Baboon Crossing Synchronization Scheme

Alessandro Monticelli, Jóhannes Helgi Tómasson, Jonas Stahl April 3, 2026

1. Description of the Solution

To solve the Baboon Crossing problem, we must ensure that baboons crossing from opposing directions (North and South) do not meet on the rope, while also adhering to a weight limit of 60 lb. We define the capacity of the rope in terms of "tokens", where the rope has a total capacity of 2 tokens. Based on the average weights (Male: 55 lb, Female: 28 lb), a male baboon requires 2 tokens and a female baboon requires 1 token.

Our synchronization protocol employs the following primitives:

- **Directional Mutual Exclusion:** To prevent baboons from opposing directions from crossing simultaneously, we use a **Lightswitch** pattern. As described in *The Little Book of Semaphores* (p. 70), this pattern allows the first baboon of a group to lock the resource (the rope) and the last one to unlock it, effectively treating a stream of baboons as a single entity.
- **Starvation Prevention:** Simply using a mutex for direction can lead to starvation if a continuous stream of baboons enters from one side. To resolve this, we use a **Turnstile** pattern, as seen in *The Little Book of Semaphores* (p. 29, 75). A baboon must pass the turnstile before locking the directional switch. If a baboon from the opposite side arrives, they wait on the turnstile, forcing subsequent arrivals from the current direction to wait, thus guaranteeing the stream will eventually conclude.
- **Atomic Capacity Acquisition:** A deadlock can occur if two males attempt to cross simultaneously and each grabs only one token (capacity = 0). To prevent this, the acquisition of capacity tokens is protected by a mutex (`ropeGrabbing`), ensuring it is atomic.

2. Proposed Pseudocode

The following pseudocode implements the logic described above. It utilizes a `LightSwitch` for each direction, a `turnstile` for starvation prevention, and a `ropeGrabbing` mutex for atomic token acquisition.

Note on Atomic Exit: In the exit protocol for the Male baboon, we utilize an atomic signal of 2 tokens (‘capacity.signal(2)‘). This removes the need for the ‘ropeGrabbing‘ mutex during exit, preventing the circular dependency (deadlock) that would occur if a Female were holding ‘ropeGrabbing‘ at the entry while a Male waited for it at the exit.

Listing 1: Semaphores and synchronization variables

```
// Global Variables
rope = Semaphore(1)          // Direction Mutex
capacity = Semaphore(2)      // Capacity tokens (Weight Limit)
turnstile = Semaphore(1)     // Avoids starvation on direction
ropeGrabbing = Semaphore(1) // Atomic acquisition of capacity tokens

northSwitch = LightSwitch() // Tracks crossing from North
southSwitch = LightSwitch() // Tracks crossing from South
```

Listing 2: Male baboon crossing

```
func Male(direction: N | S) {
    mySwitch = (direction == N) ? northSwitch : southSwitch

    entry:
        turnstile.wait()     // Wait for permission to enter
        mySwitch.lock(rope)  // Lock the rope for my direction (if first)
        turnstile.signal()   // Release for the next baboon

        ropeGrabbing.wait()  // Wait if someone else is taking capacity
            tokens
        capacity.wait(2)     // Atomic: Grab 2 tokens
        ropeGrabbing.signal()

    critical:
        // Crossing the canyon..

    exit:
        // Atomic release of 2 tokens.
        // No mutex needed here, preventing deadlock with Female entry.
        capacity.signal(2)

        mySwitch.unlock(rope)// Unlock the rope for the other side (if Last
            )
```

```
}
```

Listing 3: Female baboon crossing

```
func Female(direction: N | S) {
    mySwitch = (direction == N) ? northSwitch : southSwitch

    entry:
        turnstile.wait()
        mySwitch.lock(rope)
        turnstile.signal()

        ropeGrabbing.wait()  // Wants to take a token
        capacity.wait(1)     // Takes 1 token
        ropeGrabbing.signal()// Finished taking token

    critical:
        // Crossing the canyon...

    exit:
        capacity.signal(1)   // Release capacity token
        mySwitch.unlock(rope)// Release rope
}
```

3. Argument of Correctness

3.1 Problem Formalization and Invariants

We define the following state variables representing the number of baboons currently on the rope

- N_m: Male baboons crossing from North
- N_f: Female baboons crossing from North
- S_m: Male baboons crossing from South
- S_f: Female baboons crossing from South

The synchronization scheme must satisfy two primary invariants based on the problem constraints:

Invariant 1: Directional Mutual Exclusion
Baboons can only cross in one direction at a time. If there are baboons crossing from the North, there must be zero from the South, and vice-versa:

$$((N_m + N_f > 0) \Rightarrow (S_m + S_f = 0)) \wedge ((S_m + S_f > 0) \Rightarrow (N_m + N_f = 0))$$

Invariant 2: Capacity and Weight Limit

The rope allows a maximum of 2 tokens. A male consumes 2 tokens and a female consumes 1. The total weight usage must satisfy:

$$2(N_m + S_m) + 1(N_f + S_f) \leq 2$$

The resulting combined invariant that the system must maintain is:

$$((N_m + N_f > 0) \rightarrow (S_m + S_f = 0)) \wedge (2(N_m + S_m) + (N_f + S_f) \leq 2)$$

3.2 Proof of Constraint Compliance

We demonstrate that the solution satisfies the capacity invariant in all possible valid states:

- **Case 1 (1 Male on rope):**

$$2(1) + 1(0) = 2 \leq 2 \quad (\text{True})$$

- **Case 2 (2 Males on rope):**

$$2(2) + 1(0) = 4 \leq 2 \quad (\text{False - Impossible State})$$

  The second male blocks at `capacity.wait(2)` because only 2 tokens are available in the semaphore.

- **Case 3 (2 Females on rope):**

$$2(0) + 1(2) = 2 \leq 2 \quad (\text{True})$$

- **Case 4 (1 Male and 1 Female on rope):**

$$2(1) + 1(1) = 3 \leq 2 \quad (\text{False - Impossible State})$$

  If a Male holds 2 tokens, 0 are left; the Female waits. If a Female holds 1 token, 1 is left; the Male waits for 2 tokens (which are not available).

3.3 Liveness and Starvation Prevention

Deadlock Freedom: The potential deadlock identified in previous iterations—where a Male waiting to exit holds tokens while a Female waiting to enter holds the entry lock—is eliminated. By using `capacity.signal(2)`, the Male performs an atomic exit that does not require the `ropeGrabbing` lock.

Starvation Freedom: Starvation is prevented by the `turnstile`. If South baboons are crossing and a North baboon arrives, the North baboon waits on the turnstile. This prevents subsequent South baboons from passing the turnstile, ensuring the South stream eventually drains and the North baboon can proceed.