Mapping Finite State Machines to zk-SNARKs Using Category Theory

A talk for DEVCON V

Fabrizio Genovese, Andre Knispel, Joshua Fitzgerald

Statebox Team

8 Oct 2019, Osaka JP

arxiv.org/abs/1909.02893

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

- They are good for privacy!

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

- They are good for privacy!
- They compress information by a lot;

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

- They are good for privacy!
- They compress information by a lot;
- Blockchain people seem to love them.

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

- They are good for privacy!
- They compress information by a lot;
- Blockchain people seem to love them.

Knowing what a zk-SNARK is is not important for this talk!

What is a zk-SNARK??

zk-SNARKs are a way to prove that some computation followed some rules, without revealing anything of the computation itself.

- They are good for privacy!
- They compress information by a lot;
- Blockchain people seem to love them.

Knowing what a zk-SNARK is is not important for this talk!

What is important is to know that if you have a **boolean circuit**, then you can turn it into a zk-SNARK.

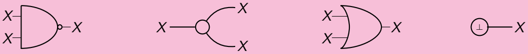
...Then what is a boolean circuit?

...Then what is a boolean circuit?

Literally, a bunch of logical gates wired together.

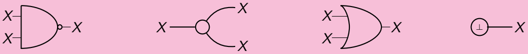
...Then what is a boolean circuit?

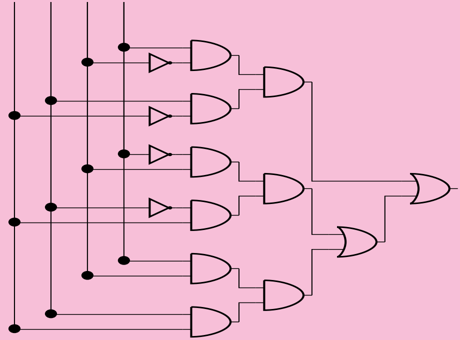
Literally, a bunch of logical gates wired together.



...Then what is a boolean circuit?

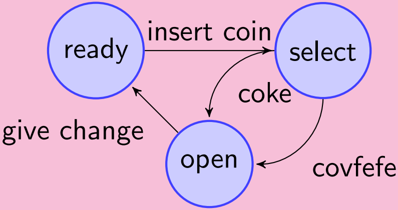
Literally, a bunch of logical gates wired together.





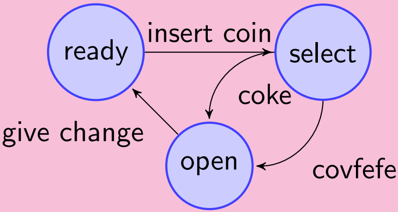
What is a Finite State Machine (FSM)?

What is a Finite State Machine (FSM)?



A FSM is some very old and cute thing in theoretical computer science.

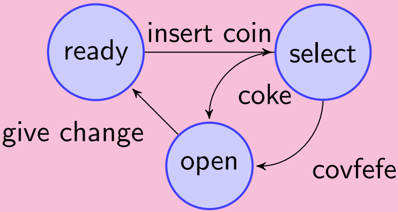
What is a Finite State Machine (FSM)?



A FSM is some very old and cute thing in theoretical computer science.

For us, a FSM will just be a graph:

What is a Finite State Machine (FSM)?

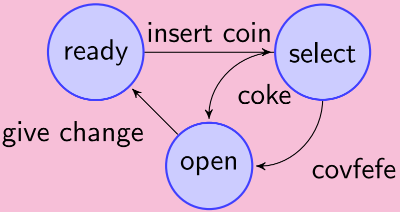


A FSM is some very old and cute thing in theoretical computer science.

For us, a FSM will just be a graph:

- Vertexes of the graphs represent states of the FSM;

What is a Finite State Machine (FSM)?

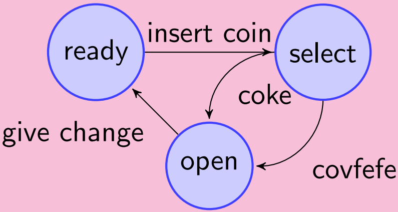


A FSM is some very old and cute thing in theoretical computer science.

For us, a FSM will just be a graph:

- Vertexes of the graphs represent states of the FSM;
- An arc between two vertexes represents an operation that mutates the state of the FSM;

What is a Finite State Machine (FSM)?

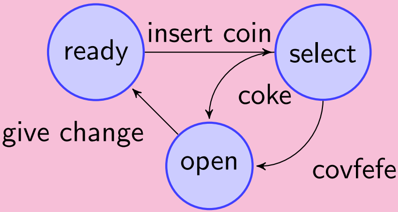


A FSM is some very old and cute thing in theoretical computer science.

For us, a FSM will just be a graph:

- Vertexes of the graphs represent states of the FSM;
- An arc between two vertexes represents an operation that mutates the state of the FSM;
- A computation that follows the rules of the FSM is just a path on the graph.

What is a Finite State Machine (FSM)?



A FSM is some very old and cute thing in theoretical computer science.

For us, a FSM will just be a graph:

- Vertexes of the graphs represent states of the FSM;
- An arc between two vertexes represents an operation that mutates the state of the FSM;
- A computation that follows the rules of the FSM is just a path on the graph.

If we can map graph paths to boolean circuits, then we can get a zk-SNARK verifying that a given computation followed the FSM rules!

Yeah but WTF is category theory anyway?

Yeah but WTF is category theory anyway?

Category theory is a very complicated way to do simple things.

Yeah but WTF is category theory anyway?

Category theory is a very complicated way to do simple things.

The reason why it's cool is that it scales better than traditional maths, so super-complicated things are actually easier when done categorically.

Yeah but WTF is category theory anyway?

Category theory is a very complicated way to do simple things.

The reason why it's cool is that it scales better than traditional maths, so super-complicated things are actually easier when done categorically.

Also, it's super related to type theory, so functional programming people love it, and functional programming is cool.

Yeah but WTF is category theory anyway?

Category theory is a very complicated way to do simple things.

The reason why it's cool is that it scales better than traditional maths, so super-complicated things are actually easier when done categorically.

Also, it's super related to type theory, so functional programming people love it, and functional programming is cool.

Category theory allows to relate different mathematical structures compositionally: That is, in a way that respects the structure of the things we are relating.

This means that...

This means that...

If we build a categorical correspondence between graphs and boolean circuits, then we can be sure that if we morph the graph the circuit will morph accordingly.

This means that...

If we build a categorical correspondence between graphs and boolean circuits, then we can be sure that if we morph the graph the circuit will morph accordingly.

Moreover, once categorified concepts are automatically related to all mathematics: If you have a categorical way to map what you care about to graphs, then you automatically have a way to map that thing to boolean circuits (and hence to get a zk-SNARK for it!)

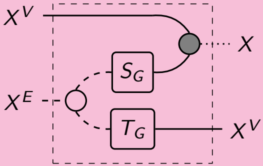
The category of paths for a graph

Each graph G can be used to build a category $\mathfrak{F}(G)$, which represents all the possible paths in the graph:

$$\begin{tikzcd}
G \arrow[r, mapsto] \arrow[d, "f"'] & \mathfrak{F}(G) \arrow[d, "\mathfrak{F}(f)"] \\
G' \arrow[r, mapsto] & \mathfrak{F}(G')
\end{tikzcd}$$

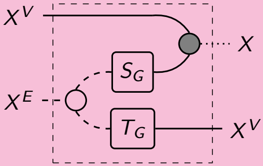
Circuits from graphs

We can use the adjacency matrix of a graph G to build a circuit:



Circuits from graphs

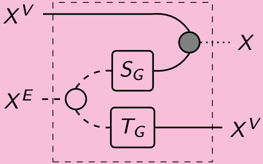
We can use the adjacency matrix of a graph G to build a circuit:



The left wires receive the enumeration of a vertex (top), and of an edge (bottom) respectively.

Circuits from graphs

We can use the adjacency matrix of a graph G to build a circuit:

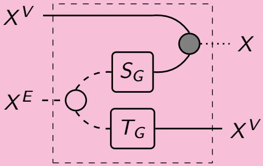


The left wires receive the enumeration of a vertex (top), and of an edge (bottom) respectively.

The wire up right returns 1 if the vertex is the source of the edge, 0 otherwise. The bottom right wire returns the target vertex of the edge.

Circuits from graphs

We can use the adjacency matrix of a graph G to build a circuit:



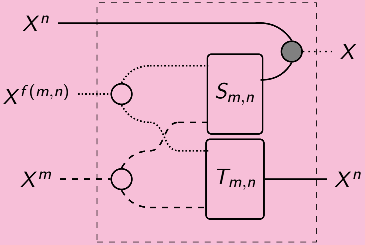
The left wires receive the enumeration of a vertex (top), and of an edge (bottom) respectively.

The wire up right returns 1 if the vertex is the source of the edge, 0 otherwise. The bottom right wire returns the target vertex of the edge.

We can clearly compose these circuits by piping them one into the other. For each category of paths $\mathfrak{F}(G)$, we are are able to get a functor – a structure preserving map – to the category of boolean circuits.

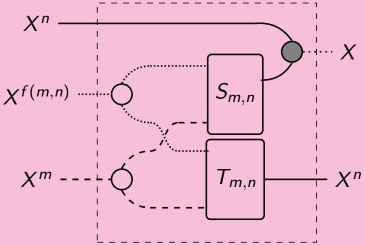
Generalizing over graphs

We can abstract over the adjacency matrix obtaining the following circuit:



Generalizing over graphs

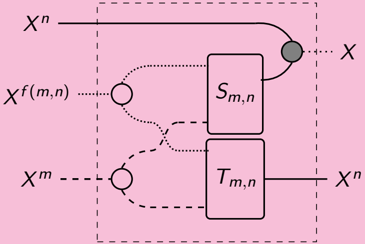
We can abstract over the adjacency matrix obtaining the following circuit:



The second wire on the left accepts a specification of an adjacency matrix of $m \times n$ bits. After having specified this, the circuit behaves like the one in the previous slide.

Generalizing over graphs

We can abstract over the adjacency matrix obtaining the following circuit:



The second wire on the left accepts a specification of an adjacency matrix of $m \times n$ bits. After having specified this, the circuit behaves like the one in the previous slide.

Again the circuits are composable and we obtain a functor from the category of graphs to the category of boolean circuits.

The power of category theory

Everything we built is compositional!!

$$
\begin{tikzcd}
 & & & \mathbb{B}^{f(m,n)}_{\mathbf{ZKP}} \arrow[rrr, equal] \arrow[dd] & & & \mathbb{B}^{f(m,n)}_{\mathbf{ZKP}} \arrow[dd] \\
 & & \mathbf{Count} \arrow[ur] \arrow[rrr] \arrow[ddd, equal] & & & \mathbb{B}^{G}_{\mathbf{Graph}} \arrow[ur] \arrow[ddd] & \\
 & & & \mathbb{B}^{f(m',n')}_{\mathbf{ZKP}} \arrow[rrr, equal] & & & \mathbb{B}^{f(m',n')}_{\mathbf{ZKP}} \\
G \arrow[r] \arrow[dd] & \mathfrak{F}(G) \arrow[uur] \arrow[rrr] \arrow[dd] & & & \mathbb{B}^{G}_{\mathbf{path}} \arrow[uur] \arrow[dd] & & \\
 & & \mathbf{Count} \arrow[uur] \arrow[rrr] & & & \mathbb{B}^{G'}_{\mathbf{Graph}} \arrow[uur] & \\
G' \arrow[r] & \mathfrak{F}(G') \arrow[uur] \arrow[rrr] & & & \mathbb{B}^{G'}_{\mathbf{path}} \arrow[uur] & &
\end{tikzcd}
$$

Take home message:

Take home message:

When you do things categorically there's a good chance that:

Take home message:

When you do things categorically there's a good chance that:

- They will be formally correct;

Take home message:

When you do things categorically there's a good chance that:

- They will be formally correct;
- You will be able to verify that your constructions satisfy some nice properties;

Take home message:

When you do things categorically there's a good chance that:

- They will be formally correct;
- You will be able to verify that your constructions satisfy some nice properties;
- They will be more easily implementable in a formally verified setting, since already well structured;

Take home message:

When you do things categorically there's a good chance that:

- They will be formally correct;
- You will be able to verify that your constructions satisfy some nice properties;
- They will be more easily implementable in a formally verified setting, since already well structured;
- You will be able to link things together without making a mess.

Take home message:

When you do things categorically there's a good chance that:

- They will be formally correct;
- You will be able to verify that your constructions satisfy some nice properties;
- They will be more easily implementable in a formally verified setting, since already well structured;
- You will be able to link things together without making a mess.

Category theory seems difficult in the beginning, but it's just a better method to do software engineering!


We are finished
THANKS
QUESTIONS PLEASE