
Reykjavik University

T-419-CADP - Assignment 1

# Baboons Crossing

Freyr Elí Sveinbjörnsson
freyr23@ru.is

Georg Ingi Einarsson
georg23@ru.is

Óliver Máni Samúelsson
olivers23@ru.is

Instructor: Marcel Kyas

January 30, 2026

# 1 Problem Setup

We have two baboon types:

$$\text{male} = 55 \text{ lb}, \qquad \text{female} = 28 \text{ lb}.$$

The rope capacity requires the total weight on the rope to satisfy

$$W \leq 60.$$

To avoid collisions, we must enforce that all baboons currently on the rope are moving in the same direction.

**Capacity implications.** A male must cross alone (since $55 + 28 > 60$). Up to two females may cross simultaneously ($28 + 28 = 56$). No other combination is allowed (e.g. 3 females $= 84 > 60$, 2 males $= 110 > 60$).

# 2 Shared State and Invariants

As in the lecture *one-lane bridge* (passing the baton), we use:

- `non`: number of baboons on the rope going from the North side,
- `nos`: number of baboons on the rope going from the South side,
- `W`: total weight currently on the rope,
- `e`: baton/mutex (binary semaphore),
- `s1, s2`: condition semaphores for the two sides,
- `d1, d2`: delay counters (number waiting on each side),
- `turn`: fairness variable used when the rope becomes empty.

All shared variables are protected by `e`. If a baboon cannot enter safely, it increments its side's delay counter (`d1` or `d2`), releases `e`, and blocks on `s1` or `s2`. The procedure `Signal()` wakes an appropriate waiting baboon (passes the baton) or releases `e` if no one can proceed.

**Safety invariants.**

$$\textbf{(C1)} \quad \texttt{non} = 0 \ \vee \ \texttt{nos} = 0 \qquad \text{(one direction at a time)}, \tag{1}$$

$$\textbf{(C2)} \quad W \leq 60 \qquad \text{(rope capacity)}. \tag{2}$$

## 3 Pseudocode

```
// weights
weight(sex):
if sex == male return 55 else return 28

// Signal: pass baton to a waiting baboon that can safely enter, else release e.
// minWeightWaitingOnSideX means: 28 if any female is waiting on that side, else 55.
Signal():
  if d1 > 0 and nos == 0 and W + minWeightWaitingOnSide1 <= 60 and
     (turn == 1 or d2 == 0) {
      d1--
      V(s1)
  } else if d2 > 0 and non == 0 and W + minWeightWaitingOnSide2 <= 60 and
      (turn == 2 or d1 == 0)
      d2--
      V(s2)
  } else {
      V(e)
  }

// Side 1 (direction counted by non)
BaboonFromSide1(sex):
  myW = weight(sex)

  P(e)

  // use while rather than if (like Marcel did in the "pass the baton" lecture) for
      more chance for the waiting baboon to go on the rope
  // if, if statement, between the moment you're signaled and the moment you actually
      run again, other baboons can enter the rope first and increase W, change turn
      or re-fill the rope with same direction

  while (nos > 0) OR (W + myW > 60) OR (non == 0 and nos == 0 and d2 != 0 and
  turn == 2) {
    d1++
    V(e)
    P(s1)
    P(e)
    // when awakened, we continue holding the baton (same pattern as lecture)
  }
  non++
  W += myW
  Signal()
  V(e)
  Cross() // outside critical section

  P(e)
  non--
  W -= myW
  if non == 0 { turn = 2 } // when rope empties for this direction, give other side
      next chance

  Signal()
  V(e)
```

```
// Side 2 (direction counted by nos) - symmetric
BaboonFromSide2(sex):
  myW = weight(sex)

  P(e)
  while (non > 0) OR (W + myW > 60) OR (non == 0 and nos == 0 and d1 != 0 and
  turn == 1) {
    d2++
    V(e)
    P(s2)
    P(e)
  }
  nos++
  W += myW
  Signal()
  V(e)
  Cross()

  P(e)
  nos--
  W -= myW
  if nos == 0 { turn = 1 }
  Signal()
  V(e)
```

# 4 Correctness Argument

### Collision avoidance (Invariant C1)

A baboon from Side 1 waits while `nos` $> 0$; a baboon from Side 2 waits while `non` $> 0$. Therefore, `non` and `nos` cannot be positive simultaneously. Hence **(C1)** holds.

### Weight limit (Invariant C2)

Before entering, each baboon checks $W + myW \leq 60$. Updates to `W` occur only while holding `e`, so the check and the update are atomic. Thus **(C2)** holds.

### As many as possible (within constraints)

The rope does not need to become empty for additional same-direction baboons to enter. Any waiting baboon that satisfies the direction and weight checks may be released by `Signal()`. This allows two females (56 lb) to cross together, while a male crosses alone.

### No starvation

The variable `turn` enforces fairness when the rope becomes empty. If both sides have waiting baboons, the side indicated by `turn` is favored both in the entry condition and in `Signal()`. When the last baboon of a direction leaves (`non==0` or `nos==0`), `turn` flips to give the other side the next opportunity. Hence, a continuous stream from one side cannot block the other indefinitely.

### Choosing a waiter to wake under the weight limit

Because weights differ, `Signal()` should wake a waiting baboon that can fit the remaining capacity. A simple policy is:

$$minWeightWaitingOnSideX = \begin{cases} 28 & \text{if any female is waiting on side } X, \\ 55 & \text{otherwise.} \end{cases}$$

This is sufficient to ensure progress whenever any waiting baboon can legally enter.