# 3-ary Collatz Grammar

by Sven Nilsen, 2021

*In this paper I introduce a 3-ary grammar for cycles the Collatz function.*

The Collatz Conjecture[1], also known as the “3n+1” problem, which states that the following function will reach `1`, regardles of which positive integer is chosen initially:

collatz(x: nat ∧ (> 0)) = if (x % 2) == 0 {x/2} else {3*x+1}

When it reaches `1`, it will produce a cycle, `1-4-2-1`.
This cycle has an orbit length `3`, which means that:

collatz . collatz . collatz <=> id          if $x \in \{1, 2, 4\}$

The `collatz$^3$` function, but for all positive integers, will be the bases of a 3-ary grammar.
Each possible branch in the `collatz$^3$` function is represented by a bit, which gives the following table:

| | **possibility** | **input** | **output** |
|---|---|---|---|
| 000 | possible[1] | even | unknown |
| 001 | possible[2] | even | even |
| 010 | possible[3] | even | unknown |
| 011 | impossible[4] | - | - |
| 100 | possible[5] | odd | unknown |
| 101 | possible[6] | odd | even |
| 110 | impossible[7] | - | - |
| 111 | impossible[8] | - | - |

Proofs are given in the Appendix.

The evenness property of inputs and outputs of possible sequences are assigned `even, odd, unknown`.

This makes it possible to derive the following Piston-Meta[2] grammar for cycles:

```
3 even = .r!([.r?(“001”) .r!({“000” “010”})])
2 odd = [.r?(“100”) ?”101”]
1 cycle = .r!([even odd])
```

A non-terminating cycle stuck in `001`, `100` or `010` leads to the `1-4-2-1` cycle, because a such cycle is decreasing for all real numbers greater than 1.

A cycle is said to start with an even branch because any number has a multiple of `2`.
With other words, the arms of the cycle are ignored, because at least one arm can be constructed.

## References:


[1] “Collatz Conjecture”
Wikipedia
https://en.wikipedia.org/wiki/Collatz_conjecture

[2] “Piston-Meta”
PistonDevelopers – A DSL parsing library for human readable text documents
https://github.com/pistondevelopers/meta

[3] “Catușkoți Existential Lift”
Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/catuskoti-existential-lift.pdf

[4] “Catușkoți Existential Path Equations”
Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/catuskoti-existential-path-equations.pdf

# Appendix

1. Possibility of ` 000`:

| | | |
|---|---|---|
| 0 | (∃{(= true)}collatz[even])(true) = true | Lemma C |
| 0 | (∃{(= true)}collatz[even])(true) = true | Lemma C |
| 0 | collatz[even](true) = unknown | Lemma A |

2. Possibility of `001`:

| | | |
|---|---|---|
| 0 | (∃{(= true)}collatz[even])(true) = true | Lemma C |
| 0 | c(∃{(= true)}collatz[even])(false) = true | Lemma D |
| 1 | collatz[even](false) = true | Lemma B |

3. Possibility of `010`:

| | | |
|---|---|---|
| 0 | (∃{(= true)}collatz[even])(false) = true | Lemma D |
| 1 | collatz[even](false) = true | Lemma B |
| 0 | collatz[even](true) = unknown | Lemma A |

4. Impossibility of `011`:

| | | |
|---|---|---|
| 0 | (∃{(= true)}collatz[even])(false) = true | Lemma D |
| 1 | collatz[even](false) = true | Lemma B |
| 1 | - | |

5. Possibility of `100`:

| | | |
|---|---|---|
| 1 | collatz[even](false) = true | Lemma B |
| 0 | (∃collatz[even]{(= true)})(true) = true | Lemma C |
| 0 | collatz[even](true) = unknown | Lemma A |

6. Possibility of `101`:

| | | |
|---|---|---|
| 1 | collatz[even](false) = true | Lemma B |
| 0 | (∃collatz[even]{(= true)})(false) = true | Lemma D |
| 1 | collatz[even](false) = true | Lemma B |

7. Impossibility of `110`:

| | | |
|---|---|---|
| 1 | collatz[even](false) = true | Lemma B |
| 1 | - | |
| 0 | collatz[even](true) = unknown | Lemma A |

8. Impossibility of `111`:

| | | |
|---|---|---|
| 1 | collatz[even](false) = true | Lemma B |
| 1 | - | |
| 1 | - | |

# Lemma A

collatz[even](true) = unknown

| | | |
|---|---|---|
| ∵ | collatz[even](true) = unknown | |
| ∴ | even . (/ 2) => \\true | Catușkoți existential lift[3] with Kleene's three-value logic[4] |
| ∴ | ∃(even . (/ 2)) <=> \true | |
| ∴ | \true <=> \true | Examples: `2 * 3`, `2 * 2 * 3` |
| ∴ | true | |
| | Q.E.D. | |

# Lemma B

collatz[even](false) = true

∵ collatz[even](false) = true
∴ (even . (\(x) = 3 * x + 1)[even → id])(false) = true
∴ (even . (+ 1) . (\(x : nat) = 3 * x)[even → id])(false) = true
∴ (even . (+ 1) . (* 3) . (\(x : nat) = x)[even → id])(false) = true
∴ (even . (+ 1) . (* 3) . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ (even . (+ 1) . (* 3) . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ ((add[even] even(1)) . even . (* 3) . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ ((eq false) . even . (* 3) . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ (not . even . (* 3) . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ (not . (mul[even] even(3)) . even . $\mathrm{id}_{nat}$[even → id])(false) = true
∴ (not . (or false) . $\mathrm{id}_{nat}$[even → even])(false) = true
∴ (not . $\mathrm{id}_{bool}$ . $\mathrm{id}_{bool}$)(false) = true
∴ not(false) = true
∴ true
Q.E.D.

# Lemma C

(∃collatz[even]{(= true)})(true) = true

∵ (∃collatz[even]{(= true)})(true) = true
∴ (\true)(true) = true
∴ true = true
∴ true
∴ Q.E.D.

# Lemma D

(∃collatz[even]{(= true)})(false) = true

∵ (∃collatz[even]{(= true)})(false) = true
∴ (\true)(false) = true
∴ true = true
∴ true
Q.E.D.