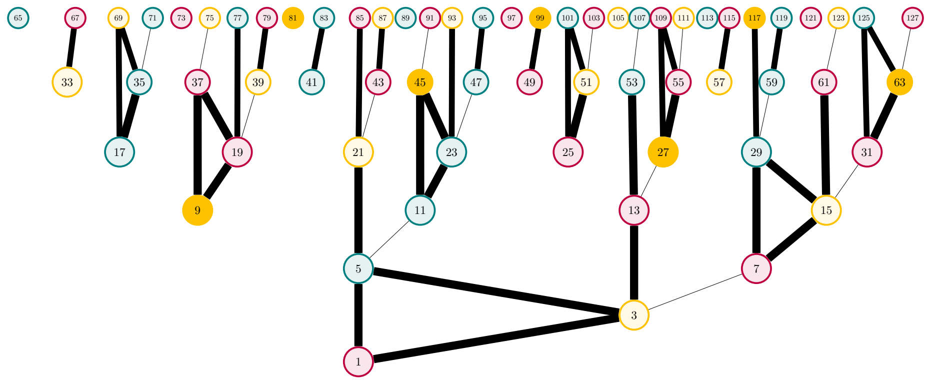
Golden Automaton in pseudocode

November 13, 2020

1 Basics

The Golden Automaton is a Quiver with a non-uniform branching factor that is traversing the binary tree over $2\mathbb{N}+1$, all the while proving the convergence of odd numbers under the Collatz dynamic.

2 The phase space



How it should look from 1 to 127

3 The fundamental rules

Let us define

- V(x):=4x+1
- S(x):=2x+1
- G(x):=2x-1
- ”type A”, a+1 is dividable by 3
- ”type B”, b is dividable by 3
- ”type C”, c-1 is dividable by 3

The following rules apply anywhere on $2\mathbb{N}+1$ but applying them from number 3 onward generates a graph we call the ”Golden Quiver”

1. $\forall x$ odd, $V(x) \equiv (x)$
2. $\forall x, k$ odd, $S^k V(x) \equiv S^{k+1} V(x)$ and $\forall x, k$ even, $S^k V(x) \equiv S^{k+1} V(x)$
3. $\forall n, y \in \mathbb{N}^2$, $\forall x$ odd non B, $3^n x \equiv y \Rightarrow \bigwedge_{i=1}^{n} (V(4^i 3^{n-i} x)) \wedge S(V(4^i 3^{n-i} x)) \equiv y$
4. $\forall n, y \in \mathbb{N}^2$, $\forall x$ odd non B, $S(3^n x) \equiv y \Rightarrow \bigwedge_{i=1}^{n} (S(4^i 3^{n-i} x) \wedge S^2(4^i 3^{n-i} x)) \equiv y$
5. $\forall n \in \mathbb{N}$, $\forall y \in \mathbb{N}$, $\forall x$ odd non B where $3^n x$ is of rank 1, $a \equiv y$, $a = G(3^n x) \Rightarrow \bigwedge_{i=0}^{n} (S^i(G(3^{n-i} x)) \wedge S^{i+1}(G(3^{n-i} x))) \equiv y$

4 Pseudocode

1. Generate a scalable phase space up to n=17, where the number on the left is $2^{n-1}+1$ and the number to the right of each line is $2^n + 1$. The figure must be in svg or any other vectorial (zoomable) format. Only generate odd numbers. Display the type A in TEAL, the type B in GOLD, the type C in PURPLE
2. compute the Rules 1 and 2 and represent each vertex they cover in SOLID BLACK
3. compute the Rules 3,4,5 from number 3 onward and represent the shortest vertices they cover (the shortest path to an already connected number) in SOLID GOLD
4. each number that is connected to the growing network by either SOLID GOLD or SOLID BLACK lines is proven to converge and can be used for further computation of rules 3,4,5.

I let you optimise the way rules 3,4 and 5 should be computed for the best possible performances, but I suggest you apply those rules to numbers in their natural order (i.e. starting by 3, then 17, which is the first new number proven by Rule 3, then 21? which is proven by rule 1, then 9, which is proven by rule 4)