# General Board Geometry

Cameron Browne, Éric Piette, Matthew Stephenson, and Dennis J. N. J. Soemers

Department of Data Science and Knowledge Engineering, Maastricht University, Paul-Henri Spaaklaan 1, 6229 EN, Maastricht, the Netherlands
{cameron.browne,eric.piette,matthew.stephenson,dennis.soemers}
@maastrichtuniversity.nl

**Abstract.** Game boards are described in the Ludii general game system by their underlying graphs, based on tiling, shape and graph operators, with the automatic detection of important properties such as topological relationships between graph elements, directions and radial step sequences. This approach allows most conceivable game boards to be described simply and succinctly.

**Keywords:** General Game Playing · Ludii · game boards · geometry.

## 1 Introduction

The Digital Ludeme Project (DLP) is a five-year research project using Artificial Intelligence techniques to improve our understanding of the development of games throughout history [2]. We are modelling the 1,000 most "important" traditional strategy games in a consistent digital format, to provide a playable database of the world's traditional games for comparative analysis.

### 1.1 Ludii

The Ludii general game system[1] [5] is a software tool developed specifically for this task, for modelling the full range of possible board games (950+ games implemented in version 1.2.8). Games are described in terms of simple *ludemes* assembled into structures to define arbitrarily complex behaviour, where each ludeme is a game-related concept implemented as a Java class (or enum attribute) in the Ludii code base [4].

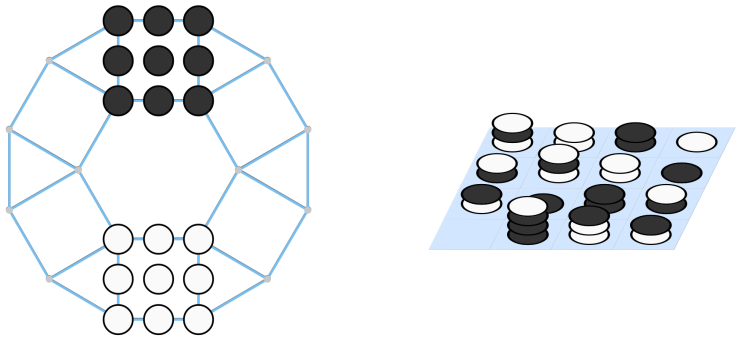
A key challenge in this task is to allow the user to describe arbitrarily complex game boards in a simple and intuitive way. This paper outlines our method for describing game boards in the Ludii grammar for general games.

[1] Ludii is available at ludii.games and the source code at github.com/Ludeme/Ludii

## 2 Game Graphs

In Ludii, the board shared by all players is represented internally as a finite graph defined by a triple of sets $\mathcal{G} = \langle V, E, C \rangle$ in which $V$ is a set of vertices, $E$ a set of edges, and $C$ a set of cells. In graph theory, a cell is more commonly called *face* and represents a region bounded by a set of edges and that contains no other vertex or edge.[2] Vertex, edge and cell are all graph elements which can refer to each other, and denote *playable sites* at which players can place components during the game:

- Let $v \in V$ denote a vertex. Then $v$ is an endpoint to each edge in $E(v)$, $C(v)$ gives the set of cells that $v$ is part of, and $V(v) = \varnothing$.
- Let $e \in E$ denote an edge. Then $V(e)$ is a set of 2 vertices that are the endpoints of $e$, $C(e)$ gives the set of cells $e$ is bounding, and $E(e) = \varnothing$.
- Let $c \in C$ denote a cell. Then $E(c)$ is the set of all the edges bounding $c$, $V(c)$ gives the set of the vertices which are the endpoints of the edges bounding $c$, and $C(c) = \varnothing$.



(a) (b)

**Fig. 1.** (b) A board game with pieces played on vertices, edges and cells. (c) A stacking board game played on cells.

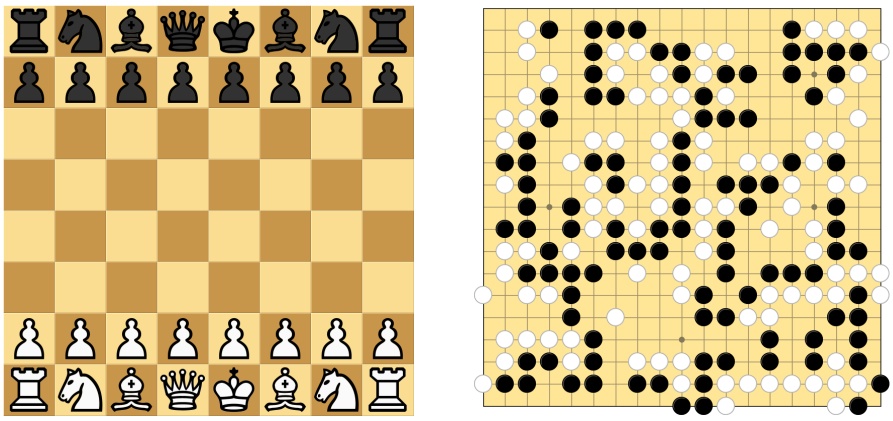
For example, Figure 1a shows a game with pieces placed on the vertices, edges and cells of the board graph. Figure 1b shows a board game played only on the cells, but in which pieces can may stack.

In any single game, components (or a stack of components) can be placed on any graph element. For this reason, we define a playable site as a triple $\langle Type, Index, Level \rangle$ in which the $Type$ can be (Vertex, Edge or Cell), the $Index$ is the number of the element and the $Level$ is the index of the element in the stack (0 meaning the ground).

[2] We sometimes use "game design" terms or definitions in lieu of stricter mathematical equivalents, in keeping with Ludii's primary purpose as a game design tool.

Any ludeme referring to a playable site has to specify each of these data. However, for convenience, Ludii uses default values. The default type of a location is Cell, except if the description of the game specifies another default site type. For levels, the default value is the top Level of the location specified.

### 2.1 Dimensions: Cells or Vertices



(a) (b)

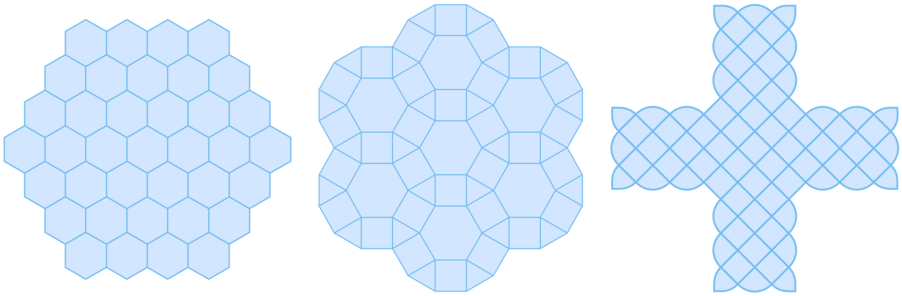
**Fig. 2.** (a) Chess (8x8 Cells). (b) Go (19x19 Vertices).

The graph is generated based on the specified board dimensions and default site type. For example, a Chess board described `(board (square 8))` (see Figure 2a) produces a square grid with 8 cells per row and column. However, a Go board described as `(board (square 19) use:Vertex)` (see Figure 2b) produces a square grid with 19 vertices per row and column. Note that if the default site type is `Edge`, the vertices are used for the dimensions.

## 3 Game Board Description

Game boards are described in the Ludii grammar [3] using the following basic EBNF syntax: `<board> ::= (board <graph>)` where the underlying `<graph>` object defines the vertices, edges and cells that make up the game board.

The user can specify the location of each vertex (and adjacencies between them as edges) to allow the description of arbitrarily complex graphs, or they can take advantage of a range of predefined tilings, shapes and graph operators for more concise descriptions (described more fully in Section 5). For example, the three game boards shown in Figure 3 are described by the following graphs (the `poly` field describes the polygonal shape of the board):



(a) (b) (c)

**Fig. 3.** Boards from tilings: (a) hexagonal, (b) semi-regular 3.4.6.4, (c) `celtic` basis.

```
(hex 4)

(tiling T3464 2)

(celtic (poly {{3 0}{3 4}{0 4}{0 7}{3 7}{3 11}{6 11}{6 7}{10 7}
               {10 5} {6 5}{6 0}}))
```

## 4 Graph Relations

For a graph $\mathcal{G} = \langle V, E, C \rangle$, two different graph elements $g_1$ and $g_2$ can have different relations:

- **Adjacent**: $g_1$ and $g_2$ are *adjacent* if and only if $(\exists e \in E(g_1) \cap E(g_2)) \vee (\exists v \in V(g_1) \cap V(g_2)) \vee (\exists c \in C(g_1) \cap C(g_2))$. In other words, two graph elements are adjacent if they share any graph element they are referring.
- **Orthogonal**: $g_1$ and $g_2$ are *orthogonal* if and only if $\exists e_1 \in E(g_1), \exists e_2 \in E(g_2), e_1 = e_2$. In other words, two graph elements are orthogonal if they share an edge.
- **Diagonal**: Two cells are considered *diagonal* if and only if:[3]
  1. They share a vertex (but not an edge) and the bisectors of the angles at that vertex in each cell are maximally opposed. Note that a cell can have multiple diagonal neighbours through a vertex if all satisfy this property. Or:
  2. If a cell has no such adjacent neighbour through a given vertex, then we allow a non-adjacent diagonal neighbour through that vertex if the two cells are coincident with the end points of some edge $E$ (which does not belong to either cell) and the bisectors of the angles at the end point in each cell are maximally opposed.

[3] This definition differs slightly from the actual implementation, but it captures the general understanding of *diagonality* between cells on a game board.

These two diagonal relationships are shown in Figure 4.
Diagonality is defined similarly for vertices, but transposing "cell" and "vertex" in the above definitions.

- **Off Diagonal**: $g_1$ and $g_2$ are off diagonal if and only if $g_1 \in C, g_2 \in C, \exists v_1 \in V(g_1), \exists v_2 \in V(g_2), v_1 = v_2, \nexists e_1 \in E(g_1), \nexists e_2 \in E(g_2), e_1 = e_2$. In other words, two cells are off diagonal if they are not diagonal, not orthogonal and they share a vertex.
- **All**: $g_1$ and $g_2$ are related if they are orthogonally, adjacently, diagonally or off diagonally related to each other.

These relationships are summarised for the regular tilings in Table 1.

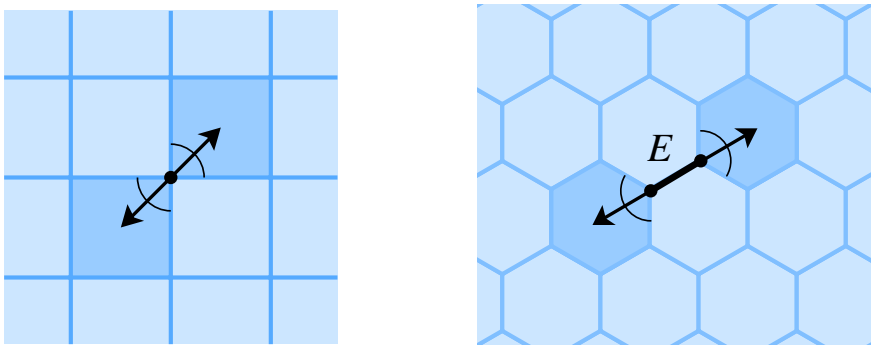



**Fig. 4.** Adjacent diagonals (left) and non-adjacent diagonals (right).

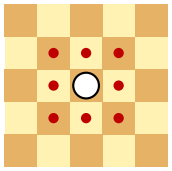
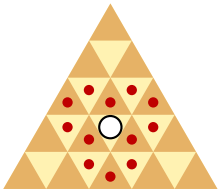
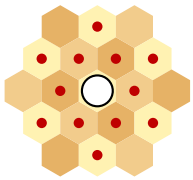
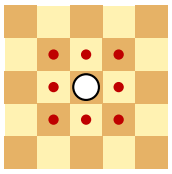
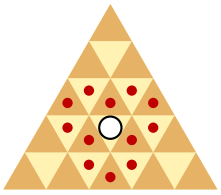
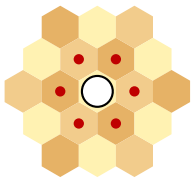
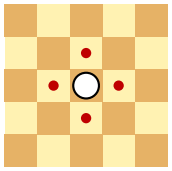
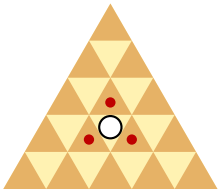
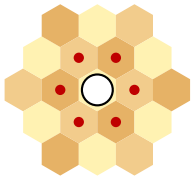
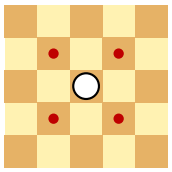
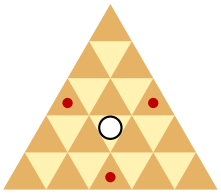
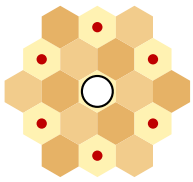
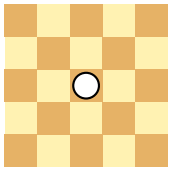
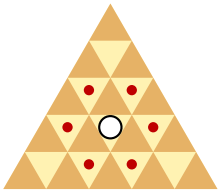
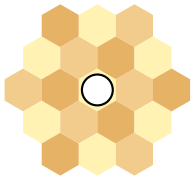
### 4.1 Directions

Ludii supports the following direction types:

- *Intercardinal* directions: `N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW.`
- *Rotational* directions: `In, Out, CW` (clockwise), `CCW` (counter-clockwise).
- *Spatial* directions for 3D games: `D, DN, DNE, DE, DSE, DS, DSW, DW, DNW and U, UN, UNE, UE, USE, US, USW, UW, UNW.`
- *Axial* directions subset (for convenience): `N, E, S, W.`
- *Angled* directions subset (for convenience): `NE, SE, SW, NW.`

Each graph element $g$ has a corresponding set of *absolute directions* $A_d$ and *relative directions* $R_d$ to associated graph elements of the same type. Absolute directions can be any of the above direction types in addition to any relation type (`Adjacent, Orthogonal, Diagonal, Off Diagonal`, or `All`).

**Table 1.** Relations for the regular tilings.

| **Relation** | **Square** | **Triangular** | **Hexagonal** |
|---|---|---|---|
| All |  |  |  |
| Adjacent |  |  |  |
| Orthogonal |  |  |  |
| Diagonal |  |  |  |
| Off-Diagonal |  |  |  |

Relative directions from an element $g$ are defined by $R_d(g, facing, rotation, relation)$ where $facing$ describes direction in which a component at $g$ is facing, $rotation$ describes to the number of rightward steps of the component at $g$, and $relation$ describes the graph relation to use at each step (`Adjacent` by default). Relative directions are: `Forward, Backward, Rightward, Leftward, FR, FRR, FRRR, FL, FLL, FLLL, BR, BRR, BRRR, BL, BLL` or `BLLL`.

For example, consider a piece on a `square` board (which involves only the eight major compass directions as adjacent relations). If the piece is facing `N` (North) with a rotation of 0, the relative direction `Forward` is the graph element immediately to the North (upwards) if such an element exists. However, if that piece is facing `E` (East) and its current rotation is 1, the relative direction `FR` (meaning "Forward Right") is the graph element immediately to the South (below) if such an element exists.

### 4.2 Steps and Walks

A *step* is a record of two related graph elements (*from* and *to*) which can be of different types and the absolute directions that describe their relationship. For example, a cell $A$ directly above another cell $B$ on a Chess board could be described as an `Adjacent`, `Orthogonal` or `N` step away.

Ludii also provide three relative step types (`F`, `L` and `R`) that allow users to define *walks* within the board graph. These correspond the standard "forward", "left" and "right" commands used in *turtle graphics* [1], as shown in Figure 5.

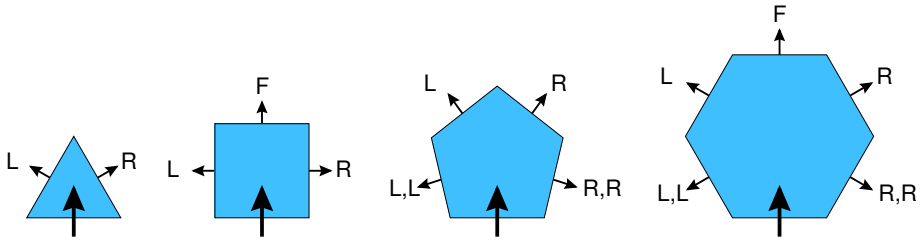



**Fig. 5.** Relative steps from various cell types.

This representation allows descriptions of piece movements to be easily transferred between different board topologies. For example, a knight move in Chess may be described as the walk `{F,F,R}` as shown in Figure 6 (left) ans this walk may be directly used on a board based on the semi-regular 3.4.6.4 tiling (Figure 6, right). Note, however, that different topologies may introduce ambiguities such as whether both right turns in the 3.4.6.4 knight move (dotted lines) should be considered valid moves or only one of them (probably the furthest reaching one). Such ambiguities should be resolved by the game designer according to the behaviour they want.

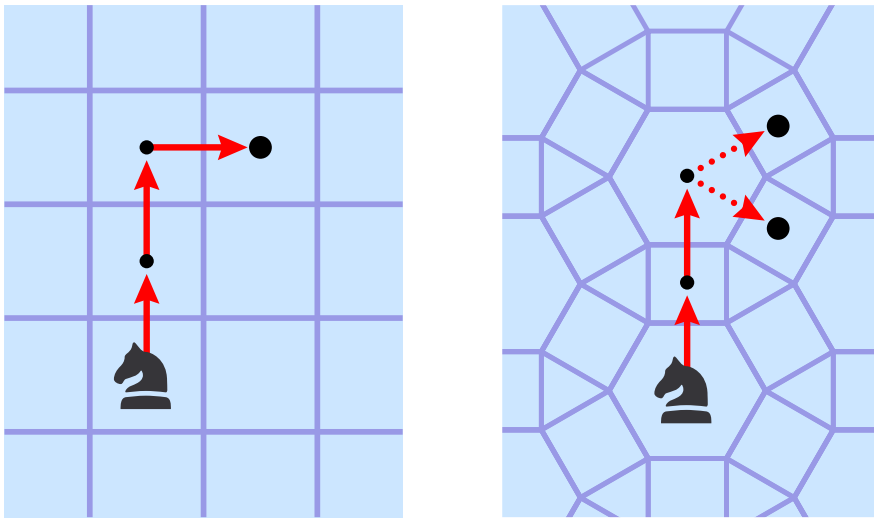


**Fig. 6.** Walk {`F,F,R`} describes knight moves on square and 3.4.6.4 tilings.

### 4.3 Radials

Many games involve piece movement through contiguous lines of cells in a direction, such as slide moves by the queen, rook and bishop pieces in Chess. Ludii automatically generates such *radials* for each playable site on the board, for convenient game description and efficient processing.

For each playable site on the board $S$, each valid step to a neighbouring graph element of the same type in an absolute direction $d$ is extended as far as possible, to produce a radial from $S$ in direction $d$. For example, Figure 7 shows `Orthogonal` radials from the shaded cell on a circular Chess board, such as a rook would move in the game Shatranj ar-Rumiya. Note that radials may bend to follow the board topology.

Radials extend step-by-step to the next step in the given absolute direction that minimises deviation in the radial's current heading. If the next step would deviate by $90^o$ or more, then the radial terminates.

Radials can *branch* where two or more steps in the current direction are equally as good.[4] For example, Figure 8 shows how an `Orthogonal` step into a triangular cell may validly continue either `L` (left) or `R` (right), and thereafter alternate {`L,R,L,R,...`} to produce branching zig-zagging radials in which the direction of each individual step is less important than the average direction of the radial overall.

[4] Still to be implemented in Ludii.

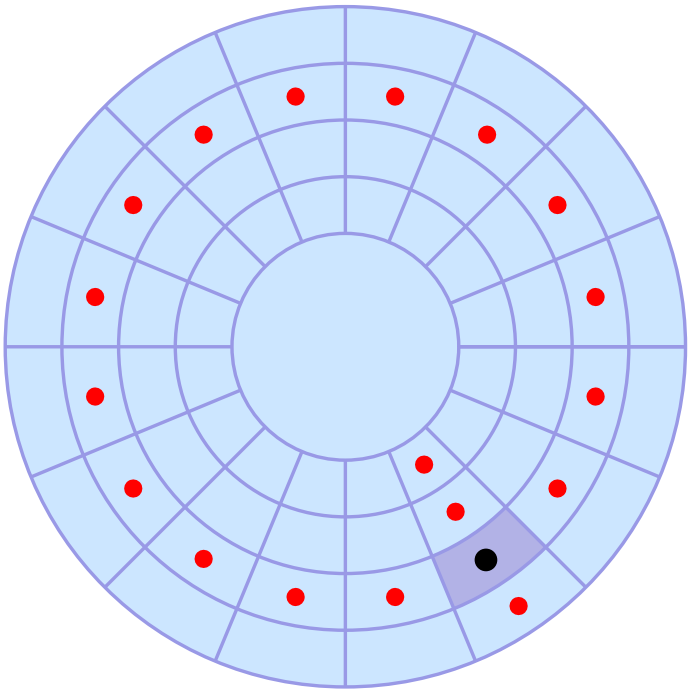


**Fig. 7.** Orthogonal radials on a circular Chess board (Shatranj ar-Rumiya).

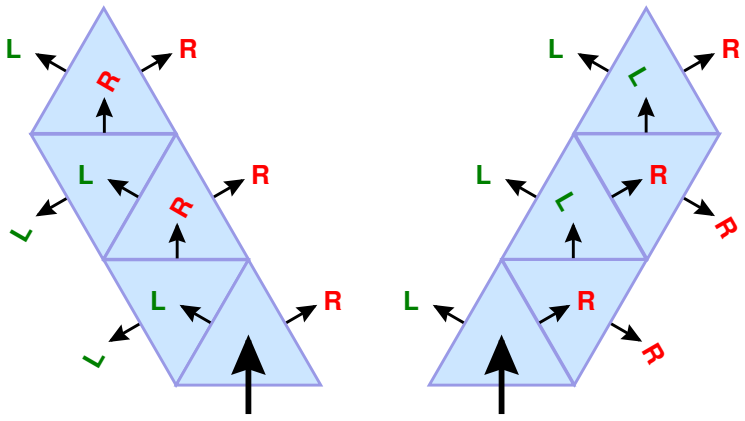



**Fig. 8.** Branching radial on a triangular grid.

## 5 Graph Operators

Graphs are initially defined by a tiling and/or shape but can then be further modified using a range of *graph operators*. The complete set of tilings, shapes and graph operators defined in the Ludii grammar is shown in Table 2. These can be used in combination to define thousands of different boards types quickly and easily. Greyed out items indicate planned future work not implemented yet.

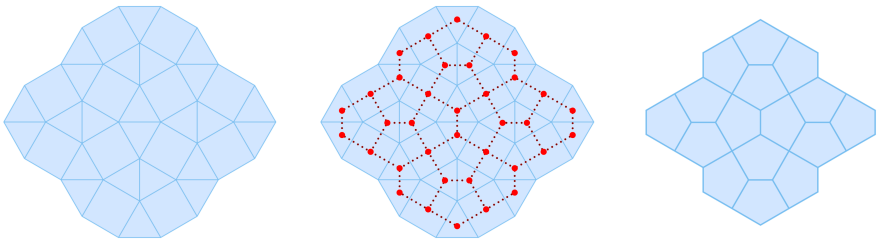
**Table 2.** Keywords in the Ludii grammar for describing game boards.

| **Tiling** | **Shape** | **Operator** |
|---|---|---|
| *Regular* | `square` | `add` |
| `square` | `rectangle` | `clip` |
| `hex` | `hexagon` | `complete` |
| `tri` | `triangle` | `dual` |
| | `wedge` | `hole` |
| *Semi-Regular* | `regular` (polygon) | `intersect` |
| `T488` (*i.e.* 4.8.8) | `poly (any polygon)` | `keep` |
| `T4612` (*i.e.* 4.6.12) | | `layers` |
| `T3464` (*i.e.* 3.4.6.4) | *Attribute* | `makeFaces` |
| `T3636` (*i.e.* 3.6.3.6) | `Star` | `merge` |
| `T31212` (*i.e.* 3.12.12) | `Diamond` | `recoordinate` |
| `T33336` (*i.e.* 3.3.3.3.6) | `Prism` | `remove` |
| `T33344` (*i.e.* 3.3.3.4.4) | | `renumber` |
| `T33434` (*i.e.* 3.3.4.3.4) | *Modifier* | `rotate` |
| | `diagonals:<DiagType>` | `scale` |
| *Custom* | `pyramidal:<boolean>` | `shift` |
| `concentric` | `limping:<boolean>` | `skew` |
| `spiral` | `fractal/recursive` | `splitCrossings` |
| `quadhex` | `lattice` | `subdivide` |
| `brick` | `projective` | `trim` |
| `celtic` | | `union` |
| `repeat` | | |

For example, the very useful `dual` operator converts a source graph into its *weak dual* defined by edges whose end points are the centroids of its adjacent cells. Figure 2 shows a `dual` operation applied to a small graph based on tiling 3.3.4.3.4 to produce the well known Cairo tiling:

```
(dual (tiling T33434 2))
```

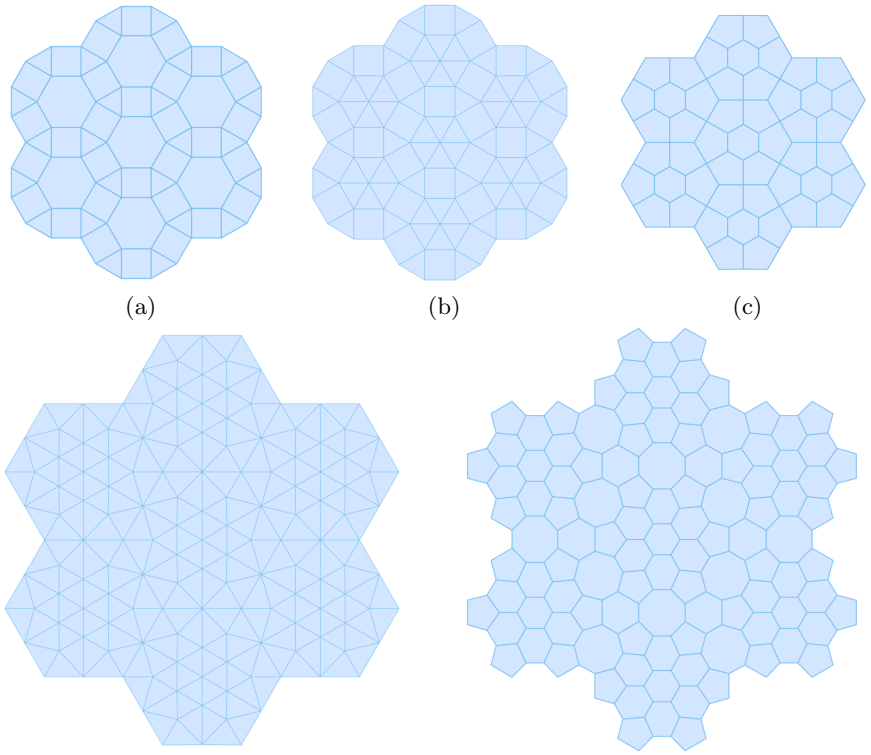
Another useful operator is `subdivide`, which subdivides all faces with $N$ or more sides into triangular sub-faces that share a central vertex (default $N = 1$). Figure 10 shows a sequence of `subdivide` and `dual` operations applied to a rhombitrihexahedral 3.4.6.4 tiling to produce a novel and exotic board design:



(a) (b) (c)

**Fig. 9.** A 3.3.4.3.4 tiling (a), its cell adjacencies (b) and its weak dual (c).

```
(dual (subdivide (dual (subdivide (tiling T3464 2) min:6))))
```



(a) (b) (c)

(d) (e)

**Fig. 10.** A 3.4.6.4 tiling (a) subdivided at $N \geq 6$ (b), its dual (c), all subdivided (d) and its dual (e).

## 6 Conclusion

The Ludii grammar provides a simple way to describe almost any conceivable game board by its underlying graph, using tiling, shape and graph operators. This approach has allowed us to model the boards of hundreds of games for the Ludii general game system, and continues to produce interesting new board designs based on simple operations.

## Acknowledgements

This research is funded by the European Research Council as part of the Digital Ludeme Project (ERC Consolidator Grant #771292) led by Cameron Browne at Maastricht University's Department of Data Science and Knowledge Engineering.

## References

1. Abelson, H., diSessa, A.: Turtle Geometry: The Computer as a Medium for Exploring Mathematics. MIT Press, Cambridge (2008)
2. Browne, C.: Modern techniques for ancient games. In: IEEE Conference on Computational Intelligence and Games. pp. 490–497. IEEE Press, Maastricht (2018)
3. Browne, C., Soemers, D.J.N.J., Piette, É., Stephenson, M., Crist, W.: Ludii language reference. https://ludii.games/downloads/LudiiLanguageReference.pdf (2020)
4. Browne, C.B.: A class grammar for general games. In: Advances in Computer Games. Lecture Notes in Computer Science, vol. 10068, pp. 167–182. Leiden (2016)
5. Piette, É., Soemers, D.J.N.J., Stephenson, M., Sironi, C.F., Winands, M.H.M., Browne, C.: Ludii – the ludemic general game system. In: Giacomo, G.D., Catala, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020). Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 411–418. IOS Press (2020)