Evolutionary Game Design

Cameron Browne and Frederic Maire

***Abstract*—It is easy to create new combinatorial games but more difficult to predict those that will interest human players. We examine the concept of game quality, its automated measurement through self-play simulations, and its use in the evolutionary search for new high quality games. A general game system called Ludi is described and experiments conducted to test its ability to synthesise and evaluate new games. Results demonstrate the validity of the approach through the automated creation of novel, interesting and publishable games.**

***Index Terms*—Artificial intelligence (AI), combinatorial game, aesthetics, game design, evolutionary search.**

I. Introduction

WHILE games research has led to many important artificial intelligence (AI) breakthroughs over the last few decades, these have generally come through the study of classics such as Chess, Go and Checkers, and used as their yardstick for success the strength of the artificial player. Little attention has been paid to measuring the quality of the games themselves or asking such questions as:

- What makes a game interesting to play?
- Can we tell if a game is likely to become a classic?

The emergence of the games industry as a commercial phenomenon makes these questions increasingly important, as record numbers of designers produce record numbers of games each year, but it can take years of play testing and commercial enquiry to determine whether a game is likely to succeed.

A tool for automatically measuring the quality of a given game could be of significant benefit to both game designers and the industry. It could reduce development time by quickly detecting flaws, and reduce the need for extensive play-testing which can require designers to prematurely reveal their prototypes to external sources. Further, such a tool could direct the automated search for new rule combinations, with a view to suggesting interesting avenues for designers to pursue or even to producing complete new games of meaningful quality.

Pell [1] states:

> If we could develop a program which, upon consideration of a particular game, declared the game to be uninteresting, this would seem to be a true sign of intelligence! So when this becomes an issue, we will know that the field has certainly matured.

With this in mind, we describe a framework called Ludi for the measurement and synthesis of combinatorial games, and three experiments designed to test the system and the validity of the following hypotheses:

I. *That there exist fundamental (and measurable) indicators of game quality.*

II. *That these fundamental indicators may be harnessed for the directed search for new high quality games.*

II. Defining Games

Of the many ways to define a game, Salen & Zimmerman make the following useful observation: *A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome* [2]. This definition was condensed from the findings of many prior studies, most of which identified the following key elements:

- rules,
- play, and
- outcome.

A game may therefore be expressed in terms of its *means*, *play* and *ends*. These three aspects are central to the design of Ludi and its underlying model of game analysis, and will provide a recurring theme throughout this paper.

A. Combinatorial Games

We focus on *combinatorial games*, which are:

- *Finite*: produce a well-defined outcome.
- *Discrete*: turn-based.
- *Deterministic*: chance plays no part.
- *Perfect information*: no hidden information.
- *Two-player*.

The two-player requirement is debatable as solitaire puzzles may constitute combinatorial games, in the sense that the puzzle solver competes against the null player and indirectly the designer who set the challenge. Multiplayer games with three or more players fall outside the scope of combinatorial play due to the social aspect of coalitions that may arise.

Manuscript received September 9, 2009. This work was supported in part by an Australian Postgraduate Award and a small grant from a Games Research scheme conducted by Microsoft Research Asia.

Cameron Browne is with the Computational Creativity Group, Department of Computing, Imperial College London, 180 Queens Gate, London, SW7 2AZ, UK (e-mail: cameron.browne@imperial.ac.uk).

Frederic Maire is with the Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane, QLD, 4000, Australia (e-mail: f.maire@qut.edu.au).

The term *game* shall henceforth refer to a two-player combinatorial game throughout this paper. Such games are an ideal test bed for the experiments as they are typically deep but described by simple, well defined rule sets.

Note that this is not a work in *combinatorial game theory* (CGT), which is concerned with the analysis of games with a view to solving them or at least finding optimal strategies [3] and developing artificial players able to challenge human experts. Within the context of this study, the artificial player is of little interest except as a means for providing self-play simulations. While it must be of sufficient strength to provide meaningful playouts, we are concerned primarily with the quality of the game itself rather than the quality of the player.

B. Ludemes

Just as a *meme* is a unit of information that replicates from one person to another [4], a *ludeme* is a game meme or unit of game information. First coined by Borvo [5], this term describes a fundamental unit of play often equivalent to a rule; ludemes are the conceptual equivalent of a game's components – both material and non-material – and are notable for their ability to pass from one game or game class to another [6].

Ludemes may be single units of information, such as the following items that describe aspects of the game board shown in Fig. 1(a):

```
(tiling square)

(size 3 3)
```

Conceptually related items may be encapsulated to form higher level compound ludemes as follows:

```
(board
  (tiling square)
  (size 3 3)
)
```

Collecting rules into such compound ludemes is a convenient way to describe games. For example, the essence of Tic-Tac-Toe may be succinctly described as follows (assuming a two-player combinatorial model):

```
(game Tic-Tac-Toe
  (board
    (tiling square)
    (size 3 3)
  )
  (win (in-a-row 3))
)
```

The concept of an entire game as an item of information may seem odd but it is valid; there exist many examples of identical games being discovered, fully formed, at similar times. The most famous case is the independent discovery of Hex by mathematicians Piet Hein and John Nash in the 1940s [7]. A more recent example is Chameleon, discovered by New Zealand and USA designers within a week of each other in 2003. Such cases may be examples of “memetic convergence” in action towards optimal designs.

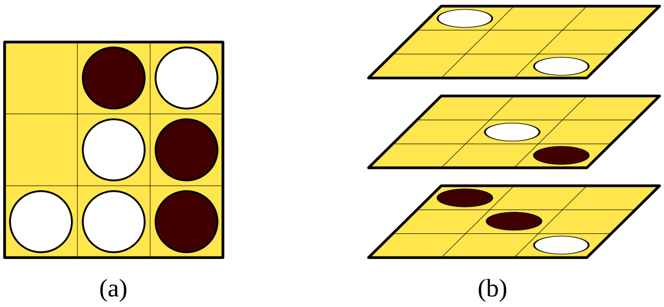


Fig. 1. Games of: (a) Tic-Tac-Toe and (b) Tic-Tac-Toe (3D) won by White.

C. Recombination Games

Given a game in its ludemic form, it is a simple matter to manipulate its rules to create variants and new games. For Tic-Tac-Toe, such modifications might include the board size:

```
(size 2 2)
```

or the target line length:

```
(win (in-a-row 2))
```

However, a moment's reflection will reveal that each of these changes break the game, by making it unwinnable in the first case and trivially winnable in the second.

Other manipulations might involve extending the board to three dimensions, as shown in Fig. 1(b):

```
(size 3 3 3)
```

or inverting the end condition to give a *misere* version:

```
(lose (in-a-row 3))
```

These variants are both more interesting but still trivially solvable, and are more notable for their novelty value than any inherent value as games. There is much room for improvement in this branch of the *N*-in-a-row family.

The difficulty of deriving an interesting game from Tic-Tac-Toe does not just stem from the fact that it is itself flawed (it is drawish if played correctly). There is the serious problem that rule sets for combinatorial games tend to be highly optimised and fragile; authors strive for the simplest rule sets that give the deepest playing experience, and the slightest change will generally break a game. As in most creative fields, it is easy to generate artificial content but much more difficult to generate artificial content of human expert quality.

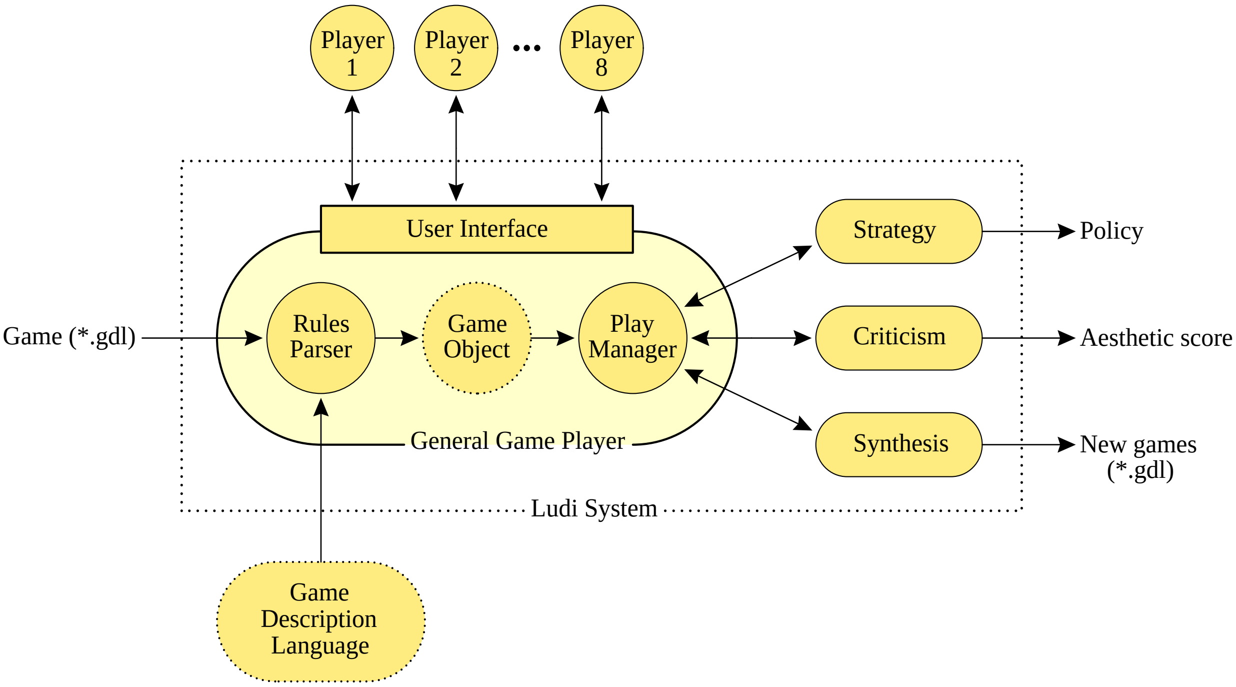


Fig. 2. Framework of the Ludi system.

Given that the rule sets of most existing games are highly optimised – certainly the well known ones – it is unlikely that such simple manipulations of a game's degrees of freedom will produce a better game in isolation. The designer would usually have tested such obvious variants and discarded them as inferior. Instead, a more promising approach is to recombine the game's rules with rules from other games and look for the *emergence* [8] of interesting, new rule combinations not previously considered. The idea that there pre-exist a multitude of games in the form of optimal rule combinations waiting to be discovered resonates strongly with the Platonist view of mathematics [9]. The question then becomes how to search this potentially huge design space effectively.

D. *Game Distance*

It can be useful to measure the distance between existing games and a newly derived rule set, in order to determine whether it constitutes a duplicate, variant, or completely new game.

The distinction between a variant and a new game is subtle, but may be achieved by representing both games as rule trees (based on their ludemic descriptions introduced above) and accumulating the total weighted difference between these two trees. Differences between rule parameters are weighted lightly whereas structural differences between the rules themselves are weighted more heavily, in inverse proportion to their depth of nesting; higher level rules generally have wider applicability and are therefore generally more important. If the total difference between the two rule sets exceeds a certain threshold value then the two games are considered to be distinct.

E. *General Game Players*

Given the possibility of creating a large number of rule sets, it would be desirable to test them automatically through self-play. General game players (GGPs) – software systems for playing a range of games well rather than any one particular game expertly – are ideal for this purpose.

GGPs were first proposed several decades ago [10] but have recently enjoyed a resurgence of interest as researchers come to realise their potential value to the gaming and broader AI communities. This includes GGP competitions run over recent years in conjunction with international AI conferences [11].

F. *Game Description Languages*

Central to any GGP is the game description language (GDL) that defines the scope of games understood by the system. There is a delicate balance between defining a GDL that is powerful and extensible enough to encompass a wide range of known and not-yet-known games, yet also efficient, elegant and comprehensible to human authors.

The most widely used GDL is probably the commercially available Zillions of Games ZRF rule language [12]. ZRF authors define games in a Lisp-like syntax using predefined keywords, and may programmatically create complex rule structures through macros. More recently, the Stanford GDL used for the AAAI GGP competitions [13] is a lower level language that defines games using first order logic.

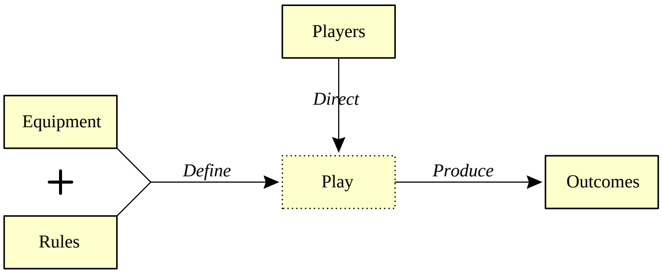


Fig. 3. The basic game model.

III. The Ludi System

Ludi is a system for playing, measuring and synthesising games within the scope of its GDL (Fig. 2). The main components of the system are:

- *GDL*: defines the scope of games.
- *GGP*: interprets games and coordinates play.
- *Strategy module*: informs move planning.
- *Criticism module*: measures game quality.
- *Synthesis module*: generates new games.

A. Ludi GDL

The Ludi GDL is a high level game description language based on the ludemic understanding of games outlined in Section II. It is structured to follow the basic *means-play-ends* model of games, extended to include the relationship between the game and its players (Fig. 3).

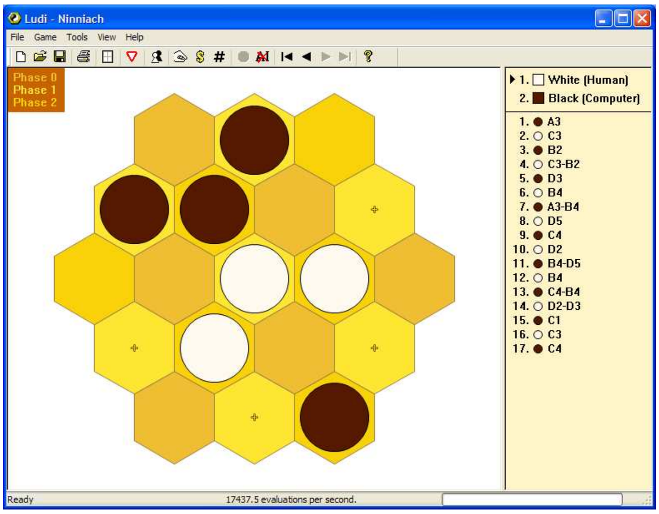
The Ludi GDL was devised with Kernighan and Pike's principles of good software design [14] in mind:

- simplicity,
- clarity,
- generality, and
- automation.

It is a higher level language than the Stanford GDL and Zillions ZRF, and although concise and conducive to human authoring and machine manipulation it lacks the universal generality of the Stanford GDL in particular. However, its hierarchical and well-defined nature makes it ideal for the intended experiments, as it is much more likely that a structured tree-based language will evolve sensible rule sets than an unstructured logic-based one. The Ludi GDL proved sufficiently rich for this intended purpose that its somewhat limited scope was not an issue.

The following example conveys the essence of the language:

```
(game Tic-Tac-Toe
  (players White Black)
  (board
    (tiling square i-nbors)
    (size 3 3)
  )
  (end (All win (in-a-row 3)))
)
```



```
▸ 1. □ White (Human)
  2. ■ Black (Computer)

 1. ● A3
 2. ○ C3
 3. ● B2
 4. ○ C3-B2
 5. ● D3
 6. ○ B4
 7. ● A3-B4
 8. ○ D5
 9. ● C4
10. ○ D2
11. ● B4-D5
12. ○ B4
13. ● C4-B4
14. ○ D2-D3
15. ● C1
16. ○ C3
17. ● C4

Ready    17437.5 evaluations per second.
```

Fig. 4. The Ludi user interface.

This game (Tic-Tac-Toe) is played between White and Black on a 3x3 square grid with orthogonal and diagonal adjacency, and is won by the player to make a line of three pieces of their colour (if any). Unless otherwise stated, it is assumed that players take turns placing a piece of their colour on an empty board cell each move.

Ludi GDL definitions closely correspond to a game's ludemic description, which is how a human designer would typically conceptualise the game. A more detailed description of the language is given in Appendix I and further examples of games defined in the GDL can be found in Appendix II.

B. Ludi GGP

The core of the Ludi system is its general game player, as shown in Fig. 2. The Ludi GGP is implemented in C++ and provides the following functionality:

- rules parser,
- game object,
- user interface, and
- play manager.

The rules parser loads and parses games defined in the Ludi GDL. If a definition is valid according to the grammar, then the corresponding ludeme tree is constructed and the single game object initialised. The game object maintains a record of the current board state and handles tasks such as the generation of legal moves and testing for terminal conditions.

The user interface (Fig. 4) presents games uniformly and anonymously so that quality judgments are made on the merits of the games themselves rather than their visual attractiveness. The interface provides a plain English translation of the current rule set and a tutorial mode to help players understand new games. In tutorial mode, legal placements are marked '+' and legal destination cells for movable pieces are similarly marked '+' when those pieces are clicked on.

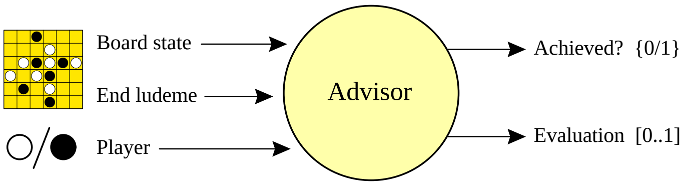


Fig. 5. Advisor model.

The play manager coordinates play for one to eight human and artificial players, although only two are used for this study. This includes move scheduling, input handling, *ko* repetition testing to avoid infinite loops, all players passing, and so on.

Moves for artificial players are planned using standard alpha beta adversarial search with move ordering, beam width reduction and iterative deepening [16]. Estimated values for non-terminal board positions are provided by the Strategy module, as follows.

C. Strategy Module

The Strategy module provides value estimates for given board positions relative to each player. This is achieved through a set of advisors working within certain policies.

Advisors

Advisors are evaluation functions that represent some narrow but rational view of the board position [17], and express whether a player's position is favorable or unfavorable within this perspective [18].

Ludi defines 20 such advisors for aspects such as mobility, proximity to goal, attacking potential, connective potential, and so on. Each advisor takes as input a board state, end condition and player colour (Fig. 5) and has the functionality to return both an estimate of that player's positional value and whether the end condition has been achieved, as required.

Policies

The contributions from each advisor are combined using a weighted linear function [15] to provide an overall value $E(s)$ for board state s:

$$E(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s) \qquad (1)$$

where w is a vector of weights and { $f_1(s)$, … , $f_n(s)$} the set of advisor functions. The weight vector w constitutes a *policy* that describes the relative importance of each advisor for that game.

Policy Optimisation

For efficiency, it is important that only those advisors relevant to the game have non-zero weight. The system is able to derive a default policy for each game based upon its rules and to optimise that policy through self-play using two-membered evolutionary search (1+1)-ES [19].

Policy optimisation threw up some surprising emergent strategies, such as a small negative weighting on stack height to disincline repetitive cycles in stacking games, which was later incorporated into default policies where appropriate. A null "fight or flight" policy is used for any player with no specified end conditions, in which pieces are moved towards enemy pieces to encourage engagement while maximizing their movement potential to allow escape if necessary.

Although a high level of play cannot be claimed for all games, this advisor/policy approach proved sufficient for exercising rule sets defined in the GDL and providing meaningful self-play simulations. All players – human and artificial – are beginners at any newly created game.

IV. Game Evaluation

The Criticism module measures games for quality through self-play simulations, based on certain aesthetic criteria.

A. Game Quality

The term *quality* in this context refers to the likelihood that a given game will be of interest to human players. While players generally know whether they enjoy a game or not, few can articulate the reasons in concrete terms. Thompson [20] took a significant step towards formalising such concepts by defining four key attributes that a game should possess:

- *Depth*: games should hold lasting interest.
- *Clarity*: their mechanics should not be confusing.
- *Drama*: hope of recovery from bad positions.
- *Decisiveness*: end quickly once a winner is certain.

Thompson also points out that games may be viewed as sequences of logic puzzles that players pose to each other, and hence a good game should readily yield such puzzle positions. Key attributes defined by other authors include interestingness [21], uncertainty [22], interaction [23] and tension [24]. Originality in design is also of paramount importance.

Just as mathematicians strive for beautiful, aesthetically meaningful abstractions [25], we are concerned not with the sensual beauty of games but their intellectual appeal; the elegance of the rules, how well they complement each other, and the quality of the competition they produce. Contrary to Ellis [26], the absence of flaws is a precondition for beauty.

B. Aesthetic Model

Birkhoff [27] describes the evaluation of visual art using functions of certain *aesthetic criteria*, an approach later extended to a complete *algorithmic aesthetics* system for the design and criticism of aesthetically meaningful visual objects by Stiny and Gips [28]. Similar principles can be applied to the aesthetic measurement of games; in this case, the interpretation of an object (game) will be a set of aesthetic measurements derived through self-play, and the output of the algorithm will be a single predicted aesthetic value. Importantly, Stiny and Gips distinguish between *constructive* and *evocative* modes for the understanding of objects within

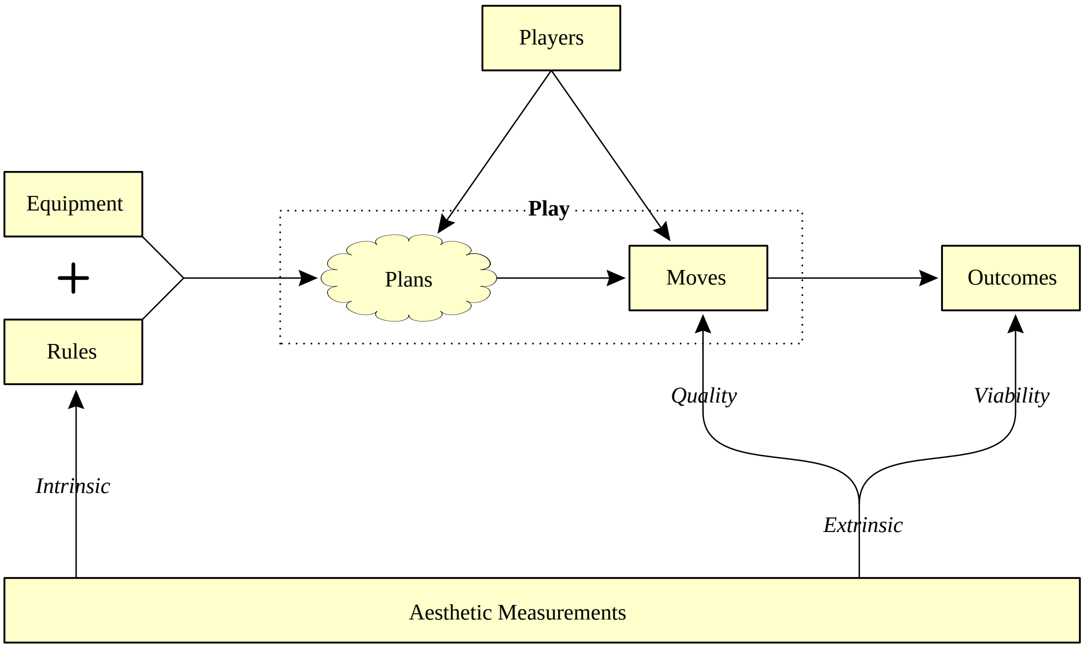


Fig. 6. Player-centric aesthetic model of games.

this system. In constructive mode objects are understood by the rules of their construction, and in evocative mode they are understood by the associations, ideas or emotions they evoke.

Figure 6 shows the player-centric aesthetic model of games devised for this study. It is based on the basic game model of Fig. 3 with the "play" component expanded to distinguish the players' strategic plans from their eventual moves. It is these plans that we would ideally like to measure as an indication of the players' engagement with the game, but this is of course impossible. However, we can measure the resulting moves that represent tangible realisations of those plans. As Perlis [29] states: "We measure our understanding (and control) by the extent to which we can mathematize an activity."

Aesthetic Criteria

The model is divided into *intrinsic* and *extrinsic* aspects, with the latter further divided into *quality* and *viability*. These define the three main categories of aesthetic criteria:

- *Intrinsic*: based on rules and equipment.
- *Viability*: based on game outcomes.
- *Quality*: based on trends in play.

Intrinsic criteria are the easiest to measure but are less useful for indicating game quality than they are for specifying personal feature preferences. Viability criteria, such as those proposed by Althöfer [21], are robust and useful for quickly detecting flaws in games. Quality criteria are the most subtle and difficult to measure, but are the key to estimating the potential worth of games that have proven viable.

Preferred Game Length

It is useful to define a preferred game length M_{pref} which is the move number at which the current game will ideally reach a natural conclusion. It might seem sensible to adjust this value on a game-by-game basis according to the estimated complexity of the game based on board size, piece count, branching factor and so on, but this has the undesirable effect of rewarding very short games on smaller boards and excessively long games on larger boards. Instead, we wish to reward deep, involved passages of play emerging from simpler rule sets and piece configurations; this follows more closely the spirit of combinatorial game design and will reduce the running time for the experiments while still allowing complex, elegant and (hopefully) interesting games to emerge.

We therefore standardised the preferred game length over all games to $M_{pref} = 60$ moves, based on the observation that two human opponents taking 30 seconds per move on average will complete such a game in 30 minutes on average, a figure that experience suggests is reasonable (note that the GGP plays much faster than this for most games).

C. *Aesthetic Measurement*

Aesthetic measurements are made for each game during a number of self-play trials *G*. The first few moves of each game are made randomly (but legally) to encourage a more thorough exploration of the move search space; random moves are not included in the aesthetic measurements. Self-play trials that exceed twice the preferred game length are abandoned as draws.

Intrinsic criteria are measured before the trials start and viability criteria measured by the outcome of each trial. Quality criteria measure trends in play during the trials through the use of *lead histories*, as shown in Figs. 7 and 8. The white and black dots indicate the relative positional strengths of both players following each move, while the thick lines represent the current difference between the eventual winner and loser.

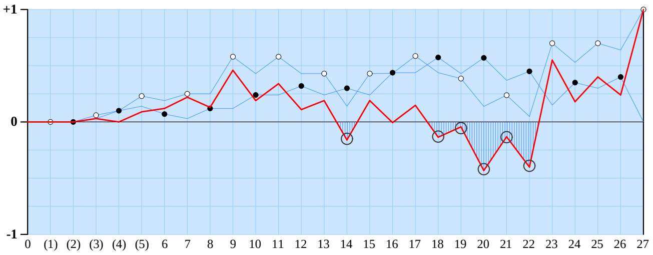
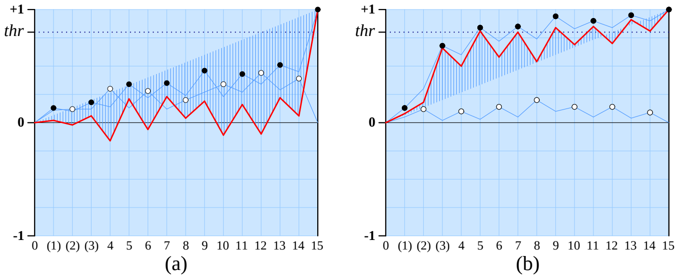


Fig. 7. Lead history of a dramatic game.



(a)

(b)

Fig. 8. Lead histories of (a) an uncertain game and (b) a certain game.

A total of 57 aesthetic criteria were implemented for Ludi, consisting of:

- 16 × intrinsic criteria.
- 11 × viability criteria, and
- 30 × quality criteria.

We do not claim that this is a universal or canonical set of features with which all games may be measured. Rather, we define a large number of measurements based on observation, experience and prior work, and experimentally determine which of these are most relevant for the task at hand.

As it is not possible to describe all 57 measurements within the space of this paper, we present a selection below by way of example. For complete descriptions of all criteria see [15].

Completion (Viability)

Games should produce more victories than draws. The completion criterion A_{comp} simply measures the total sum of games won by either player as a ratio of all games played G:

$$A_{comp} = wins / G \tag{2}$$

A low completion rate indicates a flawed game that is either drawish or tends to exceed the maximum move limit.

Duration (Viability)

Games should neither be too short nor too long. The duration criterion A_{durn} measures the average discrepancy between the total moves made per game M_g and the preferred game length M_{pref}, which is set to 60:

$$A_{durn} = 1 - \sum_{g=1}^{G} \frac{\left| M_{pref} - M_g \right|}{M_{pref}} / G \tag{3}$$

This measurement is useful for detecting pathological flaws in both directions, that is, trivial games that end within a few moves, and excessively long games that are difficult to conclude. Game length is also one of Althöfer's criteria [21].

Completion and duration are typical viability criteria: they are simple, robust, fast to measure, and good indicators of whether a game works at a fundamental level. We now outline two quality criteria for comparison.

Drama (Quality)

Players should have at least a hope of recovering from bad positions if they are to maintain a vested interest in a game. For example, Fig. 7 shows a game in which the eventual winner spends several moves in a negative (losing) position before recovering to win.

The drama criterion A_{drav} measures the degree to which the winner of each game suffers a negative lead:

$$A_{drav} = \sum_{g=1}^{G} \frac{\sum_{n=M_{gr}+1}^{M_g-1} E_w(m_n) < E_l(m_n) \begin{cases} \sqrt{E_l(m_n) - E_w(m_n)} \\ 0 \end{cases}}{\underset{[M_{gr}+1 \le n \le M_g - 1]}{count} \left(E_w(m_n) < E_l(m_n) \right)} / G \tag{4}$$

where $E_w(m_n)$ represents the board evaluation for the eventual winner at move n and $E_l(m_n)$ represents the board evaluation for the eventual loser at move n. This equation therefore measures the average number of moves that the eventual winner of each game spends in a negative position, and the severity of each such position.

Uncertainty (Quality)

The outcome of each game should remain as uncertain for as long as possible if all players are to maintain a vested interest in it. For example, Fig. 8(a) shows an uncertain game in which neither player develops a strong advantage until its conclusion, while Fig. 8(b) shows a certain game in which the eventual winner takes a strong early lead and keeps it; this will probably be a very unsatisfying game for the losing player.

The uncertainty criterion A_{uncl} measures the degree to which the estimated lead value for each move falls above or below the expected average:

$$A_{uncl} = \sum_{s=0}^{S-1} \left(\min \left(1, t - \left(\sum_{g=1}^{G} E_{co}(m_{tM_g}) \right) / G \right) \right) / S \tag{5}$$

The amount of uncertainty is indicated by the area enclosed by the lead history plot and an imaginary line drawn from (0, 0) to (M, 1). A number of samples S = 100 are made at regular intervals t across the completed game history, and the average distance measured between this interpolation line and $E_{co}(m_{tMg})$ the estimated lead value at time t. Samples that fall below the interpolation line indicate greater uncertainty.

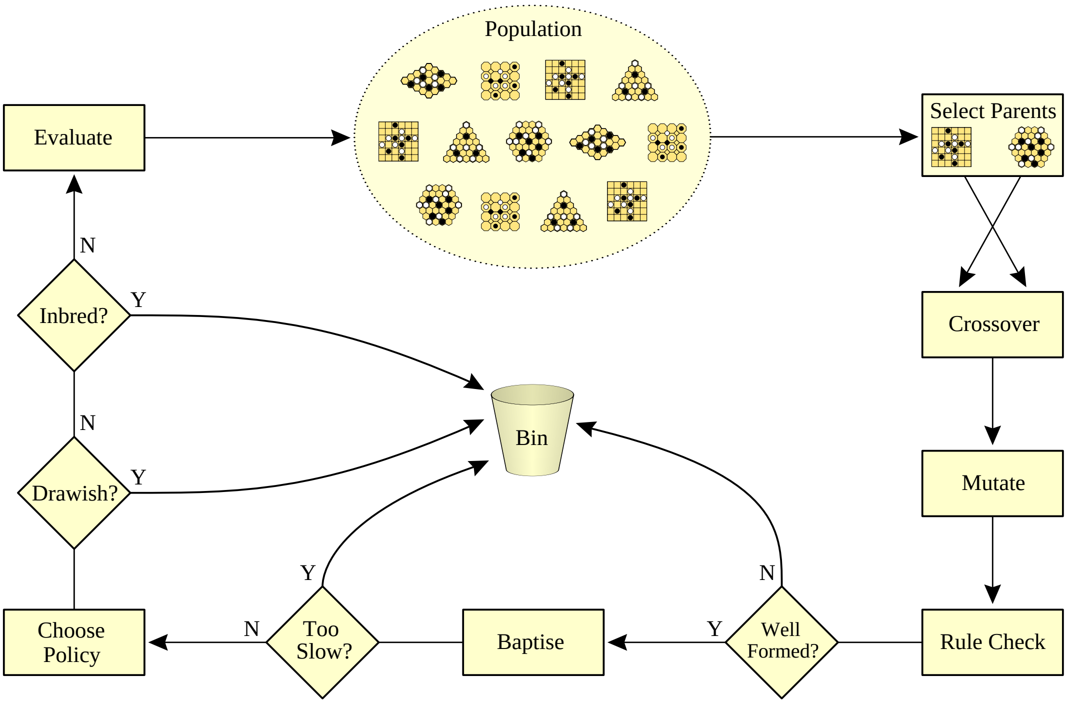


Fig. 9. The game life cycle.

It can be seen that quality criteria such as drama and uncertainty are more subtle, abstract and difficult to measure than the viability criteria; however, they are useful for estimating the quality of viable games.

D. *Criteria Efficiency*

Some of the criteria not discussed here are significantly slower to measure than others and require additional trial runs in different modes. These "slow" criteria include measurements related to puzzle detection, depth estimation and robustness to random, overly defensive and obstructive players. Full details of these modes are given in [15].

E. *Aesthetic Coefficients*

The weighted sum of all 57 criteria gives an estimated aesthetic score A_s, which is the output of the aesthetic measurement process for each game:

$$A_s = \left(\sum_{i=0}^{C-1} A_i w_i \right) + w_C \tag{6}$$

The aesthetic weightings *w* provide an aesthetic profile for each game, and were determined empirically in Experiment II (described shortly) which involved correlating aesthetic measurements for a number of games with human player rankings of those games. The bias term w_C improves the accuracy of the correlation. Criteria with a 0 weighting for a given game are not invoked for that game, for efficiency.

V. Game Synthesis

The Synthesis module creates new games through the evolution of rule sets using genetic programming methods.

Pell has previously demonstrated the automated generation of Chess-like games using a method of *constrained stochastic context-free generation*, in which the user specifies a number of parameters and games are created by making statistical choices at each decision point in the grammar according to these parameters [18]. Similarly, Rolle's Morphling allows the user to interactively experiment with rule variations of a game according to certain rules [30].

Games created by these systems fall within the *constructive mode* of understanding described by Stiny and Gips, as they are directly constructed according to certain rules [28]. On the other hand, we wish to create games using the *evocative mode* of understanding and search for those games that stimulate interest in human players without having to make assumptions about any rules necessary for their construction. For this purpose we chose an evolutionary approach to search the game design space, which balances the *exploration* of the design space with the *exploitation* of existing knowledge [31].

A. *Mating Games*

Ludi uses a standard evolutionary approach [32] to create games, except that all surviving offspring are returned to the population and never culled from generation to generation. Fitness determines order rather than survival, to encourage the emergence of novel rule combinations.

Evolution continues until a given time limit is reached or the desired number of new games created. The basic process is summarised in Fig. 9 and explained below. It is worth noting that the aim of this process is to produce a number of interesting individuals rather than a single optimal individual.

Population

The population is initialised with a number of known games in order to encourage well-formed offspring, as strictly random rule combinations are most unlikely to produce viable games. Members of the population each have a unique name and remain sorted by estimated aesthetic value at all times.

Parent Selection

For each iteration, two parents are selected from the population using *stochastic universal sampling* [33]. This method draws samples from the entire range but selects fitter individuals more often, encouraging genetic diversity while maintaining a reasonable standard of fitness.

Recombination

The ludeme trees of the two parents are then *crossed over* to produce a child game using standard genetic programming techniques [34]. One of the parents is chosen at random to act as a template [35] and elements crossed over from the second parent with 10% likelihood. Elements are only crossed over to elements with which they are compatible, giving a weak form of *strong typing* [36] that encourages the creation of well formed children. The Ludi GDL description of games as high level, hierarchical ludeme trees is ideal for this purpose.

Mutation

Elements of the child game are then visited and *mutated* with 10% likelihood. This may involve adding, changing or removing elements and/or attributes within the ludeme tree. Context-dependent constraints are again employed to ensure that changes are compatible with element type, and that deleted items are replaced with default values where needed. Some repair functions [37] are used to correct obvious errors.

Baptism

Each child is given a short name unique to the population, using a Markov chain algorithm based upon letter combination frequencies found within a list of source words [14]. Ludi uses as its input a list of Tolkien-style names from a public domain computer game [38] to produce names such as: Oroth, Galdal, Etherond, Kemeneth, Valindor, Bered, Mor, and so on.

B. *Validity Checks*

Each child game then undergoes a series of validity checks.

Rule Safety

The child is tested for *rule safety* [35] by invoking the GGP to instantiate the corresponding game object, in order to exploit its rigorous error checking systems.

Some rule optimisation would typically be performed at this stage, such as trimming impossible or irrelevant vestigial rules [39], but Ludi forgoes this step for reasons given shortly. As Perlis [29] observes: “Optimization hinders evolution.”

Speed Test

The child is timed at search plies of 1, 2, 3 and 4 and discarded if move planning exceeds 15 seconds per move. This somewhat draconian measure ensures that slower games don’t unduly hold up the evolutionary process.

One concern is that the imposed speed limit will filter out more complex games, typically those on larger boards with more pieces and greater branching factors, and risk producing simple children only. However, it was our specific intention to produce elegant games in which simple rules combine harmoniously to produce complex move decisions rather than seeking complexity in sheer volume of numbers; the validity of this approach was born out in the experimental results.

Policy Choice

Perfunctory policy choice is achieved by comparing all combinations of the child’s default policy with both parents’ policies in round-robin matches and choosing the most successful hybrid. The child is discarded if match games take too long or more than half fail to produce a winner.

Inbreeding

The child is measured for distance from each member of the population and culled if an overly close relative is found.

C. *Repopulation*

All children that survive to this point are measured for quality and inserted back into the population according to their predicted aesthetic score. These scores are only preliminary at this stage but sufficient for ordering purposes.

A by-product of the aesthetic measurement is a *viability test* that identifies games as non-viable if they:

- result in draws more often than not,
- are very unbalanced towards either player,
- have a serious first or second move advantage, or
- do not end within a reasonable number of moves.

Viable offspring have their policies optimised and are remeasured for quality as a post-processing step. They are the successful products of the evolutionary process and typically represent around 1% of all non-culled children.

Emergence

Since no rule optimisation is performed and all surviving offspring are returned to the gene pool, the population will be rife with flawed rules and broken games. This seems at odds with the general wisdom that decries the presence of introns and bloat in the population [40] but proved necessary for the success of this project.

Sanitising the population had the effect of producing offspring that were mostly just slight variations of their original ancestors, as it is highly unlikely that two given rule sets will recombine and mutate to produce a viable, distinct and superior child, due to rule fragility. Instead, it proved more effective to flood the population with flawed genetic material that would act as recessive genes and recombine in unexpected and hopefully serendipitous ways over many generations; this was where true innovation emerged during the experiments.

TABLE I
CORRELATION BY CRITERIA TYPE

Criteria	$corr_r$
All (57)	0.426
Intrinsic (16)	0.115
Quality (30)	0.437
Viability (11)	0.609
Best set (17)	0.828
Best set (16)	0.799

VI. EXPERIMENTS

Three experiments involving game ranking, measurement and synthesis were then conducted to test the hypotheses.

A. Experiment I: Game Ranking

In order to distinguish good games from bad, it is first necessary to determine which games interest human players. The aim of the first experiment was to set a yardstick of human player rankings for a database of 79 predefined sample games.

Method

Experimental subjects were presented with survey software that randomly selected two games from the database of 79, then required them to play both games against the computer and nominate which of the two they found more interesting. Paired comparisons were automatically emailed to the authors.

Subjects

57 subjects participated in the survey, recruited from online board gaming groups. All were at least 18 years of age.

Results

Rankings were induced from 628 paired comparisons by a *cross-entropy* (CE) method due to the second author [15]. The 79 sample games were ranked from most preferred to least preferred with a classification rate of 0.8997 (variance = 0.000318), indicating that the derived game rankings were consistent with almost 90% reliability.

B. Experiment II: Game Measurement

The purpose of experiment II was to determine whether the player rankings induced for the 79 sample games could be correlated with aesthetic measurements of those games.

Method

Each of the 57 aesthetic criteria were measured for the 79 sample games through automated self-play trials, conducted on two Windows desktop machines over a period of two weeks. Some supervision was required to narrow the beam search for slower games, in order to finish within the given timeframe.

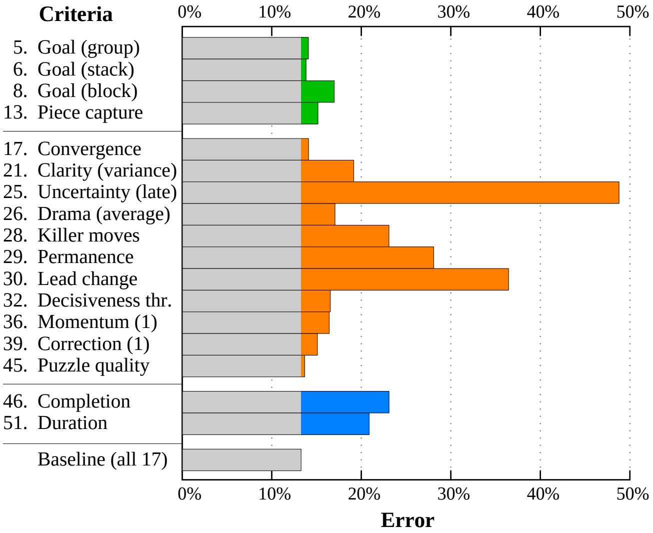


Fig. 10. Relative contributions of the best 17 predictors.

Results

The aesthetic measurements were correlated with player rankings using linear regression and standard leave-one-out cross-validation, to give $corr_r$ the correlation between ranking and aesthetic score. A set of 17 aesthetic criteria were identified as the best combination of predictors using another CE method due to the second author [15].

Table I shows a baseline ranking correlation of 0.426 with a 95% confidence interval of 0.435 [0.176 to 0.611] using all 57 criteria as predictors. It can be seen that the intrinsic criteria in isolation are poor predictors of game ranking (0.115) but that the quality criteria perform relatively well (0.437) and viability criteria even better (0.609) as predictors of game ranking.

The best set of predictors involved 17 criteria drawn from all three categories and proved to be an excellent predictor of game ranking, with a 0.828 correlation and 95% confidence interval of 0.371 [0.562 to 0.933]. The “puzzle quality” criterion was later removed to produce a best set of 16 predictors with a significant speed advantage but negligible loss of accuracy (Table I).

The relative importance of each predictor is indicated in Fig. 10, based on the increase in error when each is removed from the set. Six of these criteria stand out as most important:

- uncertainty,
- lead change (negative correlation),
- permanence,
- killer moves,
- completion, and
- duration.

The survey participants appear to prefer stable games with uncertain outcomes that end within a reasonable number of moves, and in which strong moves are reasonably permanent. Lead change is distinct from drama as it refers to lead change frequency.

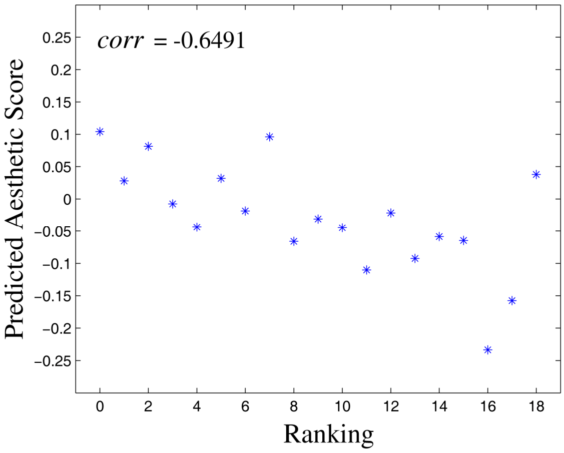


Fig. 11. Predicted score versus player ranking of synthesised games.

These results support hypothesis I: *That there exist fundamental (and measurable) indicators of game quality*, at least for this group of subjects and this set of combinatorial games.

C. Experiment III: Game Synthesis

Experiment III was designed to test whether new, viable games may be evolved from existing games, and whether aesthetic measurements may be used to reliably rank them.

Method

Using the database of 79 sample games from Experiment I as the initial population, a number of evolutionary runs were conducted on three Windows desktop machines over one week. The coefficients of the best 16 predictors were used to predict aesthetic scores for new games, which were then ranked and a follow-up survey, similar in format to that of Experiment I, conducted to evaluate these predicted rankings.

Subjects

27 subjects participated in the follow-up survey, recruited mostly from the 57 participants of Experiment I.

Results

A total of 1,389 new games were evolved from the initial population of 79 sample games and 19 deemed viable. A selection is listed in Appendix II; see [15] for complete descriptions and analyses of the evolved games.

127 paired comparisons were received for the 19 viable games, and human player rankings were induced as per Experiment I with a classification rate of 0.8283.

Figure 11 shows a plot of the predicted aesthetic scores versus actual player rankings of the new games, with a correlation of -0.6491 and 95% confidence interval of 0.577 [-0.851 to -0.274]. The relationship is negative as higher scores generally correspond to lower (i.e. better) rankings, as expected. This indicates a significant linear trend between the aesthetic measurements made by the system and player rankings for the 19 new games.

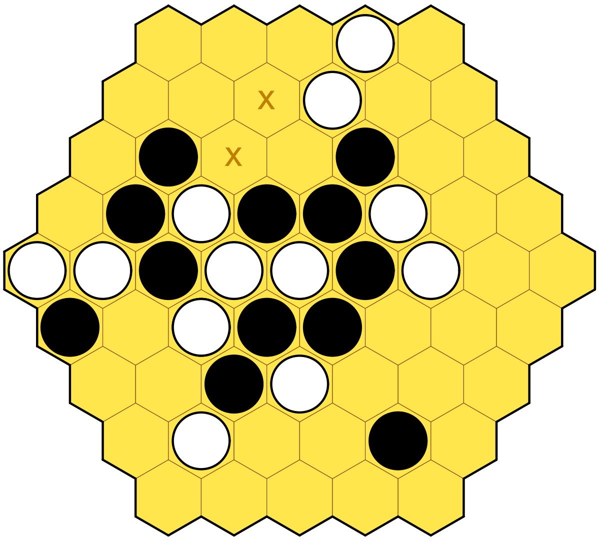


Fig. 12. Yavalath puzzle: White to play and force a win.

These results support hypothesis II: *That these fundamental indicators may be harnessed for the directed search for new high quality games*, at least in the search for new combinatorial games that this group of subjects find interesting.

VII. Discussion

The first thing to note is the general success of the approach; the system was able to correlate aesthetic measurements of games with human player rankings and hence identify those evolved games of most interest. Several of the final 19 games exhibit novel and interesting rule combinations, and those ranked #1 and #2 by human players – Ndengrod and Yavalath – have proven to be of exceptional quality and are now commercially published [41].

Ndengrod combines Go-like surround capture with a 5-in-a-row goal. This combination works well, but is a rediscovery of an existing game (Irensei) translated to the hexagonal grid.

Yavalath, however, features an innovative rule that has not previously been published: win by making 4-in-a-row but lose by making 3-in-a-row before doing so. Bearing in mind the assertion that good games should yield interesting puzzles [20], Fig. 12 shows a Yavalath puzzle that demonstrates its depth. *Hint*: Black can force a win with either move ‘x’, so White must make a counter forcing move to avoid this.

Analysis of Yavalath’s ancestry reveals that this innovative winning condition came about from the serendipitous mating of rules that were impossible in isolation. If such flawed rules had been optimised out during the evolutionary process then Yavalath would probably never have emerged.

Teiglith (#4), Elrostir (#5), Gorodrui (#7) and Valion (#16) demonstrate the approach’s usefulness even with games of average or below average appeal. While not overly successful as games, each involve interesting rule mechanisms that game authors might use as inspiration for future designs. See Appendix II for GDL descriptions of the games mentioned in this section.

Curiously, 63% of the final games (12 out of 19) featured *N*-in-a-row goals, as opposed to 29% (23 out of 79) in the initial data set. This may be a reflection of the prevalence of such games in the initial data set, the superiority of the *N*-in-a-row advisor over other advisors, or simply indicate that this is a robust rule that thrives in more contexts than others. In any event, it shows that interesting games can indeed be derived from Tic-Tac-Toe.

VIII. Conclusion

We demonstrate both the Ludi system's ability to automatically measure games for their potential to interest human players, and its ability to create new high quality games. This is the first known demonstration of automated combinatorial game design at a successful (publishable) level. However, we do not claim to have defined a canonical set of aesthetic features with which all games can be measured; rather, our results pertain to a selection of individuals for which a particular set of aesthetic measurements appear to resonate for a subset of combinatorial games definable in the Ludi GDL. Our main contribution is to demonstrate a practical approach to the problems of automated game measurement and content (rule) generation; no doubt many more such aesthetic features will be discovered in the future over a much wider range of games.

Future work might also include expanding the GDL to increase its scope and to seed the generative process with larger data sets of games, such as the thousands described online at [42]. Monte Carlo tree search methods such as UCT [43] could overcome some shortcomings of the advisor/policy approach during move planning to further increase the generality of the system. It would also be interesting to extend the system to perform not only game measurement and synthesis, but the detection of flaws such as symmetry strategies [23] and their correction.

Finally, we hope that game designers do not see in systems such as Ludi a threat to this very human endeavor. Designers can benefit tremendously from such tools and will – for the foreseeable future – hold the edge in unconstrained creativity.

Appendix I
The Ludi GDL

This Appendix provides a functional description of the key elements of the Ludi game description language (GDL). A complete formal definition of the grammar is beyond the scope of this paper, but can be found in [15].

Games are defined as recursive trees of elements of the following form:

```
(element [attributes] [(element ...)s])
```

The first item of each element is its name which must be unique within its current scope. Each element name must be either a predefined keyword or a user-defined variable, such as a named piece type, that has been previously described within the game's rule set. Italicised symbols indicate keywords, upper case symbols indicate data types, capitalised symbols indicate record types, and square brackets [] denote optional items. Approximately 200 keywords and 20 record types are defined and implemented for the language.

The root of each rule tree is the main *game* ludeme:

```
game →  (game NAME [params]
             players
             board
             rules
            [support]
        )

rules →{ [pieces] [start] [play] end }
```

These main ludeme types are now briefly described.

Players Ludeme

Specifies the player names and directions (mandatory).

```
players →(players
            {NAME|(NAME CompassDirection)}s
         )
```

Each player has a unique name and is optionally associated with a compass direction indicating direction of play. All games in this study involved two players "White" and "Black".

Board Ludeme

Specifies the board topology (mandatory).

```
board →(board [phase]
            (tiling TilingType [i-nbors])
            (shape ShapeType)
            (size UINTs)
           [(regions RegionRecords)]
        )
```

Board cells may be *phase*d to alternate colour with neighbours, such as the light and dark cells of a Chess board. Trivalent tilings require three phase colours, as shown in Fig. 4. The following tilings are supported:

- *tri*,
- *square*,
- *hex*,
- *trunc-square* (4.8.8 semiregular tiling).

Adjacency is assumed between orthogonal neighbours, while the optional *i-nbors* flag indicates adjacency between indirect (diagonal) neighbours. The following board shapes are supported:

- *tri*,
- *square*,
- *hex*,
- *rhombus*,
- *trapezium*,
- *boardless*.

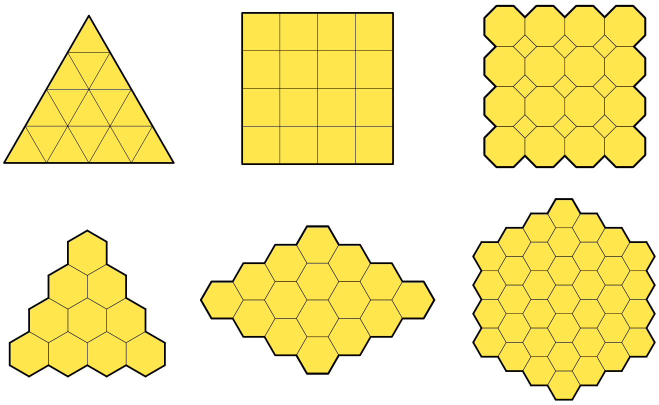


Fig. 13. Some board types of size 4.

The user defines one or more board dimensions depending on tiling and shape, and may optionally define distinct regions of cells for special purposes such as goal areas, promotion zones or connection targets. Figure 13 shows some examples of supported board types of size 4.

Pieces Ludeme

Specifies piece types and movement (optional)

```
pieces → (pieces piece_defns [NAME] )

piece_defn → (NAME PlayerType
                [(label STRING)]
                [(value INT)]
                [(state State)]
                [(flags Flags)]
                 (moves move_defns)
             )

move_defn → (move
               [(priority UINT)]
               [mandatory]
               [(label STRING)]
               [(dirn Direction)]
               [(owner PlayerType)]
               [(pre bool_function)]
                (action {pass|actions})
               [(post post_conditions)]
            )
```

Defining piece behavior is the core of the language's complexity and where most of its 200 predefined keywords are used. Each piece is identified by a unique name and associated with one or more players, and may optionally be given a label for display purposes, a value, a state and a set of flags.

Each piece must have one or more move definitions, optionally marked as mandatory and/or ordered by priority if more than one are provided. Moves may also be labeled for display purposes, for example when the current player must choose from among a choice of moves, may be associated with a direction (absolute or relative to the piece's current orientation) and may be optionally marked with an owner.

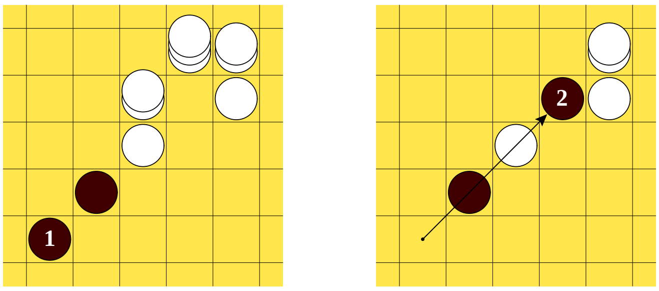


Fig. 14. A complex move.

The *pre* clause specifies preconditions that must exist for the move to occur, and takes the form of a tree of keywords combined with logical operators that may describe complex conditions. The *action* clause defines the move itself which may involve passing, adding a piece or moving a piece, possibly to stack. The *post* clause specifies postconditions that are exercised after the move, including:

- captures,
- conversions,
- rotations,
- displacement of neighbours,
- piece state updates,
- cell state updates,
- piece flag updates,
- score updates, and so on.

The following example shows how complex move descriptions may be constructed from these simple predicates.

```
(move
  (pre
    (and
      (owner from)
      (empty to)
      (line)
      (or
        (not (> (num-between empty) 0))
        (= (num-between enemy) 1)
      )
    )
  )
  (action (pop) (push))
  (post
    (capture
      (if (>= (height current) 2)) d-nbors
    )
    (inc-state)
  )
)
```

This movement rule states that the piece may be moved if it belongs to the current player and is moved to an empty board cell along a line that contains either no empty spaces or a single enemy piece (or both). The piece is popped from its current position and pushed to its new position (i.e. it is moved), possibly to stack. Following the move, all orthogonally adjacent stacks of height 2 or more (enemy stacks by default) are captured and the piece's state is incremented. The piece's state may be relevant to another move type for this piece or even other pieces. Figure 14 shows such a move being executed.

Non-linear move paths, such as knight moves, may be defined in the *pre* clause by relating the move's to and *from* cells using a sequence of turtle directions, relative to either the piece's current orientation or a global direction:

- *f* = forwards,
- *b* = back,
- *l* = left,
- *r* = right.

If no *pieces* ludeme is defined then players add a piece of their colour at an empty cell each turn by default.

Start Ludeme

Specifies the initial board state (optional).

```
start →(
            [place_clause]s
            [in_hand_clause]s
        )
```

The optional *place_clause*s specify pieces to be placed on the board before the game starts. Numbers of specific pieces may be placed at particular cells or regions, or relative to each player's home row, including specifiers for cell phase and some symmetry operations.

The optional *in_hand_clause*s specify the number of particular pieces held in-hand and off the board by each player, which may be entered into play as the game progresses.

If no *start* ludeme in specified, then the board is initially empty and players hold an infinite number of each piece type in hand.

Play Ludeme

Specifies other constraints on play (optional).

```
play →  ([can-pass])
```

The *play* ludeme specifies whether players may voluntarily pass or not. This was the only constraint on play implemented for the experiments, but other constraints of this type might include first move equaliser, progressive move counts, play order, and so on.

If no *play* ludeme is specified then players may not voluntarily pass. However, they may be forced to pass if they have no valid moves on a given turn, unless the *no-move* end condition is specified.

End Ludeme

Specifies terminating conditions (mandatory).

```
end → (end
          end_clauses
          [mover-wins|mover-loses|draw]
        )

end_clause →(PlayerType ResultType
                 Tree<end_condition>
             )
```

The *end* ludeme specifies a number of *end_clauses* and optionally how to handle the case of a move triggering end clauses for both players: *mover-wins*, *mover-loses* or *draw*n game. Each end_clause specifies a player (White, Black or All players), a result type (*win*, *lose* or *draw*) and tree of logical operators relating one or more end conditions of the following types:

- *connect*: Connect two or more target regions of the board, according to optional piece, adjacency and stacking constraints.
- *group*: Form a single connected group of a given size, according to optional piece, adjacency and stacking constraints.
- *in-a-row*: Form a consecutive line of *N* pieces, according to optional piece and adjacency constraints.
- *reach*: Reach the specified goal (cell, region or board side) with the specified number of the specified piece.
- *capture*: Capture the specified number of the specified pieces.
- *eliminate*: Eliminate the specified player.
- *score*: Reach the specified score.
- *stack*: Achieve the specified stack size under certain constraints.
- *state*: Achieve the specified cell or piece state.
- *no-move*: The current player has no moves.

The game ends as soon as any end condition is completely met, and the specified player is awarded the specified result (*win*, *lose* or *draw*).

For example, the following game ends either when White loses by forming a group of size 5 or when either player wins by having no moves and either connecting their sides of the board or forming a stack of height 4:

```
(end
  (White lose (group 5))
  (All win
        (and
           (no-move)
           (or
               (connect own-regions)
               (stack 4)
           )
        )
    )
)
```

Support Ludeme

Specifies additional metadata for the game (optional).

```
support →{
              [advisors]
              [description]
              [aim]
              [ancestry]
              [ranking]
              [viable]
              [score]
            }
```

These items fulfill the following roles:

- advisors: Defines the policy for the game as a list of relevant advisors and their relative weightings.
- description: Includes a text description of the game for help manual purposes.
- aim: Includes a text description of the aim of the game, which, together with the GGP's tutorial mode, should help new players learn the game quickly.
- ancestry: Contains information on the game's evolutionary history including its immediate parents, generation number, and average distances from its parents, members of the initial population and members of the final population.
- ranking: Contains an estimated ranking of the game within its population.
- viable: Contains the estimated viability of the game according to the viability test described in Section V.
- score: Specifies the game's estimated aesthetic score as measured by the process described in Section IV.

APPENDIX II
GDL DESCRIPTIONS OF SYNTHESISED GAMES

NDENGROD (#1)

```
(game Ndengrod
  (players White Black)
  (board (tiling hex) (shape trapezium) (size 7 7))
  (pieces
    (Piece All
      (moves
        (move
          (pre (empty to))
          (action (push))
          (post (capture surround))
        )
      )
    )
  )
  (end (All win (in-a-row 5)))
)
```

YAVALATH (#2)

```
(game Yavalath
  (players White Black)
  (board (tiling hex) (shape hex) (size 5))
  (end
    (All win (in-a-row 4))
    (All lose (and (in-a-row 3) (not (in-a-row 4))))
  )
)
```

TEIGLITH (#4)

```
(game Teiglith
  (players White Black)
  (board (tiling square) (size 7 7))
  (pieces
    (Stone All
      (moves
        (move
          (pre
            (and
              (> (group-size to) (phase to))
              (connected)
            )
          )
          (action (pop) (push))
        )
      )
    )
  )
  (start (place (Stone White) home))
  (end (All win (no-move)))
)
```

ELROSTIR (#5)

```
(game Elrostir
  (players White Black)
  (board (tiling square i-nbors) (size 5 5))
  (end (All lose (or (no-move) (in-a-row 3))))
 )
```

GORODRUI (#7)

```
(game Gorodrui
  (players White Black)
  (board (tiling hex) (shape hex) (size 3))
  (pieces
    (Stone All (state 1)
      (moves
        (move (pre (empty to)) (action (push)))
        (move
          (pre
            (and
              (enemy from) (empty to)
              (= (+ (piece-state) 1) (distance))
            )
          )
          (action (pop) (push))
          (post (inc-state))
        )
      )
    )
  )
  (start (in-hand (Stone All) 5))
  (end (All lose (no-move)))
)
```

VALION (#16)

```
(game Valion
  (players White Black)
  (board (tiling square i-nbors) (size 4 4))
  (pieces
    (Stone All
      (moves
        (move
          (pre (>= (num-nbors to enemy) 1))
          (action (push))
        )
        (move
          (pre (and (empty to) (connected)))
          (action (push))
        )
      )
    )
  )
  (start
    (place (Stone White) A1)
    (place (Stone Black) D4)
  )
  (end
    (All win (in-a-row 3))
    (All lose (or (in-a-row 4) (group)))
  )
)
```

ACKNOWLEDGMENT

Thanks to B. Pham (QUT, Brisbane) for useful guidance, I. Althöfer (Friedrich Schiller Universität, Jena) for his help and enthusiasm towards the topic, S. Colton (Imperial College London) for support, the reviewers and proofreaders.

REFERENCES

[1] B. Pell, “METAGAME in symmetric Chess-like games,” *Heuristic Programming in Artificial Intelligence 3*, H. Van den Kerik and L. Allis, Eds. Chichester: Ellis Horwood, 1992.

[2] K. Salen, and E. Zimmerman, *Rules of Play*, Cambridge: MIT Press, 2004.

[3] E. Berlekamp, J. Conway and R. Guy, *Winning Ways for your Mathematical Plays*, Volumes 1 and 2, London: Academic Press, 1982.

[4] R. Dawkins, *The Selfish Gene*, Oxford: Oxford Univ. Press, 1976.

[5] A. Borvo, *Anatomie d’un jeu de cartes: L’Aluette ou le Jeu de Vache*, Nantes: Librarie Nantaise Yves Vachon, 1977.

[6] D. Parlett. “What is a ludeme?”, *Incompleat Gamester*, 2007 [Online]. Available: http://www.davidparlett.co.uk/gamester/ludemes.html.(URL)

[7] C. Browne, *Hex Strategy: Making the Right Connections*, Natick, Massachusetts: AK Peters, 2000.

[8] J. Holland, *Emergence: From Chaos to Order*, Redwood City: Addison-Wesley, 1998.

[9] L. Horsten, “Philosophy of mathematics,” 2007 [Online]. Available: http://plato.stanford.edu/entries/philosophy-mathematics.(URL)

[10] J. Pitrat, “Realisation of a general game-playing program,” *IFIP Congress*, 2, 1570–1574, 1968.

[11] M. Genesereth, N. Love, and B. Pell, “General game playing: Overview of the AAAI competition”, *AI Magazine*, 26:2, 62-72, 2005.

[12] N. Love, T. Hinrichs, D. Haley, E. Schkufza and M. Genesereth, “General game playing: Game description language specification,” Technical Report LG-2006-01, Stanford Logic Group, 2006.

[13] J. Mallett and M. Lefler, “Zillions of Games,” 1998 [Online]. Available: http://www.zillions-of-games.com.(URL)

[14] B. Kernighan and R. Pike, *The Practice of Programming*, Reading: Addison-Wesley, 1999.

[15] C. Browne, *Automatic Generation and Evaluation of Recombination Games*, Ph.D. Thesis, FIT, QUT, Brisbane, Australia, 2008.

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., New Jersey: Prentice Hall, 2003.

[17] S. Epstein, “Deep Forks in Strategic Maps: Playing to Win,” *Heuristic Programming in Artificial Intelligence 2*, D. Levy and D. Beal, Eds. Chichester: Ellis Horwood, 1991, 189-203.

[18] B. Pell, *Strategy Generation and Evaluation for Meta-Game Playing*, Ph.D. Thesis, Trinity College, Univ. of Cambridge, UK, 1993.

[19] T. Bäck, F. Hoffmeister, and H. Schwefel, “Basic aspect of evolution strategies,” *Statistics and Computing*, 4:2, 1994, 51-63.

[20] M. Thompson, “Defining the abstract,” *The Games Journal*, 2000 [Online]. Available: http://www.thegamesjournal.com.(URL)

[21] I. Althöfer, I. (2003) “Computer-aided game inventing,” Technical Report, Friedrich Schiller Universität Jena, 2003 [Online]. Available: http://www.minet..uni-jena.de.preprints/althoefer_03/CAGI.pdf.(URL)

[22] H. Iida, K. Takahara, J. Nagashima, Y. Kajihara and T. Hashimoto, “An application of game-refinement theory to Mah Jong,” *ICEC 2004 (LNCS 3116)*, M. Rauterberg Ed. Berlin: Springer, 2004, 445-450.

[23] C. Browne, *Connection Games: Variations on a Theme*, Natick, Massachussetts: AK Peters, 2005.

[24] W. Kramer, “What makes a game good?” *The Games Journal*, 2000 [Online]. Available: http://www.thegamesjournal.com.(URL)

[25] M. Gardner, “Theory of everything,” *The New Criterion*, 23:2, 2004.

[26] H. Ellis, *Impressions and Comments*, Boston: Houghton Mifflin, 1914.

[27] G. Birkhoff, *Aesthetic Measure*, Cambridge: Harvard Univ. Press, 1933.

[28] G. Stiny and J. Gips, *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts*, Berkeley: Univ. Calif. Press, 1978.

[29] A. Perlis, “Epigrams on programming,” *SIGPLAN Notices*, 17:9, 1982, 7-13.

[30] T. Rolle, *Development of a Multi-Game Engine*, Diploma Thesis, Friedrich-Schiller-University Jena, Faculty of Mathematics and Computer Science, Jena, 2003.

[31] L. Hansheng and K. Lishan, “Balance between exploration and exploitation in genetic search”, *Wuhan University Journal of Natural Sciences*, 4:1, 1999, 28-32.

[32] T. Bäck, F. Hoffmeister, and H. Schwefel, “A survey of evolution strategies,” *Genetic Algorithms*, R. Belew and L. Booker Eds. San Mateo: Morgan Kaufman, 1991, 2-9.

[33] J. Baker, “Reducing bias and inefficiency in the selection algorithm,” *2nd Intl. Conf. on Genetic Algorithms and their Application*, Mahwah: Lawrence Erlbaum, 1987, 14-21.

[34] J. Koza, “Genetic programming as a means for programming computers by natural selection,” *Statistics and Computing*, 4, 1994, 87-112.

[35] W. Langdon, “Data structures and genetic programming,” *Advances in Genetic Programming 2*, Cambridge: MIT Press, 1996, 395-414.

[36] D. Montana, “Strongly typed genetic programming,” *Journal of Evolutionary Computation*, 3, 199-230, 1995.

[37] Z. Michalewicz and G. Nazhiyath, “Genocop III: A co-evolutionary algorithm for optimisation problems with nonlinear constraints,” *2nd IEEE Conf. on Evolutionary Computation*, 1995, 647-651.

[38] J. Greene, “Angband,” 2007 [Online]. Available: http://members.cox.net/nppangband/download.htm.(URL)

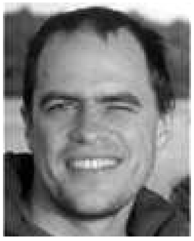
[39] G. Costikyan, “Don’t be a vidiot: What computer game designers can learn from non-electronic games”, 2005 [Online]. Available: http://www.costik.com/vidiot.html.(URL)

[40] T. Soule and J. Foster, “Code size and depth flows in genetic programming,” *2nd Conf. on Genetic Programming*, 1997, 313-320.

[41] N. Romeral Andrés, “Nestorgames,” 2009 [Online]. Available: http://www.nestorgames.com.(URL)

[42] S. Alden, “BoardGameGeek,” 2000 [Online]. Available: http://www.boardgamegeek.com.(URL)

[43] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” *15th European Conference on Machine Learning*, 2006, 282-293.



Cameron Browne received the Ph.D. degree in computer science from the Faculty of Information Technology, QUT, Brisbane, Australia, in 2008.

Currently, he is a visiting researcher at the Computational Creativity Group at Imperial College London. His research interests include AI techniques for computer artwork and game design.

Dr. Browne received an Australian Postgraduate Award for his Ph.D. studies and the Dean’s Award for Outstanding Thesis of 2008.

Frederic Maire received the Ph.D. degree in discrete mathematics from the Universite Pierre et Marie Curie, Paris 6, France, in 1993, after completing M.Sc. degrees in pure mathematics and computer science engineering in 1989.

Currently, he is a senior lecturer at the Faculty of Science and Technology, Queensland University of Technology, Brisbane, Australia. His research interests include computer vision and robotics.

Dr. Maire is a member of the IEEE.