Towards AGGI Demo

Łukasz Kaiser and Łukasz Stafiniak

CNRS & LIAFA
Paris

Artificial General Game Intelligence

…not yet AGI but near!

What is AGGI?

Goal: demonstrate any board game to a robot and have him play it well

(1) Demonstration

→

(2) Description

→

(3) Learning

→

(4) Play

What is AGGI?

Goal: demonstrate any board game to a robot and have him play it well

(1) Demonstration

→

(2) Description

→

(3) Learning

→

(4) Play

Status, i.e. what we can show and what **not**

(1) Make moves on-screen and see rules derived from pictures
No robot integration now, too weak vision algorithms at present

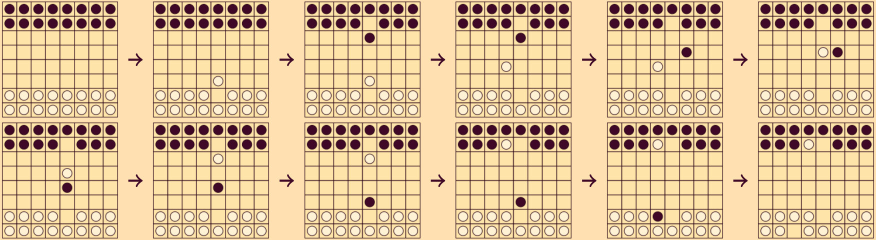
(2) Derive simple constraints automatically, use a high-level language
Only simple constraints from positive/negative pictures, no GUI

(3) Derive position evaluation function from game rules alone
Only approximately good weights learned in real-time

(4) Play the game you defined against our engine on-screen
No robot moving, the engine may be slow for high-branching games

(1) Demonstration

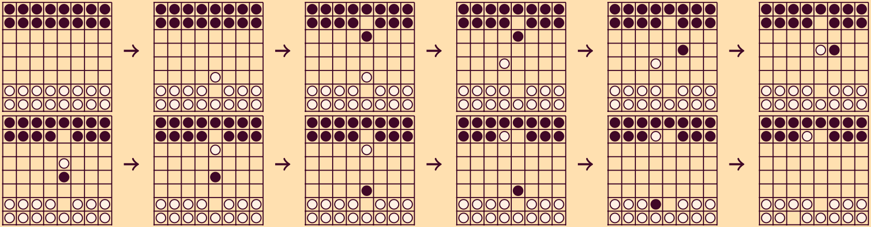
Example Input: Breakthrough moves, a sequence of screenshots



(1) Demonstration

Example Input: Breakthrough moves, a sequence of screenshots

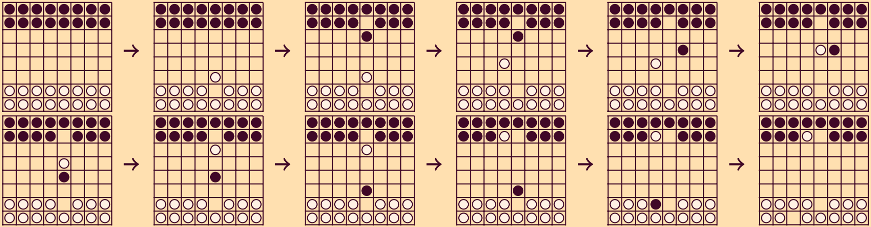
Derived Structures: segmentation of the input, row and column relations



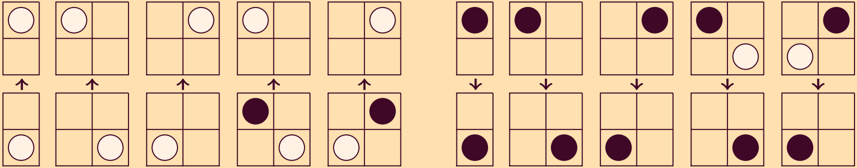
(1) Demonstration

Example Input: Breakthrough moves, a sequence of screenshots

Derived Structures: segmentation of the input, row and column relations



Derived Structure Rewriting Rules correspond **directly** to moves



(2) Description

Why that? You cannot demonstrate **everything** – sometimes you **say it**.

(2) Description

Why that? You cannot demonstrate **everything** – sometimes you **say it**.

Example: Breakthrough Winning Condition for White

Text: **Some white piece must be in the last row.**

Last row is the one for which there is no next row.

Formula: $\exists\, x\ \mathbf{White}(x)\ \wedge\ \mathbf{LastRow}(x)$

$\mathbf{LastRow}(x) \equiv \neg\exists\, y\ \mathbf{NextRow}(x, y)$

(2) Description

Why that? You cannot demonstrate **everything** – sometimes you **say it**.

Example: Breakthrough Winning Condition for White

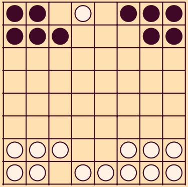
Text: **Some white piece must be in the last row.**

Last row is the one for which there is no next row.

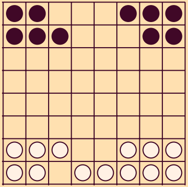
Formula: $\exists\, x\ \mathbf{White}(x)\ \wedge\ \mathbf{LastRow}(x)$

$\mathbf{LastRow}(x) \equiv \neg\exists\, y\ \mathbf{NextRow}(x, y)$

Automatic Derivation of Simple Constraints



Positive Example



Negative Example

Derived Formula: $\exists\, x\ (\mathbf{White}(x)\ \wedge\ \forall\, y\ \neg\mathbf{NextRow}(x, y))$

(3) Learning

Deriving Interesting Patterns (with examples for **Breakthrough**)

(i) When I move, what do I **add or delete**? **White added, Black deleted**

(ii) Can I **expand** the goal to an **existential conjunction**?

$$\exists x_1 \dots x_8 \left(\mathbf{NextRow}(x_1, x_2) \land \dots \land \mathbf{NextRow}(x_7, x_8) \land \mathbf{White}(x_8) \right)$$

(iii) How many of these **conjunction items** are **realized**?

(iv) Other expansions and sums over co-occurring items.

(3) Learning

Deriving Interesting Patterns (with examples for **Breakthrough**)

(i) When I move, what do I add or delete? **White added, Black deleted**

(ii) Can I expand the goal to an existential conjunction?

$$\exists x_1 \ldots x_8 \left(\mathbf{NextRow}(x_1, x_2) \wedge \ldots \wedge \mathbf{NextRow}(x_7, x_8) \wedge \mathbf{White}(x_8)\right)$$

(iii) How many of these conjunction items are realized?

(iv) Other expansions and sums over co-occurring items.

Learning Weights for Patterns

- Initially weight added (**White**) with $+\alpha$, deleted (**Black**) with $-\alpha$
- Multiply by β for realizing any goal conjunction
- Sum over configurations which realize at least γ conjunctions
- Play and Optimize the weights α, β, γ and other
  This is the learning part, important but very time-consuming

(4) Play

Visit `www.tplay.org` to check it out

Remarks

- All games except for chess use automatically derived position evaluation
- Give more time for better play (upper-left screen corner)
- Open-source (BSD license), available from `www.toss.sf.net`

Summary

- **Good representation** corresponds directly to visual input
- **Describing** additional constraints and patterns in **logic** is easy
- **Result**: demonstrate a board game and we will play it
- **Future**: let a robot do it. **We search for collaborators!**

Thank You