1 gcd

$Init : (x = M) \land (y = N)$

$NextState : ((x > y) \land (x' = y) \land (y' = x \text{`}mod\text{`} y)) \lor ((x < y) \land (y' = x) \land (x' = y \text{`}mod\text{`} x))$

$$
\begin{array}{lllllll}
(& & & & & & (1) \\
 & (& (x > y) & \land & (x' = y) & \land & (y' = x \text{`}mod\text{`} y)) \quad (2) \\
\lor & (& (x < y) & \land & (y' = x) & \land & (x' = y \text{`}mod\text{`} x))) \quad (3)
\end{array}
$$

2 TLA$^+$ Language

2.1 GCD example specification

CONSTANTS	M,N
VARIABLES	x,y

$$
\begin{array}{rrrr}
Init \triangleq & (x = M) & \land & (y = N) \\
Next \triangleq & (& (x > y & \\
 & \land & (x' = y) & \\
 & \land & (y' = x - y) & \\
\lor & (& (x < y) & \\
 & \land & (y' = x) & \\
 & \land & (x' = y - x))) &
\end{array}
$$

2.2 quicksort example

What it is: a divide-and-conquer algorithm for sorting an array in place.

$$A[0], ..., A[N-1]$$

Uses a procedure $Partition(lo, hi)$.
$Partition$ chooses $pivot$ in $lo...(hi-1)$, permutes $A[lo], ..., A[hi]$
to make $A[lo], ..., A[pivot] \leq A[pivot+1], ..., A[hi]$
and returns pivot.

2.2.1 thinking around procedures

A procedure differs from a (pure) function in that it has side-effects. Side effect here are crucial. We can think of a procedure as a function which take a hidden input, *State*, and returns a hidden output, *State′*. But then the order of execution becomes important. If our procedure is tree-recursive, we have to traverse the whole tree in the correct order to juggle the changing *State*. But there is a formalism for the order of execution, and this is a monad!
N.B.: see if the monad is a minimal formalism, or is there something more abstract that would suffice.
Now, is order of execution really important here? for example, does it matter whether we sort the lower or the higher half first? No, it does not.

¿ instance Monad State where ¿ return :: a -¿ State a ¿ return a = State a ¿ ¿ (¿¿=) :: State a -¿ (a -¿ State b) -¿ State b ¿ (¿¿=) a f = return (f a) ¿

¿ qsort :: Ord a =¿ State [a] -¿ [a] ¿ qsort as = do ¿ n ¡- length as ¿ pivot ¡- n/2 – floor ¿ ls ¡- take n as ¿ hs ¡- drop n as ¿ – order here is forced by monadic idiom, it is not a necessary part of an algorithm. ¿ result ¡- qsort ls ++ qsort hs ¿ return result

FOO

2.2.2 Parallel QuickSort specification

$$
\begin{array}{lll}
\mathit{Init} \triangleq & A & = \text{ARRAY OF LENGTH } N \\
& \wedge U & = \{(0, N-1)\} \qquad \text{set of all contigious intervals to be sorted} \\
& \wedge \mathit{pivot} & = N/2
\end{array}
$$

Partitions(B, pivot, lo, hi) $\triangleq$ the set of arrays obtained from $B$by permuting $B[lo], ..., B[hi]$ such that...

$$
\begin{array}{lll}
\mathit{Next} \triangleq & (U \neq \{\} & \\
\quad \wedge & \text{Pick any } b, t \text{ in} U : & \\
& \text{IF} & b \neq t \\
& \text{THEN} & \text{pick any } p \text{ in } b..(t-1) \\
& U' & \\
& = U \text{with} < b, t > \text{removed and} < b, p-1 > \text{and} < p, t > \text{added} & \\
& \wedge & (x' = y) \\
& \wedge & (y' = x - y) \\
\quad \vee & (& (x < y) \\
& \wedge & (y' = x) \\
& \wedge & (x' = y - x)))
\end{array}
$$

3 copy-pasted example of various alignment commands

$$
\frac{\begin{array}{c}(x_1 x_2) \\ \times (x_1' x_2')\end{array}}{(y_1 y_2 y_3 y_4)} \tag{4}
$$

$$
f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ -(n+1)/2 & \text{if } n \text{ is odd} \end{cases}
$$

$$
A = \left(\int_t XXX \right. \\ \left. YYY \ldots \right) \tag{5}
$$