Evaluating Chess-like Games Using Generated Natural Language Descriptions

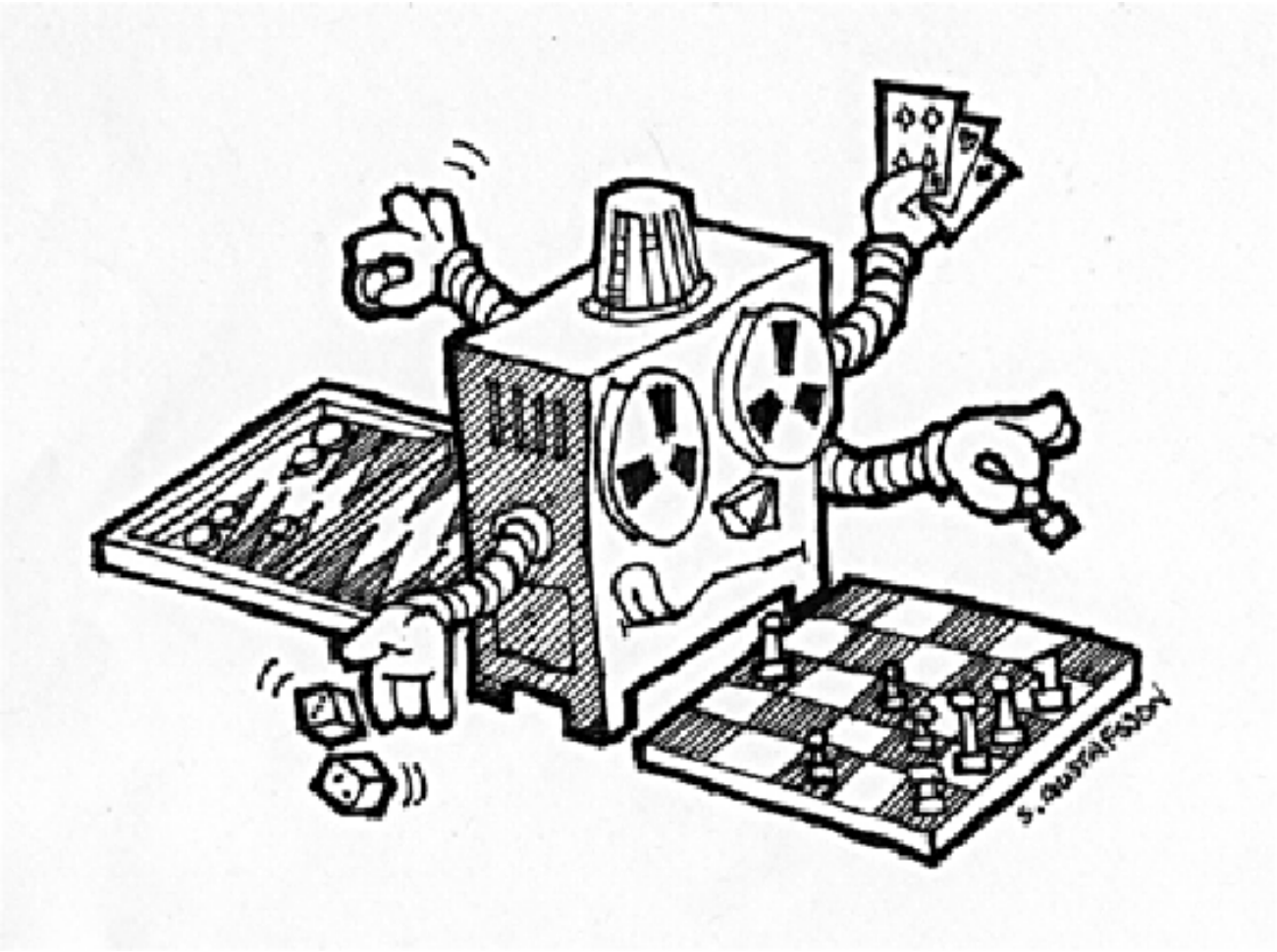
Jakub Kowalski, Łukasz Żarczyński, Andrzej Kisielewicz

University of Wrocław, Poland

ACG

05.07.2017

General Game Playing (GGP)



General Game Description Languages

- 1968 Chess-like games *(Pitrat)*
- 1992 METAGAME *(Pell)*
- 1997 Gala *(Koller, Pfeffer)*
- 1998 Zillions of Games *(Mallett, Lefler)*
- 2005 GDL *(Genesereth, Love, Pell)*
- 2008 Ludi *(Browne)*
- 2010 GDL-II *(Thielscher)*
- 2010 Toss *(Kaiser, Stafiniak)*
- 2011 Strategy Game Description Language *(Mahlmann et al.)*
- 2012 **Simplified Boardgames** *(Björnsson)*
- 2013 Card Game Description Language *(Font et al.)*
- 2013 Video Game Description Language *(Ebner et al.)*
- 2015 rtGDL *(Kowalski, Kisielewicz)*

METAGAME (B. Pell; 1992)

```
DEFINE man
  MOVING
    MOVEMENT
      LEAP
      <1,1> SYMMETRY {side}
        END MOVEMENT
  END MOVING
  CAPTURING
    CAPTURE
      BY {hop}
      TYPE [{opponent} any piece]
      EFFECT remove
      MOVEMENT
        HOP BEFORE [X = 0]
            OVER [X = 1]
            AFTER [X = 0]
        HOP_OVER [{opponent} any piece]
        <1,1> SYMMETRY {side}
      END MOVEMENT
    END CAPTURE
  END CAPTURING
  PROMOTING
    PROMOTE_TO king
  END PROMOTING
  CONSTRAINTS continue captures
END DEFINE
```

```
DEFINE king
  MOVING
    MOVEMENT
      LEAP
      <1,1> SYMMETRY {forward side}
        END MOVEMENT
  END MOVING
  CAPTURING
    CAPTURE
      BY {hop}
      TYPE [{opponent} any piece]
      EFFECT remove
      MOVEMENT
        HOP BEFORE [X = 0]
            OVER [X = 1]
            AFTER [X = 0]
        HOP_OVER [{opponent} any piece]
        <1,1> SYMMETRY {forward side}
      END MOVEMENT
    END CAPTURE
  END CAPTURING
  CONSTRAINTS continue captures
END DEFINE
```

Simplified Boardgames (Y. Björnsson; 2012)

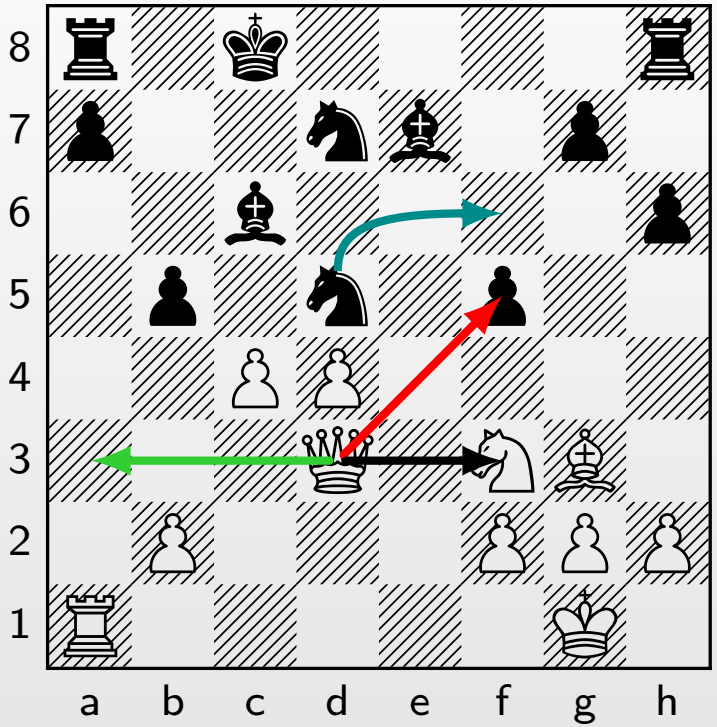
- Turn based; two players; zero-sum games;
- Rectangular board; fixed initial position; max one piece per square;
- One piece movement per turn;
- *Capturing* only at destination square;
- Winning conditions:
  - reaching a *goal* square using a certain piece,
  - captured some number of opponent's pieces;
- *Draw* occurs when the preset maximum game length is reached.

Set of piece's move rules

Regular language over an alphabet Σ containing triplets $(\Delta x, \Delta y, on)$:

- Δx, Δy are relative column/row distances;
- $on \in \{e, p, w\}$ describes the content of the destination square:
  - e – **e**mpty square,
  - p – square occupied by an o**p**ponent piece,
  - w – square occupied by an o**w**n piece.

Move examples



[1]

- ♕d3-a3: $(-1, 0, e)(-1, 0, e)(-1, 0, e)$
- ♕d3-f5: $(1, 1, e)(1, 1, p)$
- ♞d5-f6: $(2, 1, e)$
- ♕d3-f3: ~~$(1, 0, e)(1, 0, w)$~~

[1] Deep Blue vs Garry Kasparov, 1997, Game 6, Move 19 (last)

Simplified Chess example

```
<--BOARD-->
8 8
rnbqkbnr
pppppppp
........
........
........
........
PPPPPPPP
RNBQKBNR
```

```
<--GOALS-->
200 &
@P 0 7, 1 7, 2 7, 3 7, 4 7, 5 7, 6 7, 7 7 &
@p 0 0, 1 0, 2 0, 3 0, 4 0, 5 0, 6 0, 7 0 &
#K 1 &
#k 1 &
```

```
<--PIECES-->
P (0,1,e) + (-1,1,p) + (1,1,p) +
  (0,1,e)(0,5,e)(0,-4,e) + ... &
N (2,1,e) + (2,-1,e) + (-2,1,e) + (-2,-1,e) + (1,2,e)... &
R (0,1,e)^* + (0,1,e)^*(0,1,p) + (0,-1,e)^* + ... &
```

Natural Language Decriptions

Natural Language Decriptions for fairy chess pieces

Goal

Procedurally generate natural language descriptions of arbitrary piece, given its rules as a Simplified Boardgames regular expression.

Motivation

- Producing game manuals for procedurally generated games.
- Evaluation of games based on the simplicity of their rules.

PCG in Simplified Boardgamess

- Existing mechanisms that generate game rules evaluate mainly ,,strategic properties" of the games
- This is enough for the AI GGP purposes – computers are less interested in the complexity of given regular expressions than humans

Describing moves by searching for analogies

T. R. Dawson *Theory of Movements*

- The basis of the fairy chess movement patterns
- Division into leap, ride, and hop
- Strict base for the METAGAME rules

knight $\equiv \langle 2, 1 \rangle$-leaper
king $\equiv \langle 1, 0 \rangle$ and $\langle 1, 1 \rangle$-leaper
rook $\equiv \langle 1, 0 \rangle$-rider

Analogies

- To describe more general Simplified Boardgames rules, we can refer to this theory
- We can describe any piece as a composition of simpler analogies to known pieces / partial moves

queen $\equiv$ moves like rook or bishop
lance $\equiv$ moves as rook but only forward
bishight $\equiv$ moves forward as a bishop and backward as a knight

Describing moves by searching for analogies

Classes

Class describes the most basic type of movements. Class contains:

- Type: *leaper* or *rider*
- Translation vector: $\langle \Delta x, \Delta y \rangle$

Operators

Operators are used to restrict the moves within some class. Example operators are:

- forwards, backwards, sideways, only capture, without capture, max times, only odd, only even, ...

Move descriptions

Usually pieces can be described by the sum of classes combined with the operators:
lance $\equiv \langle 1, 0 \rangle$-rider forwards
bishight $\equiv \langle 1, 1 \rangle$-rider forwards + $\langle 2, 1 \rangle$-leaper backwards

Describing moves by searching for analogies

Composite cases

Some types of pieces (e.g. bent riders) cannot be described in a concise way using only sum of restricted classes.

- Composite classes allows to chain (class,operators) pairs
- Preceeding class defines a vector that can be used as a reference point for operators (e.g. outwards, inwards).

griffon $\equiv \langle 1,1 \rangle$-leaper + $\langle 1,1 \rangle$-leaper $\rightarrow \langle 1,0 \rangle$-rider outwards

Searching procedure

- We start with the set of all words within the language
- Greedy search is used to find ,,best" (class, operators) to describe a subset of moves
- Described words are removed from the language
- Procedure is repeated until all words are described
- Finally, repairing run simplify the obtained descriptions

Evaluation

Evaluation function

Evaluating vectors

$$f(\langle x, y \rangle) = 1 + \max(|x|, |y|) + \min(dist_{+}(x, y), dist_{\times}(x, y)). \quad (1)$$

Evaluating operators

$f(o)$	o
0	none
1	backwards, forwards
2	sideways, only capture, without capture, outwards max times, min times, not horizontal
3	exactly times, over own piece instead
5	only odd, only even

Evaluation function

Evaluating classes with operators

$$f(\langle x, y\rangle, [o]) = f(\langle x, y\rangle) \cdot (1 + \sum_i f(o_i)) \quad (2)$$

Evaluating composite descriptions

- Product of all components evaluated using (2)

Evaluating single move (special case)

- Product of all included vectors, evaluated using (1)

Evaluating games

$\#$ – number of piece's ocurrences in the initial position
k – number of defined types of pieces

$$f([p]) = (\sum_{i=1}^{k} \#p_i \cdot f(p_i))/(\sum_{i=1}^{k} \#p_i) \cdot (10 + k).$$

Experiments

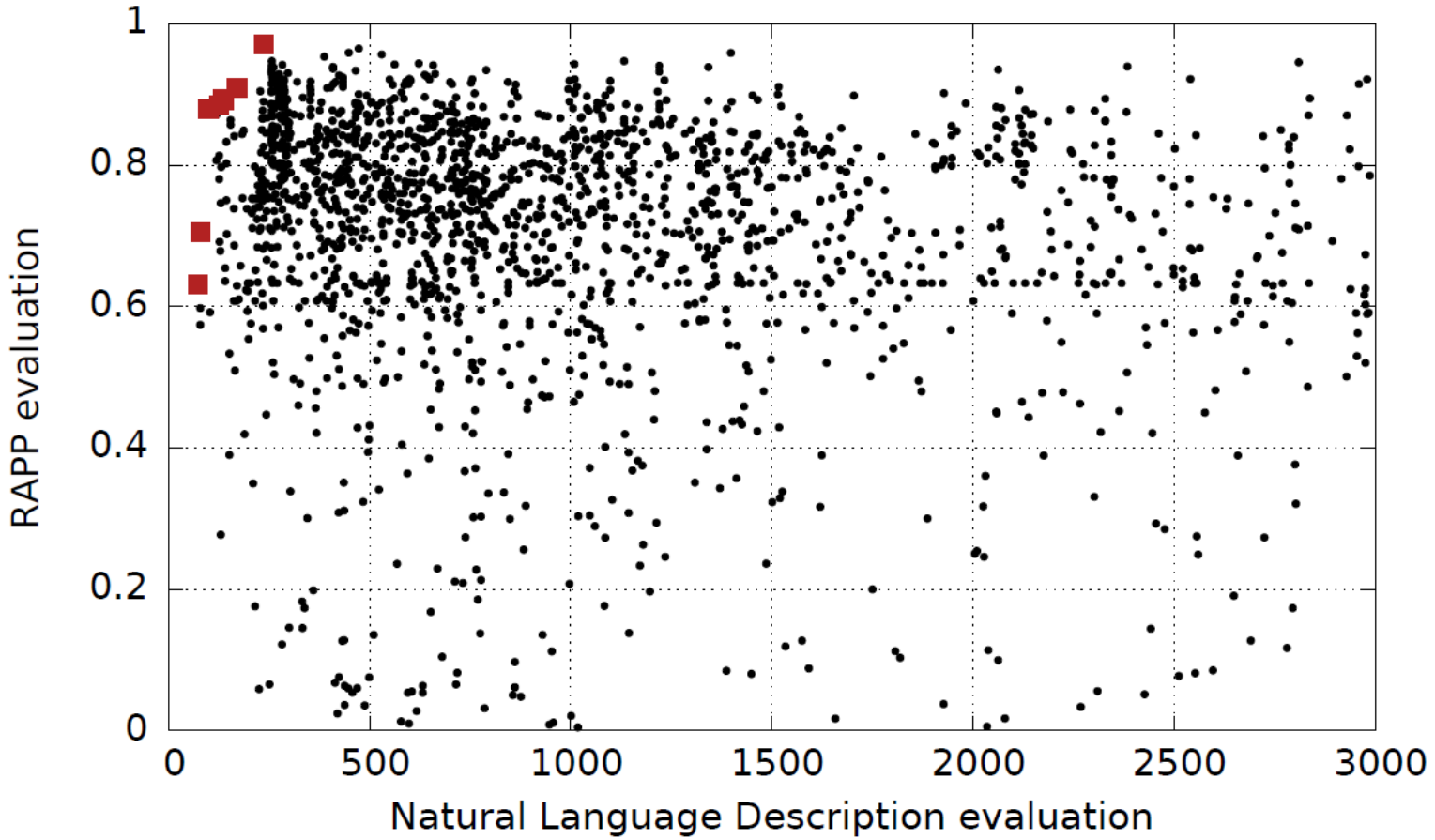
Evaluating pieces

Piece	Score	Description
queen	6	Rides in every direction
short rook	12	Rides max 4 times horizontally or vertically
lance	4	Rides vertically forward
centaur	14	Leaps 1 in every direction or over vector (1,2)/(2,1)
griffon	78	Moves 1 diagonally or moves 1 diagonally and then rides outwards vertically or horizontally
hippogriff	120	Moves 1 diagonally and then rides minimum 2 times outwards horizontally or vertically
duke	60	Rides without capturing diagonally and then leaps outwards 1 vertically or horizontally
moa	36	Moves 1 diagonally and then leaps outwards 1 vertically or horizontally
buffalo	28	Leaps over vector (3,2)/(2,3) or (3,1)/(1,3) or (1,2)/(2,1)

Evaluating games

Game	NLD evaluation
shatranj	166
chess	168
CWDA Colorbound Clobberers	196
CWDA Remarkable Rookies	222
CWDA Nutty Knights	248
SIMB 30-P4	253
tamerlane chess	376
RAPP The Legacy of Ibis	387

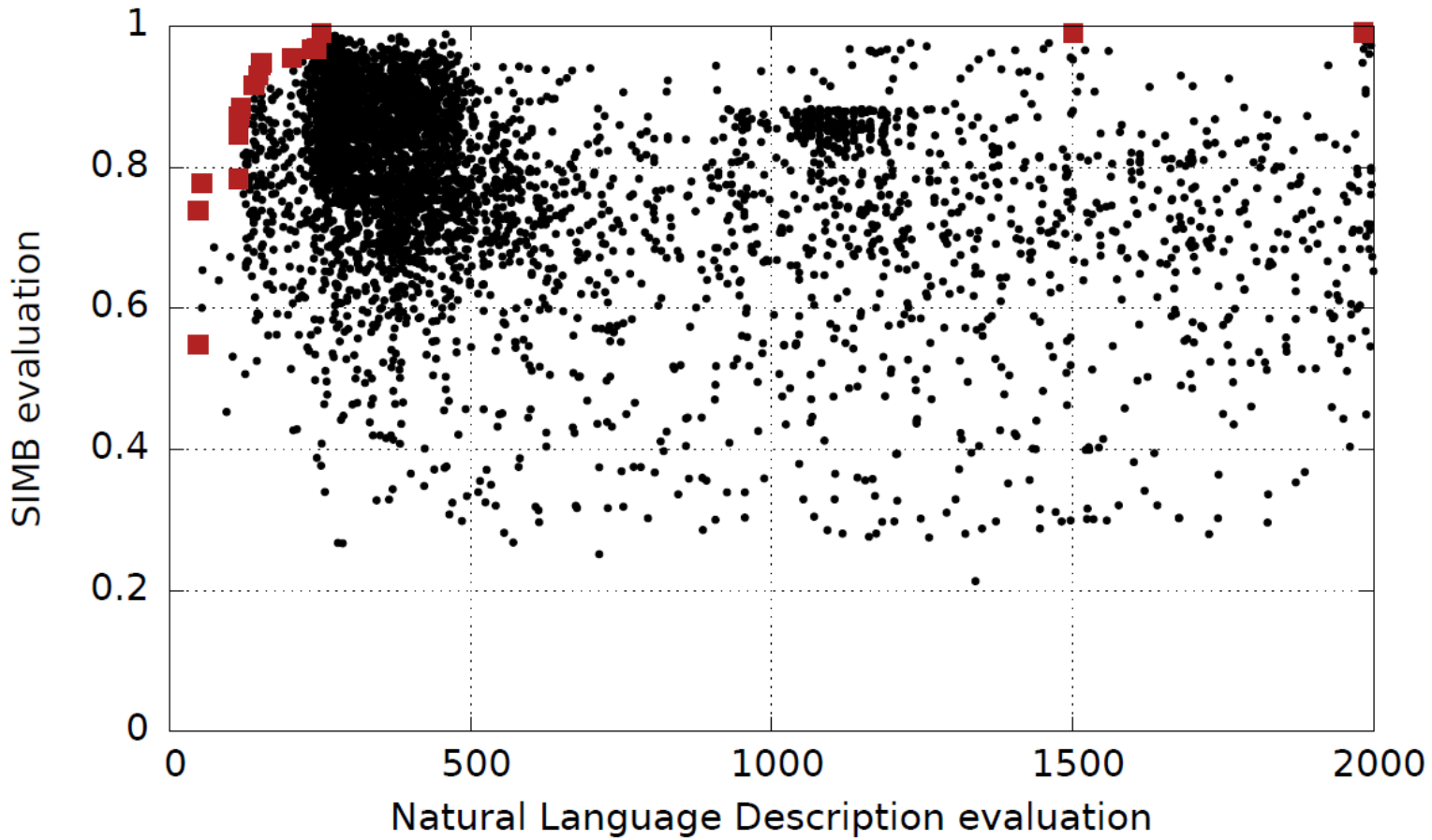
Evaluating procedurally generated games: RAPP



Evolving Chesslike Games Using Relative Algorithm Performance Profiles (J. Kowalski, M. Szykuła; 2016)

- Pareto front: 8 games
- Shown 71% of 2581 generated games

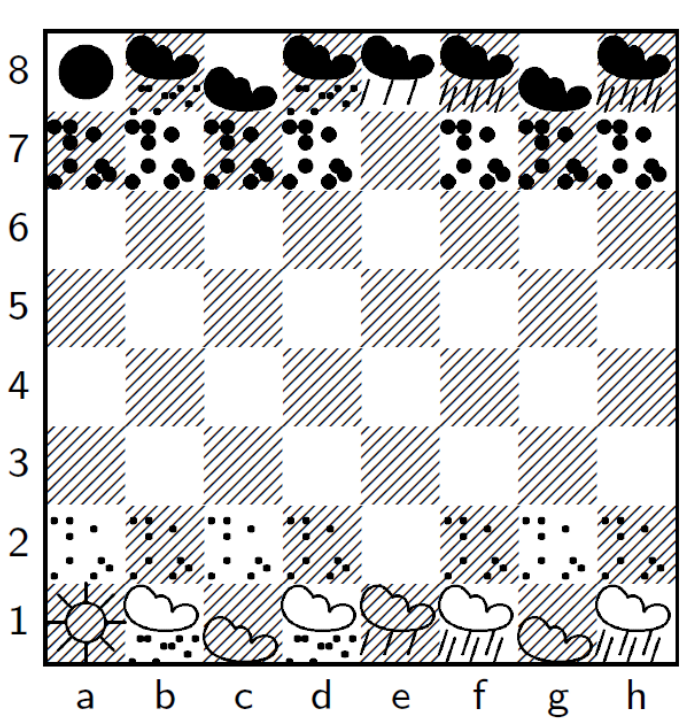
Evaluating procedurally generated games: SIMB



Procedural Content Generation for GDL Descriptions of Simplified Boardgames (J. Kowalski, M. Szykuła; 2015)

- Pareto front: 17 games
- Shown 54% of 8002 generated games

Example game



(16) Moves 1 backwards vertically
Captures 1 forward vertically

(18) Leaps 2 forward diagonally
Captures 1 vertically or horizontally

(12) Leaps 1 backwards diagonally
Leaps forward over vector (1,2)

(8) Leaps 2 forward vertically
Leaps 1 vertically

(16) Leaps 1 backwards diagonally
Captures 1 horizontally or diagonally

(10) Leaps 1 backwards diagonally
Captures 1 diagonally

- RAPP score: 0.971
- NLD score: 236
- King: ☼; Pawn: ⁘
- Turnlimit: 109

Future Work

Online Game Manager

Goal

Provide (chess-like) General Game Playing challange for human players

- Allows human players and AI's to play Simplified Boardgames via the online game manager
- Contains predefined GGP agents that can play any given boardgame
- Contains a set of procedurally generated games
  (more advanced: allows to generate new games on the fly)
- Describes and visualizes the game rules and the game course in a human readable way
- Collects data about the user's preferences regarding generated content

Automated rule teaching mechanism

Improve natural language descriptions generation

- Make descriptions look more naturally and less artificial
- Define a layered set of references that can be used within the rules description (e.g. use rook instead of $\langle 1, 0 \rangle$-rider)
- Tune weights used to evaluate the quality of the descriptions

Visualizing piece movements

- Acompany description with generated visualization describing piece's rules
- (advance: highlight-based correspondence between the text and the graphics)

Showing advanced usage examples

- Develop mechanism that will generate game situations showing usage of a particular piece

Generation of piece shapes

Motivation

- Procedurally generated games should not use existing shapes of human-defined pieces for the visualization purposes
- Thus, the additional mechanism generating shapes is required

Method

- The piece shape is defined as a list of Bézier curves
- We use evolutionary algorithms to obtain ,,good" shapes
- Evolution works in two phases
- Phase one is devoted to create piece base family shape (that ,,guarantees" that pieces used in one game will be easily destinguished for the pieces of other games)
- Phase two generates final shapes of every piece used by a given game
- Evaluation of shape depends on the properties of the image (gravity center, spline length, self-similarities, etc.), and the properties of the corresponding piece (strength, simplicity of the rules, role).

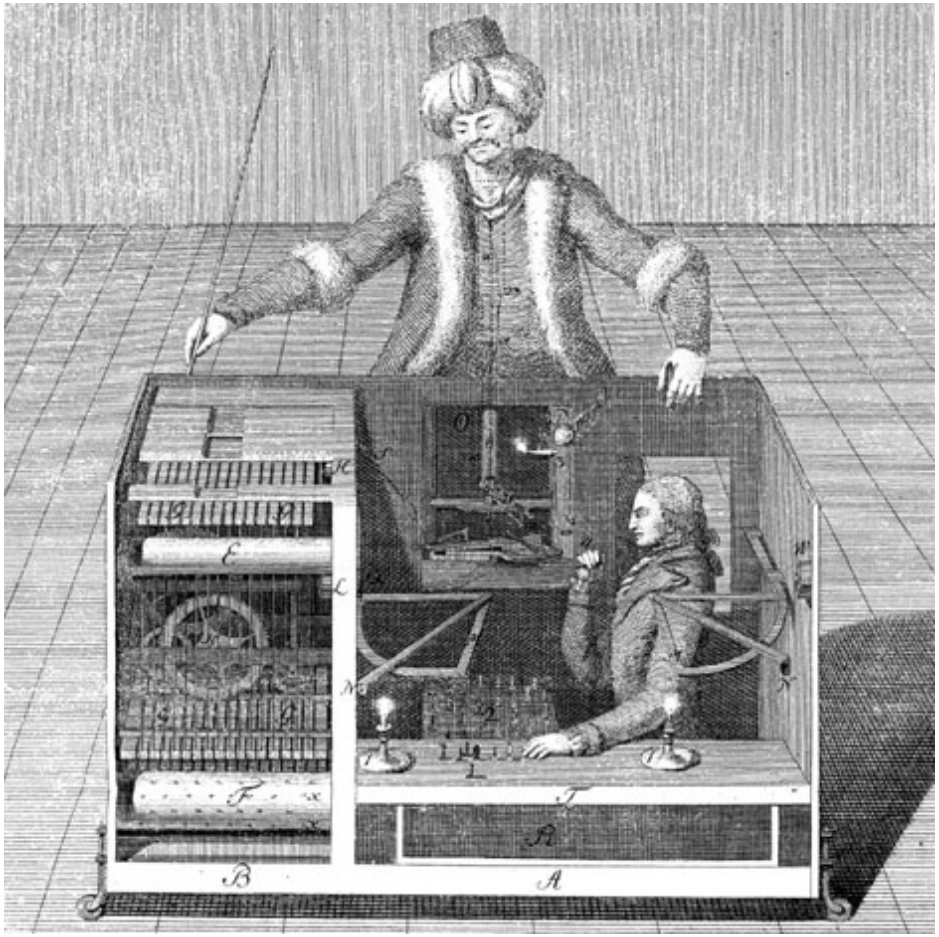
Extending Simplified Boardgames to Regular Boardgames

Regular Boardgames (J. Kowalski, J. Sutowicz, M. Szykuła; 2017)

- We further generalize the method of describing game rules using the regular expressions.
- Our goal is to provide maximal expressiveness under the one common set of rules, without introducing special cases.
- The construction emerged from an analysis of popular chess variants, checkers, go, tic-tac-toe, exchange games, games for more than two players; and a try to generalize the concepts they use.

Main features

- A possibility to remove existing pieces or produce new pieces during a move (chess promotions, checkers capture).
- Move priorities
  (capturing maximal number of pieces in checkers; en passant).
- Regular expressions as terminal conditions
  (tic-tac-toe *three in a row*).
- A lot of minor syntax and semantic issues, like macros or arbitrary selection of the next player after a move.



Thank you