Transition Diagrams: Two Process Mutual Exclusion

```
  proc(id, other, turn)
    while(TRUE) {
n:       ⟨ b:= TRUE ; turn = (id + 1) % 2; ⟩
w:       wait until( !other.b | turn = id );
c:       b := FALSE;
    }
```

Consider the definition of a process `proc` defined here. A `proc` is parameterized by an `id`, the other process `other`, and the id of the process whose turn it is initially, `turn`, which is a shared variable. The relevant lines of code are labelled `n` (non-critical), `w`(waiting), and `c` (critical). Initially, `b` is `FALSE`.

Consider two processes, P_0 and P_1 as defined below. The code for each process is shown, annotated with addtional subscripts for convenience. The processes run concurrently with a single-process scheduler, with a non-deterministic choice of which process to run next. Each line runs atomically.

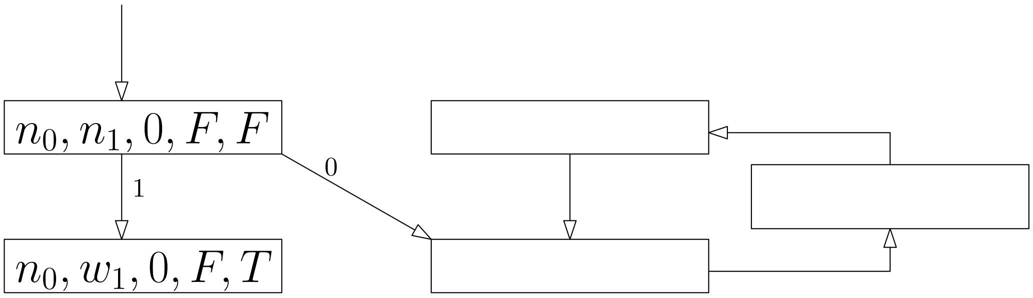
Process P_0 = `proc(0,` P_1`, 0)`

```
     while(TRUE) {
n_0:      ⟨ b_0:= TRUE ; turn = 1; ⟩
w_0:      wait until( !b_1 | turn = 0 )
c_0:      b_0 := FALSE;
     }
```

Process P_1 = proc(1, P_0, 0)

```
     while(TRUE) {
n_1:      ⟨ b_1:= TRUE ; turn = 0; ⟩
w_1:      wait until( !b_0 | turn = 1 );
c_1:      b_1 := FALSE;
     }
```

A state of the whole system, which describes the composed states of both processes is given by the tuple $(pc_0, pc_1, turn, b_0, b_1)$, where $pc_i \in \{n_i, w_i, c_i\}$ is the program counter of P_i, $turn \in \{0, 1\}$ is the value of the shared variable `turn`, and $b_i \in \{$`TRUE, FALSE`$\}$ is the value of local variable `b` of P_i.


$n_0, n_1, 0, F, F$
0
1
$n_0, w_1, 0, F, T$