Simulating Evolution:
Challenges and Insights

Paul Valiant

Brown University

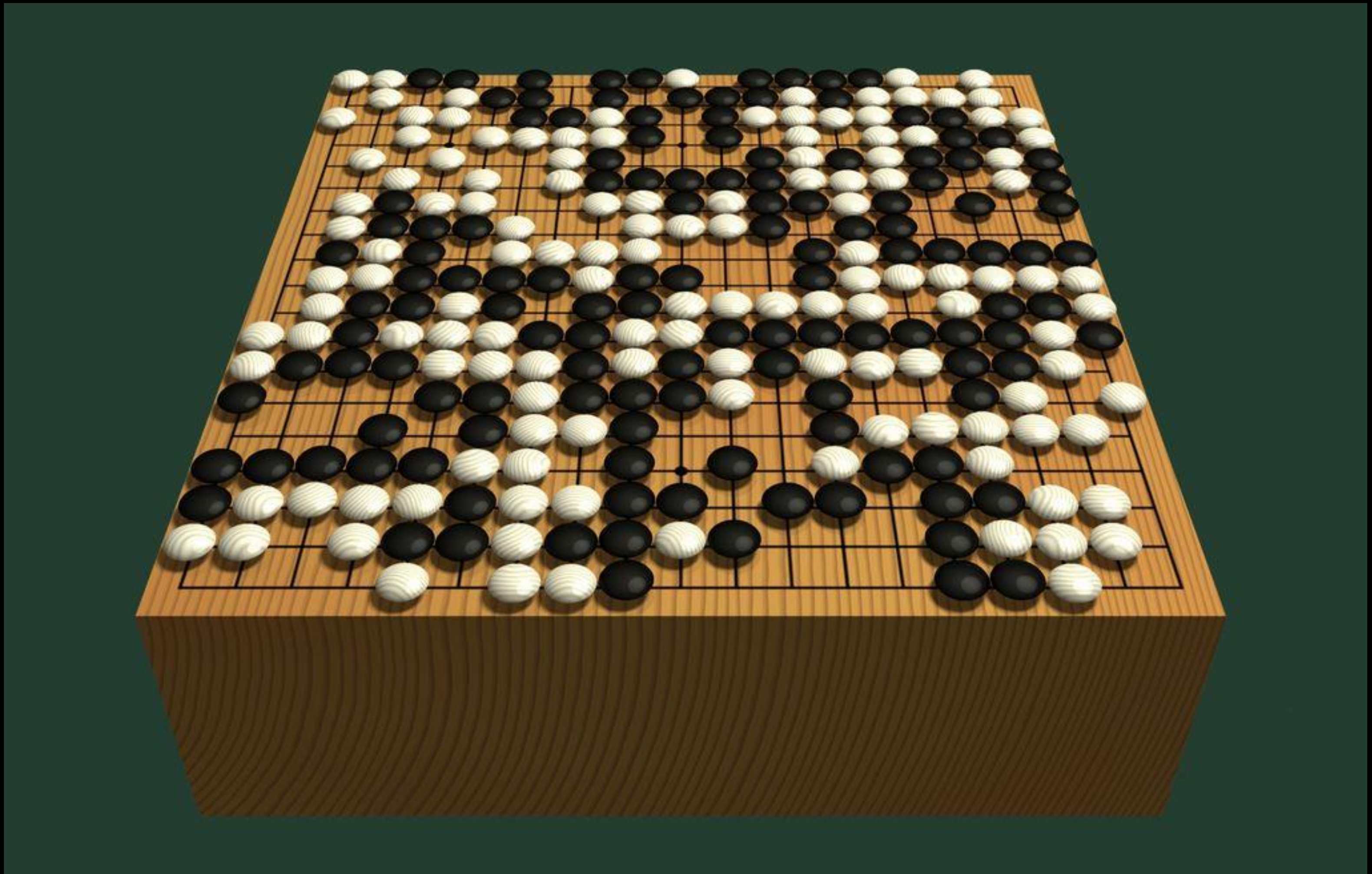
“Evolution will occur whenever and wherever three conditions are met: replication, variation (mutation), and differential fitness (competition)"

--Encyclopedia of Evolution

One missing component:

“DNA is code”

A rich setting for artificial life: the game of Go



Go: the laws of physics of virtual life

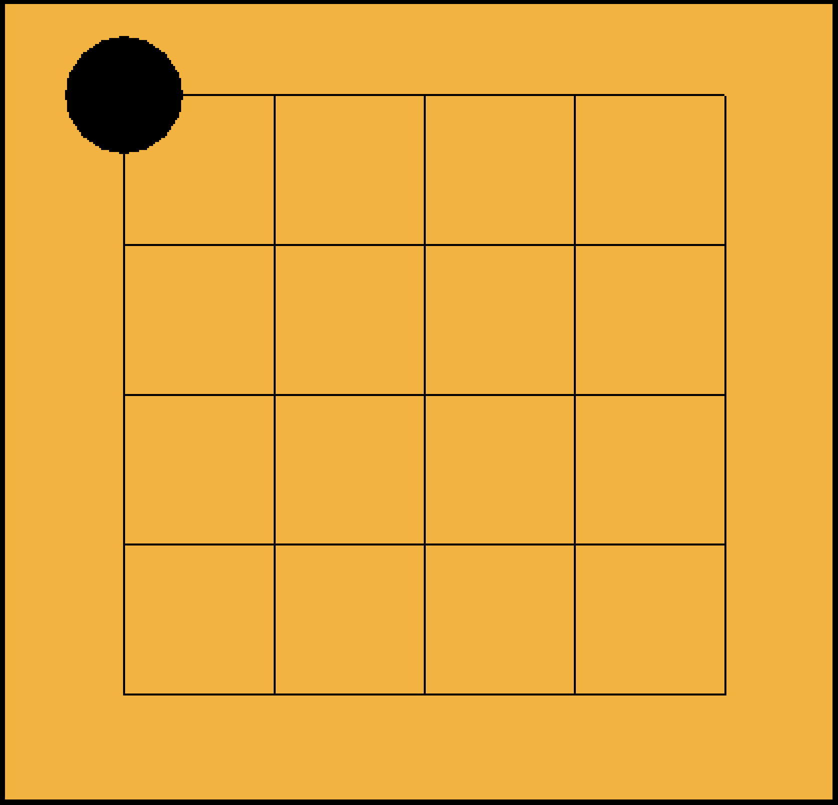
- Two creatures alternate turns.
- When your turn starts, you “wake up” in the top left of the board, and can move around, sense your environment.
- Wherever you are at the end of your turn is interpreted as your move.
- Your behavior is coded for by your DNA, modifying and interacting with your internal state (which may persist across turns)

Go folklore: however good you are, if you play a master, she can give you simple advice that will noticeably improve your play → accessible beneficial mutation

Trivial Creatures

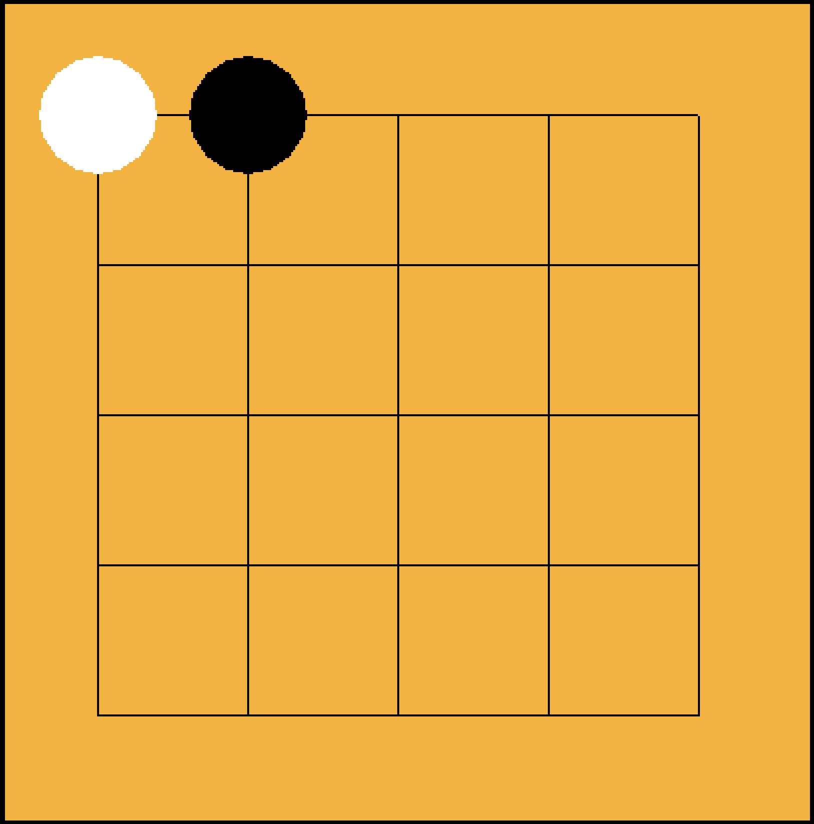
Black = {}

White = {}



The first step

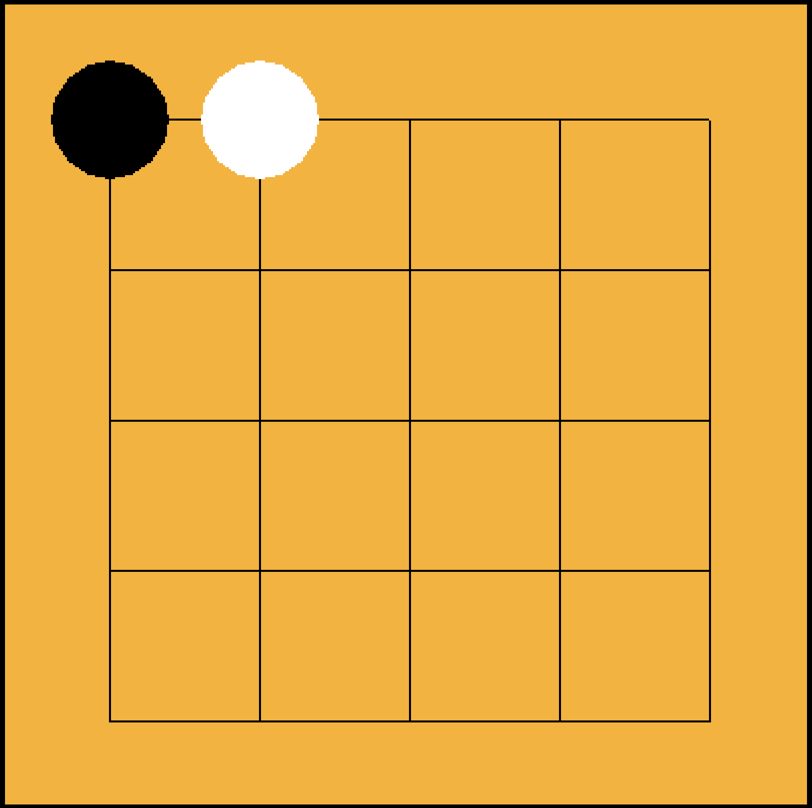
Black = {move right}   White = {}



The first step

Black = {}

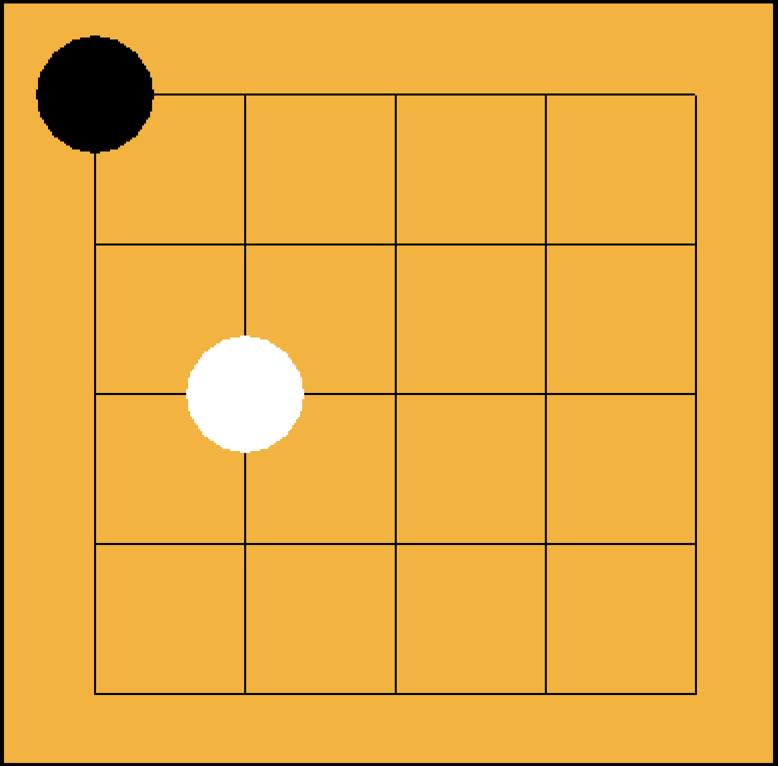
White = {move right}



The first step

Black = {}

White = {move right
move down
move down}



The second step

Must behave differently in different circumstances to get multiple stones on the board

Black = { Look
If empty:
Halt
Move right
}

White = {move right
move down
move down}

Two options: sensory input; randomness

Interlude: junk DNA and mutations

Black = { Look

JUNK

If empty:

Halt

JUNK

Move right

JUNK

}

To enable future growth, helps to have richer mutation toolkit than simply point mutations: deletions, insertions, and copying help significantly

Many steps: control flow

To get more different behaviors than the number of instructions: control flow – i.e. loops

```
Black = { 1. Look
             If empty:
                Halt
             Move right
            Goto 1
             }
```

```
White = {move right
         move down
         move down}
```

Natural variants move vertically or diagonally

Running into a wall

The next innovation to look for has the effect of a "nested loop", but this is very hard to evolve

Black = { Find the first empty square left to right;
If we run into a wall, go down a row and look for the first empty square right to left;
If we run into a wall, go down a row and repeat }

Instead: function calls (gene regulation/promotion)

A: Random walk:
1. Move left
Move right
Promote A
Goto 1

B: Random walk:
2. Move up
Move down
Promote B
Goto 2

C: Look
If empty:
    Halt

Modularity is hard to encourage

A: Random walk:
1. Move left
Move right
Promote A
Goto 1

B: Random walk:
2. Move up
Move down
Promote B
Goto 2

C: Look
If empty:
    Halt

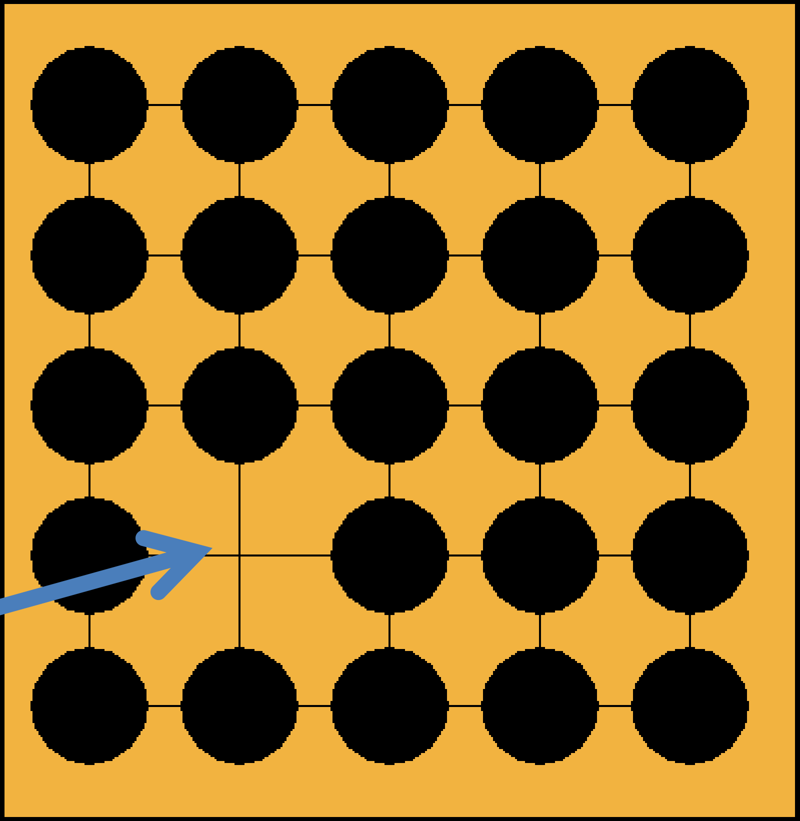
X:
1. Move up
Move down
Move left
Move right
Look
If empty:
    Halt
Promote X
Goto 1

Other features

- Memory/Synchronization/Communication
- Ecology: many smaller islands with different migration rates to the mainland
- Diploid genomes: two versions of each gene, random one is run each time it is called
- Fitness reward for using more genes
- Sex and speciation: opponents can agree to mate instead of competing at Go
- Eric Siggia: reduce population 100x but play 100 games instead of 1
- Daniel Fisher: ecology of beginning, middle, and end game, games are between teams of 3 players from separate populations.

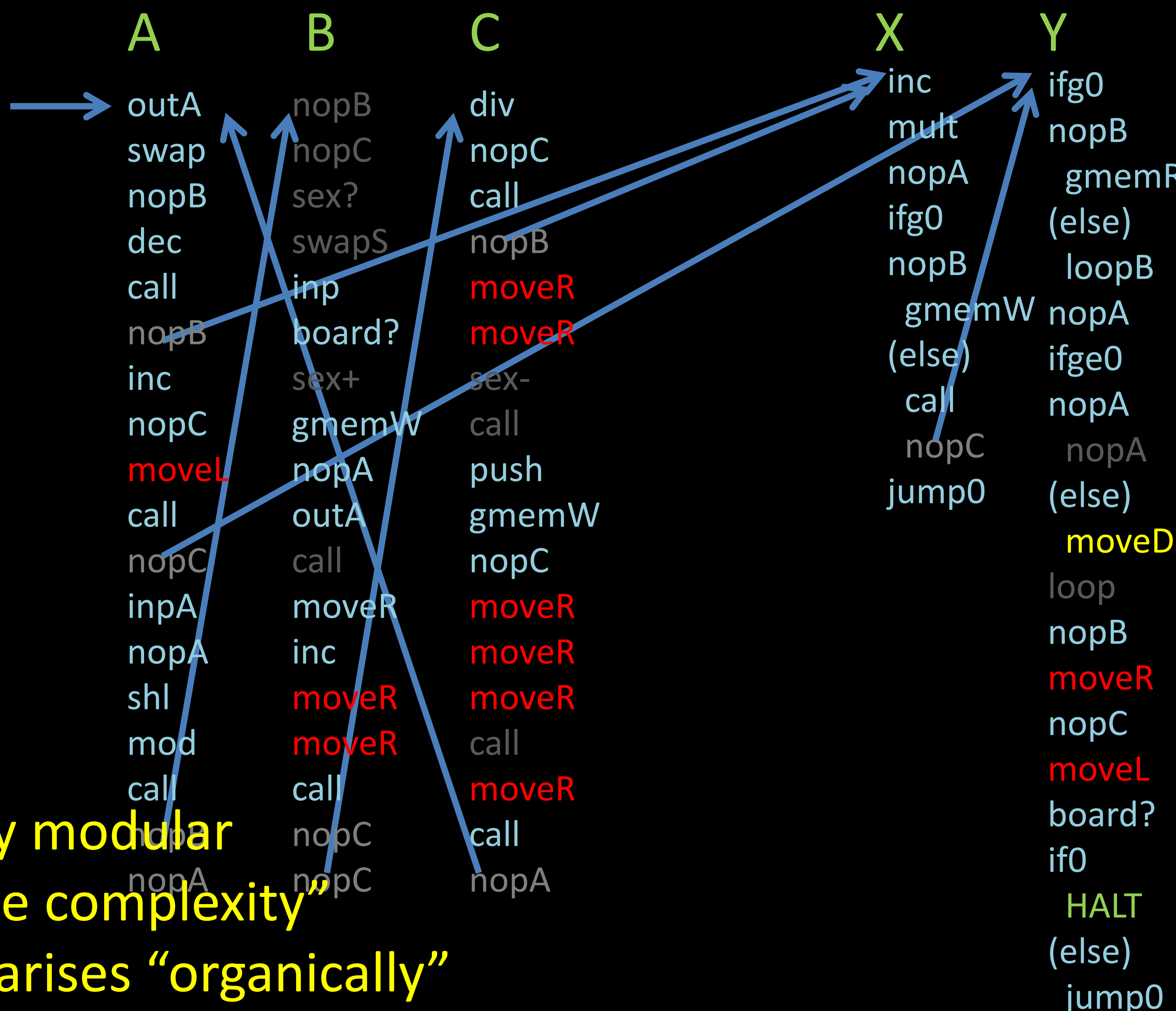
Non-Go games

"Find a hole puzzle":



Move:

Return: 4th row



Intriguingly modular

"Irreducible complexity"

Reliability arises "organically"

Only evolvable <u>after</u> Go

Further Directions

Creatures that play many games (Go + “find the hole” + …)

→ With Spencer Gordon and Roie Levin

“Soft” failures – creature has a mortality rate instead of a lifespan; function calls can spawn 0-2 copies instead of 1

“Development” as key to genetic control over more intricate traits

Why is modularity so hard??