Waterjugs (***)

We have 3 jugs of water, capacity: 8, 5, 3 liters resp. Initially there are 8 liters of water in the first jug, the other two jugs are empty.
Wanted end-situation: 4 liters in the first jug, 4 liters in the second jug, 3rd jug empty.

The jugs are irregularly shaped and unmarked; no water can be spilled; each step pouring water from one jug to another stops when either the source jug is empty or the destination jug is full, whichever happens first.

Use Z3 to find a solution (and present the answer in human readable form).

Variables:

MaxJug:

Function $MaxJug$ represent the capacities as a $MaxJug\colon\ J \to Z^+$, where $MaxJug(a)$ gives the capacity of $a \in J$, where $J = \{1,2,3\}$, each jug has a specific capacity:

- Jug 1 has a capacity of 8 liters.
- Jug 2 has a capacity of 5 liters.
- Jug 3 has a capacity of 3 liters.

Q:

Function Q represents amount of water in Jug at any step $Q\colon J \times Z^+ \to Z^+$.

N:

Const N represents number of steps to solve the puzzle $N \in Z^+$. Solving the problem involves gradually reducing the variable until the task becomes unsat.

Functions:

Pour:

The pour function $Pour\colon J \times J \times J \times Z^+ \to Bool$, defines the condition for water to be transferred from $Jug\ a$, to $Jug\ b$, at time t.

The body of the $Pour$ function consists of a conjunction of several conditions:

- **Change in First Jug:**

$$Q(j1,t) \neq Q(j1,t+1)$$

This condition asserts that the quantity of water in the second jug (**j2**) changes between time steps **t** and **t+1**.

- **Change in Second Jug:**

$$Q(j2,t) \neq Q(j2,t+1)$$

This condition asserts that the quantity of water in the second jug (**j2**) changes between time steps **t** and **t+1**.

- **Change in Second Jug:**

$$Q(j3,t) = Q(j3,t+1)$$

This condition asserts that the quantity of water in the second jug (**j2**) changes between time steps **t** and **t+1**.

- **Pouring Condition:**

$$Q(j1,t+!) = 0 \vee Q(j2,t+1) = MaxJug(t2)$$

This condition represents the core pouring logic. It asserts that either the first jug (**j1**) is empty after pouring (indicating all water was poured out of it), or the second jug (**j2**) is full after pouring (indicating it was filled to its maximum capacity, **maxJug(j2)**).

The function is represented as

$$\begin{gathered} Pour(j1,j2,j3,t) \equiv (Q(j1,t) \neq Q(j1,t+1) \\ \wedge Q(j2,t) \neq Q(j2,t+1) \\ \wedge Q(j3,t) = Q(j3,t+1) \\ \wedge Q(j1,t+!) = 0 \vee Q(j2,t+1) = MaxJug(t2)) \end{gathered}$$

Assertions:

- **Conservation of Water**: The total amount of water remains constant throughout the operations.

$$\forall t \in Z^{+}: \sum_{j \in J} Q(j,t) = 8$$

- **Capacity Constraint**: The amount of water in each jug at any time cannot exceed its capacity.

$$\forall j \in J, \forall t \in Z^{+}: 0 \leq Q(j,t) \leq Capacity(j)$$

- **Pouring Operations**: At each time step t*t*, we can pour water from one jug to another, following the pour function's rules.

$\forall t \in Z^{+}(0 \leq t \leq N) \rightarrow \exists(j1, j2, j3) \in J$:
$j1 \neq j2 \neq j3) \wedge (1 \leq j1 \leq 3) \wedge (1 \leq j2 \leq 3) \wedge (1 \leq j3 \leq 3) \wedge Pour(j1, j2, j3, t)$

Start State:

$$Q(1,0) = 8$$
$$Q(2,0) = 0$$
$$Q(3,0) = 0$$

End State:

$$Q(1, N) = 4$$
$$Q(2, N) = 4$$
$$Q(3, N) = 0$$

Z3 SMT Solution

code:

```
(declare-fun maxJug(Int) Int)

(declare-fun q (Int Int) Int)

(declare-const N Int)

(define-fun pour ((j1 Int) (j2 Int) (j3 Int) (t Int)) Bool
   (and
      (distinct
         (q j1 t)
         (q j1 (+ t 1))
      )
      (distinct
         (q j2 t)
         (q j2 (+ t 1))
      )
      (=
         (q j3 t)
         (q j3 (+ t 1))
      )
      (or
         (= (q j1 (+ 1 t)) 0)
         (= (q j2 (+ 1 t)) (maxJug j2))
      )
   )
)
(assert (= (maxJug 1) 8))
(assert (= (maxJug 2) 5))
(assert (= (maxJug 3) 3))

(assert (= (q 1 0) 8))
(assert (= (q 2 0) 0))
(assert (= (q 3 0) 0))

(assert (= (q 1 N) 4))
(assert (= (q 2 N) 4))
(assert (= (q 3 N) 0))

(assert
   (forall ((t Int))
```

```
      (=> (>= t 0)
         (= 8 (+ (q 1 t) (q 2 t) (q 3 t)))
      )
   )
)

(assert
   (forall ((t Int))
      (=> (>= t 0)
         (and
            (>= (q 1 t) 0)
            (>= (q 2 t) 0 )
            (>= (q 3 t) 0 )

            (<= (q 1 t) (maxJug 1))
            (<= (q 2 t) (maxJug 2) )
            (<= (q 3 t) (maxJug 3))
         )
      )
   )
)

(assert
   (forall ((t Int))
   (implies
      (<= 0 t N)
      (exists ((j1 Int) (j2 Int) (j3 Int))
         (and
            (distinct j1 j2 j3)

            (<= 1 j1 3)
            (<= 1 j2 3)
            (<= 1 j3 3)

            (pour j1 j2 j3 t)
         )
      )
   )
   )
)
```

```
(assert
    (<= 0 N 7)
)
(check-sat)
(get-value (
  (q 1 0)
  (q 2 0)
  (q 3 0)

  (q 1 1)
  (q 2 1)
  (q 3 1)

  (q 1 2)
  (q 2 2)
  (q 3 2)

  (q 1 3)
  (q 2 3)
  (q 3 3)

  (q 1 4)
  (q 2 4)
  (q 3 4)
  (q 1 5)
  (q 2 5)
  (q 3 5)
  (q 1 6)
  (q 2 6)
  (q 3 6)
  (q 1 7)
  (q 2 7)
  (q 3 7)
))
```

Output:

```
(((q 1 0) 8)
 ((q 2 0) 0)
 ((q 3 0) 0)
```

```
((q 1 1) 3)
((q 2 1) 5)
((q 3 1) 0)
((q 1 2) 3)
((q 2 2) 2)
((q 3 2) 3)
((q 1 3) 6)
((q 2 3) 2)
((q 3 3) 0)
((q 1 4) 6)
((q 2 4) 0)
((q 3 4) 2)
((q 1 5) 1)
((q 2 5) 5)
((q 3 5) 2)
((q 1 6) 1)
((q 2 6) 4)
((q 3 6) 3)
((q 1 7) 4)
((q 2 7) 4)
((q 3 7) 0))
```

Python Visual:

```
Time 0 to 1: Jug 1 poured 5 units to Jug 2. State: Jug1=3, Jug2=5, Jug3=0
Time 1 to 2: Jug 2 poured 3 units to Jug 3. State: Jug1=3, Jug2=2, Jug3=3
Time 2 to 3: Jug 3 poured 3 units to Jug 1. State: Jug1=6, Jug2=2, Jug3=0
Time 3 to 4: Jug 2 poured 2 units to Jug 3. State: Jug1=6, Jug2=0, Jug3=2
Time 4 to 5: Jug 1 poured 5 units to Jug 2. State: Jug1=1, Jug2=5, Jug3=2
Time 5 to 6: Jug 2 poured 1 units to Jug 3. State: Jug1=1, Jug2=4, Jug3=3
Time 6 to 7: Jug 3 poured 3 units to Jug 1. State: Jug1=4, Jug2=4, Jug3=0
```