# Algorithm Y

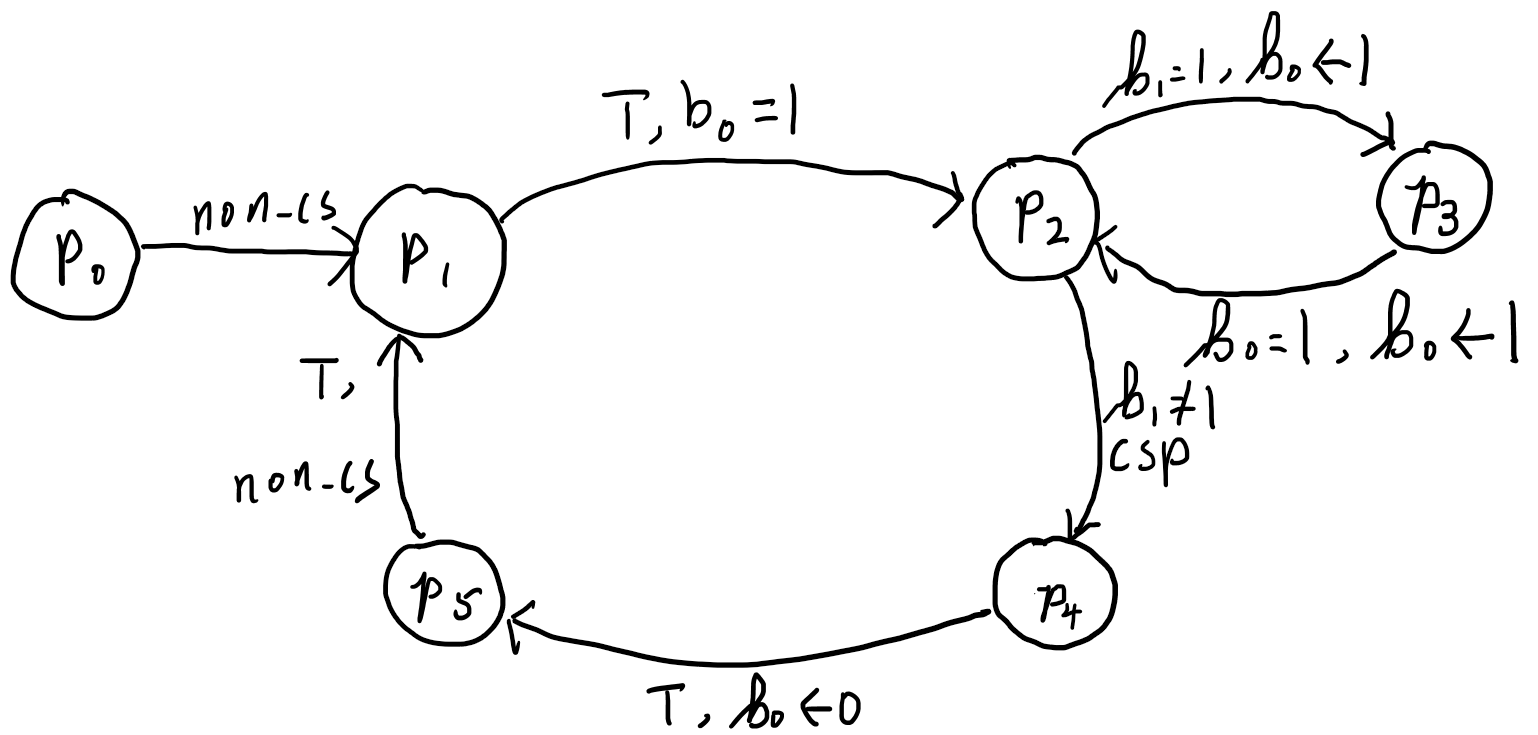
## Part 1

### Assess the desiderata

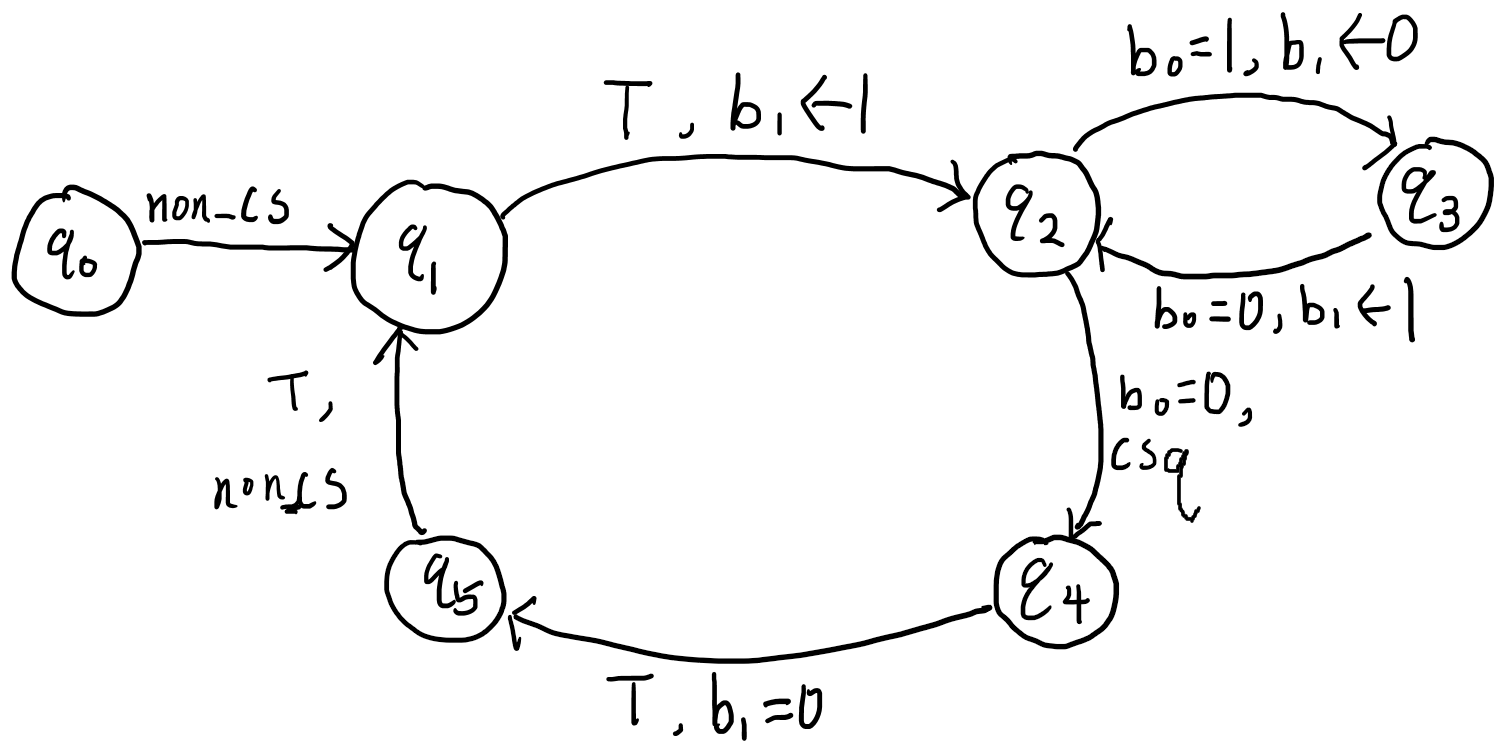
1. Mutual exclusion
   * Passed – using LTL claim **mutex**
2. Eventual entry
   * Passed for P – using **waitp**
   * Failed for Q – using **waitq**
3. Absence of deadlock
   * Passed – by selecting *invalid endstats (deadlock)* option under Safety tab
4. Absence of unnecessary delay
   * Passed – using **absp** and **absq**

## Part 2

### Transition Diagram



Transition diagram for P



Transition diagram for Q

### Assertion network

:

:

:

:

:

:

:

:

:

:

:

:

In this assertion network, and are the locations that represent the critical sections. Take the conjunction of and we have:

Which can never be true, which means that only one critical section can be alive at the same time.

#### Proof of inductive

:

: . After f, the value of becomes 1.

: , . After f, the value of is still 1

: . After f, is still 1.

: , f has no effect on .

: After f, the value of becomes 0.

: .

:

: . After f, the value of becomes 1.

: , . After f, the value of becomes 0.

: . After f, is becomes 1.

: , f has no effect on .

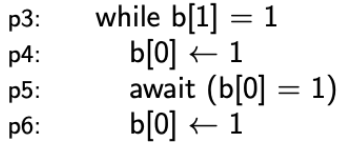
: After f, the value of becomes 0.

: .

Thus, we the assertion network is proved inductive.

## Part 3

Code line to can be directly replaced by , because this line is represented by in the transition diagram. The cycle in the diagram can be replaced by a self-pointing cycle without changing the behaviour of the process.



Superfluous code piece

?ASd


Fragment of transition diagram

In addition, according to the transition network, the location and both specifies . So, the changes will not affect the behaviour of the algorithm.

Thus, code to can be simplified as:

: await b[1] = 0