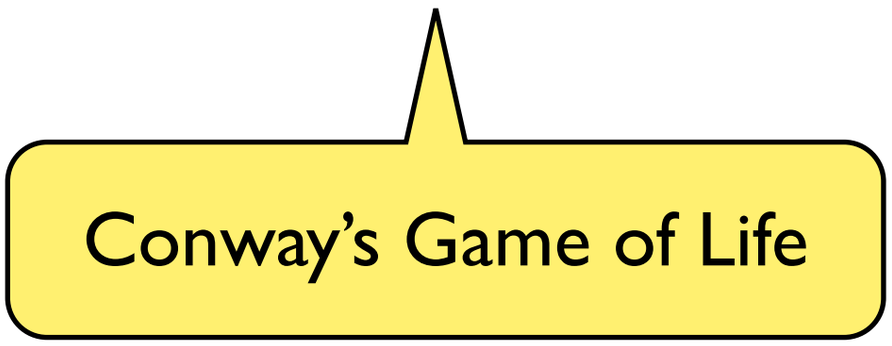
ITP 2025

# GOL in GOL in HOL

Magnus O. Myreen, Mario Carneiro

Chalmers

# GOL in GOL in HOL



# GOL in GOL in HOL

Conway's Game of Life

… simulated in Conway's Game of Life

# GOL in GOL in HOL

Conway's Game of Life

… simulated in Conway's Game of Life

… and this is verified in the HOL prover.

This talk:

## *What is Conway's Game of Life (GOL)?*

rules, gliders, space ships, useful collisions,
circuits in GOL, GOL in GOL

## *Formal verification of circuits in GOL*

symbolic simulation for individual gates,
compositional reasoning about (circuits in) GOL

This talk:

*What is Conway's Game of Life (GOL)?*

rules, gliders, space ships, useful collisions,
circuits in GOL, GOL in GOL

*Formal verification of circuits in GOL*

symbolic simulation for individual gates,
compositional reasoning about (circuits in) GOL

# Rules of Conway's Game of Life

We define a GOL state as a set $S \subseteq \mathbb{Z}^2$, where $(i, j) \in S$ means that $(i, j)$ is alive in state $S$.

# Rules of Conway's Game of Life

We define a GOL state as a set $S \subseteq \mathbb{Z}^2$, where $(i, j) \in S$ means that $(i, j)$ is alive in state $S$.

GOL's next-state function

$$\begin{aligned}\text{step } S \stackrel{\text{def}}{=} \{\ (i,j) \mid \text{if } (i,j) \in S \text{ then live\_adj } S\ i\ j \in \{2,3\}\\ \text{else live\_adj } S\ i\ j = 3\}\end{aligned}$$

# Rules of Conway's Game of Life

We define a GOL state as a set $S \subseteq \mathbb{Z}^2$, where $(i, j) \in S$ means that $(i, j)$ is alive in state $S$.

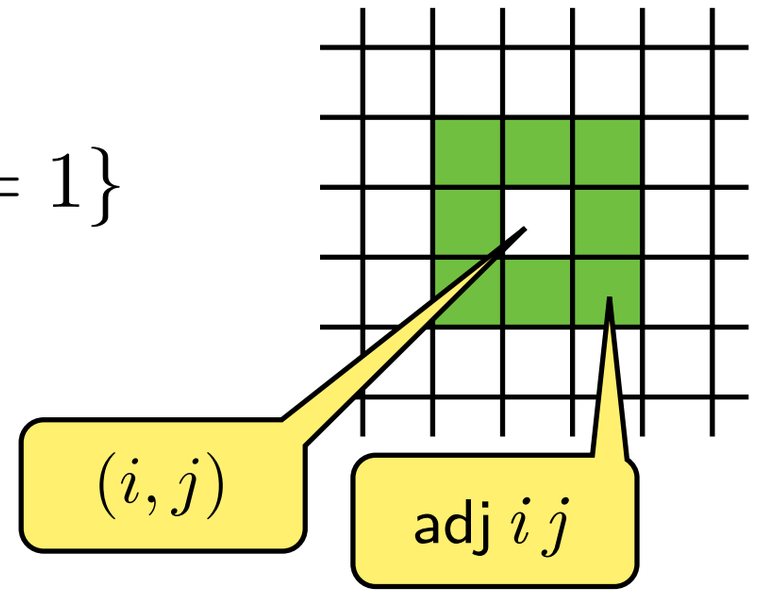
GOL's next-state function

$$\mathsf{step}\ S \stackrel{\mathsf{def}}{=} \{\ (i,j) \mid \text{if } (i,j) \in S \text{ then } \mathsf{live\_adj}\ S\ i\ j \in \{2,3\} \text{ else } \mathsf{live\_adj}\ S\ i\ j = 3\}$$

where

$$\mathsf{adj}\ i\ j \stackrel{\mathsf{def}}{=} \{(i', j') \mid \max(|i' - i|, |j' - j|) = 1\}$$

$$\mathsf{live\_adj}\ S\ i\ j \stackrel{\mathsf{def}}{=} \mathsf{card}\ (S \cap \mathsf{adj}\ i\ j)$$



# Demos

GOL rules, gliders, space ships, useful collisions

# Blog posts

WRITING

## Improved Logic Gates on Conway's Game of Life - Part 3

by Nicholas Carlini 2021-03-23

---

This is the third in a series of posts ([1], [2], [4], [5]) implementing digital logic gates on top of Conway's Game of Life, with the final goal of designing a fully functional CPU.

link to blog

# GOL in GOL?

Can we build a "mega cell" circuit in GOL such that:

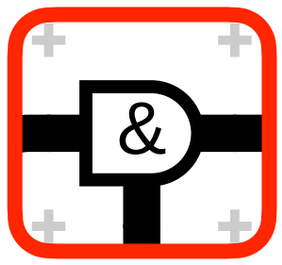
if the entire space is tiled with these mega cells,
the joint behaviour is to simulate GOL?

*Yes!*

Each mega cell needs be a circuit that behaves like a GOL cell. How do we build such a circuit?

Our mega cell design

Each mini-tile is a gate:




Mini-tile: 150x150 GOL cells

# This talk:

## *What is Conway's Game of Life (GOL)?*

rules, gliders, space ships, useful collisions,

circuits in GOL, GOL in GOL

## *Formal verification of circuits in GOL*

symbolic simulation for individual gates,

compositional reasoning about (circuits in) GOL

# This talk:

## *What is Conway's Game of Life (GOL)?*

rules, gliders, space ships, useful collisions,
circuits in GOL, GOL in GOL

## *Formal verification of circuits in GOL*

symbolic simulation for individual gates,

compositional reasoning about (circuits in) GOL

# This talk:

## *What is Conway's Game of Life (GOL)?*

rules, gliders, space ships, useful collisions,
circuits in GOL, GOL in GOL

## *Formal verification of circuits in GOL*

symbolic simulation for individual gates,

compositional reasoning about (circuits in) GOL

# Compositional reasoning about GOL

***Intuition:***

Patterns that are far from each other do not affect each other.

***In the ITP:***

## Define area of influence:

$$\mathsf{infl}\, S \stackrel{\mathsf{def}}{=} \{\, (i', j') \mid \exists\, i\, j.\, (i, j) \in S \wedge \mathsf{max}\,(|i' - i|, |j' - j|) \leq 1 \,\}$$

## Prove:

$$\vdash \mathsf{infl}\ S \cap \mathsf{infl}\ S' = \emptyset \Rightarrow \mathsf{step}\ (S \cup S') = \mathsf{step}\ S \cup \mathsf{step}\ S'$$

# What about communication?

***Intuition:***

A logic gate hands over signals to the next gate.

***In the ITP:***

$$\begin{aligned}
&\text{io_step } c\ S_1\ S_3 \stackrel{\text{def}}{=} \\
&\quad \exists\, S_2. \\
&\qquad \text{infl } S_1 \subseteq c.\text{area} \wedge \text{step } S_1 = S_2 \wedge \\
&\qquad S_2 \cap c.\text{assert_area} = c.\text{assert_content} \wedge \\
&\qquad S_3 = c.\text{insertions} \cup (S_2 - c.\text{deletions})
\end{aligned}$$

configuration

specifies owned area

GOL next-state function

assertions

insertion of input

deletion of output

# Runs of GOL with I/O

We can run many steps:

$$\text{io\_steps}\ 0\ c\ n\ S_1\ S_2 \stackrel{\text{def}}{=} S_1 = S_2$$

$$\text{io\_steps}\ (\text{Suc}\ k)\ c\ n\ S_1\ S_3 \stackrel{\text{def}}{=} \exists\, S_2.\ \text{io\_step}\ (c\ n)\ S_1\ S_2 \wedge \text{io\_steps}\ k\ c\ (n+1)\ S_2\ S_3$$

an infinite stream of configurations

A successful run is:

$$\text{run}\ c\ S \stackrel{\text{def}}{=} \forall\, k.\ \exists\, S'.\ \text{io\_steps}\ k\ c\ 0\ S\ S'$$

can run any number of steps without error

# Circuits

We build the notion of circuit on top 'run':

$$\text{circuit\_run}\ area\ ins\ outs\ init \stackrel{\text{def}}{=}$$
$$\quad \text{run}\ (\text{circ\_mod}\ area\ ins\ outs)\ init \wedge$$
$$\quad \text{circ\_mod\_wf}\ area\ ins\ outs$$

Example gate specification:

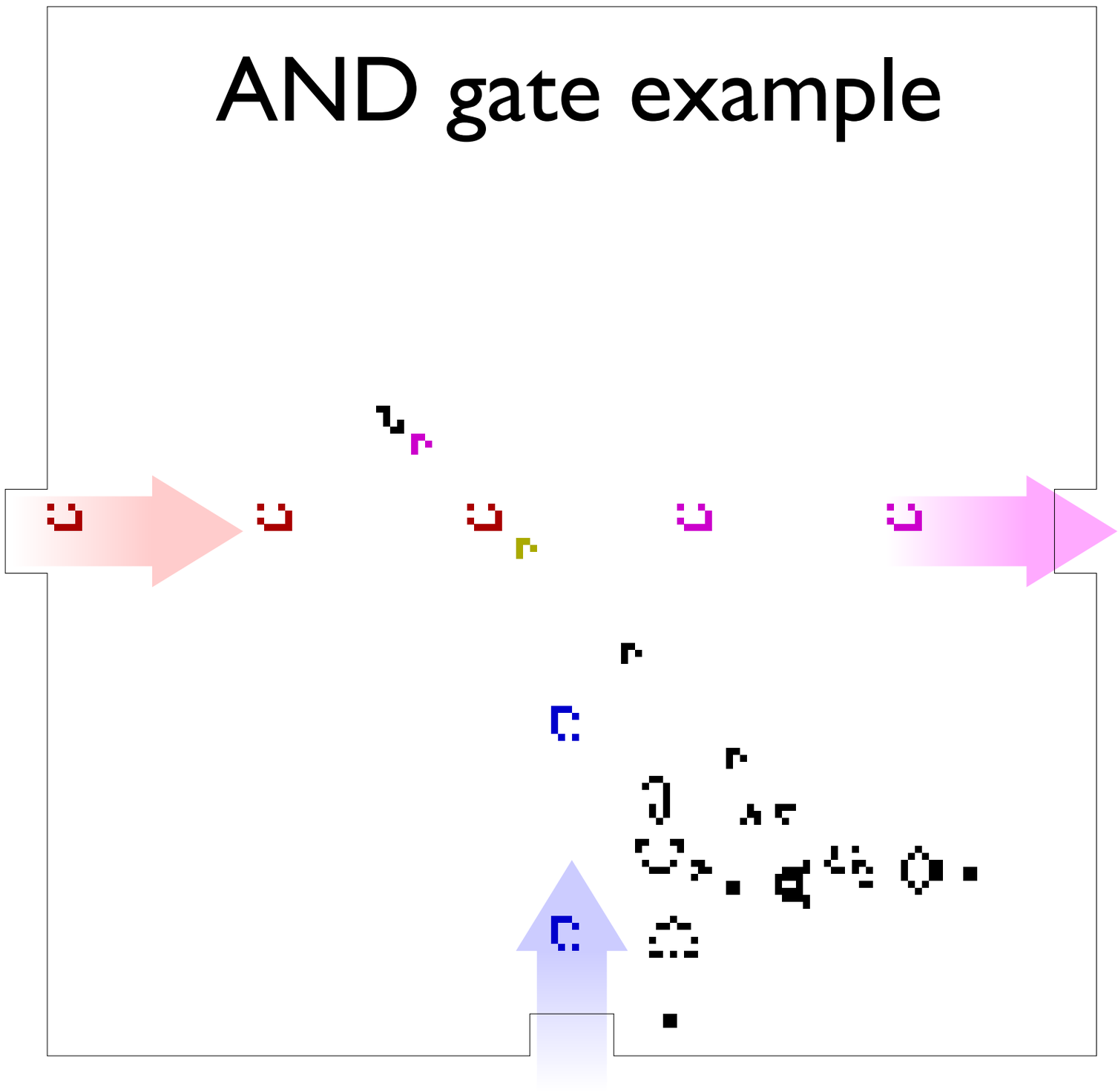
an infinite stream of booleans

another infinite stream

$$\vdash \text{circuit\_run}\ \{(0,0)\}\ \{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b)\}$$
$$\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]}\}\ \text{and\_gate\_pattern}$$

the pointwise conjunction of the two streams, delayed by 5 steps

# AND gate example



$\vdash$ circuit_run $\{(0,0)\}$ $\{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b)\}$
$\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]}\}$ and_gate_pattern

# AND gate example

***How is a theorem like this proved in HOL?***

$$\vdash \mathsf{circuit\_run}\ \{(0,0)\}\ \{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b)\}$$
$$\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]}\}\ \mathsf{and\_gate\_pattern}$$

***Answer:***

1. We define a symbolic simulator that runs circuit io_step.
2. We use an external program to find a symbolic fixed-point.
3. We run the symbolic simulator for 60 GOL steps in logic to confirm that the given symbolic state is a fixed point.

# Composing two circuits

circuit 1

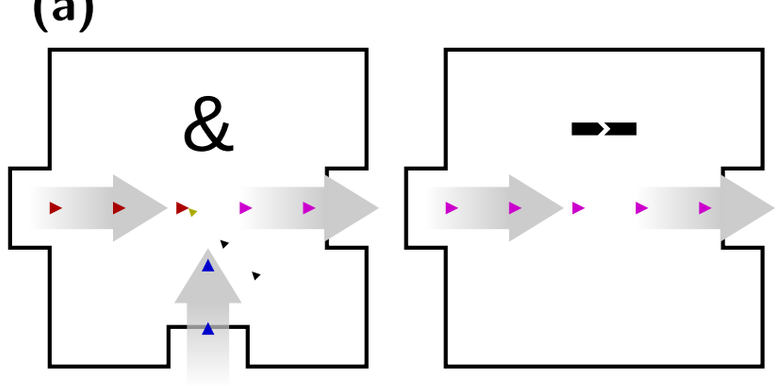
circuit 2

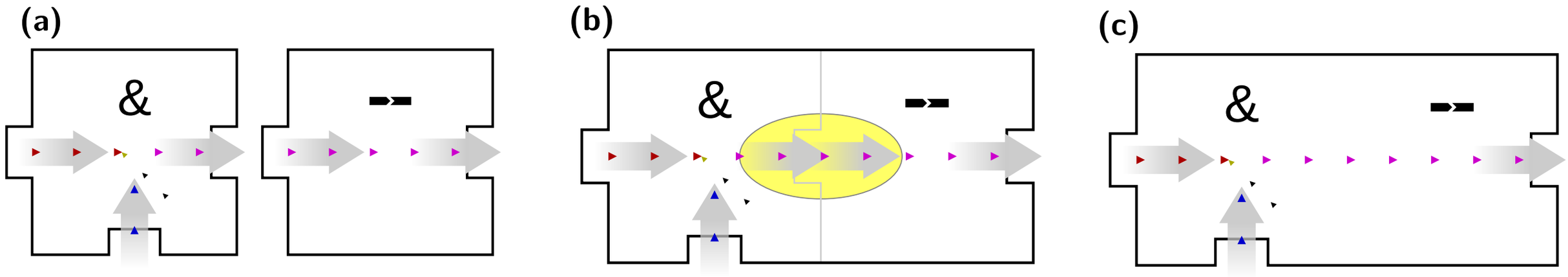
$$\vdash \text{circuit\_run } a_1\ ins_1\ outs_1\ init_1 \wedge \text{circuit\_run } a_2\ ins_2\ outs_2\ init_2 \wedge a_1 \cap a_2 = \emptyset \wedge$$
$$(\forall\, p\ d\ r.$$
$$((p,d,r) \in ins_1 \wedge p - d \in a_2 \Rightarrow (p,d,r) \in outs_2) \wedge$$
$$((p,d,r) \in outs_1 \wedge p + d \in a_2 \Rightarrow (p,d,r) \in ins_2) \wedge$$
$$((p,d,r) \in ins_2 \wedge p - d \in a_1 \Rightarrow (p,d,r) \in outs_1) \wedge$$
$$((p,d,r) \in outs_2 \wedge p + d \in a_1 \Rightarrow (p,d,r) \in ins_1)) \Rightarrow$$
$$\text{circuit\_run } (a_1 \cup a_2)\ (ins_1 \cup ins_2)\ (outs_1 \cup outs_2)\ (init_1 \cup init_2)$$

result is the union of their components

area, inputs, outputs, initial states

# Composition example

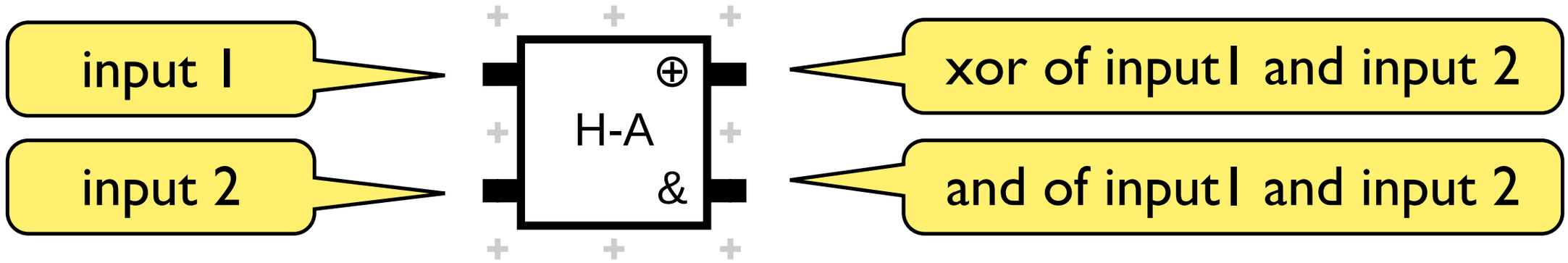





1. $\vdash$ circuit_run $\{(0,0)\}$ $\{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b)\}$ AND gate spec
   $\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]}\}$ and_gate_pattern
2. $\vdash$ circuit_run $\{(0,0)\}$ $\{((-1,0),\mathsf{E},a)\}$ $\{((1,0),\mathsf{E},a^{[5]}\}$ $\emptyset$ wire spec
3. $\vdash$ circuit_run $\{(2,0)\}$ $\{((1,0),\mathsf{E},a)\}$ $\{((3,0),\mathsf{E},a^{[5]}\}$ $\emptyset$ translate (2)
4. $\vdash$ circuit_run $\{(2,0)\}$ $\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]})\}$ substitute (3)
   $\{((3,0),\mathsf{E},(a^{[5]} \wedge b^{[5]})^{[5]}\}$ $\emptyset$
5. $\vdash$ circuit_run $\{(0,0),(2,0)\}$ compose (1,4)
   $\{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b),((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]})\}$
   $\{((1,0),\mathsf{E},a^{[5]} \wedge b^{[5]}),((3,0),\mathsf{E},(a^{[5]} \wedge b^{[5]})^{[5]}\}$
   and_gate_pattern
6. $\vdash$ circuit_run $\{(0,0),(2,0)\}$ $\{((-1,0),\mathsf{E},a),((0,1),\mathsf{N},b)\}$ internalize (5)
   $\{((3,0),\mathsf{E},a^{[10]} \wedge b^{[10]}\}$ and_gate_pattern

# Problem

The GOL "mega cell" circuit uses half-adders:




## Too precise specification:

$\vdash$ circuit_run $\{(0,0),(2,0),(0,2),(2,2)\}$ $\{((-1,0),\mathsf{E},a),((-1,2),\mathsf{E},b)\}$
$\{((3,0),\mathsf{E},(a^{[15]} \wedge ((a^{[12]} \wedge \neg b^{[18]}) \vee (\neg a^{[12]} \wedge b^{[15]} \wedge \neg b^{[18]}))) \vee (\neg a^{[15]} \wedge (a^{[12]} \vee b^{[15]})))$
$((3,2),\mathsf{E},a^{[17]} \wedge b^{[15]})\}$
half_adder_gate_pattern

xor of *a* and *b* ?

kind of if you squint very very hard….

# Mario to the rescue

Mario developed a layer on top that talks about stable signals:

$\vdash$ circuit_run' $\{(0,0),(2,0),(0,2),(2,2)\}$ $\{((-1,0),\mathsf{E},A),((-1,2),\mathsf{E},B)\}$
$\{((3,0),\mathsf{E},A^{[15]} \oplus B^{[18]}),((3,2),\mathsf{E},A^{[17]} \wedge B^{[15]}\}$
half_adder_gate_pattern

this is the stable output

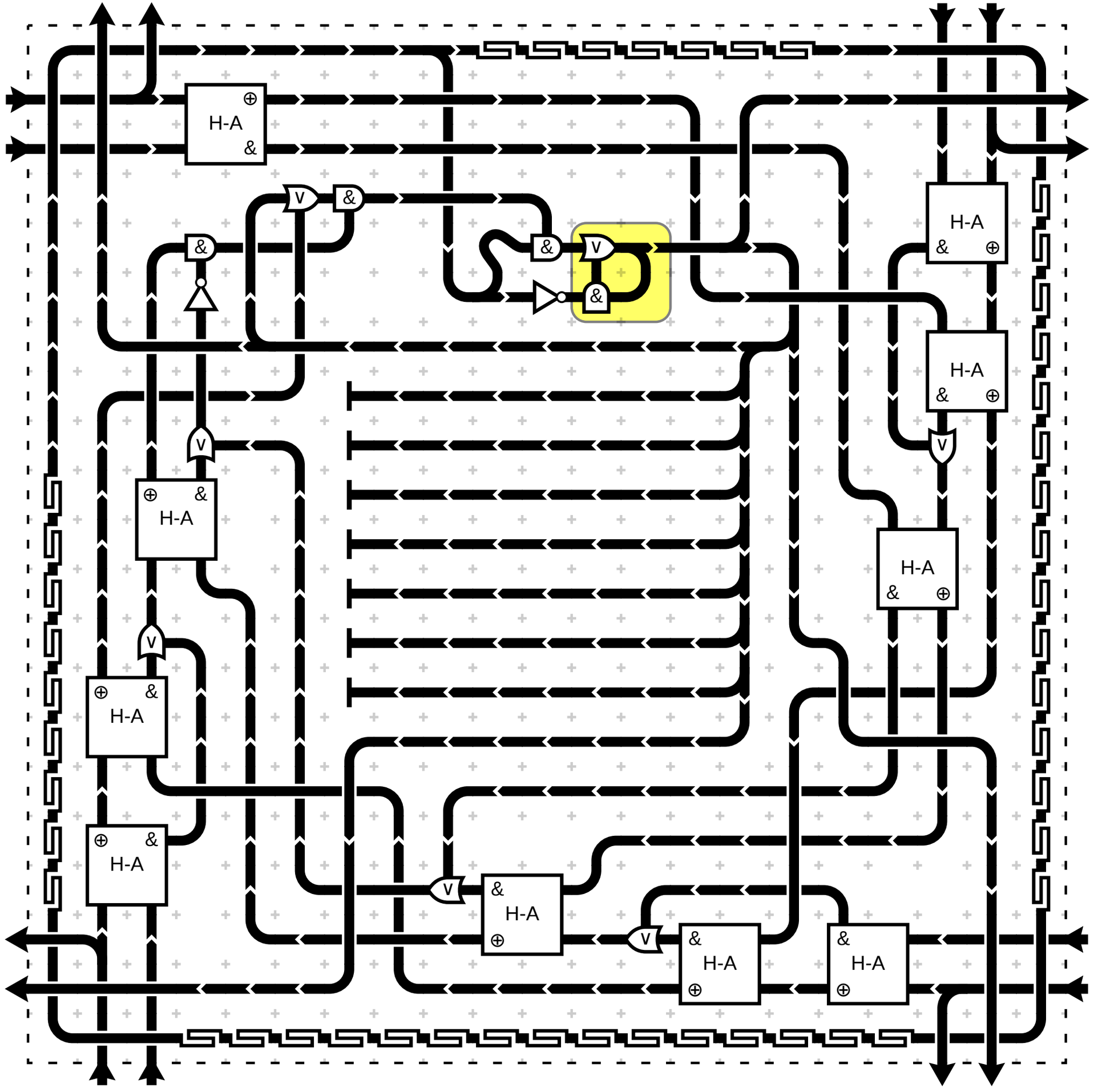
Too precise specification:

$\vdash$ circuit_run $\{(0,0),(2,0),(0,2),(2,2)\}$ $\{((-1,0),\mathsf{E},a),((-1,2),\mathsf{E},b)\}$
$\{((3,0),\mathsf{E},(a^{[15]} \wedge ((a^{[12]} \wedge \neg b^{[18]}) \vee (\neg a^{[12]} \wedge b^{[15]} \wedge \neg b^{[18]}))) \vee (\neg a^{[15]} \wedge (a^{[12]} \vee b^{[15]})))$
$((3,2),\mathsf{E},a^{[17]} \wedge b^{[15]})\}$
half_adder_gate_pattern

xor of *a* and *b* ?

kind of if you squint very very hard….

We proved:



## *Top-level correctness theorem:*

$$\vdash \forall\, n\ S.\ \mathsf{step}^{n}\ S = \mathsf{read\_mega\_cells}\ (\mathsf{step}^{n \times 60 \times 586}\ (\mathsf{build\_mega\_cells}\ S))$$

For any initial state *S* …

… we can build a mega cell tiling …

… such that a run of the mega cells …

… results in state from which we can read …

… what a direct run of the GOL steps would have produced.

# The End — *Questions?*

## 6 Conclusion, Related Work and Future Work

In this paper, we have demonstrated that it is possible to formally verify circuits built in GOL and we have verified a circuit that implements GOL itself inside GOL. To the best of our knowledge, this is the first work to formally verify, in an interactive theorem prover (ITP), constructions in a cellular automata. The formalization is roughly 9 500 LOC.

There has been significant prior work on formalizing more traditional models of computation in ITPs, e.g., Turing machines [11, 2, 20, 6], register machines [12, 3], $\lambda$-calculus [16, 13, 10, 9], $\mu$-recursive functions [5] and more [17]. We refer to Forster [8] for more in depth discussions on computability in ITPs. Rule 110 [7] is another simple universal CA.

In future work, it would be interesting to explore ITP proofs connecting GOL with more traditional forms of computability. Also, the tools used here could be generalized to prove other GOL circuits, other cellular automata, as well as low level hardware correctness proofs.