Game Description Languages:
the Good, the Bad, and the Ugly

Yngvi Björnsson

Department of Computer Science

Reykjavik University

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Overview

- Purpose of GDLs
- Properties of GDLs
- Birds-eye-view of
  - GDL
  - RBG
  - Ludii (language)
- Comparing
  - Pros and cons
  - Good, bad, ugly

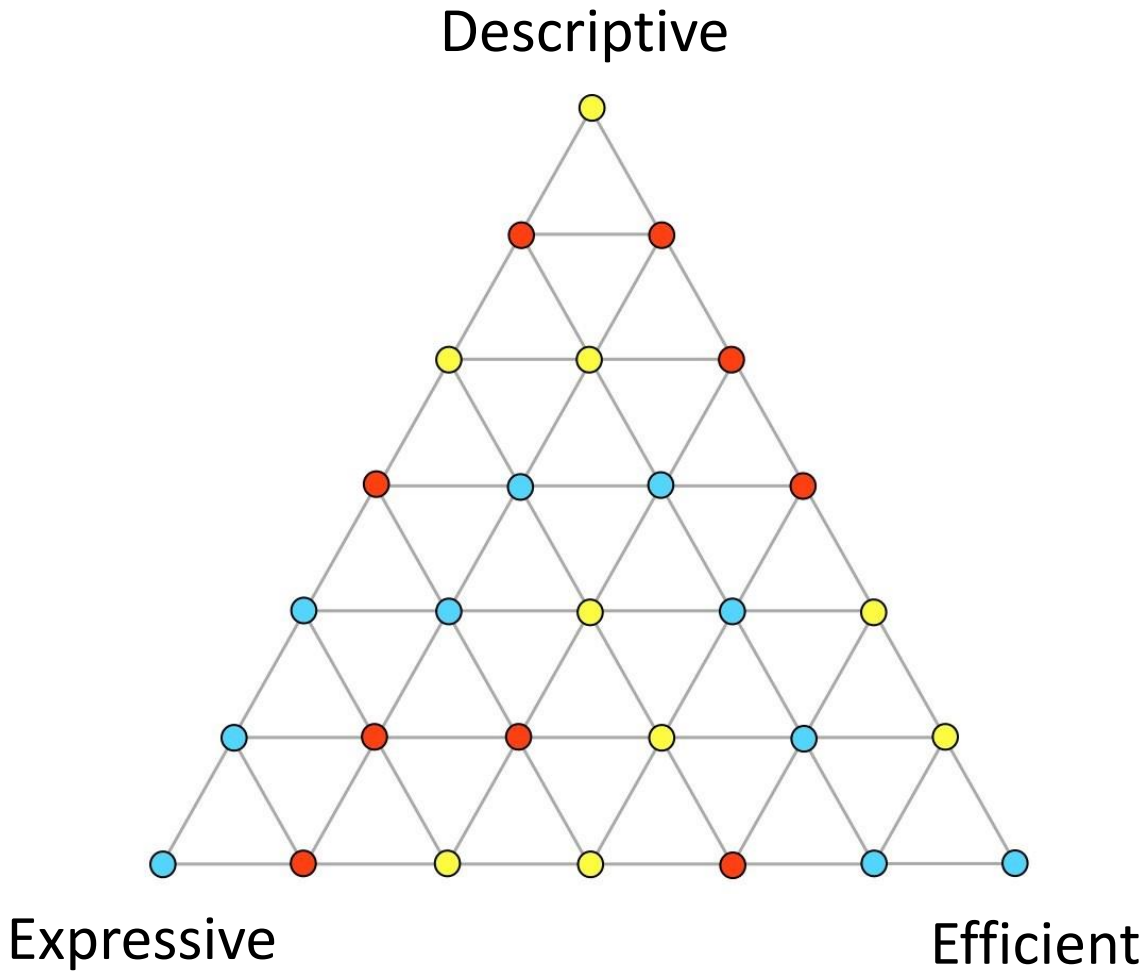
Purpose of GDLs

- Describe games
  - Preferably, in a concise and intuitive way using (mostly) human-understandable terms.
- But for what purpose?
  - Non-ambiguous recording of game rules
  - AI research
    - Game-playing agents
    - Evolution of games
  - Writing commercial games
  - Who is writing up the games?

Image source: https://www.greenbiz.com/

Desirable Properties

- Expressive
  - What type of games can be described?
  - Computationally universal?
- Descriptive
  - Concise and intuitive
  - Human readable
  - Declarative?
- Efficient
  - For computers to play



Abstract Strategy (Board) Games

- What types of games to describe?
  - Table-top vs. board vs. …
  - Number of players
  - Deterministic vs. non-deterministic
  - Perfect vs. hidden information
  - Finite vs. Infinite
  - Turn-based vs. real-time
  - …
- What aspects of a game to describe?
  - Connect-4
    - Drop from top or place in top-most empty?
  - Chess
    - Terminal conditions?
    - 3 (5)-fold repetition, 50 (75)-move rule?
    - How castling is performed? (king first, same hand)

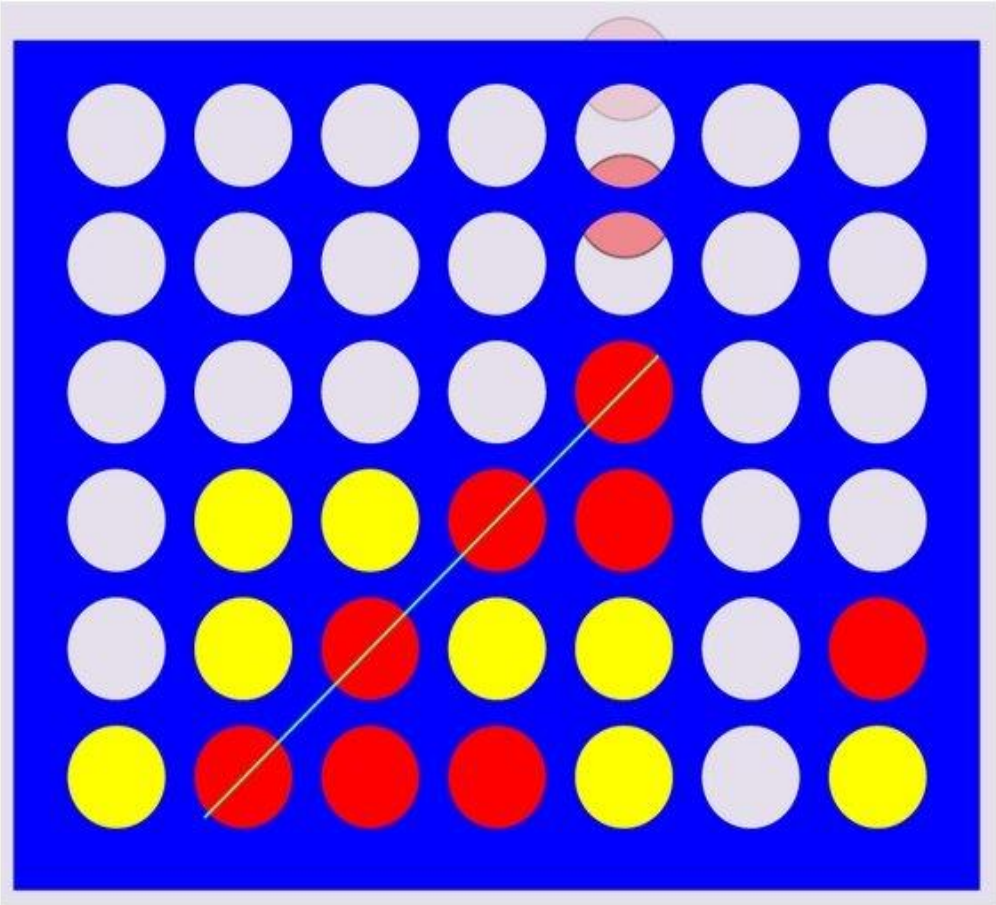


Image source: BCI Could Make Old Two-Player Games Even More Fun: A Proof of Concept with “Connect Four”

Game Description Language(GDL)(Genesereth et al., 2005)

- The official language of the GGP competitions (2005-2016)
- Logic-Based
  - Datalog -> KIF -> GDL
- Expressiveness
  - Deterministic, perfect information (allows simultaneous moves), finite.
  - GDL II/III
- Write custom reasoning backends
  - From-scratch resolution based
  - Prolog-based
  - PropNet-based

```
(role white)
...
(init (cellHolds 1 1 white))
...
(init (control white))
(<= (legal white (move ?x ?y1 ?x ?y2))
   (true (control white))
   (true (cellHolds ?x ?y1 white))
   (succ ?y1 ?y2)
   (cellEmpty ?x ?y2))
...
(<= (next (cellHolds ?x2 ?y2 ?player))
   (role ?player)
   (does ?player (move ?x1 ?y1 ?x2 ?y2)))
...
(<= (goal white 100) whiteWin)
(index 1) (index 2) (index 3) (index 4) ...
(succ 1 2)  (succ 2 3)  (succ 3 4)  (succ 4 5)  ...
```

GDL (cont.)

- Good
  - Declarative, well-known formalism
    - Although requires an understanding of logic-based languages (e.g., Prolog)
  - Runs on many platforms
- Bad
  - No game-elements to build on
  - No arithmetic
  - ”Requires” writing a new game from scratch
  - (Not expressive enough)
  - Inefficient reasoning
    - Albeit (later) alleviated by PropNet for smaller games
- Ugly
  - Not up to the task intended for, and contributed to the demise of the GGP competitions

Regular Board Games (RBG) (Kowalski et al., 2019)

- Regular Language
  - Idea of using a regular language from *Simple Board Games* (2012)
  - All game aspects (more or less) described using *regular expressions*
    - *Impressive!*
- Describes
  - Deterministic, perfect-information, finite board game
  - Non-stacking of pieces
- Implementation
  - C++
  - Provides an interpreter and a compiler

```
#players = white(100), black(100) // 0-100 scores
#pieces = e, w, b
#variables = // no additional variables
#board = rectangle(up,down,left,right,
         [b, b, b, b, b, b, b, b]
         [b, b, b, b, b, b, b, b]
         [e, e, e, e, e, e, e, e]
         [e, e, e, e, e, e, e, e]
         [e, e, e, e, e, e, e, e]
         [e, e, e, e, e, e, e, e]
         [w, w, w, w, w, w, w, w]
         [w, w, w, w, w, w, w, w])
#anySquare = ((up* + down*)(left* + right*))
#turn(me; myPawn; opp; oppPawn; forward) =
  anySquare {myPawn}      // select any own pawn
  [e] forward ({e} + (left+right) {e,oppPawn})
  ->> [myPawn]            // keeper continues
  [$ me=100] [$ opp=0]    // win if the play ends
  (   {! forward} ->> {}
// if the last line then end
    + {? forward} ->opp) // otherwise continue
#rules =  ->white (
    turn(white; w; black; b; up)
    turn(black; b; white; w; down)
  )* // repeat moves alternatingly
```

RBG (cont.)

- Good
  - Declarative, well-defined formalism
    - Although requires an understanding of regular expressions.
  - Efficient
- Bad
  - Who writes game descriptions?
  - (Not expressive enough)
- Ugly
  - Syntax
    - alien for non-cs

```
#castlingKingMove(forward; backward; color; oppColor) =
    {$ color~KingMoved==0}
    {! isAttackedBy(oppColor; forward; backward)}
    (
        right {empty} {! isAttackedBy(oppColor; forward; backward)}
        right {empty} [color~King]
        right pickUpPiece(color~RookUnmoved)
        left^2
      + left {empty} {! isAttackedBy(oppColor; forward; backward)}
        left {empty} [color~King]
        left {empty} // Rook can pass through an attacked square
        left pickUpPiece(color~RookUnmoved)
        right^3
    )
    [color~RookMoved]
```

Ludii Language (Browne, Piette et al., 2020)

- Ludii is a general game-playing system
  - Part of the Digital Ludeme Project
  - Uses a formal language for describing the games (the Ludii language)
  - Ludemes are a fundamental concept.
- Describes also, most notably:
  - hidden information and non-deterministic games
  - Less restriction on the board
    - (compared to RBG, e.g. site stacking)
- Implemented in Java
  - A Python wrapper exists
- Used in ICGA Game Olympiad Competitions

```
(game "Breakthrough"
    (players {(player N) (player S)})
    (equipment {
        (board (<Tiling:type> <Board:size>))
        (piece "Pawn" Each
            (or {
                "StepForwardToEmpty"
                (move
                    Step
                    (directions {FR FL})
                    (to if:(or
                            (is Empty (to))
                            (is Enemy (who at:(to)))
                        )
                        (apply (remove (to)))
                    )
                )
            })
        )
        (regions P1 (sites Top))
        (regions P2 (sites Bottom))
    })
    (rules
        (start {
            (place "Pawn1" (expand (sites Bottom)))
            (place "Pawn2" (expand (sites Top)))
        })

        (play (forEach Piece))

        (end (if "ReachedTarget" (result Mover Win)))
    )
)
```

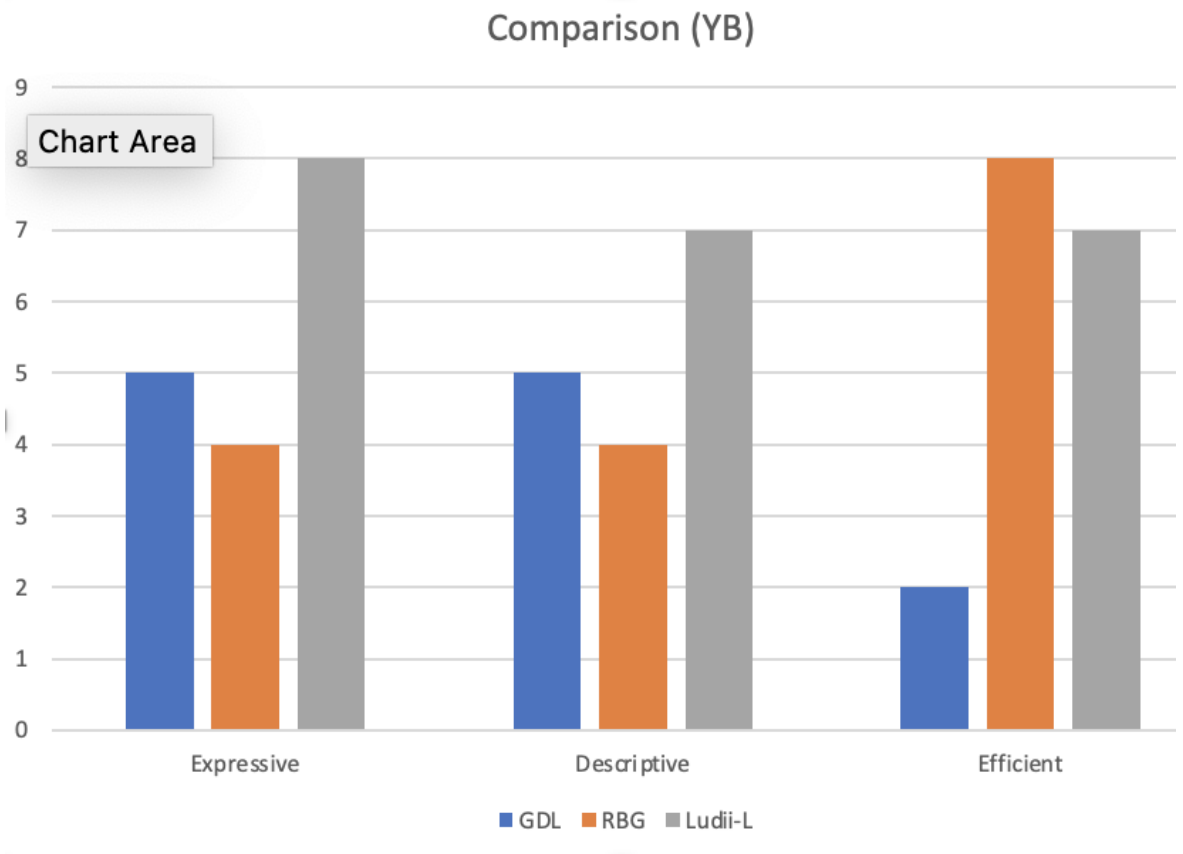
Ludii (cont.)

- Good
  - Expressive, descriptive, and efficient
  - Ludemes as building-blocks
- Bad
  - Framework vs. language?
    - Heavy dependency on Java
      - (mobile iOS?)
    - Is Java part of Ludii's GDL?
  - Building-blocks
    - without using Java (define macros)
- Ugly
  - API (unnecessarily?) bloated
    - Tempting to add and add more.
  - Efficiency vs. descriptiveness decisions
    - Multi-step-moves

```
(game "Amazons"
   (players 2)
   (equipment {
      (board (square 10))
      (piece "Queen" Each (move Slide (then (moveAgain))))
      (piece "Dot" Neutral)
   })
   (rules
      (start {
         (place "Queen1" {"A4" "D1" "G1" "J4"})
         (place "Queen2" {"A7" "D10" "G10" "J7"})
      })
      (play
         (if (is Even (count Moves))
            (forEach Piece)
            (move Shoot (piece "Dot0"))
         )
      )
      (end (if (no Moves Next)  (result Mover Win) ) )
   )
)
```

Comparison (objective)

- Expressive
  - Ludii-L can describe hidden-information, non-deterministic, (real-time) games.
- Descriptive
  - Ludii-L most human-readable
    - But understanding relies on having good (and correct!) documentation
- Efficient
  - RGB



Conclusions

- Is there room for yet another GDL?
  - Expressive
    - Enough to describe most relevant games
  - Descriptive
    - Mostly declarative, using a well-established formalism
  - Efficient
    - black-box optimizations
  - Also:
    - Module-based (minimize footprint)
    - Minimum dependency on platform/programming language
    - Extensible
    - Well-defined standardized execution model?
- Is there a need for yet another GDL?