This note is an introduction to invariants. During the execution of an algorithm, it changes state over time. This makes proofs about algorithms more challenging. Invariants can help. An *invariant* of a process that evolves over time is a property that holds throughout the lifetime of the process.

Diagonal robot. Here's an example from MCS Section 6.2.1. A robot sits on an infinite 2-dimensional integer grid. Its location is initially $(0, 0)$. At each time step, the robot moves one step diagonally: from its current location (i, j) to any of the four locations $\{(i+a, j+b) : a, b \in \{-1, 1\}\}$.

Theorem 1. *The robot can't reach* $(0, 1)$.

Proof (long form).

1. Consider an arbitrary sequence X of moves by the robot, starting at (0,0). *block: ∀*
2. Say a location (i, j) is *even* if $i + j$ is even, and *odd* otherwise.
3. Consider the following invariant: *The robot's location is even.*
4. The invariant is initially true, when the robot is located at $(0, 0)$.

5.1. Consider any move M in the sequence X. *proof pattern: ∀*

5.2.1. Assume the invariant holds just before move M. *block: if then*

5.2.2. Let (i, j) and $(i + a, j + b)$ denote the robot's locations before and after the move.

5.2.3. So $i + j$ is even, and $a, b \in \{-1, 1\}$.

5.2.4. Then $i + a + j + b = (i + j) + (a + b)$ where $a + b \in \{-2, 0, 2\}$.

5.2.5. Since $i + j$ is even, and $a + b$ is even, the robot's location after the move is even.

5.3. By Block 5.2, if the invariant holds before move M, it holds after.

6. By Block 5, for each move in X, if the invariant holds before M, then it holds after.
7. Since the invariant holds initially, it follows (by induction) that it holds throughout X.
8. It follows that the robot can't reach the location $(0, 1)$, as that location is odd. □

Hyper-robot. (Similar to MCS die-hard example.) A hyper-robot is located initially at $(24, 9)$. At each time step, it moves from its current location (i, j) to location $(j, i - j)$ or $(j, i + j)$.

Theorem 2. *The hyper-robot can't reach a location* (i, j) *where* $i = 1$.

Proof (long form).

1. Consider any such sequence X of moves by the robot. *block: ∀*
2. Define $S = \{(i, j) : (i, j)$ is a location where i and j are multiples of three$\}$.
3. Consider the following invariant: *The robot's location is in* S.
4. The invariant is initially true, when the robot is located at $(24, 9)$.

5.1. Consider any move M in the sequence. *block: ∀*

5.2.1. Assume the invariant holds just before move M. *block: if then*

5.2.2. Let (i, j) be the robot's location just before the move.

5.2.3. So (i, j) is in S, that is, i and j are multiples of three.

5.2.4. After the move, the location is either $(j, i - j)$ or $(j, i + j)$.

5.2.5. Since $i - j$ and $i + j$ must are also multiples of three, the new location is in S.

5.2.6. So the invariant holds after move M.

5.3. By Block 5.2, if the invariant holds before move M, it holds after.

6. By Block 5, for each move in X, if the invariant holds before, it holds after.
7. Since the invariant holds initially, it is maintained, so the robot can't reach any location $(1, j)$. □

Euclid's GCD algorithm. Here's an example of using an invariant to prove an algorithm correct. Recall that the *greatest common divisor of two integers* a and b is the largest integer d such that a and b are both (integer) multiples of d. (Don't forget the tricky boundary cases for zero. Zero is a multiple of every integer, so the greatest common divisor of zero and any integer a is a itself. And zero is the only multiple of zero, so the greatest common divisor of zero and zero is zero.)

Here is an iterative version of Euclid's algorithm for computing the greatest common divisor:

input: integers (a, b) such that $a \geq b \geq 0$ and $a > 0$

1. while $b > 0$:
2.1. $(a, b) = (b, a \bmod b)$
3. return a

In each iteration, b decreases by at least 1, but remains non-negative. So the algorithm terminates.

For intuition, try simulating the algorithm on $(a, b) = (24, 9)$, the start location of the hyper-robot in the previous example. As the algorithm iterates, think of (a, b) as the location of the hyper-robot: moving with each step from current location (a, b) to $(b, a \bmod b)$ (simulating several sequential moves of the hyper-robot). Following the reasoning for the robot, it maintains the invariant that a and b are multiples of 3. More generally, it maintains the invariant that the current a and b are multiples of the greatest common divisor d of the original input pair (a, b). You can see this by using the same reasoning as for the hyper-robot.

The trickier part is showing that, eventually, the robot reaches the location $(d, 0)$ where d is the g.c.d. of a and b. For example, if it starts at $(24, 9)$ it stops at location $(3, 0)$ and returns 3.

Before we give the proof, here are a definition and two lemmas. Let $\mathbb{Z}$ denote the set of all integers. Given integers a and b, let $I(a, b)$ denote the set of *integer combinations* of a and b: $I(a, b) = \{i \cdot a + j \cdot b : i, j \in \mathbb{Z}\}$. ($I(a, b)$ is also known in mathematics as an *ideal*.)

You can skip the proofs of the lemmas if you want, and focus on the proof of the theorem.

Lemma 1. *For any non-negative integers a, b, and d, if $I(a, b) = I(d, 0)$, then d is the greatest common divisor of a and b.*

Proof of Lemma 1.

1. Consider arbitrary non-negative integers a, b, d such that $I(a, b) = I(d, 0)$.
2. By inspection, $I(d, 0)$ is $\{i \cdot d : i \in \mathbb{Z}\}$, the set of integer multiples of d.
3. Since $a, b \in I(a, b) \subseteq I(d, 0)$, it follows that a and b are integer multiples of d.
4. That is, *d is a common divisor of a and b.*
5. To finish we verify that no common divisor of a and b is larger than d.
6.1. Consider an arbitrary common divisor d' of a and b.
6.2. Since d' is a divisor of a and b, it also a divisor of each integer combination of a and b.
6.3. Since $d \in I(d, 0) \subseteq I(a, b)$, it follows that d is an integer combination of a and b.
6.4. By the previous two steps d' is a divisor of d. Since $d \geq 0$, it follows that $d' \leq d$.
7. By Block 6, no common divisor of a and b is larger than d.
8. By this and Step 4, d is the greatest common divisor of a and b. □

By Lemma 1, to find the g.c.d. of (a, b), it suffices to find d such that $I(a, b) = I(d, 0)$. We will show that this is what the algorithm does. But how? The key observation is that, with each move, it reduces the coordinates of (a, b), *but doesn't change the set $I(a, b)$:*

Lemma 2. *For any positive integers a and b, we have $I(a, b) = I(b, a \bmod b)$.*

Proof of Lemma 2.

1. Consider arbitrary positive integers a and b.
2. Note that $a \bmod b \in I(a, b)$, as $a \bmod b = a - qb$ for $q = \lfloor a/b \rfloor$.

3.1. Consider an arbitrary $c \in I(b, a \bmod b)$.
3.2. Fix $i, j \in \mathbb{Z}$ such that $c = i \cdot b + j \cdot (a \bmod b)$.
3.3. Using q from Step 2, $c = i \cdot b + j \cdot (a - qb) = j \cdot a + (i - q) \cdot b \in I(a, b)$.

4. By Block 3, $I(b, a \bmod b) \subseteq I(a, b)$. Next we show the other direction.

5.1. Consider an arbitrary $c \in I(a, b)$.
5.2. Fix $i, j \in \mathbb{Z}$ such that $c = i \cdot a + j \cdot b$.
5.3. Then $c = i \cdot a + j \cdot b = (j + iq) \cdot b + i \cdot (a - qb) = (j + iq) \cdot b + i \cdot (a \bmod b) \in I(b, a \bmod b)$.

6. By Block 5, $I(a, b) \subseteq I(b, a \bmod b)$.
7. By Steps 4 and 6, $I(a, b) = I(b, a \bmod b)$.

□

Theorem 3. *The algorithm is correct. Given any integers a and b with $a \geq b \geq 0$ and $a > 0$, the algorithm above returns the greatest common divisor of a and b.*

Proof of Theorem 3.

1. Consider the execution of the algorithm on any input (a, b) with $a \geq b \geq 0$ and $a > 0$.
2. The next notation lets us refer unambiguously to the variables a and b as they change over time.
3. Let $(a_0, b_0) = (a, b)$ denote the input values, at the start.
4. Let T denote the number of iterations. Inspecting the algorithm, $b_T = 0$ and $b_t > 0$ for $0 \leq t < T$.
5. For $t \in \{1, 2, \ldots, T\}$ let (a_t, b_t) denote the value of the variables (a, b) after iteration t.
6. Consider the following invariant, maintained for each $t \in \{0, 1, \ldots, T\}$:

$$I(a_t, b_t) = I(a_0, b_0) \quad \text{and} \quad a_t \geq b_t \geq 0, \text{ with } a_t > 0.$$

7. (In other words, the ideal $I(a, b)$ doesn't change as the algorithm updates a and b.)
8. By inspection of the invariant, it holds initially (for $t = 0$). Next we show that it is maintained.

9.1. Consider any iteration $t \in \{1, 2, \ldots, T\}$, and assume the invariant holds before iteration t.
9.2. That is, $I(a_{t-1}, b_{t-1}) = I(a_0, b_0)$, and $a_{t-1} \geq b_{t-1} \geq 0$, with $a_{t-1} > 0$.
9.3. By inspection of the algorithm, $(a_t, b_t) = (b_{t-1}, a_{t-1} \bmod b_{t-1})$ and $b_{t-1} > 0$.
9.4. This, Lemma 2, and Step 9.2 imply $I(a_t, b_t) = I(a_{t-1}, b_{t-1}) = I(a_0, b_0)$.
9.5. Step 9.3 also implies $a_t = b_{t-1} > b_t \geq 0$.
9.6. So the invariant holds after iteration t.

10. The invariant holds before the first iteration (for $t = 0$), and, by Block 9, if the invariant holds before any iteration, it holds after the iteration. So (inductively) the invariant holds throughout.
11. Let $d = a_T > 0$ be the value returned by the algorithm. By the termination condition $b_T = 0$.
12. By the invariant, $I(a_0, b_0) = I(a_T, b_T) = I(d, 0)$.
13. So, by Lemma 1, d is the greatest common divisor of a_0 and b_0.
14. So the algorithm is correct.

□