

XXIII<sup>rd</sup> Board Game Studies Colloquium (BGS), April 2021, Paris

# Everything's a Ludeme



European Research Council

Cameron Browne  
*Digital Ludeme Project*  
Maastricht University





XXIII<sup>rd</sup> Board Game Studies Colloquium (BGS), April 2021, Paris

# (Almost) Everything's a Ludeme



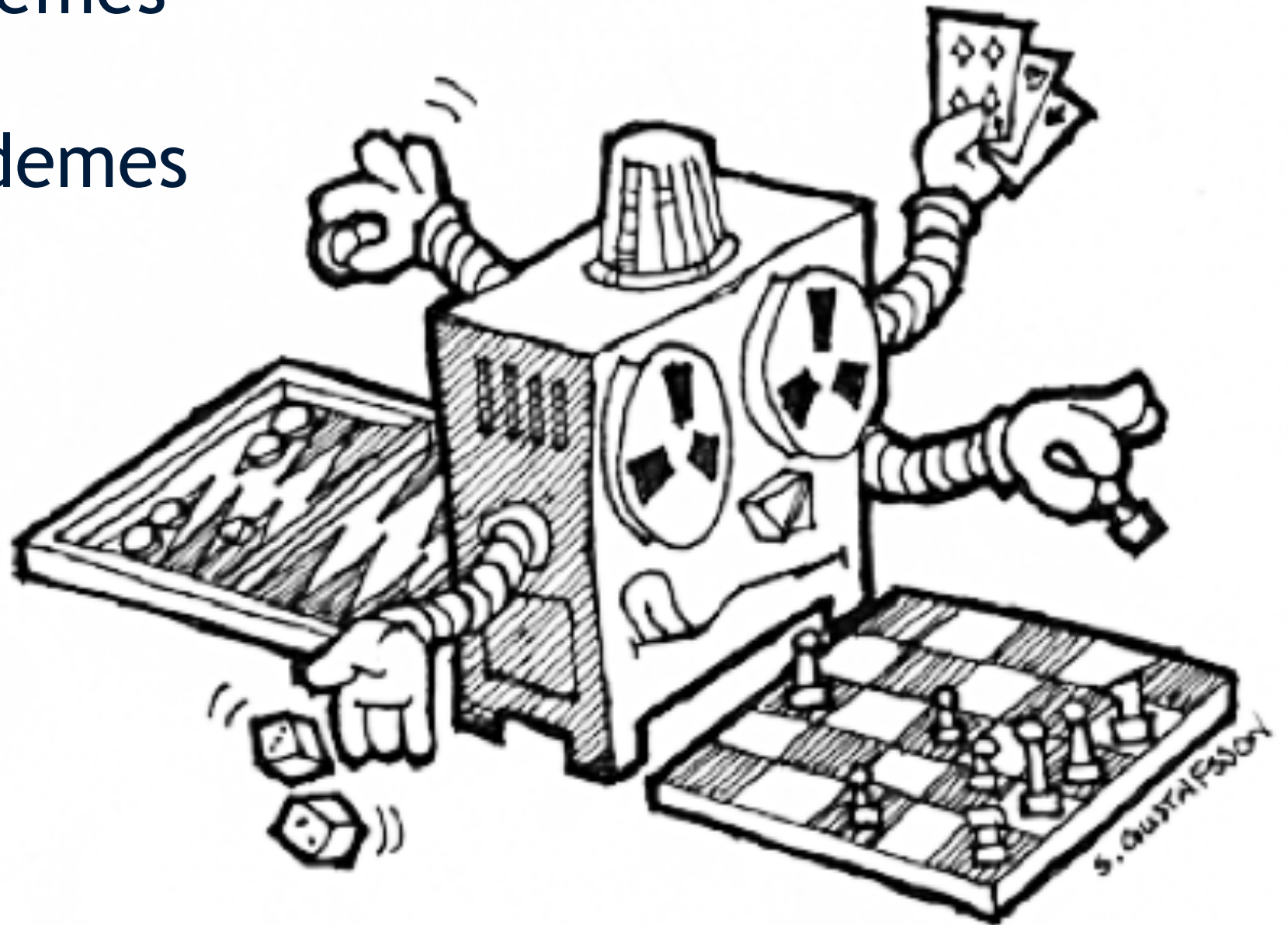
European Research Council

Cameron Browne  
*Digital Ludeme Project*  
Maastricht University



# Overview

- Defining “ludeme”
- Implementing ludemes
- Understanding ludemes

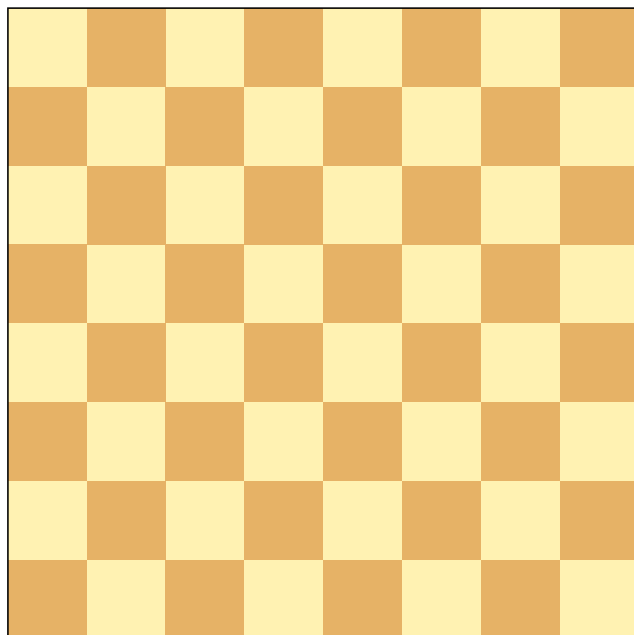


# Defining “Ludeme”

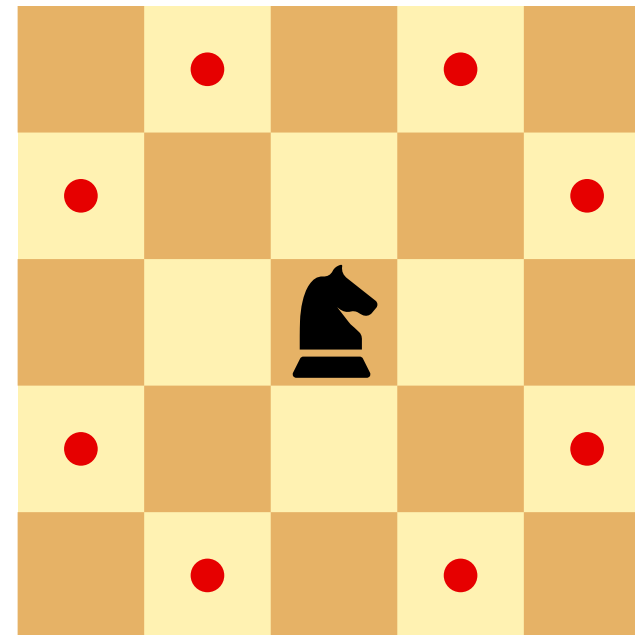
David Parlett (2006) “What’s a Ludeme?”, *BGS*

- “*an element of play, comparable to, but distinct from, a game component or instrument of play*”
- “*ludemic meme*”
- “*pass from one game... to another*”
- “*game elements... are ludemes only if they are contrastive*”

e.g.



Chess board



Knight moves



# Timeline

- 1970** Pierre Berloquin uses *ludeme* in interview  
T. Depaulis in *Foundations of Digital Archaeoludology* (2019)
- 1976** Richard Dawkins defines *meme* in *The Selfish Gene*
- 1977** Alain Borvo uses *ludeme* in *L'alurette, ou le jeu de vache*
- 1990** David Parlett uses *ludeme* in *Oxford Guide to Card Games*
- 2004-09** Ludi
- 2005** Video game designers reinvent *ludeme*
- 2006** David Parlett defines *ludeme* in “What’s a Ludeme?”
- 2018-23** Digital Ludeme Project + Ludii

# Memes

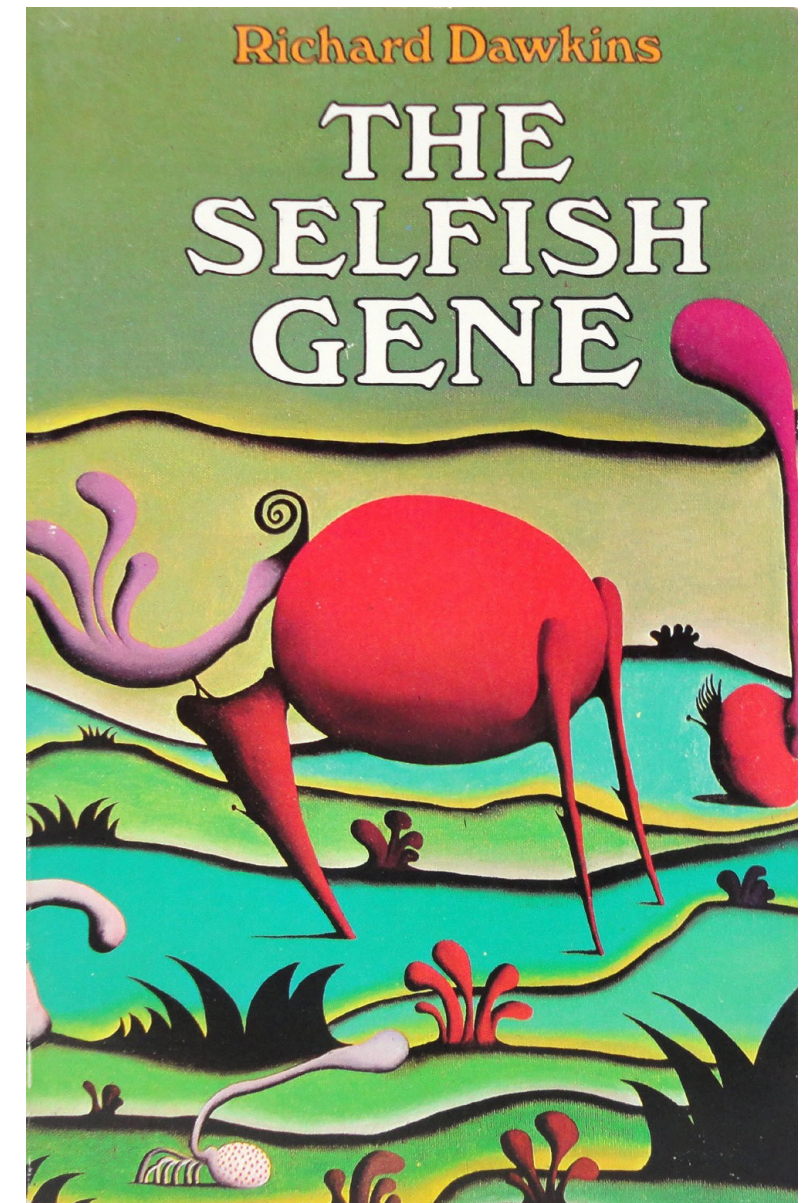
Richard Dawkins (1976) *The Selfish Gene*, pp.189-201

*Meme:*

- *“unit of cultural transmission”*
- *“propagate... via... imitation”*
- *“can be sub-divided into components... separate memes”*

e.g. Beethoven's Ninth Symphony  
Darwin's Theory of evolution

“Ludeme” came first!





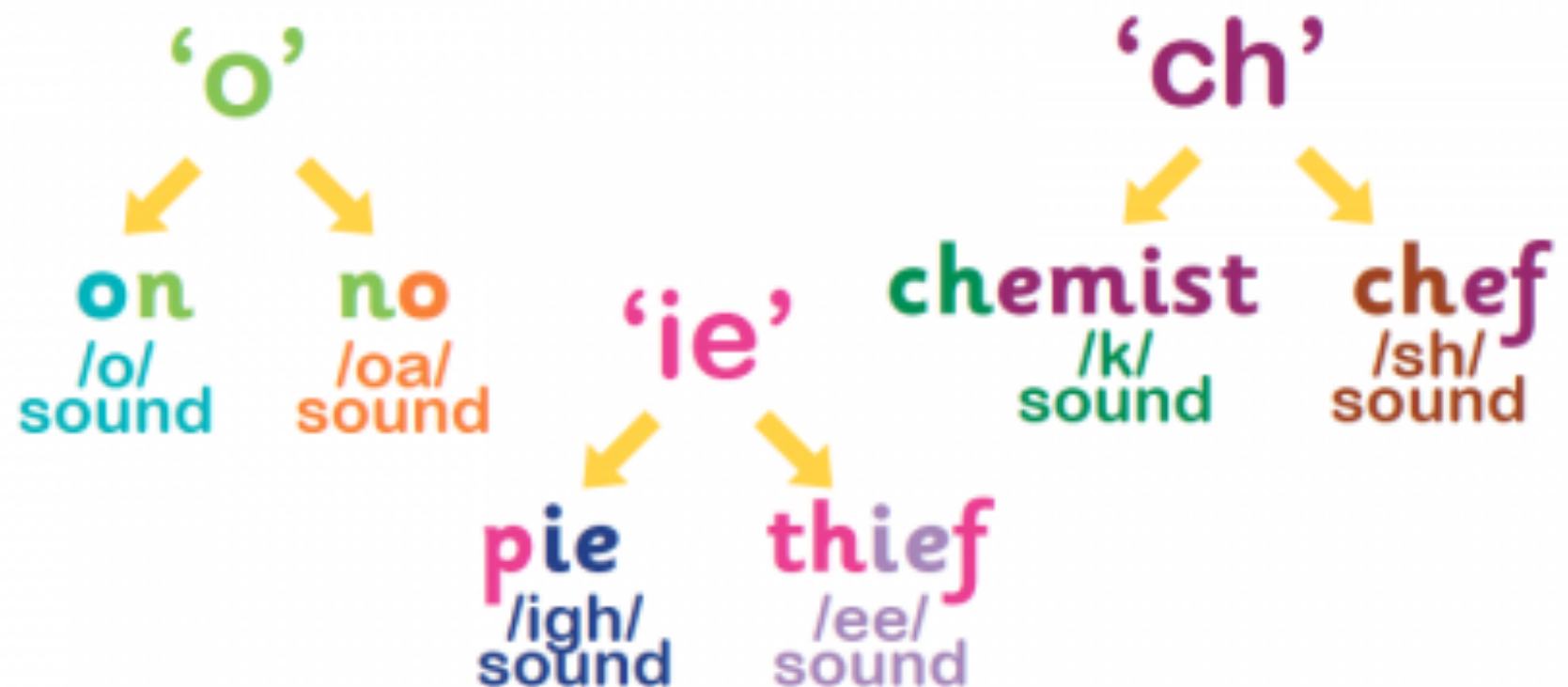
# Other \*emes in Linguistics

Nöth (1995): *Emic units* reduce variant forms to abstract units

- **Phoneme** (1873): Smallest unit of sound in speech
- **Morpheme** (1880): Smallest meaningful unit in a language
- **Grapheme** (1986): Smallest meaningful unit in a writing system

All are:

- Transferable
- Contrastive
- Minimal units



# Video Game Ludemes

Cousins (2005) “Low-Level Game Design”

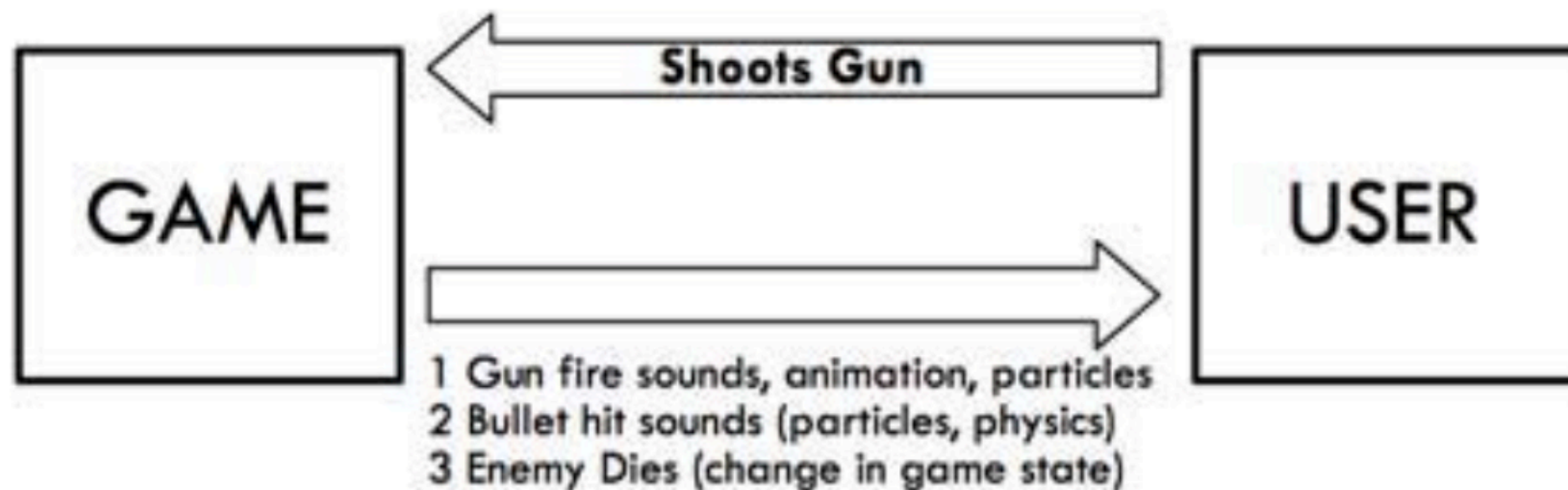
- “Atom” = Smallest loop of interaction

Koster (2005) *A Theory of Fun*

- Ludeme = “Atom”

Bojin (2010) “Ludemes and the Linguistic Turn”

- Ludeme = Smallest loop of engagement





# Two Models

Two basic ways to understand “ludeme”

**1. Memetic Model**

**1. Emic Model**

# Two Models

Two basic ways to understand “ludeme”

1. **Memetic Model**



1. **Emic Model**



Discrete unit



# Two Models

Two basic ways to understand “ludeme”

## 1. Memetic Model



## 1. Emic Model



Discrete unit

Transferable

# Two Models

Two basic ways to understand “ludeme”

## 1. Memetic Model



## 1. Emic Model



Discrete unit

Transferable

Contrastive



# Two Models

Two basic ways to understand “ludeme”

## 1. Memetic Model



## 1. Emic Model



Discrete unit

Transferable

Contrastive

Can sub-divide

# Nested Ludemes

“Hop over adjacent piece”



# Nested Ludemes

“Hop over adjacent piece”

“Hop over adjacent piece to flip it”



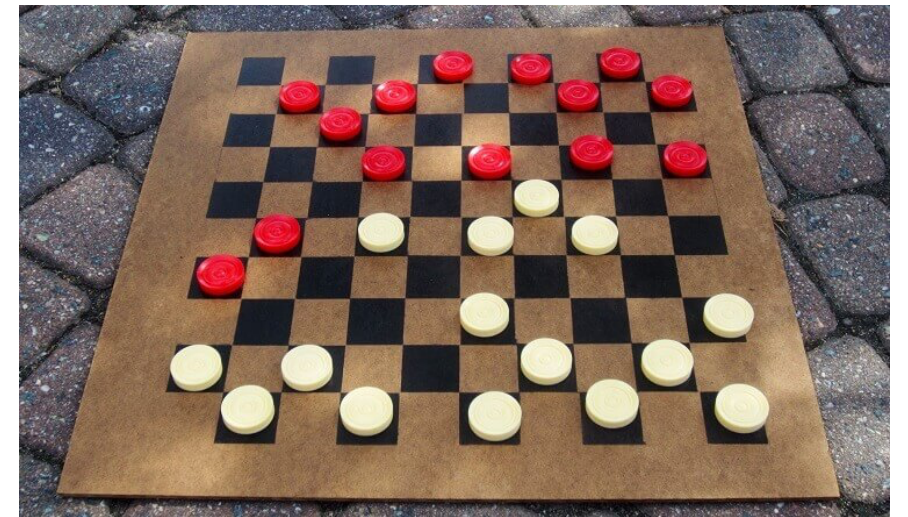


# Nested Ludemes

“Hop over adjacent piece”

“Hop over adjacent piece to flip it”

“Hop over adjacent enemy piece to capture it”



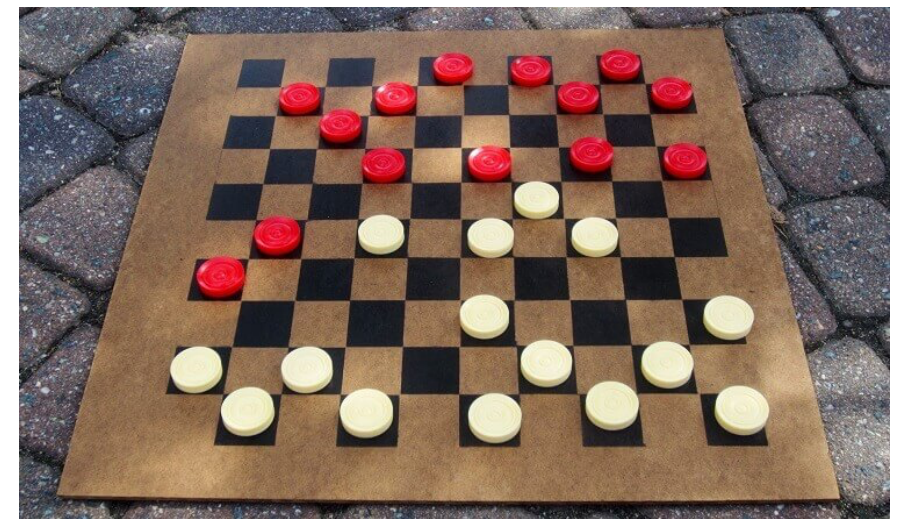


# Nested Ludemes

“Hop over adjacent piece”

“Hop over adjacent piece to flip it”

“Hop over adjacent enemy piece to capture it”



Nested ludemes:

- Provide contrast at different levels
- Qualifier within effect within move within game

# Granularity

## 1. Atomic ludemes

- Minimal units
- Can't be further sub-divided

## 2. Compound ludemes

- Ludeme structures
- “Ludemeplexes”
- Built from simpler ludemes

Can embed (sub)ludemes in ludemes

- In a contrastive way



# Ludi

## Ludi program

- Ph.D. thesis (2004-9)
- “Ludemic approach”
- Model simple board games
- Evolve rule sets

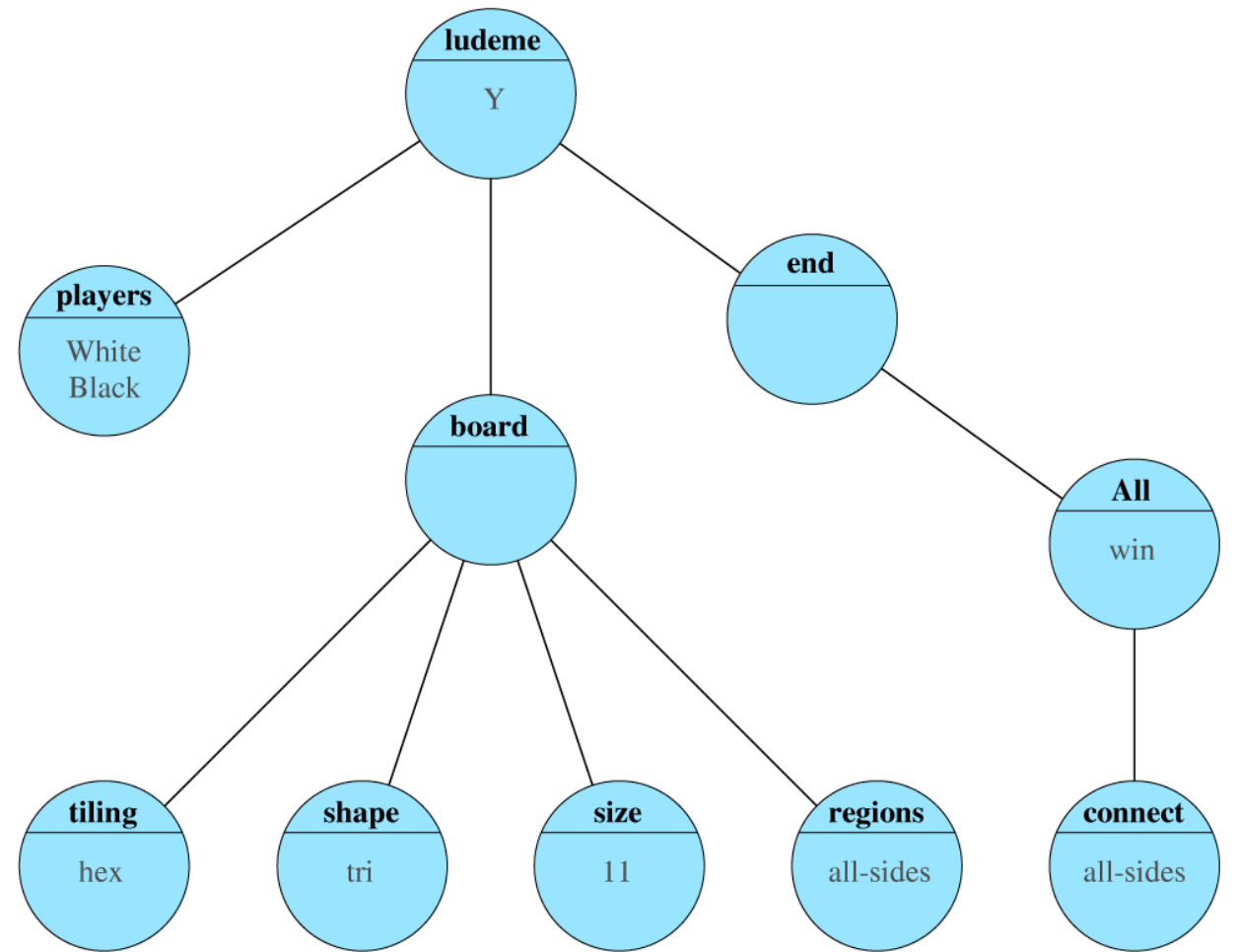
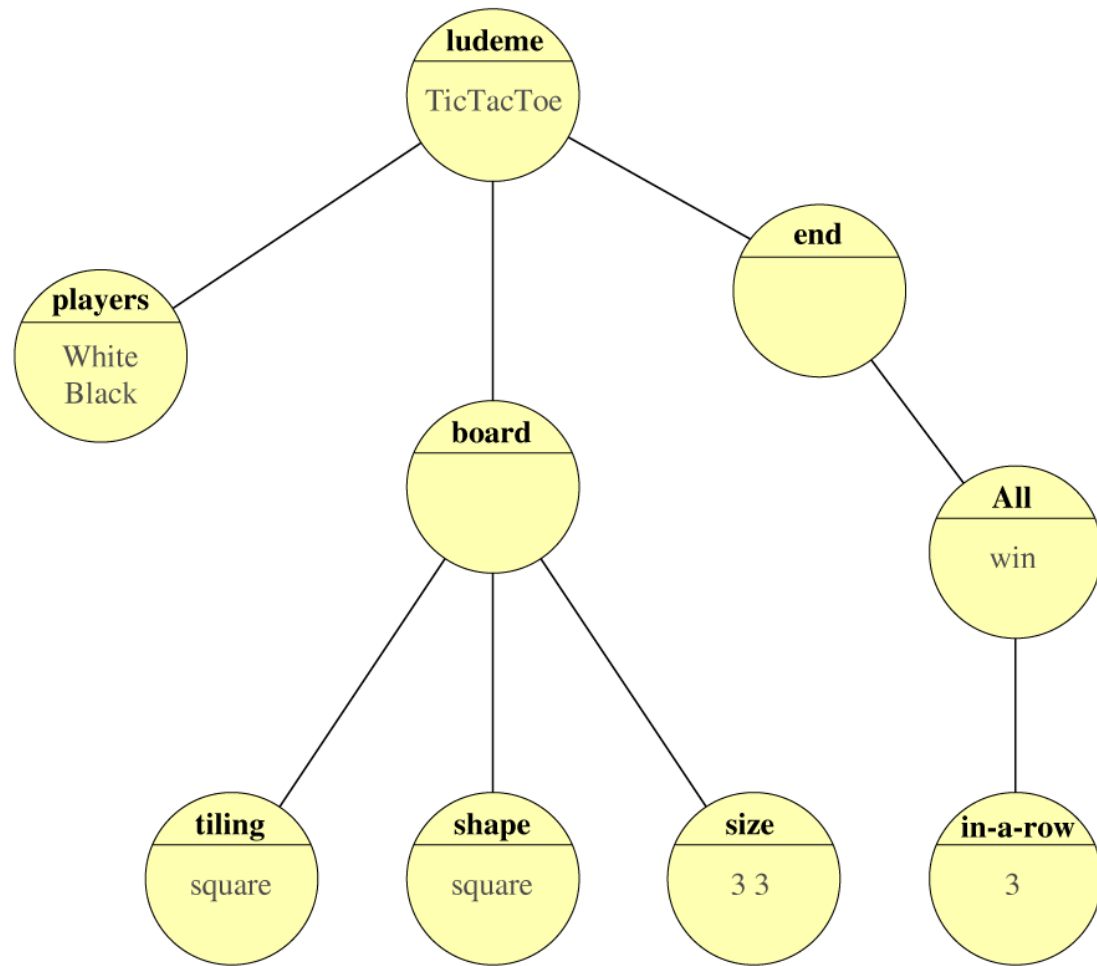
```
(game Tic-Tac-Toe
  (board
    (tiling square)
    (size 3 3)
  )
  (win (in-a-row 3))
)
```

Found interesting new games  
e.g. Yavalath and Pentalath



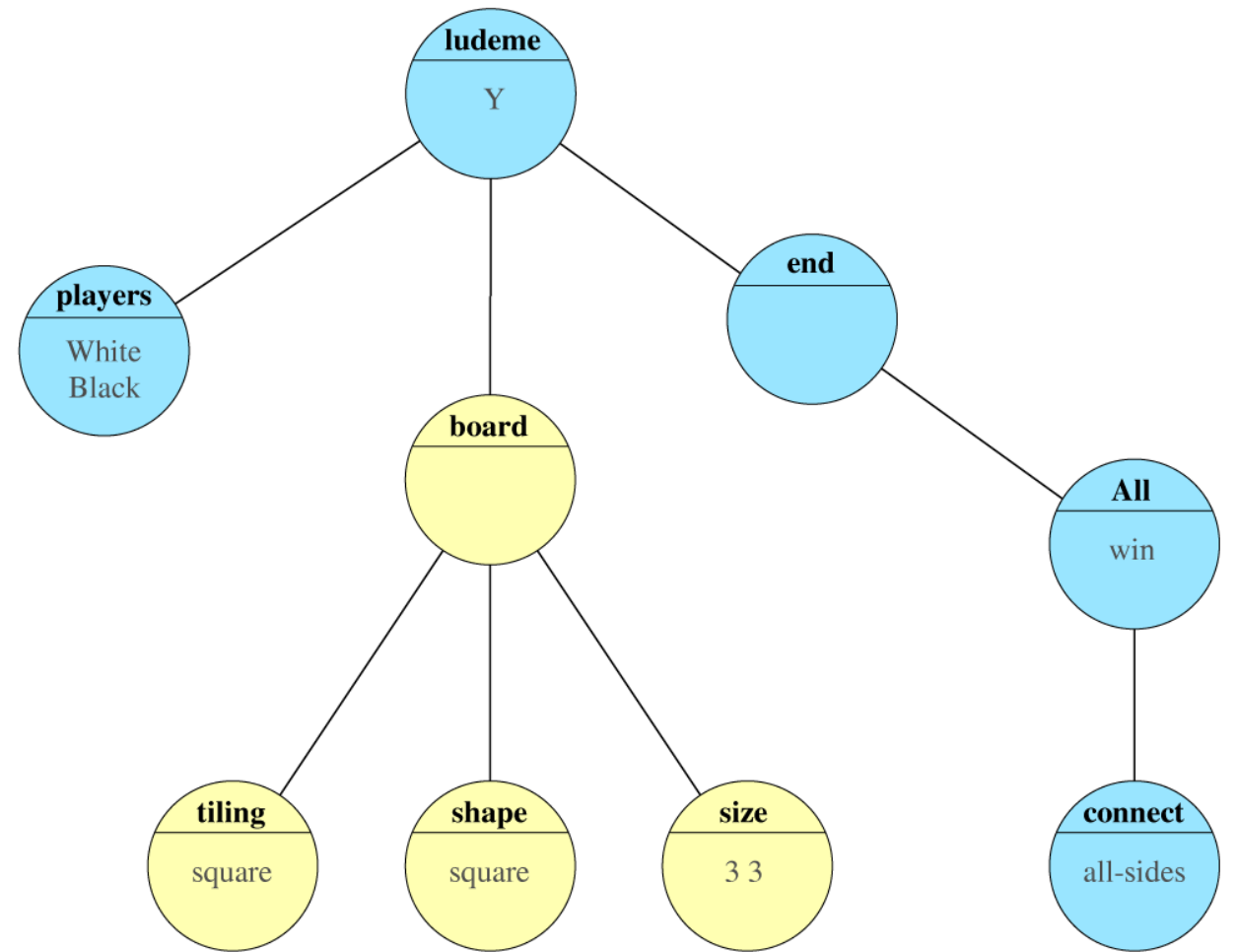
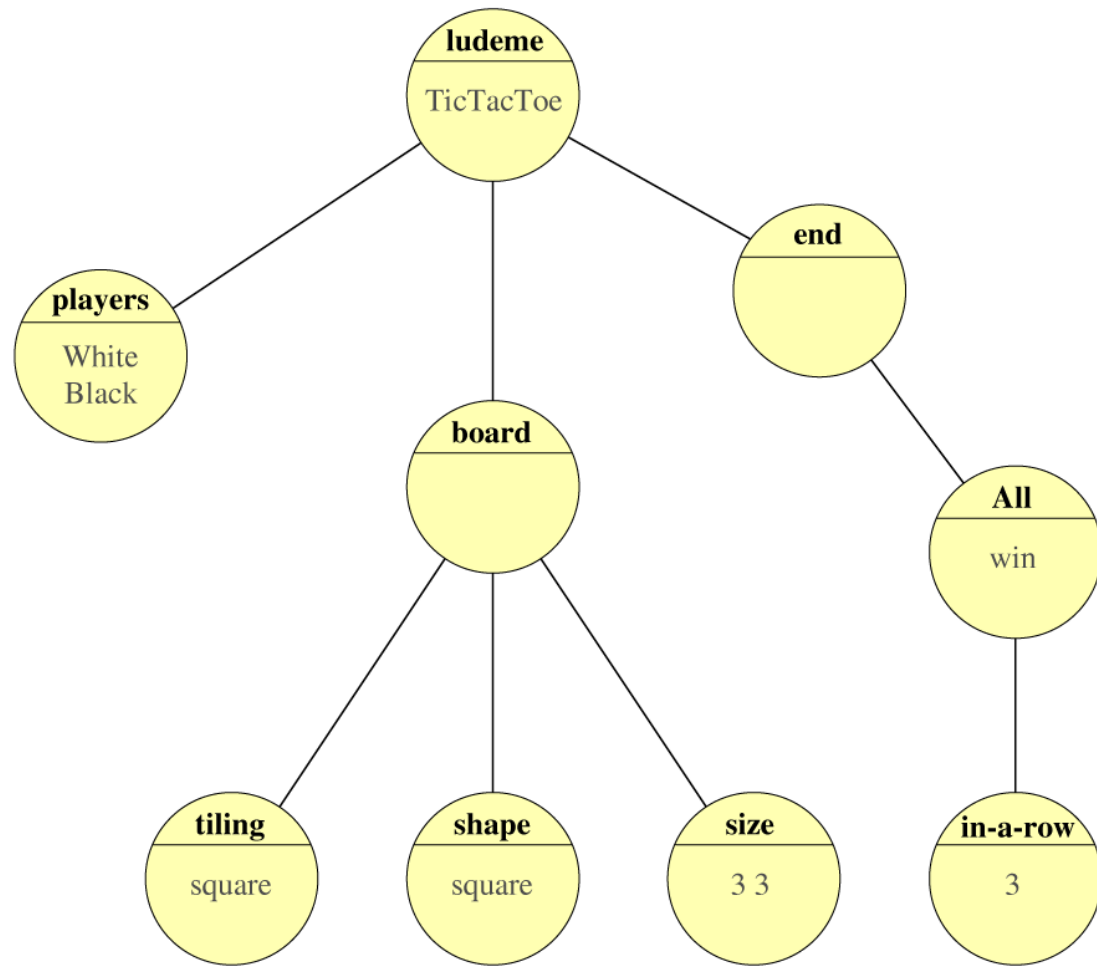
# Evolving Games

## 1. Select two games



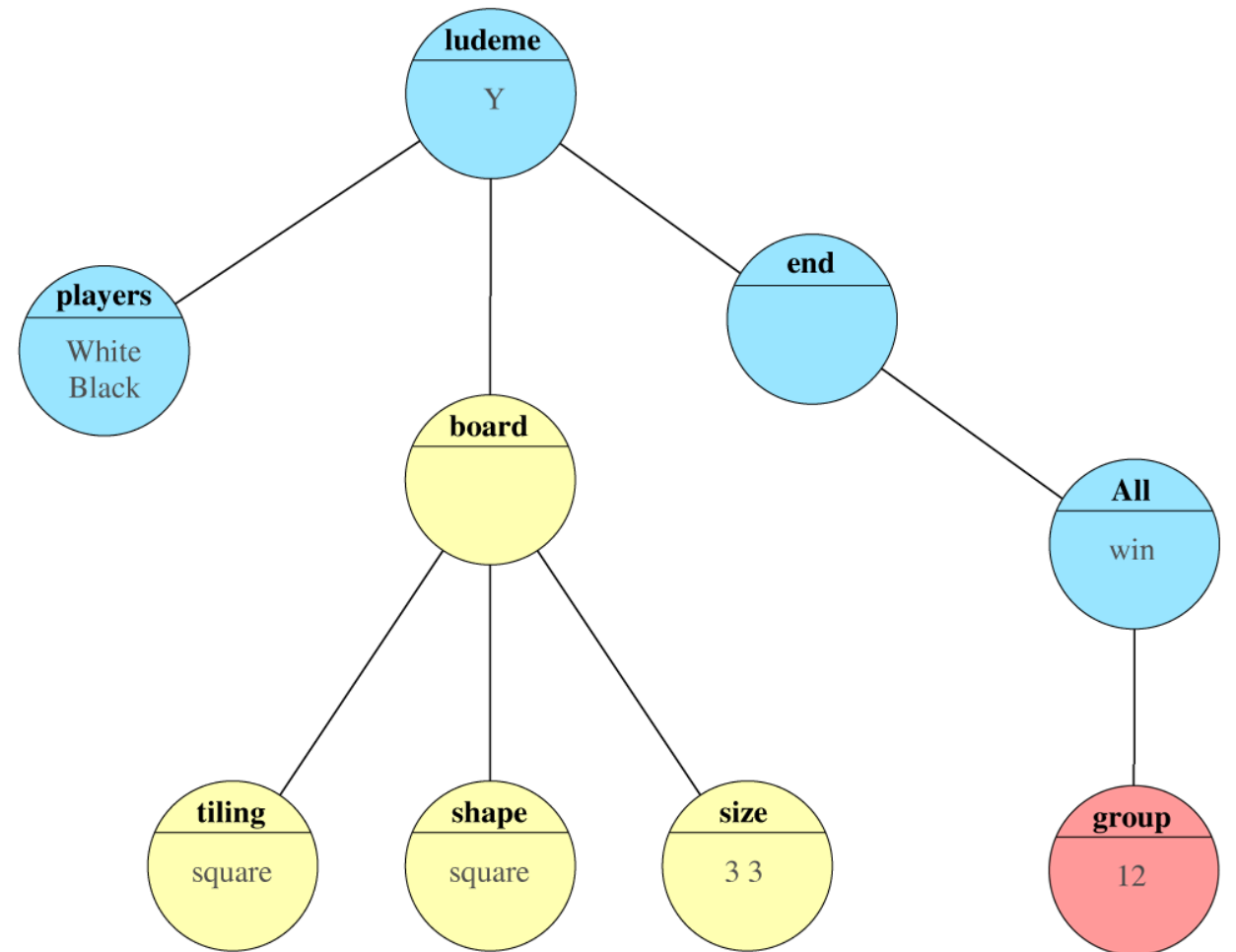
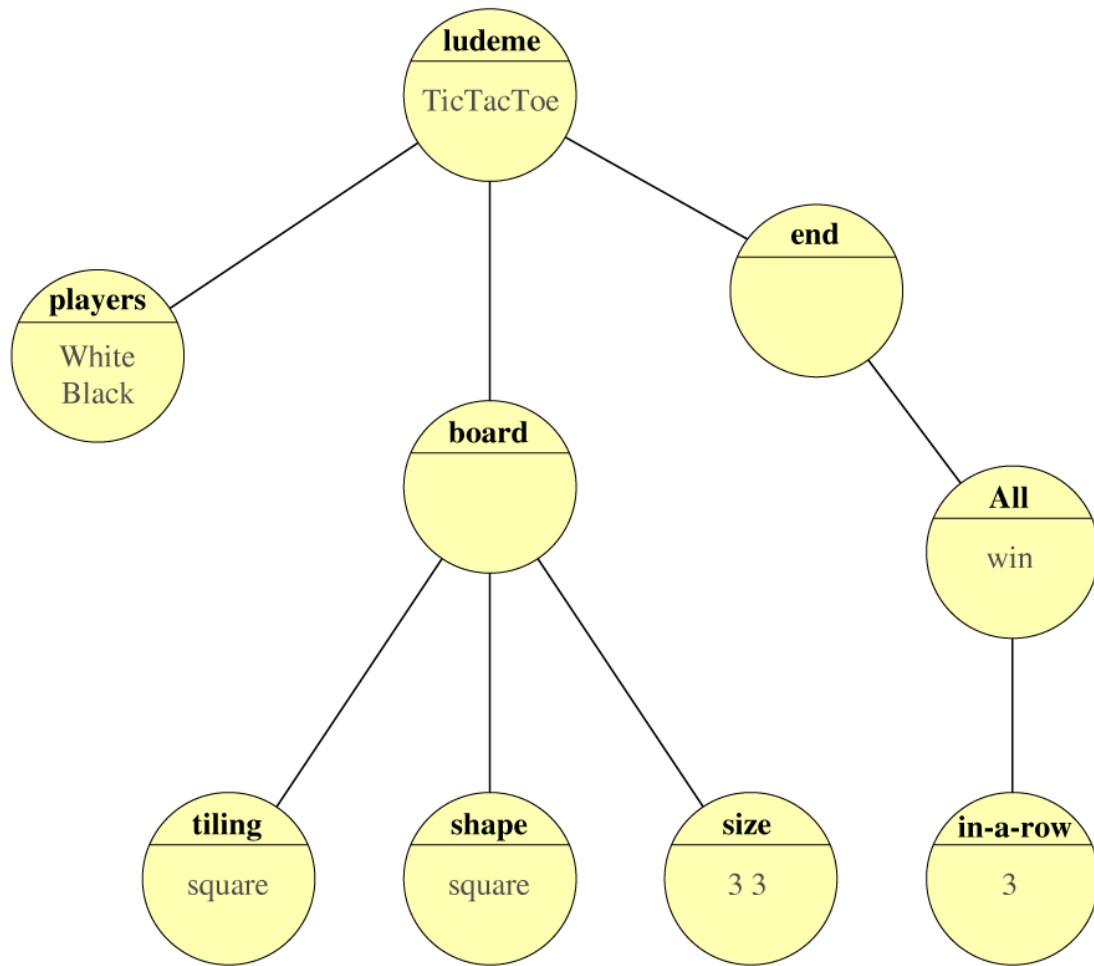
# Evolving Games

## 2. Cross over sub-trees



# Evolving Games

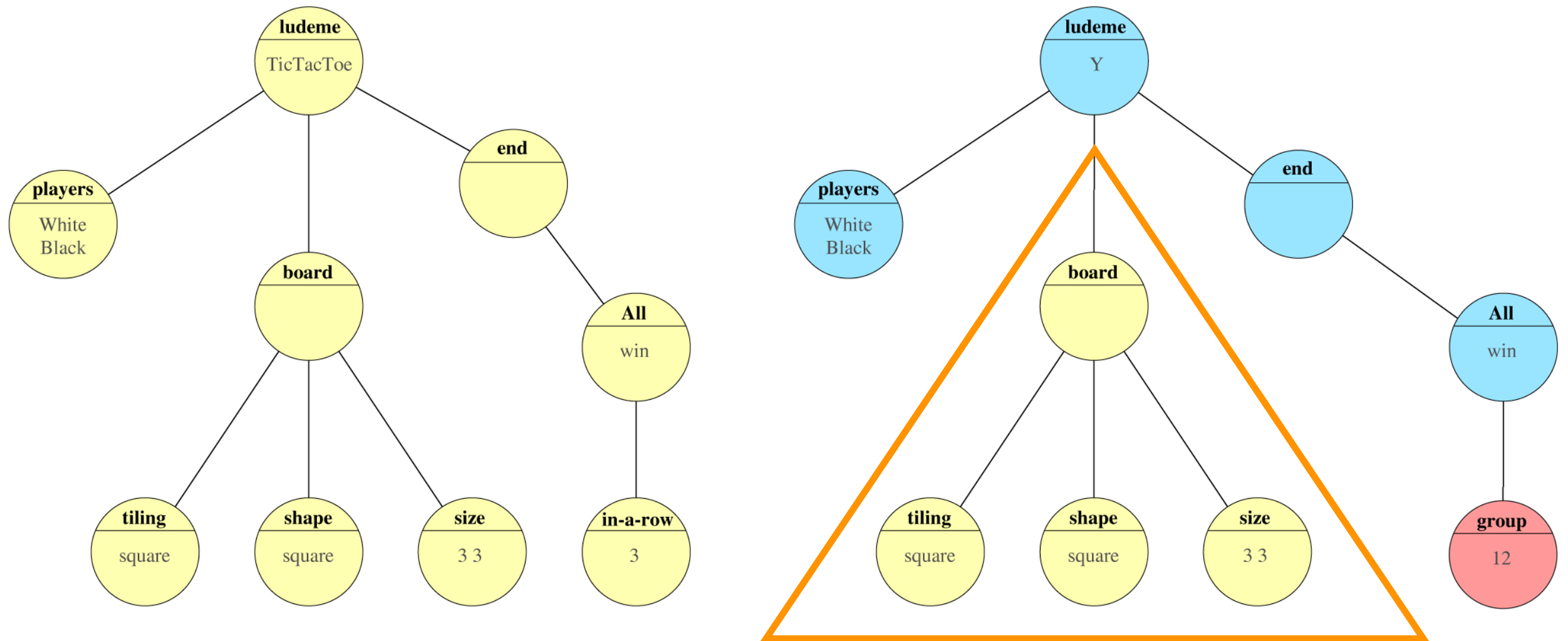
## 3. Mutate nodes





# Evolving Games

Sub-trees transferable as discrete units



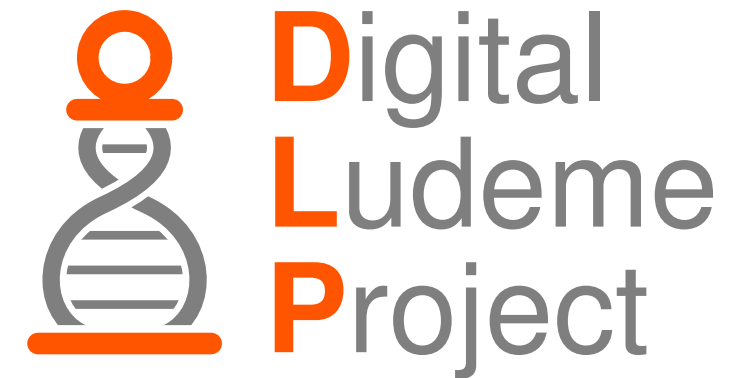
Every node and sub-tree is a potential ludeme

- If contrastive within context

# Ludii

## Digital Ludeme Project (2018-2023)

- Model 1,000 key historical games
- Map relationships through ludemes



## Ludii program

- General game system
- Similar “ludemic” approach
- Bigger and better!

```
(game "Tic-Tac-Toe"
  (players 2)
  (equipment {
    (board (square 3))
    (piece "Disc" P1)
    (piece "Cross" P2)
  })
  (rules
    (play (move Add
      (to (sites Empty))))
    (end (if (is Line 3)
      (result Mover Win)))
  )
)
```

# Ludii Language

Game descriptions composed of symbols:

## 1. Class names

Lowercase

566 Java classes

## 2. Attributes

Uppercase

652 enum constants

## 3. Values

Numbers, strings,  
True/False, ...

```
(game "Tic-Tac-Toe"  
  (players 2)  
  (equipment {  
    (board (square 3))  
    (piece "Disc" P1)  
    (piece "Cross" P2)  
  })  
  (rules  
    (play (move Add  
      (to (sites Empty))))  
    (end  
      (if (is Line 3)  
        (result Mover Win))  
    )  
  )  
)
```

# Example: Queens

Game of the Amazons (Queens don't capture):

```
(piece "Queen" Each
  (move Slide)
)
```

Chess (Queens capture):

```
(piece "Queen" Each
  (move Slide
    (to
      if: (is Enemy (who at: (to)))
      (apply (remove (to)))
    )
  )
)
```



# Example: Queens


Game of the Amazons (Queens don't capture):

```
(piece "Queen" Each  
  (move Slide)  
)
```

Chess (Queens capture):

```
(piece "Queen" Each  
  (move Slide  
    (to  
      if: (is Enemy (who at: (to)))  
      (apply (remove (to)))  
    )  
  )  
)
```

Expression  
Contrastive  
Compound ludeme



# Example: Queens

Game of the Amazons (Queens don't capture):

```
(piece "Queen" Each  
      (move Slide)  
)
```

Chess (Queens capture):

```
(define "CaptureTo"  
  (to if: (is Enemy (who at: (to))) (apply (remove (to))))))  
  
(piece "Queen" Each (move Slide "CaptureTo"))
```

Move into a “define”

- Wraps expression into discrete unit
- Gives it a name

# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if:(is Enemy (who at:(to))) (apply (remove (to))))))
```

```
(piece "Queen" Each (move Slide "CaptureTo"))  
(piece "Bishop" Each (move Slide Diagonal "CaptureTo"))  
(piece "Rook" Each (move Slide Orthogonal "CaptureTo"))
```

# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if: (is Enemy (who at: (to))) (apply (remove (to))))))
```

```
(piece "Queen" Each (move Slide "CaptureTo"))  
(piece "Bishop" Each (move Slide Diagonal "CaptureTo"))  
(piece "Rook" Each (move Slide Orthogonal "CaptureTo"))
```

↑  
Attributes  
Contrastive  
Atomic ludemes

Dawkins' "unit-memes"



# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if:(is Enemy (who at:(to))) (apply (remove (to))))))
```

```
(piece "Queen" Each (move Slide "CaptureTo"))  
(piece "Bishop" Each (move Slide Diagonal "CaptureTo"))  
(piece "Rook" Each (move Slide Orthogonal "CaptureTo"))
```



Attributes

Contrastive

Atomic ludemes

A lot of shared symbols...

Wrap into another define

# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if:(is Enemy (who at:(to))) (apply (remove (to))))))
```

```
(define "Slider"  
  (piece #1 Each (move Slide #2 "CaptureTo")))
```

```
("Slider" "Queen" Adjacent)  
("Slider" "Bishop" Diagonal)  
("Slider" "Rook" Orthogonal)
```

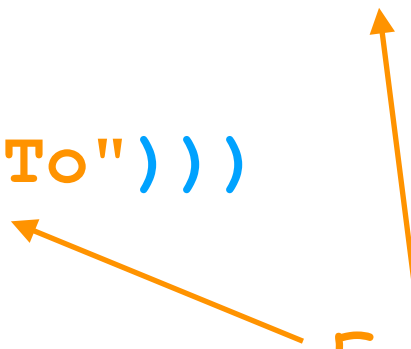
Parameterised (nested) defines

- Combine elements from different points
- Can pass expressions as parameters

# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if: (is Enemy (who at: (to))) (apply (remove (to))))))  
  
(define "Slider"  
  (piece #1 Each (move Slide #2 "CaptureTo")))  
  
("Slider" "Queen" Adjacent)  
("Slider" "Bishop" Diagonal)  
("Slider" "Rook" Orthogonal)
```



Expressions  
(compound)

# Example: Chess Pieces

Chess pieces:

```
(define "CaptureTo"  
  (to if:(is Enemy (who at:(to))) (apply (remove (to))))))  
  
(define "Slider"  
  (piece #1 Each (move Slide #2 "CaptureTo")))  
  
("Slider" "Queen" Adjacent)  
("Slider" "Bishop" Diagonal)  
("Slider" "Rook" Orthogonal)
```

Attributes  
(atomic)

Classes  
(atomic)

Expressions  
(compound)

Ludemes can be:

- **Symbols** (atomic)
- **Expressions** (compound)

# Values are not Ludemes

## Values

- Numbers, strings, True/False, etc.
- Are not ludemes in themselves
- Only meaningful in the context of an expression

```
(square 8)
```

```
(square 9)
```

```
(is Line 3)
```

```
(is Line 5)
```

```
(= (what at: (to)) (id "Pawn" Enemy) )
```

```
(= (what at: (to)) (id "Rook" Enemy) )
```



# Where are the Ludemes in Ludii?

## 1. Semantic

- Java Code
- Implementation

```
public class Is extends Ludeme
{
    public static BooleanFunction
    construct
    (
        final IsLineType isType,
        final IntFunction length
    )
    {
        // Java code
    }
}
```

# Where are the Ludemes in Ludii?

## 1. Semantic

- Java Code
- Implementation

```
public class Is extends Ludeme
{
    public static BooleanFunction
    construct
    (
        final IsLineType isType,
        final IntFunction length
    )
    {
        // Java code
    }
}
```

## 2. Syntactic

- Grammar
- Rules + clauses

`<is> ::= (is Line <int>)`

# Where are the Ludemes in Ludii?

## 1. Semantic

- Java Code
- Implementation

```
public class Is extends Ludeme
{
    public static BooleanFunction
    construct
    (
        final IsLineType isType,
        final IntFunction length
    )
    {
        // Java code
    }
}
```

## 2. Syntactic

- Grammar
- Rules + clauses

`<is> ::= (is Line <int>)`

## 3. Symbolic

- Descriptions
- Symbols

`(is Line 3)`

# Where are the Ludemes in Ludii?

## 1. Semantic

- Java Code
- Implementation

```
public class Is extends Ludeme
{
    public static BooleanFunction
    construct
    (
        final IsLineType isType,
        final IntFunction length
    )
    {
        // Java code
    }
}
```

## 2. Syntactic

- Grammar
- Rules + clauses

`<is> ::= (is Line <int>)`

Class (atomic)

Attribute (atomic)

## 3. Symbolic

- Descriptions
- Symbols

`(is Line 3)`

Expression (compound)

# Formal Definition

$R$	Set of known <code>&lt;rule&gt;</code> in the grammar
$A$	Set of known <code>Attribute</code> in the grammar
$D$	Set of known game descriptions (*.lud)
$d_a, d_b, \dots$	Specific game descriptions (complete or fragment)
$L$	Set of potential ludemes



# Formal Definition

$l \in L$       Ludeme  $l$  is in the set of potential ludemes  $L$

## Discrete Unit

$l \in R$        $l$  is a known **class** in  $R$

$l \in A$        $l$  is a known **Attribute** in  $A$

$(\exists d_n)[l \in d_n]$        $l$  is an **expression** in known description  $d_n$

## Transfer

$d_x \otimes l = d_y$       Applying  $l$  to description  $d_x$  gives  $d_y$

## Contrast

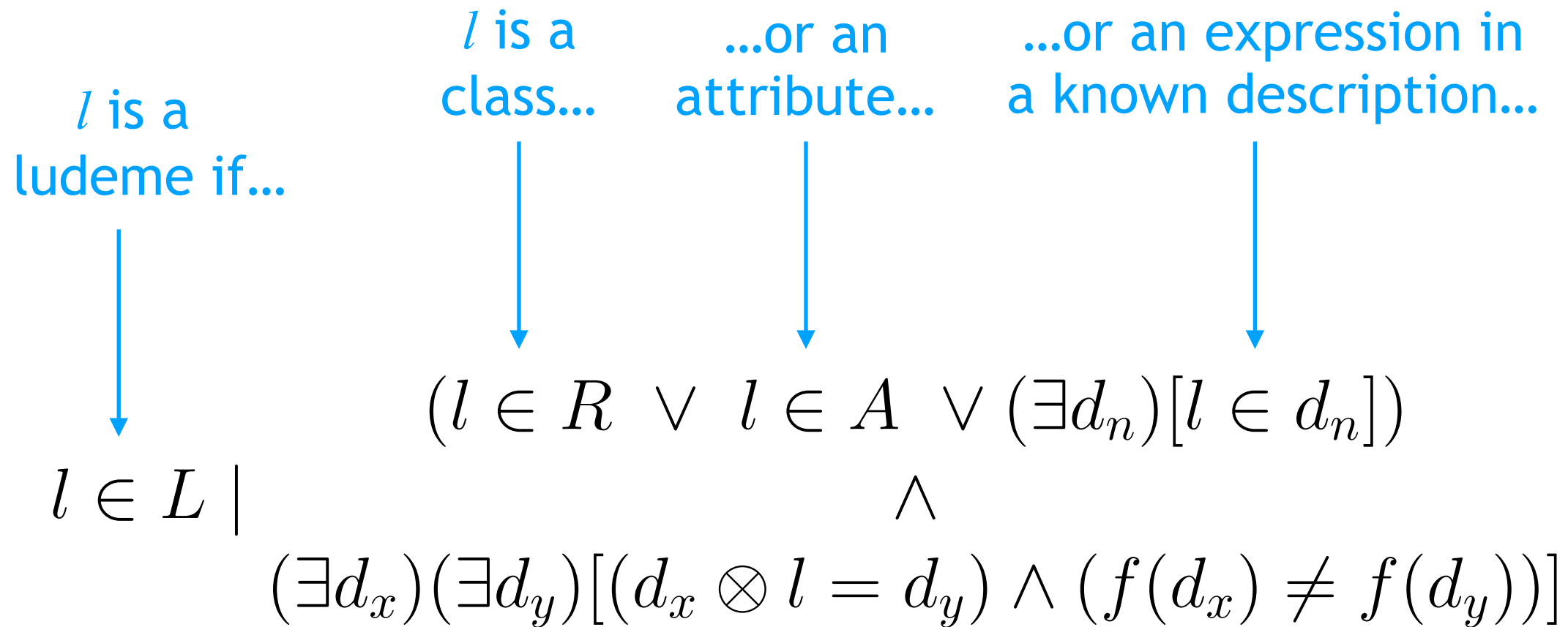
$f(d_x)$       Function of play defines legal moves for  $d_x$

$f(d_x) \neq f(d_y)$        $d_x$  and  $d_y$  are functionally different

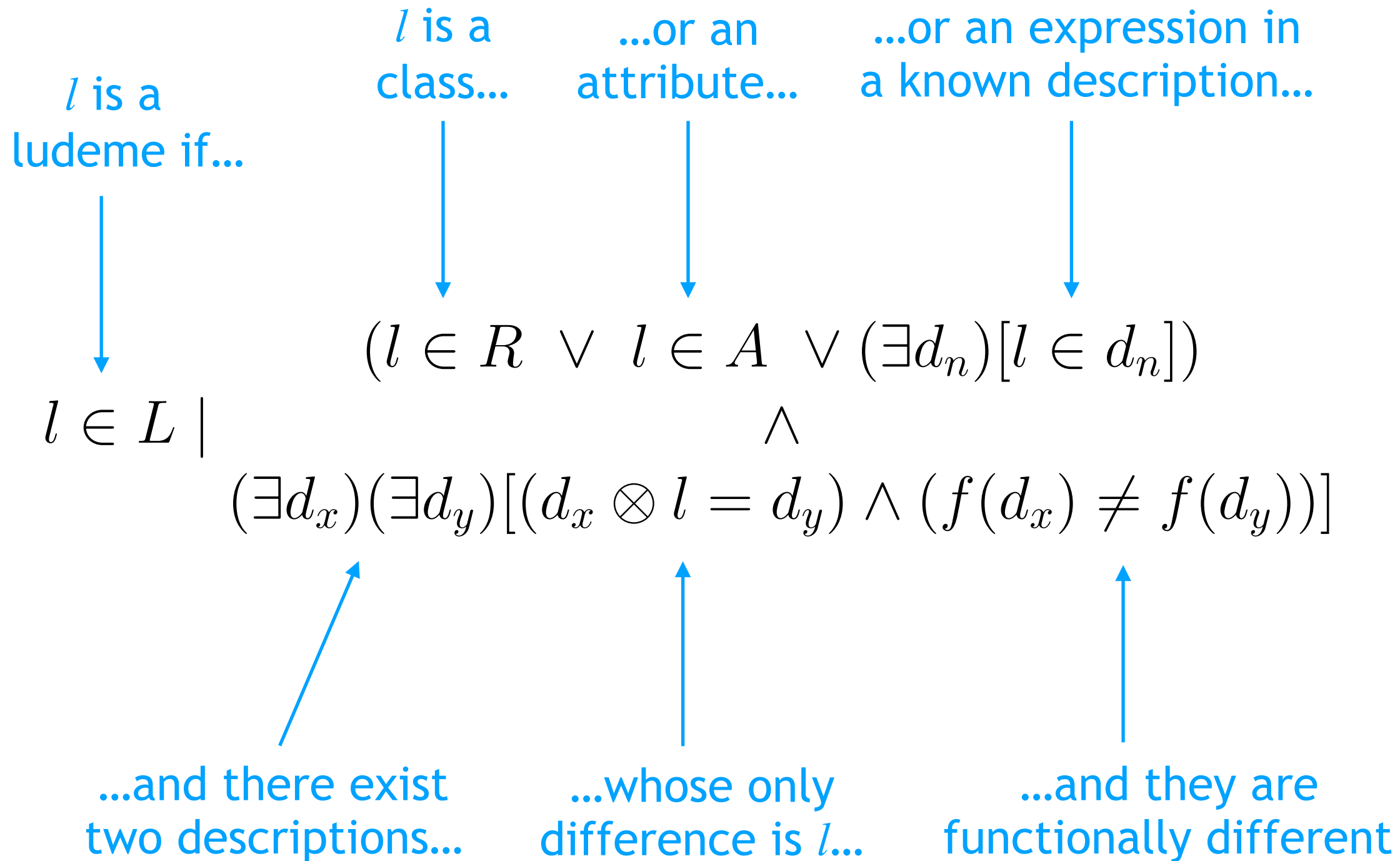
# Formal Definition

$$l \in L \mid \begin{array}{l} (l \in R \vee l \in A \vee (\exists d_n)[l \in d_n]) \\ \wedge \\ (\exists d_x)(\exists d_y)[(d_x \otimes l = d_y) \wedge (f(d_x) \neq f(d_y))] \end{array}$$

# Formal Definition



# Formal Definition



# Informal Definition

A ludeme is a:

- Discrete unit of information (atomic or compound)
- Can be transferred between games
- Changes the function of a game

Very similar to Parlett's 2006 definition

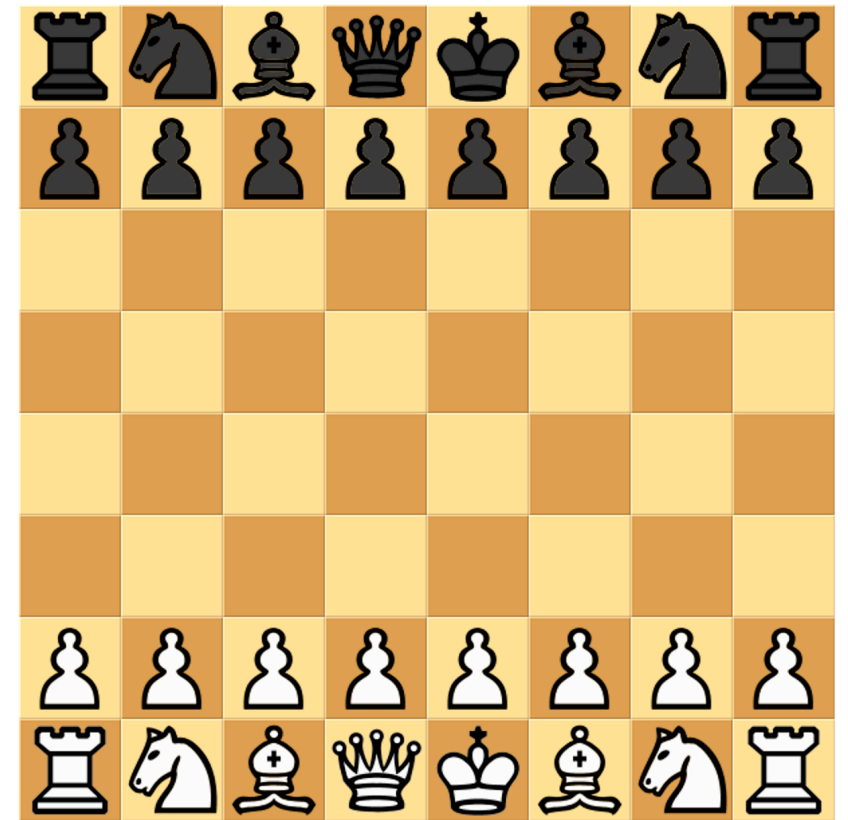
- Same conclusion through different routes
- One obvious casualty...



# What's Not a Ludeme?

Back to “Chess board”...

```
(game
  (board (square 8) )
)
(metadata
  (graphics
    (board Style Chess)
  )
)
```



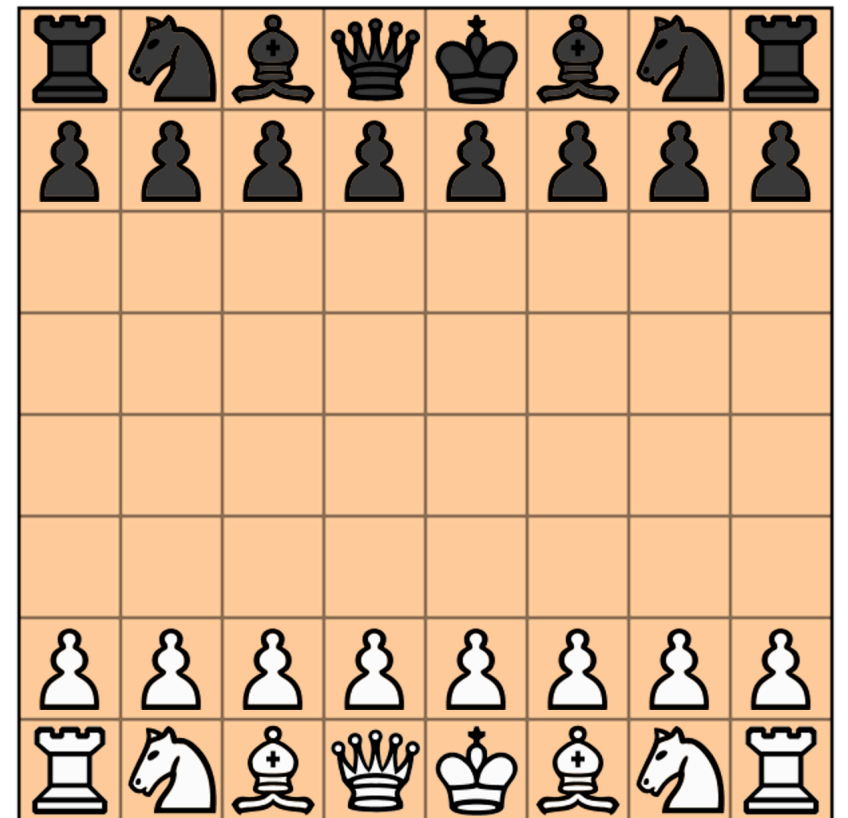
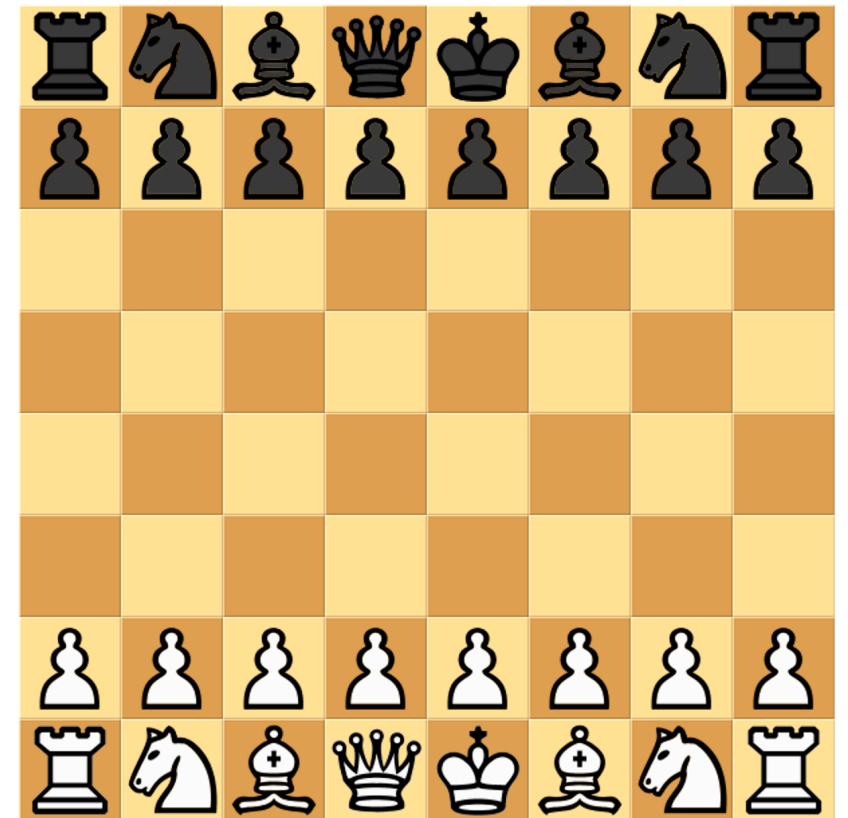
# What's Not a Ludeme?

Back to “Chess board”...

```
(game
  (board (square 8) )
  (metadata
    (graphics
      (board Style Chess)
    )
  )
)
```

Checker not part of game logic

- Only visually contrastive
- More a meme than a ludeme



# Can Games Be Ludemes?

e.g. Chameleon

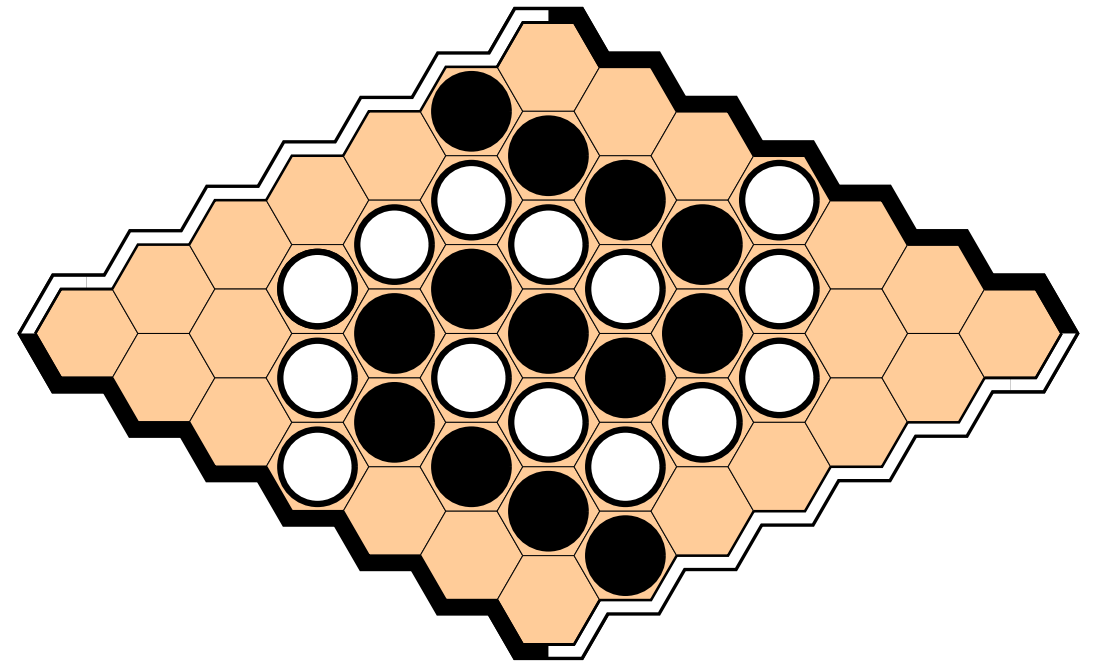
Invented by:

- Randy Cox (USA) 5 Nov. 2003
- Bill Taylor (NZ) Late Nov. 2003

+ Can be transferred as discrete unit

- Exists within context of Hex

Not really a ludeme itself



Played as per Hex, except:  
1. Play either colour per turn  
2. Connect own sides with path of either colour

# Games As Sub-Components

## Backgammon

- Played in Tavli cycle:
  1. Portes (Backgammon)
  2. Plakoto
  3. Fevga/Moultezim



# Games As Sub-Components

## Backgammon

- Played in Tavli cycle:
  1. Portes (Backgammon)
  2. Plakoto
  3. Fevga/Moultezim



## Rock-Paper-Scissors

- Game as pre-game decider

## Games are:

- Transferable as discrete units
- Contrastive within context





# Games As Sub-Components

In Ludii

- Can import any game as a sub-game in a match

```
(match "Tavli"
  (players 2)
  (games {
    (subgame "Portes" next:1 result:(score Mover))
    (subgame "Plakoto" next:2 result:(score Mover))
    (subgame "Fevga" next:0 result:(score Mover))
  })
  (end {
    (if (>= (matchScore P1) 5) (result P1 Win))
    (if (>= (matchScore P2) 5) (result P2 Win))
  })
)
```

# Games as Memes

Chess metaphor for

- Strategic thought
- Mental acumen
- Making sacrifices



Corporate logos

Avatars/icons/artworks

- Metaphor for “games” or “play”

“<Sport> is Chess at 100 miles an hour”

# Games as Memes

## *The Seventh Seal*

- The Knight challenges Death to Chess
- Delay own death

## Chess as a metaphor

- Battle of wits

## Chess as a meme

- Not a ludeme!



# Thanks To

Cesco Reale

Stephen Tavener

Ludii team:

- Eric Piette
- Matthew Stephenson
- Walter Crist
- Dennis Soemers

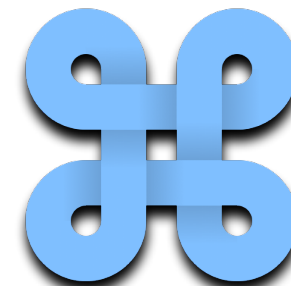


European Research Council



Digital  
Ludeme  
Project

<http://ludeme.eu>



<http://ludii.games>

# Ludemes are Contrastive

Ludemes should affect play

With all other factors held constant, changing a ludeme should change the function of the game.

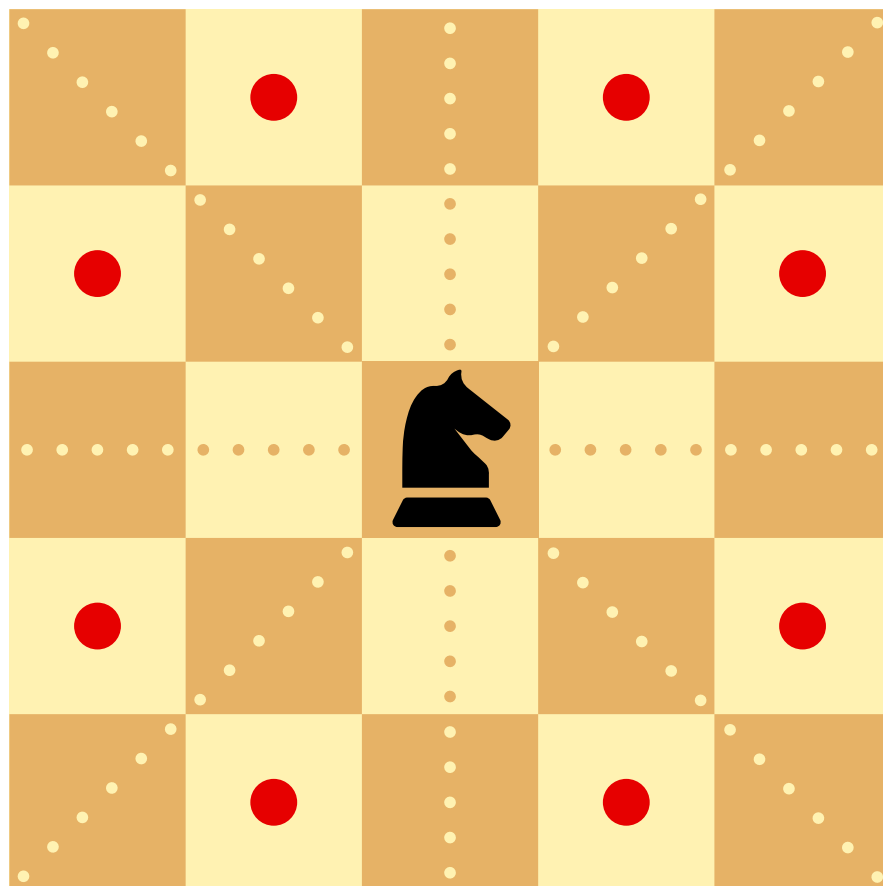
Different descriptions giving same behaviour

- Same ludeme

# Context Is Important

Knight moves can be:

1. “L” shaped walk {FFL / FFR} x 4
2. Closest non-adjacent cell of different colour
3. Closest cell not in orthogonal or diagonal line
4. Closest cell of different colour not in orthogonal line



## Canalisation

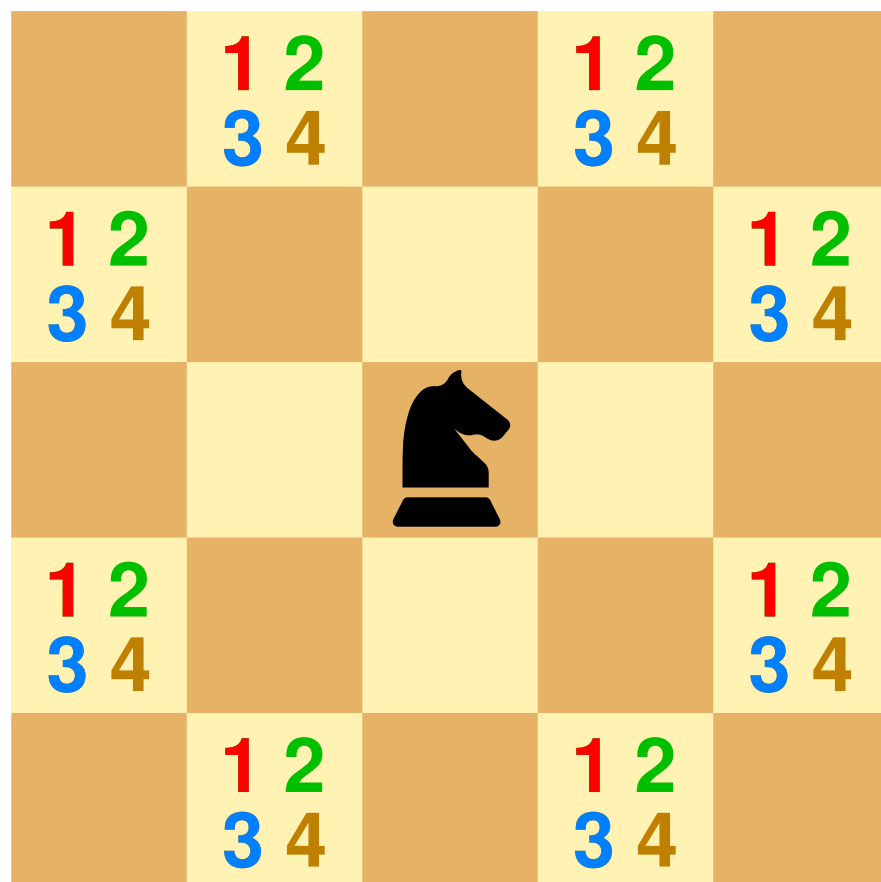
- Different genotypes (rules)
- Same behaviour



# Context Is Important

Knight moves can be:

1. “L” shaped walk {FFL / FFR} x 4
2. Closest non-adjacent cell of different colour
3. Closest cell not in orthogonal or diagonal line
4. Closest cell of different colour not in orthogonal line



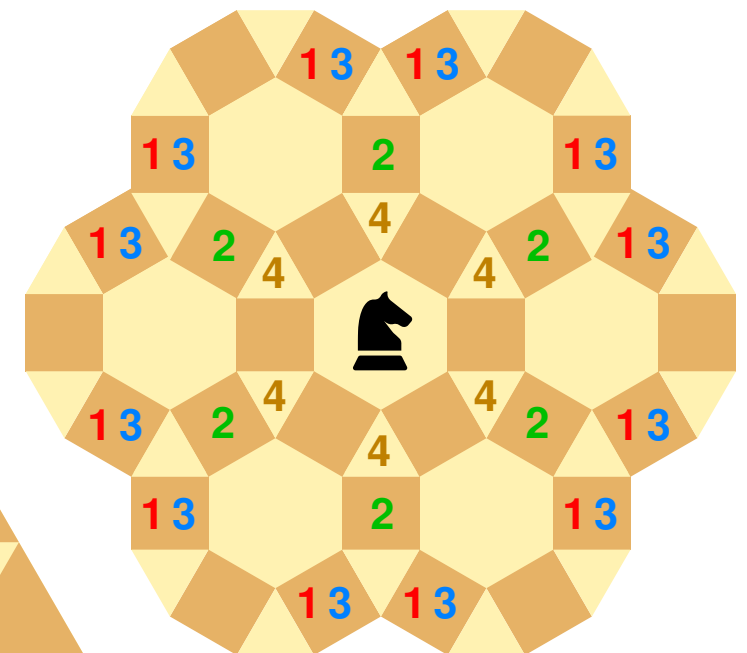
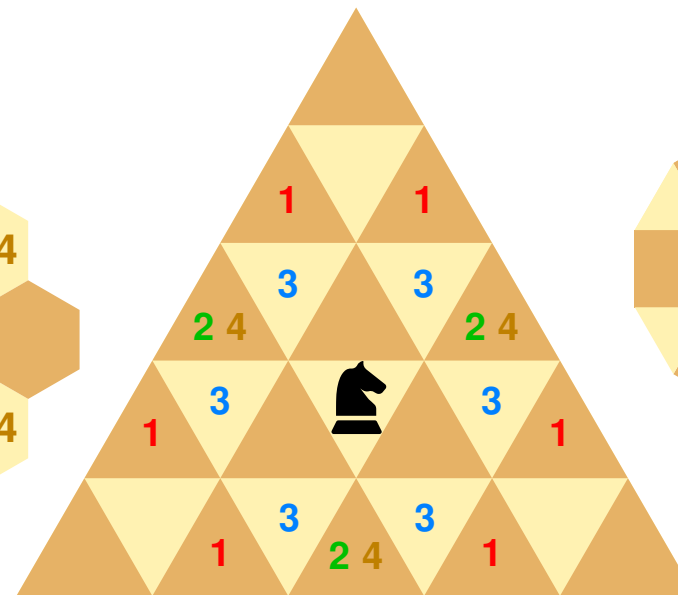
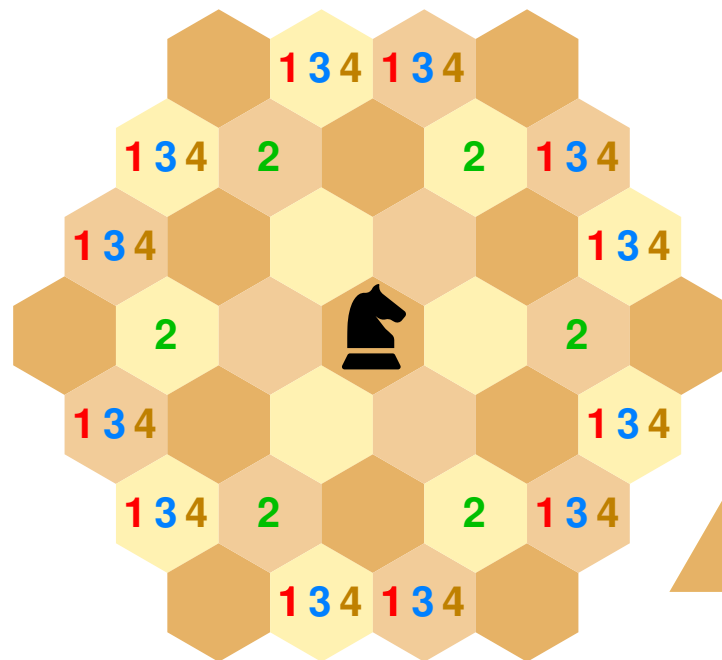
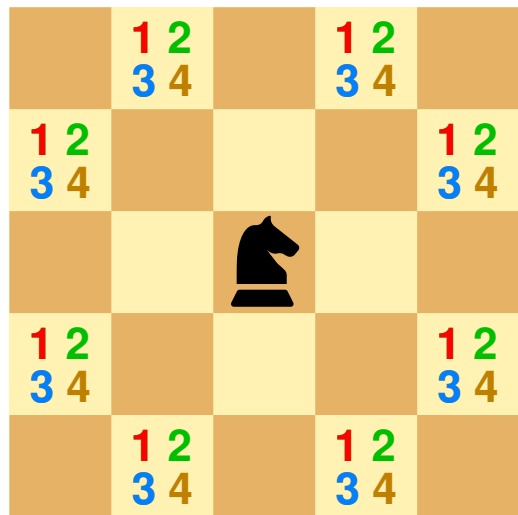
## Canalisation

- Different genotypes (rules)
- Same behaviour

# Context Is Important

Knight moves can be:

1. “L” shaped walk {FFL / FFR} x 4
2. Closest non-adjacent cell of different colour
3. Closest cell not in orthogonal or diagonal line
4. Closest cell of different colour not in orthogonal line



Contrastive in different geometries = ludemes

# Ludemes are Transferable

Ludemes can be transferred

- Verbally
- Through writing and illustration
- By example

Transcend language and cultural barriers

- Games are social lubricants (Crist *et al.*, 2016)

Digital ludemes

- Implement in software
- Transferred digitally (between game descriptions)

# Games as Memes

Bill and Ted choose different games

- Battleship
- Twister
- Electric Football

Different types of uncertainty

- Luck
- Dexterity
- Coordination

Games as metaphors

Games as memes  
(not ludemes)

