110 Rules for Prometheus

Brian Brazil
Founder

Robust Perception

Rule 110

~~110 Rules~~ for Prometheus

Brian Brazil
Founder

Robust Perception

Who am I?

- One of the core developers of Prometheus
- Founder of Robust Perception
- Primary author of Reliable Insights blog
- Contributor to many open source projects
- Ex-Googler, after 7 years in the Dublin office

You may have heard of me :)

Robust **Perception**

Looking Back..

Last year I gave a lightning talk on "An Exploration of the Formal Properties of PromQL" demonstrating that PromQL was Turing Complete via Conway's Life.

But maybe you missed that. Or found it a bit ***too*** formal.

So let's have another go.

Rule 110

Rule 110 is a linear cellular automata. It's one dimensional compared, to Conway's Life's two dimensions. Also Turing Complete.

It follows the following rule on each iteration:

111	110	101	100	011	010	001	000
0	1	1	0	1	1	1	0

The bottom number is 110 in binary.

Doing this in PromQL

We could create a `state` metric with 0s and 1s and then do something custom to visualise how it changes over time.

Sounds like a lot of work.

But we already have things that visualise changes in state over time, such as the expression browser.

Could we make that work?

New in Prometheus 2.0

We could have the values for the various cells be 0 if it's dead, or 1/2/3/4/etc. according to the cell number if it's alive.

Even better would be if we could show the gaps as gaps - which we can do now with Prometheus 2.0 staleness and expression browser updates!

So thanks to whoever implemented those changes!

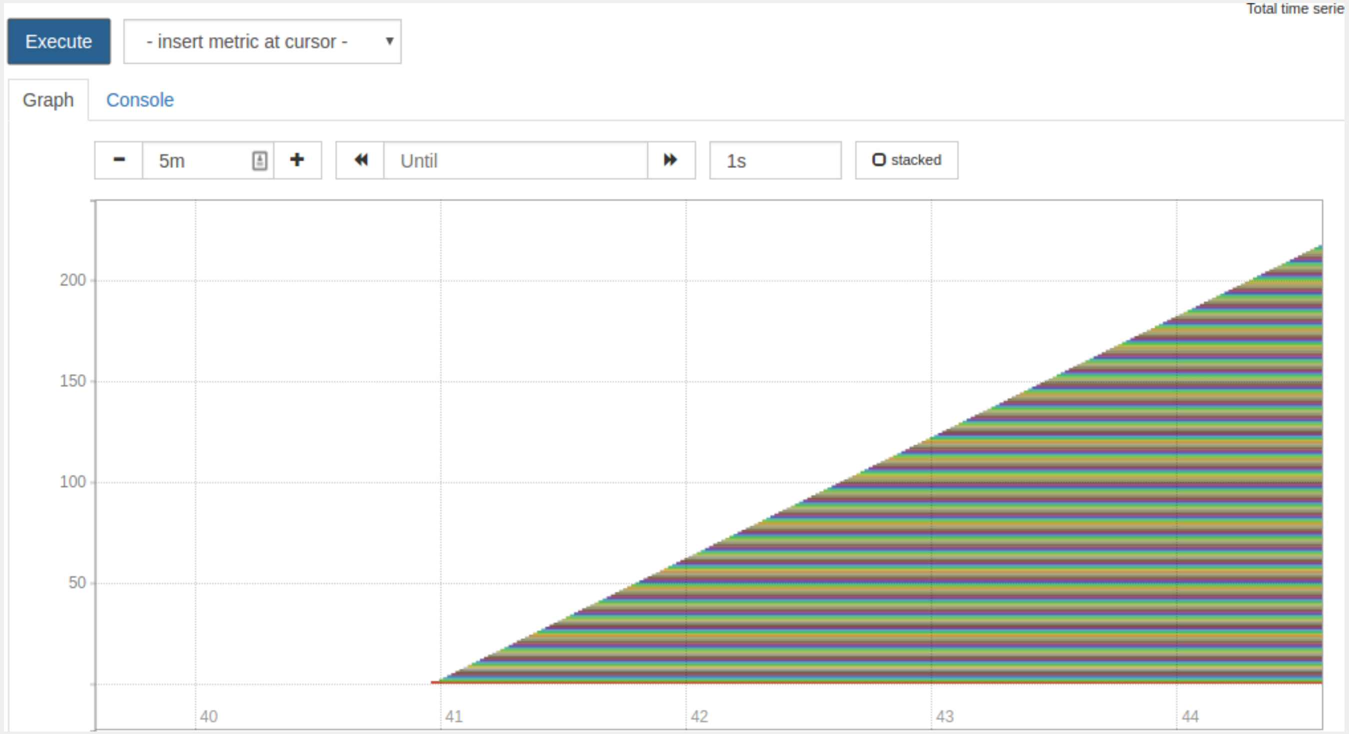
Start small...

```
init{x="1"} = 1

state =
    state
  or
    (label_replace(state, "x", "1$1", "x", "^(.*)$")) + 1
  or
    init
```

Rule 110 only grows left, so don't need to worry about negative numbers.

Start small...



Robust Perception

Da Rules - 011

```
(
    state
  unless
    label_replace(state, "x", "1$1", "x", "^(.*)$")
  and
    label_replace(state, "x", "$1", "x", "^1(.*)$")
)
```

Keep alive if the left cell missing/empty and the right cell is present/alive.

Robust **Perception**

Da Rules - 110 & 010

```
(
    state
  unless
    label_replace(state, "x", "$1", "x", "^1(.*)$")
)
```

Keep the cell alive if the cell to the right is missing/empty.

Robust **Perception**

Da Rules - 101 & 001

```
(
    (label_replace(state, "x", "1$1", "x", "^(.*)$")) + 1
  unless
    state
)
```

Change cell is alive it's currently missing/empty, and the cell to the right is present/alive.

As this cell is missing, we create the cell labels from the right cell.

Robust **Perception**

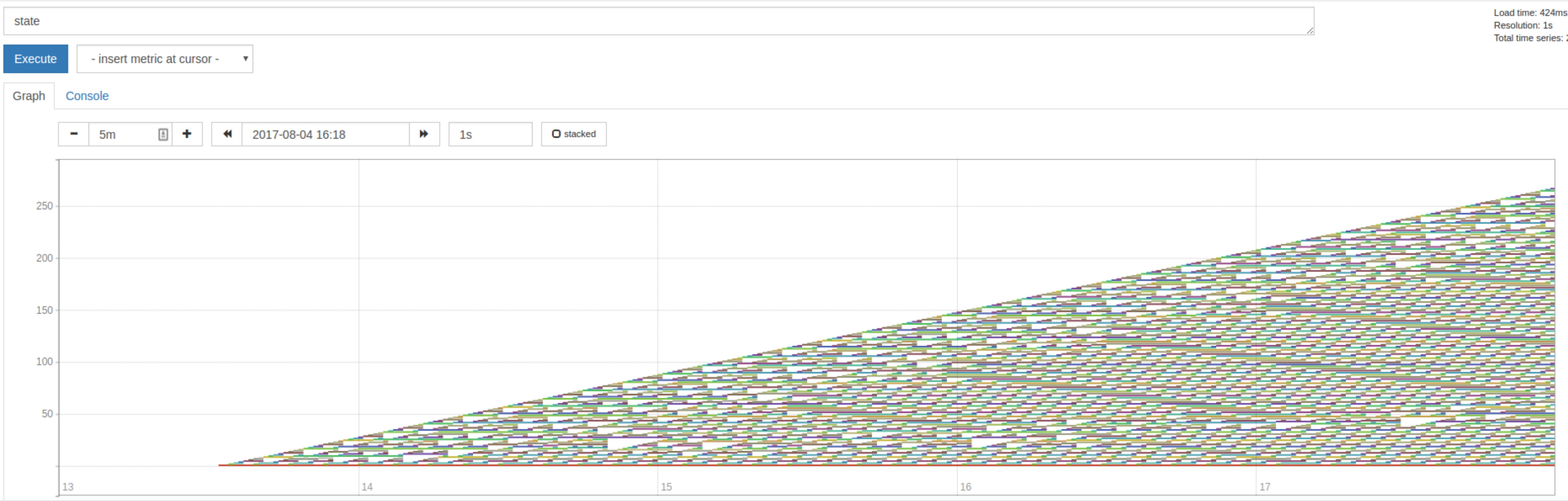
Da Rules - Other cases

Finally we `or` these three expressions together.

All the other rules produce empty, so no need to mention them.

We also `or` in `init` to get things started. The rightmost cell is meant to stay 1 forever, so this is fine.

Result!


state

Load time: 424ms
Resolution: 1s
Total time series: 2

Robust Perception

What have we learned?

- PromQL is still Turing Complete
- Rule 110, which is Turing Complete, was implementable in PromQL
- Does who know what "Turing Tarpit" means are utterly unsurprised by this
- New staleness can be used for crazy inadvisable things
- Brian may know some things about PromQL

Resources

Robust Perception Blog: www.robustperception.io/blog

Queries: prometheus@robustperception.io

Robust **Perception**