MODULE $GCD1$

Our goal here is to prove that an algorithm solves the problem that it claims to solve. Namely to prove a particular form of *Euclid*'s algorithm, but using TLA+ instead of mathematics. See the paper "*Euclid* Writes an Algorithm: A Fairy Tale" by *Leslie Lamport* for more depth.

EXTENDS *Integers*, *TLC*

CONSTANTS M, N

First, let's define the divisibility relation. We say a divides b, written a $\mid b$, if there's some natural number c in $[1, q]$ $s.t.$ $b = ac$.

$p \mid q \triangleq \exists\, d \in 1 \,..\, q : q = p * d$

$Divisors(q)$ will denote the set of all divisors of q, including 1 and q

$Divisors(q) \triangleq \{d \in 1 \,..\, q : d \mid q\}$

$Maximum(S)$ equals the maximal element x in S such that $x \geq$ for all y

$Maximum(S) \triangleq \text{CHOOSE } x \in S : \forall\, y \in S : x \geq y$

$GCD(p, q)$ is defined then as the maximum of the intersection of the divisors of p and q; that is, the largest divisor common to

$GCD(p, q) \triangleq Maximum(Divisors(p) \cap Divisors(q))$

Now *Euclid*'s algorithm can be expressed. It takes natural numbers x and y as input, and stores the result in x and y as well.
We use the *GCD* expression above the verify the result with an assertion.

```
--algorithm Euclid{
    variables x ∈ 1 .. M, y ∈ 1 .. N, x0 = x, y0 = y ;
    {
        while ( x ≠ y ) {
            if ( x < y ) {
                y := y − x ;
            } else {
                x := x − y ;
            }
        } ;
        assert x = GCD(x0, y0) ∧ y = GCD(x0, y0)
    }
}
```

The TLA+ toolbox will translate the above *PlusCal* program into a TLA+ one below:

BEGIN TRANSLATION ($chksum(pcal) =$ "6fcb1b8f" $\land$ $chksum(tla) =$ "d8b1bdba")

VARIABLES $x, y, x0, y0, pc$

$vars \triangleq \langle x, y, x0, y0, pc \rangle$

$Init \triangleq$ Global variables
$\land\, x \in 1 \,..\, M$
$\land\, y \in 1 \,..\, N$
$\land\, x0 = x$
$\land\, y0 = y$
$\land\, pc =$ "Lbl_1"

$Lbl\_1 \triangleq \land\, pc =$ "Lbl_1"

$$
\begin{array}{l}
\land \text{IF } x \neq y \\
\quad \text{THEN } \land \text{IF } x < y \\
\qquad\qquad \text{THEN } \land y' = y - x \\
\qquad\qquad\qquad\;\; \land x' = x \\
\qquad\qquad \text{ELSE } \land x' = x - y \\
\qquad\qquad\qquad\;\; \land y' = y \\
\qquad\quad \land pc' = \text{"Lbl\_1"} \\
\quad \text{ELSE } \land Assert(x = GCD(x0,\, y0) \land y = GCD(x0,\, y0), \\
\qquad\qquad\qquad \text{"Failure of assertion at line 29, column 9."}) \\
\qquad\quad \land pc' = \text{"Done"} \\
\qquad\quad \land \text{UNCHANGED } \langle x,\, y \rangle \\
\land \text{UNCHANGED } \langle x0,\, y0 \rangle
\end{array}
$$

Allow infinite stuttering to prevent deadlock on termination.

$Terminating \triangleq pc = \text{"Done"} \land \text{UNCHANGED } vars$

$$Next \triangleq Lbl\_1 \\ \qquad\quad \lor Terminating$$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{"Done"})$

END TRANSLATION

* Modification History
* Last modified *Thu Jan* 04 09:03:01 *EST* 2024 by *sca*
* Created *Thu Jan* 04 06:30:29 *EST* 2024 by *sca*