig. 7. Full reasoning chain, analogous to the prose interpretation found in Fig. 4. The rules invoked in the reasoning are chain are found in purple. As can be seen, numerous rules and subfacts might need to be invoked to build up to larger pieces of knowledge, such as the chain building up to **Reading 2a** "The outcome labeled 'learn' is favorable" which builds off of three separate game definitions and invokes **Rules 1**, **2**, and **3**.

A. *Other Formalizations*

In addition to the example we have described in detail, we have also formalized and derived reasoning about a handful of other games, including Julien Thiennot's *Cookie Clicker*,[3] the classic arcade games *Pong* and *Kaboom!*, and four games of our own design.

Cookie Clicker is a resource-driven game that does not rely on spatial movement logics, where the goal is to bootstrap automated cookie-production systems that escalate to a constantly growing tower of upgrades and achievements. The key dynamics in the game are feedback loops and investment—e.g., spending a relatively small amount of cookies to purchase an upgrade leads to the permanent faster production of cookies. We formalize mechanics such as *clicking on a cookie increases cookies*, *time passing causes cookies to increase by "cookies per second,"* and *buying a producer increases cookies per second by that producer's production rate, subtracts cost from number of cookies, and increases the subsequent cost of the producer*. As there is no extrinsically stated goal in *Cookie Clicker*, we have to supply cultural knowledge to the encoding in order to provide a basis for understanding the game. Namely, we add the fact that *cookies are good* to represent the likely player interpretation of cookies as desirable.

The system determines that it costs cookies to increase the cookies-per-second, which then leads to the creation of more cookies, creating a positive feedback loop. Due to the fact that clicking on the cookie is unrestricted, that clicking generates cookies, and that cookies are a restriction on buying grandmas and farms, it deduces that a player must bootstrap by first clicking on cookies to be able to buy a producer. Given the positive feedback loop associated with cookies-per-second and that cookies are good, the system reasons that buying a producer represents an investment. Finally, given that both grandmas and farms require and consume cookies and the initialized values for their costs, it reasons that the player must make a choice of where to allocate their cookies, with grandmas having the lower initial cost of the two (although an indeterminate ordering on an arbitrary time scale).

We tried our analysis on several other game encodings, including *Pong, Kaboom,* and games of our own design. The kind of knowledge we are able to generate for these games includes the following.

1) For *Pong,* we derive that the game is a symmetric competition, that the two players are antagonists of each other, and that each player will try to hit the ball with their respective paddle.
2) For *Kaboom!,* we derive facts such as that the bomber is an antagonist, bombs harm the player, the bucket harms bombs, the player will attempt to cause the bombs to

[3]http://orteil.dashnet.org/cookieclicker/

collide with the bucket, and the player will attempt to avoid causing the bombs to collide with the bottom of the screen. We also deduce that difficulty increases monotonically.

3) For our own games, we encode reasoning analogous to design specifications, such as "a player must perform a risky hand–eye coordination task to keep their cool" and "the player must make time-sensitive decisions about resource allocation whose consequences have a tradeoff between personal benefit and global cost."

For external validation, we look at the work of Treanor *et al.* [25] who performed a reading of *Kaboom!*, wherein they found such things as:

1) "Because *Kaboom!* is unwinnable (like many early arcade-style games, the difficulty just keeps progressing until inevitable defeat), this quest to protect the world from damage is ultimately hopeless."
2) "We know that bombs are destructive and that the player's goal is to intercept them with diffusing buckets."

Our readings decisively capture the second of these; for the first, we deduce that difficulty increases indefinitely and the game is unwinnable, but our system fails to overlay the additional aesthetic meanings of hopelessness and protection.

As for *Pong,* the closest thing to a "gold standard" reading may be the single instruction on the *Pong* arcade cabinet: "Avoid missing ball for high score" [26].

Our system successfully reconstructs this meaning: The ball hitting the opposing side leads to an increase in score, the only method of interaction is hitting the ball, therefore, "Avoid missing ball for high score."

Overall, we found that human-generated dynamics and critical readings of these games matched the interpretations found by our system, lending support to its ability to provide useful interpretations of existing game artifacts. To the extent that our system missed certain readings, we are confident that the reasoning used to arrive at those readings is authorable; however, it remains an open research question whether a finite, static body of reasoning principles can ever encapsulate the breadth of reasoning principles employed by human game critics.

We also analyzed the expressive range of our system by utilizing the generation half of *Gemini*. Limiting the scope of our generator to games involving two or fewer distinct classes of entities and at most two resources, we were able to generate over 500 000 distinct readings.

VI. Discussion

We have demonstrated a novel approach to the problem of relating game mechanics to higher level meanings based on formalizing a theory of proceduralist readings. We introduced two novel and distinct knowledge formalisms: the specification language Cygnus, in which we express mechanics, and the meaning-level design intent language. Cygnus represents a substantial language design effort, evading simple characterization due to its dual role of supporting meaning-level interpretation while also unambiguously mapping onto an executable game (making it feasible to write a compiler back-end). The meaning-level design intent language is informed by microrhetorics [27] and proceduralist readings, and allows for generative variation through a many-to-many relationship: specifications may give rise to multiple meanings, and a single meaning may have many instantiations as a specification.

Currently, we assess our work as forming a good account of meaning for games with resource transactions, movement and collision logics, and graphical communication affordances (including, notably, some high-level representation of continuous physics, which we did not illustrate in our previous examples). Thus, we can already express simple arcade games, platformers, and incremental games in a uniform way, which we assessed would have been very difficult in existing game formalisms, such as the VGDL [28]. Furthermore, within our general framework of conditions, results, and communicative affordances, we essentially express games in terms of their operational logics [29], free from any particular style or genre of game, which grants us the ability to scale to reasoning about a large range of other common and innovative mechanical idioms at little extra authoring cost. For instance, we expect that we could easily codify the mechanics of a game, such as *Papers, Please* [1], which escapes easy representation in formalisms like VGDL.

ASP has been an excellent tool for prototyping our bidirectional, nonpipelined system since we can simultaneously test our rules in the context of generation and game understanding, refining them in light of mutually beneficial rules. On the other hand, ASP is not a perfect match for proceduralist meaning derivations in that it only produces stable models, not structured proofs. Since meaning derivations are isomorphic to logical proofs, a sensible alternative might be using the Elf programming language [30], which produces proofs as well as answers. However, moving to a formalism like Elf would lose ASP's benefits of generating a complete stable model, pipeline freeness, and ability to specify complex constraints involving negation.

Currently, there are two main limitations to the analysis portion of *Gemini*: 1) it is focused on single-screen 2-D action games, and 2) it only performs static analyses. The first is an authorship challenge, as Cygnus and *Gemini* are capable of supporting other styles of games, but given our generation domain/the domain of previous analyses, authoring out-of-domain rules has not been focus. The second does limit the scope of readings possible. While we are able to develop rules that handle limited dynamics (e.g., **Rule 6** in Fig. 7), large-scale dynamics, like the types of properties found by *Ludi* [31], are likely beyond the scope of *Gemini*. We note that running simulations as part of the ASP process is possible, as it has been done by Zook and Riedl [15] and Smith *et al.* [32], but as with 1) it presents an authoring challenge.

In general, while *Gemini* as currently authored is capable of analyzing and generating a wide range of games, there is always likely to be a gap between the space of all possible analyses/games and what can be produced by *Gemini* due to the authoring challenges. We note that the general process of authoring for *Gemini* is incrementally easier than for a system like *Game-o-Matic* due to our ability to utilize previous readings to either build up new readings or codify new ways of producing a certain reading. For instance, **Rule 3** in Fig. 7 is only one

possible way of producing the reading of a favorable outcome and we could write other rules that would have the same reading (e.g., an outcome is favorable if it increases the health of the player or if it decreases the health of an antagonist).

Evaluating our game analysis engine beyond comparison to existing readings could potentially include an expert evaluation, requesting critical readings of games from leading scholars, formalizing their arguments, and comparing the shapes of the derivation trees to the ones our system generates. We could also compare the shapes of generated meaning derivations internally, i.e., carry out an expressive range analysis [33] over variables, such as proof tree width, depth, and number of rules in the reasoning principle base used, repeated, or omitted. Furthermore, we could use these properties of meaning derivations as characteristics of games themselves to reason over or optimize for: Perhaps we want games with the fewest possible meanings but whose deepest meaning derivation is as deep as possible, or some other combination of constraints.

In summary, we have presented a novel approach to computational game interpretation based on a formalization of procedural reasoning, resulting in a static analysis relating mechanic-based game definitions to high-level meanings. This paper represents an effort to combine mechanics-level game specification, cultural knowledge, and operational logics in one formalism, resulting in a highly expressive reasoning space.

Acknowledgment

The authors would like to thank M. Treanor and M. Guzdial for helpful discussion of the ideas behind this paper.

References

[1] L. Pope, "Papers, please," Aug. 2013. [Online]. Available: http://papersplea.se/
[2] P. Pedercini, "Unmanned," 2012. [Online]. Available: http://unmanned.molleindustria.org/
[3] I. Bogost, *Persuasive Games: The Expressive Power of Videogames*. Cambridge, MA, USA: MIT Press, 2007.
[4] A. Delwiche, "From the green berets to America's army: Video games as a vehicle for political propaganda," in *The Players Realm: Studies on the Culture of Video Games and Gaming*. Jefferson, NC, USA: McFarland and Company, 2007, pp. 91–109.
[5] J. Zagal, "Ethically notable videogames: Moral dilemmas and gameplay," in *Proc. Int. Conf. Digit. Games Res. Assoc. Found. Digit. Games*, 2009, pp. 1–9.
[6] J. Orwant, "EGGG: Automated programming for game generation," *IBM Syst. J.*, vol. 39, no. 3/4, pp. 782–794, 2000.
[7] M. J. Nelson and M. Mateas, "Towards automated game design," in *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*. New York, NY, USA: Springer-Verlag, 2007, pp. 626–637.
[8] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proc. IEEE Symp. Comput. Intell. Games*, 2008, pp. 111–118.
[9] R. Hunicke, M. Leblanc, and R. Zubek, "MDA: A formal approach to game design and game research," in *Proc. Challenges Games AI Workshop, 19th Nat. Conf. Artif. Intell.*, 2004, pp. 1–5.
[10] M. Treanor, B. Schweizer, I. Bogost, and M. Mateas, "Proceduralist readings: How to find meaning in games with graphical logics," in *Proc. 6th Found. Digit. Games*, 2011, pp. 115–122.
[11] D. Ventura, "Mere generation: Essential barometer or dated concept," in *Proc. 7th Int. Conf. Comput. Creativity*, 2016, pp. 1–8.
[12] M. Treanor, B. Blackford, M. Mateas, and I. Bogost, "Game-o-Matic: Generating videogames that represent ideas," in *Procedural Content Gener. Workshop Found. Digit. Games Conf.*, 2012, pp. 1–8.
[13] A. M. Smith and M. Mateas, "Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games," in *Proc. IEEE Conf. Comput. Intell. Games*, 2010, pp. 273–280.
[14] M. Cook and S. Colton, "Ludus Ex Machina: Building a 3D game designer that competes alongside humans," in *Proc. 5th Int. Conf. Comput. Creativity*, 2014, vol. 380, pp. 1–9.
[15] A. Zook and M. O. Riedl, "Automatic game design via mechanic generation," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 530–536.
[16] M. Cook and S. Colton, "A rogue dream: Automatically generating meaningful content for games," in *Proc. AIIDE Workshop Exp. Artif. Intell. Games*, 2014, pp. 1–6.
[17] M. Cook and S. Colton, "From mechanics to meaning and back again: Exploring techniques for the contextualisation of code," in *Proc. AIIDE Workshop Artif. Intell. Game Aesthetics*, 2013, pp. 1–5.
[18] G. Kuhlmann, K. Dresner, and P. Stone, "Automatic heuristic construction in a complete general game player," in *Proc. 21st Nat. Conf. Artif. Intell.*, Jul. 2006, pp. 1457–62.
[19] S. Schiffel and M. Thielscher, "Fluxplayer: A successful general game player," in *Proc. 22nd Nat. Conf. Artif. Intell.*, 2007, vol. 2, pp. 1191–1196.
[20] D. Cook, "The chemistry of game design," 2007. [Online]. Available: Gamasutra.com
[21] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. Cambridge, MA, USA: MIT Press, 2004.
[22] C. Martens, "Ceptre: A language for modeling generative interactive systems," in *Proc. 11th Artif. Intell. Interact. Digit. Entertainment Conf.*, 2015, pp. 1–7.
[23] K. T. Stolee and T. Fristoe, "Expressing computer science concepts through Kodu game lab," in *Proc. 42nd ACM Tech. Symp. Comput. Sci. Educ.*, 2011, pp. 99–104.
[24] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, "Clingo = ASP + control: Preliminary report," *Theory and Practice of Logic Programming*, vol. 14, p. Online Supplement, 2014.
[25] M. Treanor, M. Mateas, and N. Wardrip-Fruin, "Kaboom! is a many-splendored thing: An interpretation and design methodology for message-driven games using graphical logics," in *Proc. 5th Int. Conf. Found. Digit. Games*, 2010, pp. 224–231.
[26] A. Alcorn, "Pong," 1972.
[27] M. Treanor, B. Schweizer, I. Bogost, and M. Mateas, "The micro-rhetorics of game-o-matic," in *Proc. Int. Conf. Found. Digit. Games*, 2012, pp. 18–25.
[28] T. Schaul, "A video game description language for model-based or interactive learning," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–8.
[29] M. Mateas and N. Wardrip-Fruin , "Defining operational logics," in *Proc. Digit. Games Res. Assoc.*, 2009, pp. 1–8.
[30] F. Pfenning, "Elf: A language for logic definition and verified metaprogramming," in *Proc. 4th Annu. Symp. Logic Comput. Sci.*, 1989, pp. 313–322.
[31] C. Browne, *Evolutionary Game Design*. Berlin, Germany: Springer Science & Business Media, 2011.
[32] A. M. Smith, M. J. Nelson, and M. Mateas, "Computational support for play testing game sketches," in *Proc. Artif. Intell. Interact. Digit. Entertainment*, 2009, pp. 1–6.
[33] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Proc. Workshop Procedural Content Gener. Games*, 2010, Art. no. 4, pp. 1–4.

Authors' photographs and biographies not available at the time of publication.