```
bool in1 = false, in2 = false;
int last = 1;
process CS1 {
  while (true) {
    last = 1; in1 = true;     /* entry protocol */
    ⟨await (!in2 or last == 2);⟩
    critical section;
    in1 = false;              /* exit protocol */
    noncritical section;
  }
}
process CS2 {
  while (true) {
    last = 2; in2 = true;     /* entry protocol */
    ⟨await (!in1 or last == 1);⟩
    critical section;
    in2 = false;              /* exit protocol */
    noncritical section;
  }
}
```

Figure 3.5 Two-process tie-breaker algorithm: Coarse-grained solution.

NO MUTUAL EXCL.! ⇒

ANDREWS STYLE OF "OPERATIONAL PROOFS"
NOT WATERTIGHT.

```
bool in1 = false, in2 = false;
int last = 1;

process CS1 {
  while (true) {
    last = 1; in1 = true;      /* entry protocol */
    ⟨await (!in2 or last == 2);⟩
    critical section;
    in1 = false;               /* exit protocol */
    noncritical section;
  }
}

process CS2 {
  while (true) {
    last = 2; in2 = true;      /* entry protocol */
    ⟨await (!in1 or last == 1);⟩
    critical section;
    in2 = false;               /* exit protocol */
    noncritical section;
  }
}
```

in1 = true ; last = 1;

in2 = true ; last = 2;

Figure 3.5 Two-process tie-breaker algorithm: Coarse-grained solution.

Peterson's algm.

NO MUTUAL EXCL ! ⇒ after exec. in1 = true other process blocked after which one ends up in critical section

ANDREWS STYLE OF "OPERATIONAL PROOFS" NOT WATERTIGHT.

Correct version 1