Algorithm Y

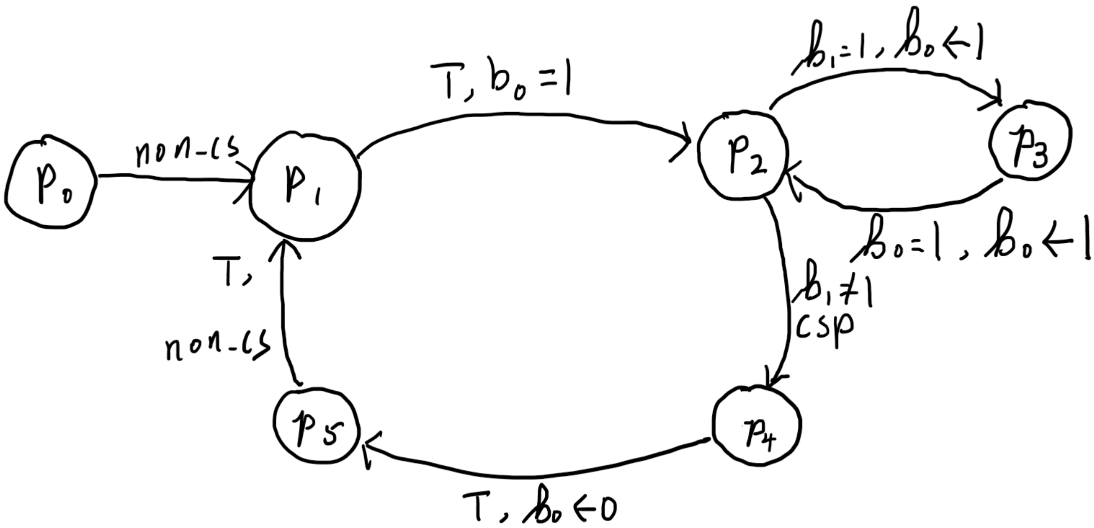
Part 1

Assess the desiderata

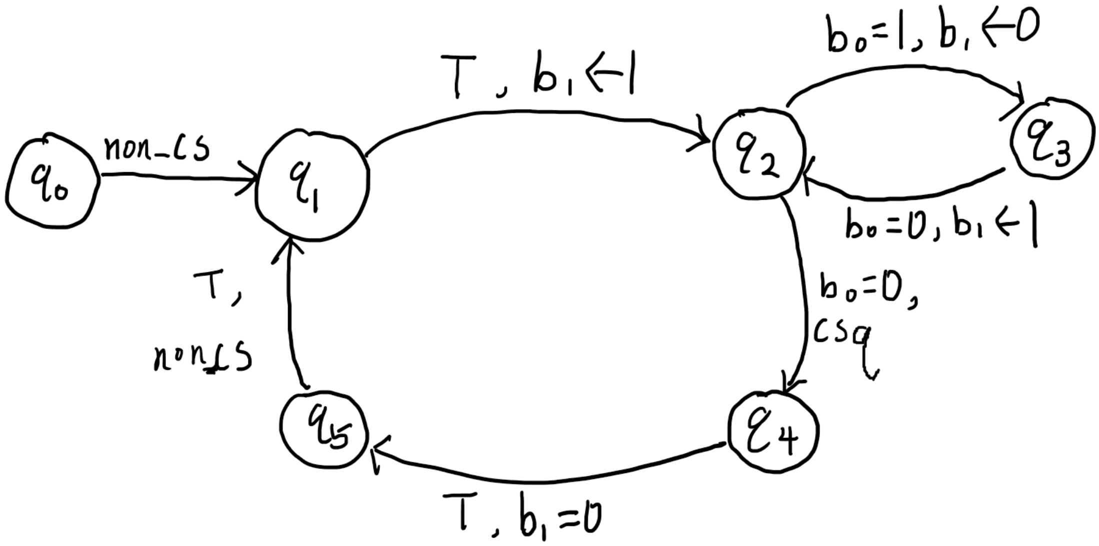
1. Mutual exclusion
   - Passed – using LTL claim **mutex**
2. Eventual entry
   - Passed for P – using **waitp**
   - Failed for Q – using **waitq**
3. Absence of deadlock
   - Passed – by selecting *invalid endstats (deadlock)* option under Safety tab
4. Absence of unnecessary delay
   - Passed – using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

Part 2

Transition Diagram



Transition diagram for P



Transition diagram for Q

Assertion network

p_0: $b_0 = 0 \wedge b_1 = 0$
p_1: $b_0 = 0$
p_2: $b_0 = 1$
p_3: $b_0 = 1$
p_4: $b_0 = 1 \wedge b_1 \neq 1 \wedge P@p_4$
p_5: $b_0 = 0$

q_0: $b_0 = 0 \wedge b_1 = 0$
q_1: $b_1 = 0$
q_2: $b_1 = 1$
q_3: $b_0 = 1 \wedge b_1 = 0$
q_4: $b_0 = 0 \wedge Q@q_4$
q_5: $b_1 = 0$

In this assertion network, p_4 and q_4 are the locations that represent the critical sections. Take the conjunction of p_4 and q_4 we have:

$$b_0 = 1 \wedge b_1 \neq 1 \wedge P@p_4 \wedge b_0 = 0 \wedge Q@q_4$$
$$\Leftrightarrow (b_0 = 0 \wedge b_0 = 1) \wedge b_1 \neq 1 \wedge P@p_4 \wedge Q@q_4$$
$$\Leftrightarrow \bot \wedge b_1 \neq 1 \wedge P@p_4 \wedge Q@q_4$$
$$\Leftrightarrow \bot$$

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

Proof of inductive

$p_0 \to p_1$: $p_0 \wedge g: b_0 = 0 \wedge b_1 = 0 \Longrightarrow b_1 = 0, f: non\_cs$
$p_1 \to p_2$: $p_1: b_0 = 0, g = \top$ $f: set\ b_0 = 1$. After f, the value of b_0 becomes 1.
$p_2 \to p_3$: $p_2 \wedge g: b_0 = 1 \wedge b_1 = 1$, $f: set\ b_0 = 1$. After f, the value of b_0 is still 1
$p_3 \to p_2$: $p_3 \wedge g: b_0 = 1 \wedge b_0 = 1 \Longrightarrow b_0 = 1, f: set\ b_0 = 1$. After f, b_0 is still 1.
$p_2 \to p_4$: $p_2 \wedge g: b_0 = 1 \wedge b_1 \neq 1 \Longrightarrow b_0 = 1 \wedge b_1 \neq 1, f: csp$, f has no effect on b.
$p_4 \to p_5$: $p_4: b_0 = 1, g: \top, f: set\ b_0 = 0.$ After f, the value of b_0 becomes 0.
$p_5 \to p_1$: $p_t \wedge g: b_0 = 0 \wedge \top \Longrightarrow b_0 = 0. f: non\_cs$

$q_0 \to q_1$: $q_0 \wedge g: b_0 = 0 \wedge b_1 = 0 \Longrightarrow b_0 = 0, f: non\_cs$
$q_1 \to q_2$: $q_1: b_1 = 0, g = \top$ $f: set\ b_1 = 1$. After f, the value of b_1 becomes 1.
$q_2 \to q_3$: $q_2 \wedge g: b_1 = 1 \wedge b_0 = 1$, $f: set\ b_1 = 0$. After f, the value of b_1 becomes 0.
$q_3 \to q_2$: $q_3 \wedge g: b_1 = 0, f: set\ b_1 = 1$. After f, b_0 is becomes 1.
$q_2 \to q_4$: $q_2 \wedge g: b_1 = 1 \wedge b_0 = 0 \Longrightarrow b_0 = 0, f: csq$, f has no effect on b.
$q_4 \to q_5$: $q_4: b_0 = 0, g: \top, f: set\ b_1 = 0.$ After f, the value of b_1 becomes 0.
$q_5 \to q_1$: $q_t \wedge g: b_0 = 0 \wedge \top \Longrightarrow b_0 = 0. f: non\_cs$

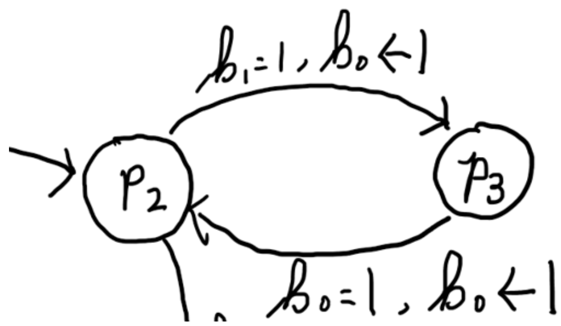
Thus, we the assertion network is proved inductive.

Part 3

Code line $p4$ to $p6$ can be directly replaced by $skip$, because this line is represented by p_3 in the transition diagram. The cycle $p_2 \to p_3 \to p_2$ in the diagram can be replaced by a self-pointing cycle without changing the behaviour of the process.

```
p3:    while b[1] = 1
p4:        b[0] ← 1
p5:        await (b[0] = 1)
p6:        b[0] ← 1
```

Superfluous code piece



Fragment of transition diagram

In addition, according to the transition network, the location p_2 and p_3 both specifies $b_0 = 1$. So, the changes will not affect the behaviour of the algorithm.
Thus, code $p3$ to $p6$ can be simplified as:

$p3$: await b[1] = 0