Formal Game Grammar and Equivalence

Paul Riggins
Berkeley Center for Theoretical Physics
University of California
Berkeley, CA, USA
priggins@berkeley.edu

David McPherson
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA, USA
david.mcpherson@eecs.berkeley.edu

***Abstract*—We develop methods to formally describe and compare games, in order to probe questions of game structure and design, and as a stepping stone to predicting player behavior from design patterns. We define a grammar-like formalism to describe finite discrete games without hidden information, allowing for randomness, and mixed sequential and simultaneous play. We make minimal assumptions about the form or content of game rules or user interface. The associated game trees resemble hybrid extensive- and strategic-form games, in the game theory sense. By transforming and comparing game trees, we develop equivalence relations on the space of game systems, which equate games that give players the same meaningful agency. We bring these together to suggest a method to measure distance between games, insensitive to cosmetic variations in the game logic descriptions.**

***Index Terms*—game grammar, game representation, game tree, equivalence relations, similarity measures, mathematical ludology**

I. Introduction

Games are notable among artistic media in that the interactive systems underlying them can often be precisely defined. This has enabled great mathematical progress in understanding game-centric decision-making processes, through efforts in game theory and artificial intelligence (AI) (e.g., [1]), but this progress has been largely isolated from other, "softer" subfields of game studies [2], [3]. In particular, little mathematical attention has been given to questions of game design, even while the methods and vocabulary used by designers has become increasingly sophisticated and systematic (e.g., [4]–[8]). The precise definability of games, however, could also be used to formally explore questions of interest to ludologers and designers: How much does a game mechanic matter for overall gameplay? What's the best user interface to reflect the underlying rules? Can we predict behavior in one game based on behavior in a similar game?

In [9], we proposed the study of mathematical ludology, aiming to bring mathematical attention to questions like these by formally exploring the space of games and their properties. This is, in some sense, a mathematically formal continuation of "game grammar" efforts begun in the game design community to atomize and interrogate game designs (e.g., [6], [10]–[12]). In this paper[1] we especially develop and explore formal notions of equivalence and similarity between games.

The ability to compare games for equivalence or similarity could be useful for studying relationships, building taxonomies, transferring behavioral learning (perhaps AI learning) from one game to another, or maybe even developing approximate game theoretic solutions for similar games. A key challenge, however, lies in the many ways the same game can be described, even with the same formalism. For instance, we are aware of one other complementary work-in-progress aiming to measure distance between games: the Digital Ludeme Project, advancing archaeoludology [13]. The proposed method, using edit distance between rule trees, is powerful but sensitive to cosmetic differences in the game rules: two different descriptions could describe identical games, yet have nonzero distance. The senses of equivalence and similarity we propose here are specifically designed to avoid this aesthetic sensitivity.

The key **contributions** of this paper are

1) We develop a grammar-like formalism to describe finite, discrete games (Sec. II), more flexible than game theoretic methods and more tailored for abstract structural analysis than general gameplaying (GGP) approaches.
2) After constructing game trees (Sec. III), we develop equivalence relations on the spaces of game trees and game systems (Sec. IV).
3) We bring these together to suggest a way to measure game similarity—insensitive to cosmetic variations in rule descriptions—by sampling the game state spaces and checking for equivalence of partial game trees (Sec. V).

II. Underlying Game Systems

The present paper will focus on the essential logic of games, what we will call the *(underlying) game system.*[2] This is the base of a game description hierarchy described in [9] (see Fig. 1), and in particular does not specify the information available to (or hidden from) players, or the form of the user interface—these are naturally quite important, but also add many complexities. An underlying game system provides the game logic as an omniscient referee might see it—as a (nondeterministic) game of perfect information—which is interesting in its own right and a useful foundation to build on. In contrast to game theoretic approaches, we do not include player preferences or payoffs (i.e., how individual players value different outcomes) in our game descriptions. Thus we draw a formal separation between game and players, adhering more closely to the colloquial understanding of a game.

[1]Most of the content in this paper first appeared in the preprint [9].
[2]Comparable to the "constituative [*sic*] rules" of [4].

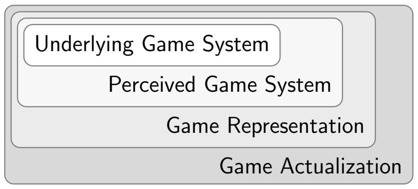


Fig. 1. A game description hierarchy (see [9]). Underlying Game Systems contain the essential logic of a game. (These are the focus of this paper.) Perceived Game Systems add an information layer, specifying what information about the game is given to each player, and how this interacts with the game logic. Game Representations add a user interface specification, how the game is represented to players, e.g., visually. Game Actualizations are the real-world realizations, like a physical chess set or software. Player preferences, skill levels, etc., are relegated to Player Models, outside these game descriptions.

Specifically, we will focus on games that are discrete in time and space and that have a finite state space; Def. 1 can describe the game system of any such game. This captures the majority of board and card games, and many video games. Def. 1 is not the only way to describe a game system, but it captures those minimal elements essential to the logic of a game, while also making formally clear what agency each player has versus what is outside their control.

In particular, Def. 1 provides: the players ($\mathcal{P}$), the game state space ($\mathbb{S}$, factorized via $\mathcal{T}$), initial game states ($\mathcal{S}_0$), decisions available to players ($\mathcal{D}$), when those decisions are legal (L), the possible consequences of those decisions (C), how the game state is changed as a result ($\mathcal{A}$), and possible game outcomes ($\mathcal{O}, \Omega$). It allows for mixed sequential and simultaneous play, deterministic or nondeterministic, and makes no additional assumptions about the content of games, not even presuming the existence of boards, pieces, or alternating turns.

Definition 1 An *(underlying) game system* $\mathcal{G}$ with n players is a 9-tuple $\mathcal{G} = \langle \mathcal{P}, \mathcal{T}, \mathcal{S}_0, \mathcal{D}, \mathcal{A}, C, L, \mathcal{O}, \Omega \rangle$, where:

- $\mathcal{P} = (1, \ldots, n)$ is a list of *players*, agents which may make decisions in the game.
- $\mathcal{T} = (T_1, \ldots, T_m)$ is a finite list of finite sets, called *substate tracks*. The set of *game states* $\mathbb{S}$ is given by $\mathbb{S} = T_1 \times \cdots \times T_m$.
- $\mathcal{S}_0 \subset \mathbb{S}$ is a set of *initial conditions*.
- $\mathcal{D}$ is a set of *decisions*, the choices which players may make in order to influence (but not directly change) the game state. This is extended with the *null decision* $0 \notin \mathcal{D}$ to form $\mathcal{D}_0 \equiv \mathcal{D} \cup \{0\}$.
- $\mathcal{A}$ is a set of *actions* $a : \mathbb{S} \to \mathbb{S}$, which can directly modify the game state.
- C is a *consequence function* $C(d_0^n, s)$ which takes a decision tuple $d_0^n \in \mathcal{D}_0^n$ (i.e., one possibly null decision per player) and state $s \in \mathbb{S}$ and returns a nonempty set of *consequences*: a set of pairs $(\mathfrak{p}_a, a)$, where $\mathfrak{p}_a \in (0, 1]$ is a non-zero probability and a is an action or product of actions. The sum of probabilities in the set must equal 1. These are the consequences of decisions, which may be outside any one player's control.
- $L : \mathcal{P} \times \mathcal{D} \to 2^{\mathbb{S}}$ is a *legality function*, which returns a (possibly empty) subset of $\mathbb{S}$ for each player $p \in \mathcal{P}$ and decision $d \in \mathcal{D}$, reflecting when that player can make that decision. The *legal set of decisions for player p at state s* is the set $L_p(s) \equiv \{d \in \mathcal{D} : s \in L(p, d)\}$. A decision $d \in L_p(s)$ is *legal* for p at s, and *illegal* otherwise.
- $\mathcal{O}$ is the set of *outcomes* that can result from the game.
- Ω is an *outcome function* $\Omega : \mathcal{S}_{\text{ter}} \to \mathcal{O}$, where $\mathcal{S}_{\text{ter}} \equiv \{s \in \mathbb{S} : L_p(s) = \varnothing \text{ for all } p \in \mathcal{P}\}$ is the set of *terminal game states*. Intuitively, $\mathcal{S}_{\text{ter}}$ is the set of game states at which no legal decisions can be made, so the game ends and the result is computed by Ω.

The separation of decisions (player choices) from actions (changes to the game state) via consequence functions provides a formal separation between what individual players can and cannot control. Consequence functions are necessary to handle random chance and simultaneous play. In both cases, each player can influence play by their chosen decision, but the ultimate effect on the game state (the consequent action) is determined probabilistically and/or after considering the simultaneous decisions of other players (see Alg. 1 and Ex. 1). In sequential, deterministic games, it is possible to have a one-to-one mapping between decisions and actions—individual players can directly determine game state changes—and so consequence functions are conceptually superfluous.

The following algorithm can be used (by players) to play any game with an underlying game system described by Def. 1:

Algorithm 1: Gameplay Algorithm

```
1 All players agree on some $s_0 \in \mathcal{S}_0$. Let the current state
    be $s' = s_0$.
2 While $s'$ is not a terminal state ($s' \notin \mathcal{S}_{\text{ter}}$), repeat:
3     Each player $p$ selects one decision from their respective
        legal set $L_p(s')$ at the current state. If $L_p(s') = \varnothing$ for
        a player $p$, then $p$ is assigned the null decision: $d_p = 0$.
4     Compute the set of consequences from the decision
        tuple and the current state: $c = C((d_1, \ldots, d_n), s') =$
        $\{(\mathfrak{p}_1, a_1), \ldots, (\mathfrak{p}_m, a_m)\}$. Randomly pick a consequent
        action from $c$, where $a_j$ is chosen with probability $\mathfrak{p}_j$.
5     Compute the new game state $s'' = a_j s'$. Let $s' = s''$.
6 The game is over ($s' \in \mathcal{S}_{\text{ter}}$). Compute the outcome $\Omega(s')$.
```

If this algorithm can always be faithfully executed for a game system, that system is *complete* [9], and its game tree(s) can be built (see Sec. III).

Here is an example of a tic-tac-toe variant, with basic notation described along the way. Its game tree is illustrated in Fig. 3. More complex games (or even this one) can benefit from richer notation in order to write them more compactly. Some notational suggestions and examples can be found in [9]; developing notation is not the purpose of the present paper.

Example 1 (Tic-Tac-Toe with random start) Two players play tic-tac-toe (or "3-to-15"), randomly choosing who starts.

2	7	6
9	5	1
4	3	8

Fig. 2. "Magic square" correspondence between tic-tac-toe and 3-to-15.

(States.) We write $(v)_t$ to express the subset of $\mathbb{S}$ where track T_t takes value v; product and sum notation on these subsets denote intersection and union, respectively.

(Actions.) By $a : S \mapsto (v_1)_1 \cdots (v_k)_k$, we mean that for any state in the subset $S \subset \mathbb{S}$, the action $a \in \mathcal{A}$ changes the value of track T_1 to v_1, and so on to track T_k, acting as the identity on any tracks not appearing on the right-hand side. It also acts as the identity on any state $s \notin S$. E.g., if $a : \mathbb{S} \mapsto (1)_a$ and we have some state $s = (2)_a(3)_b$, then $a \cdot s = (1)_a(3)_b$.

(Outcome functions.) We take $\Omega : S \mapsto \omega$ to mean $\Omega(z) = \omega$ for all terminal states $z \in S \subset \mathbb{S}$.

We'll break up the game description for ease of exposition. There are two players and 10 available decisions: one for each space, plus the coin flip. There are 10 tracks to record the turn and spaces, and the initial condition is an empty board.

$$\begin{aligned}
&\mathcal{P} = \{\text{X}, \text{O}\}, \quad \mathcal{D} = \{1, \dots, 9, \text{flip}\} \\
&\mathcal{T} : \ T_{\text{turn}} = \{\text{start}, \text{X}, \text{O}\}, \\
&\qquad T_i \quad = \{-, \text{X}, \text{O}\} \ \text{ for } \ i \in \{1, \dots, 9\} \\
&\mathcal{S}_0 = (\text{start})_{\text{turn}}(-)_1 \cdots (-)_9
\end{aligned}$$

From the initial condition, both players can only legally (L) choose the decision "flip". As a consequence (C) of this joint decision, the value of track T_{turn} becomes X or O with a 50-50 chance, via the action "X first" or "O first".

$$\begin{aligned}
&L(p, \text{flip}) = (\text{start})_{\text{turn}}, \quad p \in \mathcal{P} \\
&C((\text{flip}, \text{flip}), \mathbb{S}) = \{(1/2, \text{X first}), (1/2, \text{O first})\} \\
&\mathcal{A} \supset \{\text{X first, O first}\}, \quad p \text{ first} : (\text{start})_{\text{turn}} \mapsto (p)_{\text{turn}}
\end{aligned}$$

Players then alternate (enforced by $L(p, i)$ and the "turn" track), picking unclaimed ("$-$") spaces via the decisions $1, \dots, 9 \in \mathcal{D}$ until either no more spaces are available or someone wins.

$$\begin{aligned}
&L(p, i) = (p)_{\text{turn}}(-)_i \ \overline{E}, \qquad i \in \{1, \dots, 9\} \\
&C((i, 0), \mathbb{S}) = (1, \text{X}_i \cdot \text{next}), \ \ C((0, i), \mathbb{S}) = (1, \text{O}_i \cdot \text{next}) \\
&\mathcal{A} \supset \{\text{X}_1, \text{O}_1, \dots, \text{X}_9, \text{O}_9\}, \quad p_i : (-)_i \mapsto (p)_i \\
&\mathcal{A} \supset \{\text{next}\}, \ \ \text{next} : (\text{X})_{\text{turn}} \mapsto (\text{O})_{\text{turn}}, \ (\text{O})_{\text{turn}} \mapsto (\text{X})_{\text{turn}}
\end{aligned}$$

The game then ends in victory or draw, respectively. Note we've defined an auxiliary set E to compactly capture winning ending states. (The complement $\overline{E}$ appears in $L(p, i)$, so play can only legally proceed if a victory state has not been reached.)

$$\begin{aligned}
&E = E_{\text{X}} \cup E_O, \qquad\qquad E_p = \sum_{\substack{i,j,k \in \{1, \dots, 9\} \\ i+j+k=15}} (p)_i(p)_j(p)_k \\
&\mathcal{O} = \{\text{X wins}, \text{O wins}, \text{draw}\} \\
&\Omega : E_p \mapsto p \text{ wins}, \quad \text{otherwise} \mapsto \text{draw}
\end{aligned}$$

In tic-tac-toe, the ending states correspond to three-in-a-row board states. In 3-to-15, these correspond to integer triples that sum to fifteen. These games may be played with different user interfaces, but they share the same essential logic (see Fig. 2), and thus the same underlying game system.

A. Why not use an existing game description formalism?

Game theoretic descriptions are too limited in the games they can practically express. Strategic- and extensive-form games respectively describe simultaneous and sequential games well, but lose important nuance when trying to mix the two [14], [15]. Additionally, complex game descriptions are intractable with game theoretic formal descriptions. Generally, either the full game can be written explicitly as a strategic, extensive, or combinatorial game (all intractable, e.g., for chess), or else game theorists rely on ad hoc natural language description and reader familiarity to communicate the rules before proceeding to analysis (e.g., [16]). We want a way to formally, and tractably, describe game rules even for complex games; a grammar-like formalism is better suited to this task.

We also desire a total conceptual separation between game and player descriptions; game theory does not make this separation. This separation is also why we have included randomness in the game description itself—in contrast to typical game theory or GGP formalisms, which invoke an extra fictional player who behaves randomly [17]–[19].

GGP descriptions like GDL [19]–[21], RBG [22], or Ludii [18] are designed especially for efficient software implementation and AI methods, often with particular classes of games in mind. Ludii and RBG have compact notation, but require the construction of a game board (i.e., visual user interface) integrated with the rules, distinguishing them most naturally as game representations (see Fig. 1) suited to traditional board games; we desire a lower-level description to study and compare game rules, which makes fewer assumptions. GDL is very generic, like we might prefer, but can be intractably verbose, with an opaque state space. Our Def. 1 bears some formal similarity to GDL-II without hidden information, and is just as expressive, but offers a simpler specification of state space, a different treatment of randomness, and we permit a more compact and extensible notation.

III. Game Trees

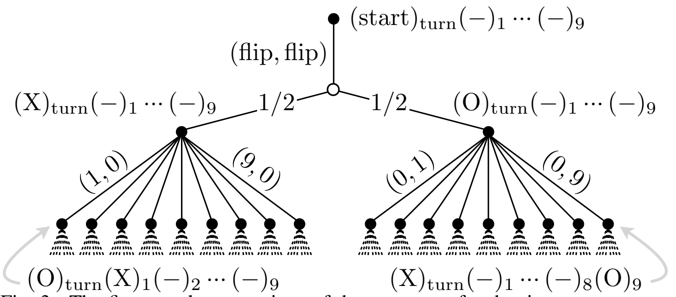
A complete game system from Def. 1 can be used to generate a (possibly infinite) game tree or a finite (possibly nondeterministic) game automaton. These are useful for visualizing and analyzing the game systems, as well as for making connection with existing work in game theory and AI. They are graphical representations of Alg. 1: each playthrough from that algorithm identifies a path from an initial node to a terminal node in a tree or automaton. We will focus on game trees in this paper.

Definition 2 The *game trees* of a complete game system $\mathcal{G}$ is a set of game trees $\tau(\mathcal{G}) \equiv \{\tau(\mathcal{G}|s_0) : s_0 \in \mathcal{S}_0\}$, one for each initial condition. Each game tree $\tau(\mathcal{G}|s_0)$ is built via Alg. 2.

Each tree resulting from Alg. 2 has the following structure: *State nodes* have assigned states and outgoing *decision edges*, which have assigned decision tuples. These decision edges lead to either new state nodes, or to unlabeled *chance nodes* which have outgoing *chance edges* labeled with probabilities. These chance edges lead to new state nodes. State nodes may be further subdivided into *single-player nodes*, in which only one player has legal decisions available (the node *belongs* to that player); *multiplayer nodes*, in which multiple players have legal

Algorithm 2: Game Tree Construction, for $\tau(\mathcal{G}|s_0)$

```
1  Draw a root node, assigned the initial state $s_0$.
2  **While** not all leaves have assigned outcomes, repeat:
3      **For** each leaf node $w$ in the current tree, with assigned
       state $s(w)$ but no assigned outcome, do the following:
4          Let $s = s(w)$. **If** $s \in \mathcal{S}_{\text{ter}}$, **then** assign the outcome
           $\Omega(s)$ to $w$, and stop for node $w$. **Else**, proceed:
5          Generate all legal decision tuples at state $s$ from the
           legal set $L_p(s)$ for each player: $D_0^n(s) \equiv \{(d_1, \ldots, d_n)$:
           $d_p = 0$ if $L_p(s) = \varnothing$, else $d_p \in L_p(s)\}$.
6          **For** each tuple $d_0^n \in D_0^n(s)$, do the following:
7              Draw a child node $w'$ below $w$, with a directed edge
               from $w$ to $w'$. Assign $d_0^n$ to this edge.
8              Compute the set of consequences $c' = C(d_0^n, s)$.
9              **If** $c' = \{(1, a)\}$, **then** assign the state $a \cdot s$ to $w'$. **Else**:
10             **For** each probability-action pair $(\mathfrak{p}_i, a_i) \in c'$, draw a
               child node $w_i''$ with a directed edge from $w'$ to $w_i''$.
               Assign $\mathfrak{p}_i$ to this edge, and the state $a_i \cdot s$ to $w_i''$.
```



$(\text{start})_{\text{turn}}(-)_1 \cdots (-)_9$
(flip, flip)
$(\text{X})_{\text{turn}}(-)_1 \cdots (-)_9$
1/2
1/2
$(\text{O})_{\text{turn}}(-)_1 \cdots (-)_9$
(1, 0)
(9, 0)
(0, 1)
(0, 9)
$(\text{O})_{\text{turn}}(\text{X})_1(-)_2 \cdots (-)_9$
$(\text{X})_{\text{turn}}(-)_1 \cdots (-)_8(\text{O})_9$

Fig. 3. The first couple generations of the game tree for the tic-tac-toe system in Ex. 1, with only a few state nodes and decision edges labeled, for brevity. Solid nodes are state nodes, while the open node is a chance node. The root state node has the initial state from $\mathcal{S}_0$. The first decision edge with tuple (flip, flip) leads to a chance node, with outgoing chance edges both with probability $1/2$. Subsequent decisions claim numbers for a player and toggle the turn track, e.g., $(1, 0)$ claims 1 for X; from the resulting state $(\text{O})_{\text{turn}}(\text{X})_1(-)_2 \cdots (-)_9$, there are 8 legal decision tuples $(0, 2), \ldots, (0, 9)$ as the game tree continues, which would each claim a number for player O.

decisions available; and *terminal state nodes*, which correspond to terminal states and have outcomes assigned to them.

A game tree generated by Alg. 2 is a sort of hybrid between game theoretic extensive- and strategic-form games (e.g., see [17]), except with no information sets. On single-player nodes, individual players choose an outgoing edge to follow. To capture simultaneous play, multiplayer nodes act as strategic ("matrix") form games: several players must simultaneously make a decision, which is then evaluated by an umpire to choose an outgoing edge to follow. (We describe this strategic-form game with a *decision matrix*, see Def. 8.) This avoids the ambiguities inherent in the information set construction of simultaneous play in extensive form games [23], which relies on sequential moves with hidden information, and can be experimentally different from true simultaneous play [14], [15]. Ultimately, an outcome is produced at a terminal node.

IV. Agency Equivalence

The game system Def. 1 is deliberately flexible, in order to describe any discrete game. However, its flexibility means that there are several ways to express the "same" game. Here we develop two precise senses of this "sameness": *game tree equivalence (up to relabeling)* and *agency equivalence*. This is a useful precursor to measuring distance between games, where we might wish equivalent games to have zero distance.

Game tree equivalence up to relabeling (Def. 11 and Sec. IV-A) matches game systems if they produce the same game trees, with some variation in aesthetic labeling: e.g., the names of decisions or states can differ, but probabilities cannot.

Agency equivalence (Def. 3) matches game systems if they offer players the same agency, that is, the same sorts of meaningful choices with the same sorts of consequences. We will define this by performing a series of reductions on game trees to prune spurious differences, declaring two game systems equivalent if their reduced trees match. There are four kinds of differences that we consider spurious for this purpose; we describe them heuristically here, then formally in Sec. IV-B:

A *bookkeeping subtree* (Def. 12) is a portion of a game tree where there is only one decision available at each state. Though there may be randomness involved, play continues on the subtree inevitably, without any chance for player influence. Consider a version of tic-tac-toe, in which players must declare end-of-turn after placing a symbol (X, end-of-turn, O, end-of-turn, ...). From the standpoint of player agency, we do not consider this meaningfully different from standard tic-tac-toe, in which turns automatically advance (X, O, X, O, ...).

A *single-player subtree* (Def. 13) is a portion of the game tree where the same single player makes several deterministic decisions in a row. There is no difference in options or outcomes if the player makes these decisions one at a time or all at once. For instance, pawn promotion in chess could be split into two steps with an intermediate state (move, then promote), or lumped into one (move-and-promote). We do not consider these different from the standpoint of player agency.

A *symmetry-redundant subtree* (Def. 14) is a portion of the game tree that is unnecessary because it duplicates a sibling subtree. Because of the board symmetry in tic-tac-toe, starting in one corner versus another corner leads to substantively the same remaining decisions for the rest of the game—even though the precise game states are different. Even if a version of tic-tac-toe forbid players from starting in three of the corners, and three of the sides, we would consider it identical to standard tic-tac-toe from the standpoint of meaningful player agency.

Finally, a *decision matrix redundancy* (Def. 15) occurs when a player has two decisions at a state that would have identical results—it really does not matter which one they pick. This player would have the same agency if they had only one of those decisions available. Putting all these together:

Definition 3 We say two game systems $\mathcal{G}$ and $\mathcal{G}'$ are *agency equivalent* if their respective game trees[3] can be made equiv-

[3]For now, this and following definitions can only be usefully applied to finite game trees, though infinite trees could be truncated and similarly compared.

alent up to relabeling (Def. 11) by performing the following reductions, as many times as necessary, in any order:

- Bookkeeping subtree reduction (Def. 12)
- Single-player subtree reduction (Def. 13)
- Symmetry-redundant subtree reduction (Def. 14)
- Decision matrix redundancy reduction (Def. 15)

The following subsections flesh out the technical details for these two senses of equivalence. Note, the reductions mentioned in Def. 3 produce *reduced game trees*, editing the game trees produced from Alg. 2 to remove non-essential information. The term "game tree" should be understood below to refer to both reduced and unreduced game trees.

A. Game Tree Equivalence up to Relabeling

To start, let's delete all tree labels and match what remains:

Definition 4 A *stripped game tree*, denoted $\langle T \rangle$, is a game tree T with all labels removed; only the arrangement of nodes and edges remains.

Definition 5 (Structural equivalence) Two game trees T, T' (or game systems $\mathcal{G}, \mathcal{G}'$) are *structurally equivalent* if the stripped trees are equal $\langle T \rangle = \langle T' \rangle$ (or if the sets of stripped game trees are equal $\langle \tau(\mathcal{G}) \rangle = \langle \tau(\mathcal{G}') \rangle$).

This establishes a bijective *structural correspondence* $f : n \mapsto n'$, similarly $f : e \mapsto e'$, between the labelled nodes and edges of corresponding trees T and T' (or $t \in \tau(\mathcal{G})$ and $t' \in \tau(\mathcal{G}')$). Several such correspondences may be possible (e.g., because a tree is symmetric).

This is sufficient to say that the arrangement of nodes and edges is the same. However, some labels do contain important content that distinguishes two game systems in substance, not just aesthetics. In particular, we want to see that corresponding probabilities are the same, players have the same kinds of choices available, and that the outcomes are similarly distinct. Comparing probabilities and outcomes is straightforward:

Definition 6 (Matching probabilities) For each chance edge e in a game tree, let $\mathsf{p}(e)$ be the assigned probability. Two structurally equivalent game trees T, T' (or game systems $\mathcal{G}, \mathcal{G}'$) with structural correspondence $f : T \to T'$ are said to have *matching probabilities* if $\mathsf{p}(e) = \mathsf{p}(f(e))$ for all chance edges $e \in T$ $(\in \tau(\mathcal{G}))$.

Definition 7 (Similarly distinct outcomes) Take two structurally equivalent game trees T, T' (or game systems $\mathcal{G}, \mathcal{G}'$) with correspondence f, let O, O' be the sets of all distinct outcomes assigned to their respective terminal nodes, and let $\Omega(z)$ be the outcome assigned to a terminal node z. We say T and T' (or $\mathcal{G}$ and $\mathcal{G}'$) have *similarly distinct outcomes* if there exists a bijective map $o : O \to O'$ such that $\Omega(z) = o(\Omega(f(z)))$ for all terminal nodes $z \in T$ $(\in \tau(\mathcal{G}))$.

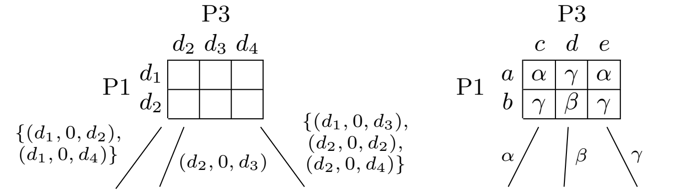
Confirming that players have the same kinds of decisions along the way is more involved, at least formally. We want single-player nodes to still be single-player nodes with the same number of choices, and multiplayer nodes to still be multiplayer nodes with the same interaction between each player's simultaneous decisions. In essence, we want the same strategic-form game to be played at each internal state node, as described in Sec. III. First let us define the decision matrix, which describes these strategic ("matrix") form games:

Definition 8 (Decision matrix) Let $\mathcal{G}$ be a game system with players $\mathcal{P} = (p_1, \ldots, p_n)$. Let w be a non-terminal state node in a game tree $T \in \tau(\mathcal{G})$ with assigned state s and outgoing decision edges E. Each player p has a set of legal choices $\ell_p(w)$ they can select to influence the edge followed. Let $\ell_p^0(w) = \ell_p(w)$ unless $\ell_p(w)$ is empty, in which case $\ell_p^0(w) = \{0\}$, with 0 the null choice. The *decision matrix* at node w is a map $D_w : \ell_{p_1}^0(w) \times \cdots \times \ell_{p_n}^0(w) \to E$ of decision tuples to edges.

A game tree produced freshly from Alg. 2 has $\ell_p(w) = L_p(s)$, the usual legal set (see Def. 1), but game tree reductions or transformations may adjust ℓ (e.g., see Defs. 13 and 15).

In a tree with $\ell_p(w) = L_p(s)$ and each edge $e \in E$ labeled with a set of one or more unique decision tuples, D_w simply maps each decision tuple $(d_{p_1}, \ldots, d_{p_n}) \in \mathcal{D}_0^n$ to the edge with that tuple. An edge might obtain multiple tuples, even though Alg. 2 only assigns one to each edge, due to something like a symmetry-redundant subtree reduction (see Def. 14).

For example, Fig. 4a is a sample decision matrix with outgoing edges as it might appear in a reduced game tree, for a 3-player game with $\mathcal{P} = (\mathrm{P1}, \mathrm{P2}, \mathrm{P3})$. The alternative labeling Fig. 4b helps clarify the structure of the joint decisions. Omitting the inactive P2, who has no legal choices:



P3
d_2 d_3 d_4
P1
d_1
d_2
$\{(d_1, 0, d_2), (d_1, 0, d_4)\}$
$(d_2, 0, d_3)$
$\{(d_1, 0, d_3), (d_2, 0, d_2), (d_2, 0, d_4)\}$
P3
c d e
P1
a α γ α
b γ β γ
α
β
γ

Fig. 4. Left: An example decision matrix in a reduced game tree (e.g., after reduction by Def. 13 or Def. 14). Right: The same matrix, relabeled.

We say that two decision matrices match (in the context of a game tree) if there is a self-consistent way to relabel the players, decisions, and edges such that the relabeled players making the relabeled decisions lead to the relabeled edges:

Definition 9 (Matching decision matrices) Take two game trees T and T' that are structurally equivalent with correspondence f. Let $w \in T$ and $w' = f(w) \in T'$ be corresponding non-terminal state nodes, with states s, s', decision matrices $D_w, D_{w'}$, and sets of players $\mathcal{P}, \mathcal{P}'$, respectively.

If possible, define a bijective *player correspondence* $\pi : \mathcal{P} \to \mathcal{P}'$, with associated bijective maps $\lambda_{\pi,p} : \ell_p^0(w) \to \ell_{\pi(p)}^0(w')$ $(p \in \mathcal{P})$, which correspond choices in the two games. Together, these furnish a unique map between the decision tuples, $\lambda_\pi : d_0^n \mapsto (d_0^n)'$, i.e., between the domains of D_w and $D_{w'}$.

The decision matrices D_w and $D_{w'}$ are said to *match*, denoted $D_w \sim D_{w'}$, if there exists at least one such π and set $\{\lambda_{\pi,p}\}$ such that the matrices map to corresponding edges: i.e., $D_w(d_0^n) = e \in T$ and $D_{w'}(\lambda_\pi(d_0^n)) = f(e) \in T'$ for all d_0^n in the domain of D_w.

See Fig. 5 for examples. Figs. 4a and 4b could also be said to match, since they only differ in choice labeling, if the node and edges were placed to correspond in two structurally equivalent trees. Note that we may relabel the decisions for each player separately (e.g., $d_2 \rightarrow b$ for P1 in Fig. 4, but $d_2 \rightarrow c$ for P3), and we may even relabel the null decision.

We can now generalize beyond individual decision matrices to game trees and systems (see Fig. 5). Whatever relabeling is necessary to make the decision matrices match, we demand at least that the player relabeling is the same everywhere. It is unimportant if the decision labels vary from matrix to matrix.

Definition 10 (Trees with matching decision matrices) Take two structurally equivalent game trees T, T' (or game systems $\mathcal{G}, \mathcal{G}'$) with structural correspondence $f : T \rightarrow T'$ and sets of players $\mathcal{P}$ and $\mathcal{P}'$. We say these trees (or systems) have *matching decision matrices* if there exists at least one bijective player correspondence $\pi : \mathcal{P} \rightarrow \mathcal{P}'$ such that all corresponding decision matrices match with respect to π—i.e., $D_w \sim D_{f(w)}$ for all internal state nodes $w \in T$ $(\in \mathcal{G})$ with π as the player correspondence. (See Def. 9. The associated decision mappings $\{\lambda_{\pi,p}\}$ may be different for each node in the tree.)

Finally, let us put all of this together to give a broadly useful sense of equivalence between game trees, which respects everything about them except for the specific labels chosen to represent players, states, decisions, and outcomes:

Definition 11 (Equivalence up to relabeling) We say that two game trees T, T' are *equivalent up to relabeling* (or that game systems $\mathcal{G}, \mathcal{G}'$ are *game tree equivalent up to relabeling*) if T and T' (or $\mathcal{G}$ and $\mathcal{G}'$) are structurally equivalent and have matching probabilities, matching decision matrices, and similarly distinct outcomes, all with respect to the same structural correspondence f.

If all labels additionally happen to be identical, then the game trees (or game systems) are simply *equivalent* (or *game tree equivalent*). If only some labels are additionally identical, we might say (using outcomes as an example) that two trees t and t' are equivalent up to relabeling and with identical outcomes. This means that if t and t' have structural correspondence $f : t \rightarrow t'$ and $\Omega(z)$ gives the outcome assignment of terminal node $z \in t$, then $\Omega(z) = \Omega(f(z))$ for all z.

It is worth noting that Def. 11, in not distinguishing between the content of outcome labels, does not distinguish whether an outcome might be good or bad for a player. The normal and *misère* versions of a game have opposite win/lose conditions, for instance, but would be considered equivalent up to relabeling.

B. Game Tree Reductions

To establish agency equivalence from Def. 3, we need to prune those differences between trees that are not meaningful from the standpoint of player agency. Here we describe the relevant transformations to reduce bookkeeping subtrees, single-player subtrees, symmetry-redundant subtrees, and decision matrix redundancies, as heuristically described in Sec. IV.

Definition 12 A *bookkeeping subtree* is a subtree of a game tree rooted at a state node r and with state nodes as leaves, which has exactly one decision edge proceeding out of r and each interior state node. Players cannot influence play in this subtree. Such a subtree may be reduced as follows (see Fig. 6):

Case 1: If there are no chance nodes in the subtree:

1) There is only a single leaf, with state s. Replace the entire subtree with a state node with state s.

Case 2: If there are chance nodes in the subtree:

1) Let G be the set of all paths from r to the subtree leaves. Let $l(g)$ be the final state node in each $g \in G$ (i.e., the leaves).
2) Assign each path g a probability $\mathfrak{p}(g)$ given by the product of the probabilities on the chance edges in g.
3) Then, if the parent r' of the subtree root r is …
   a) *Case 2a:* … a state node: Replace r with a new chance node c.
   b) *Case 2b:* … a chance node c: proceed to step 4. (Note r has an incoming chance edge with probability $\mathfrak{p}_r$.)
   c) *Case 2c:* … nonexistent (r is the root of the game tree): Replace the child of r with a new chance node c.
4) Taking the chance node c from step 3, delete all nodes and edges between c and the leaves, non-inclusive.
5) Draw new chance edges between c and each leaf $l(g)$, labeled by the corresponding probabilities $\mathfrak{p}(g)$, or $\mathfrak{p}_r \cdot \mathfrak{p}(g)$ in Case 2b.

Definition 13 A *single-player deterministic subtree* is a subtree of a game tree rooted at a state node r and with state nodes as leaves, without any chance nodes, in which all nodes belong to a single player (except perhaps the leaves). Only that player has any meaningful decisions in this subtree, and they could just as well be made all at once. Such a subtree may be reduced as follows (see Fig. 7):

1) Let G be the set of all paths from r to its leaves. Let $l(g)$ be the final state node in each $g \in G$ (i.e., the leaves).
2) Delete all nodes and edges between r and the leaves, non-inclusive.
3) Draw a new decision edge between r and each leaf $l(g)$, labeled by the sequence of decision tuples in g.

This reduction also changes the domain of the decision matrix D_r at r (see Def. 8): The legal choices $\ell_p(r)$ available to the single player p at r are now the set of decision tuple sequences from each $g \in G$ (i.e., the labels on the new decision edges proceeding from r), *not* the canonical legal set L_p.

Definition 14 A *symmetry-redundant subtree* is a subtree t of a game tree rooted at a state or chance node r and extending to all its descendants, that is equivalent up to relabeling, and with identical players and outcomes, to a subtree t' rooted at a sibling node r' (and extending to all of its descendants). (A lone terminal node z is also considered a symmetry-redundant subtree if its outcome $\Omega(z)$ is identical to one of its siblings.) The symmetry-redundant subtree t may be reduced as follows:

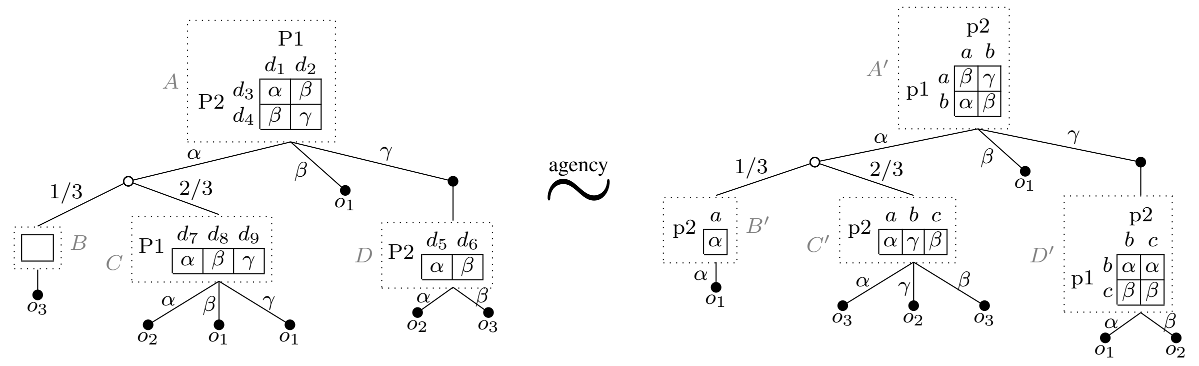


Fig. 5. These are two full game trees, with state labels suppressed, outcomes o_i on each terminal node, probabilities (1/3 and 2/3) on chance edges, and decision edges labeled with decision matrix outcomes instead of sets of decision tuples, for brevity (e.g., as Fig. 4b relabels Fig. 4a). Decision matrices are illustrated on all non-terminal state nodes, with inactive players not shown (like in Fig. 4a). Some (but not all) of the decision matrices match: $A \sim A'$, $B \not\sim B'$, $C \sim C'$, and $D \not\sim D'$. However, all of them match after reduction by Def. 15: in particular, $B \overset{\text{red.}}{\sim} B'$ and $D \overset{\text{red.}}{\sim} D'$ (the blank matrix B is a decision matrix with empty domain, see Def. 15). In fact, B (D) is the reduced form of B' (D'), after some relabeling. Thus, after these reductions, the *trees have matching decision matrices* (Def. 10) under the player correspondence P1 $\leftrightarrow$ p2 and P2 $\leftrightarrow$ p1. In fact, because the probabilities also match and the outcomes are similarly distinct, the two trees are agency equivalent (Def. 3).

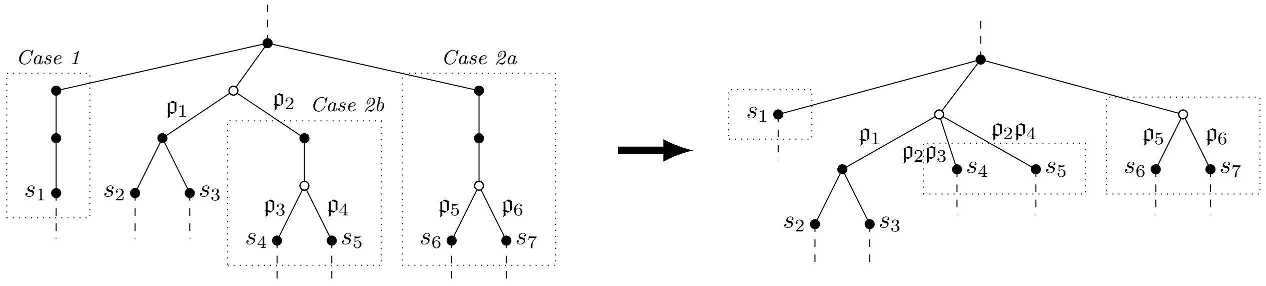

Case 1
Case 2a
Case 2b
$\mathfrak{p}_1$ $\mathfrak{p}_2$ $\mathfrak{p}_3$ $\mathfrak{p}_4$ $\mathfrak{p}_5$ $\mathfrak{p}_6$
s_1 s_2 s_3 s_4 s_5 s_6 s_7
$\mathfrak{p}_2\mathfrak{p}_3$
$\mathfrak{p}_2\mathfrak{p}_4$

Fig. 6. Bookkeeping subtree reduction example. Dotted lines highlight the bookkeeping subtrees before and after being reduced, exemplifying the different cases in Def. 12. Solid dots are state nodes, circles are chance nodes, and chance edges are labeled with probabilities $\mathfrak{p}_i$. The labels on most state nodes and all decision edges have been omitted. Dashed lines connect to other parts of the game tree.

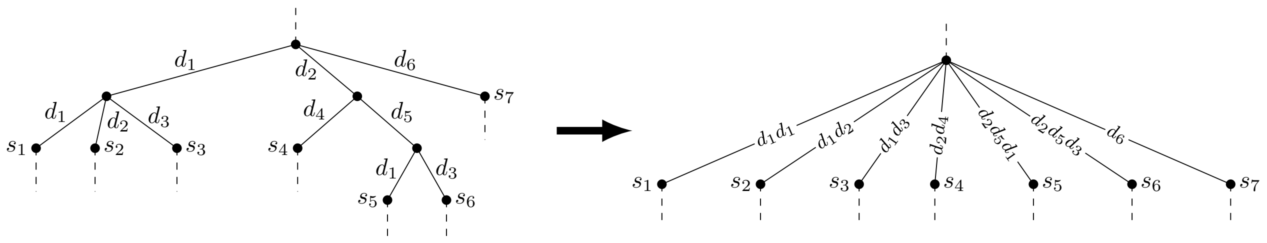

d_1 d_2 d_3 d_4 d_5 d_6
s_1 s_2 s_3 s_4 s_5 s_6 s_7
d_1d_1 d_1d_2 d_1d_3 d_2d_4 $d_2d_5d_1$ $d_2d_5d_3$ d_6

Fig. 7. Single-player subtree reduction example, illustrating Def. 13. All state nodes, except possibly the leaves, belong to the same player. Internal state node labels have been omitted. Edges have been labeled only with this player's decisions, for brevity, since all other players take the null decision everywhere. Dashed lines connect to other parts of the game tree.

1) Let e, e' be the edges ingoing to r, r'. If e and e' are …
   a) *Case 1:* …decision edges with assigned tuple sets $d(e)$ and $d(e')$ (if only one tuple, consider it a set of size 1). Replace the tuple set on e' with $d(e') \cup d(e)$.
   b) *Case 2:* …chance edges with assigned probabilities $\mathfrak{p}(e)$ and $\mathfrak{p}(e')$: Replace the probability on e' with $\mathfrak{p}(e')+\mathfrak{p}(e)$.
2) Delete the entire subtree t rooted at r, and the edge e.
3) (For *Case 2:*) If e' now has assigned probability 1, this chance edge is superfluous, as is the chance node parent c' of r'. Delete e', and replace c' by moving r' (with subtree attached) into its place.

Note symmetry-redundant subtrees may commonly occur when the sibling nodes r and r' are assigned the same state, e.g., when two different decisions lead to the exact same game state.

Several decision tuples may end up on a single edge (e.g., Fig. 4a) because of symmetry-redundant subtree reductions, which may lead to redundancies in decision matrices—if they were not redundant already. Consider Fig. 4b, for instance, which relabels Fig. 4a. It is clear that P3 gains no additional agency by having choice e available in addition to choice c. We can eliminate such meaningless redundancies in decision matrices by removing duplicate rows and columns:

Definition 15 A *decision matrix redundancy* occurs when a decision matrix $D_w : \ell^0_{p_1} \times \cdots \times \ell^0_{p_n} \to E$ contains more than one choice for some player p which lead to the same result. That is, when there exist distinct choices $a, b \in \ell_{p_i}$ for some p_i such that $D_w(d_1, \ldots, d_{i-1}, a, d_{i+1}, \ldots, d_k) = D_w(d_1, \ldots, d_{i-1}, b, d_{i+1}, \ldots, d_k)$ for all possible choices $d_j \in \ell_{p_j}$. The choices a and b are redundant.

To eliminate redundancies and unnecessary bookkeeping distinctions, a *reduced decision matrix* can be produced as follows (see Fig. 5 for examples):

1) If there exist two redundant choices $a, b \in \ell_{p_i}$ for some p_i, delete one: $\ell_{p_i} \to \ell_{p_i} \setminus \{b\}$. Repeat until no redundancies remain for any player.
2) If all players have only a single (possibly null) choice remaining ($|\ell_p^0| = 1$), there must only be a single edge e in the image of D_w. We may define $D_w : \varnothing \to \{e\}$.

Any two corresponding decision matrices with empty domains are said to match in the sense of Def. 9. Step 2 is not strictly necessary for establishing equivalence, but reflects that in a bookkeeping subtree, it does not matter which player(s) are given the task of executing the bookkeeping.

V. Discussion: Towards Game Similarity

We have proposed a grammar-like formalism to describe finite discrete game systems without hidden information, along with equivalence relations on this space of games, that are insensitive to cosmetic variations in game rules. Developing measures of game equivalence and similarity will be important for formally interrogating the design of games, and as a steppingstone towards predicting player behavior from design patterns. More broadly, we hope that such efforts may help connect game design and mathematical experts, enriching the many applications of games by exploring formal analogues to the rich tools and vocabulary used in game design today.

However, for complex games it may be impractical to check equivalence by drawing, reducing, and comparing full game trees. One possible way forward is to learn to transform and compare the grammars directly, using Defs. 3 and 11 as guidance for what those transformations must accomplish.

We could also move beyond game equivalence, to consider game similarity. Suppose a ludologer supplies a mapping ψ between the state spaces $\mathbb{S}, \mathbb{S}'$ (and perhaps also players and outcomes) of two game systems $\mathcal{G}, \mathcal{G}'$. Then the systems could be compared by sampling states $s \in \mathbb{S}$, computing some function (e.g., a partial game tree) at s and $\psi(s) \in \mathbb{S}'$, comparing the function values (e.g., assigning 1 if the partial trees are equivalent after appropriate reductions, 0 otherwise), and averaging the results. This would give a quantitative measure of game similarity (e.g., between 0 and 1), and another way to check equivalence (e.g., if similarity $= 1$). Even if only a fraction of the states are randomly sampled, a confidence interval could be estimated for the computed similarity.

There are several details to work out here in the calculation, interpretation, and likelihood estimates of such a similarity measure, which we leave for future work. For instance: in contrast to comparing full game trees, this similarity method could sample many states not legally accessible in standard play, so it would compare game rules beyond just legal gameplay. Also, since state spaces for complex games can be gargantuan, it is unclear how quickly a similarity estimate would converge. Nevertheless, when combined with the intuitive and technical guidance of the equivalence relations Def. 3 and Def. 11, such a sampling method has the potential to practically measure distances between games, without sensitivity to cosmetic variations in the rule descriptions, and with minimal input from ludologers. We look forward to exploring this and other game similarity measures in future work.

Acknowledgments

We thank Stephen Crane, Marquita Ellis, Ryan Janish, Will Johnson, Kiran Lakkaraju, Kweku Opoku-Agyemang, Stephen Phillips, and Ben Wormleighton for useful discussions. We also thank Vlaada Chvátil for designing *Mage Knight* [24], the board game that inspired this research. This work is supported in part by the Office of Naval Research under the Embedded Humans MURI (N00014-13-1-0341) as well as a Philippine-California Advanced Research Institutes (PCARI) grant.

References

[1] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018.
[2] E. Melcer, T.-H. D. Nguyen, Z. Chen, A. Canossa, M. S. El-Nasr, and K. Isbister, "Games research today: Analyzing the academic landscape 2000-2014," in *Proc. 10th Int. Conf. Foundations of Digital Games*, 2015.
[3] E. Melcer and K. Isbister, "Toward understanding disciplinary divides within games research," in *Proc. Int. Conf. Foundations of Digital Games*. ACM Press, 2017.
[4] K. S. Tekinbaş and E. Zimmerman, *Rules of play: game design fundamentals*. MIT Press, 2003.
[5] R. Koster, *A theory of fun for game design*. Paraglyph Press, 2005.
[6] E. Adams and J. Dormans, *Game mechanics: advanced game design*. New Riders, 2012.
[7] J. Schell, *The art of game design: a book of lenses*, 3rd ed. Taylor & Francis, 2019.
[8] G. Engelstein and I. Shalev, *Building blocks of tabletop game design: an encyclopedia of mechanisms*. Taylor & Francis, 2019.
[9] P. Riggins and D. McPherson, "Tools for mathematical ludology," 2019, arXiv:1912.03295 [cs.AI].
[10] B. Cousins, "Elementary game design," *Develop Magazine*, pp. 51–54, 2004.
[11] R. Koster, "A grammar of gameplay," 2005, Game Developers Conf.
[12] B. Stéphane. (2006) A game grammar. [Online]. Available: http://www.stephanebura.com/diagrams/
[13] C. Browne, "Modern techniques for ancient games," in *2018 IEEE Conf. Computational Intelligence and Games (CIG)*. IEEE, 2018.
[14] R. Cooper, D. V. DeJong, R. Forsythe, and T. W. Ross, "Communication in coordination games," *Q. J. Econ.*, vol. 107, no. 2, pp. 739–771, 1992.
[15] P. J. Hammond, "Beyond normal form invariance: First mover advantage in two-stage games with or without predictable cheap talk," in *Rational Choice and Social Welfare*. Springer, 2008, pp. 215–233.
[16] J. Beck, *Combinatorial games: tic-tac-toe theory*. Cambridge University Press, 2008.
[17] E. Rasmusen, *Games and information: an introduction to game theory*, 4th ed. Blackwell Pub, 2007.
[18] É. Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne, "Ludii - the ludemic general game system," 2019, arXiv:1905.05013 [cs.AI].
[19] M. Thielscher, "A general game description language for incomplete information games," in *Proc. 24th AAAI Conf. Artificial Intelligence*, ser. AAAI'10. AAAI Press, 2010, pp. 994–999.
[20] N. Love, T. Hinrichs, and M. Genesereth, "General game playing: Game description language specification," Stanford Logic Group, Computer Science Department, Tech. Rep. LG-2006-01, 2006.
[21] M. Thielscher, "GDL-III: A description language for epistemic general game playing," in *Proc. 26th Int. Joint Conf. Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, pp. 1276–1282.
[22] J. Kowalski, M. Mika, J. Sutowicz, and M. Szykuła, "Regular boardgames," *Proc. AAAI Conf. Artificial Intelligence*, vol. 33, pp. 1699–1706, 2019.
[23] G. Bonanno, "Set-theoretic equivalence of extensive-form games," *International Journal of Game Theory*, vol. 20, no. 4, pp. 429–447, 1992.
[24] V. Chvátil, *Mage Knight*. WizKids, 2011, board game.