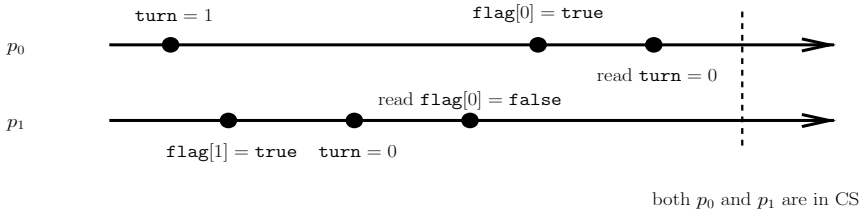
SLR206: Solutions for Quiz 1

1 2-process Peterson's algorithm

Suppose that p_0 executes the first two lines of its algorithm in the reverse order:

1. `turn` = 1;
2. `flag`[0] = `true`;

Then the following execution scenario is possible:



(Note that we do not care about the order in which the first two lines are executed by p_1.)

Here p_0 sets `turn` to 1, then p_1 sets `turn` to 0, `flag`[0] to `true` (the order in which these two operations are performed does not matter) reads `false` in `flag`[0] and proceeds to the critical section. Then p_0 reads 0 in `turn` and also proceeds to the critical section—a contradiction.

2 Tournament

The safety (mutual exclusion) part is implied by the fact that a process can only enter the critical section if it gets in the critical section of the 2-process algorithm at the root node of the tournament tree. As the 2-process algorithm ensure the mutual exclusion property, at most one process can be in its critical section at a time.

To prove starvation-freedom, suppose, by contradiction that a process p_i is blocked forever in the trying section of some node C (2-process mutual exclusion algorithm) in the tournament tree. Without loss of generality, assume that no process that no process is blocked closer to the root than p_i. Thus, every process that may obstruct p_i will eventually reach the critical section of the root node and, eventually, release all the locks it has grabbed on the way, including the one of p_i.[1] Thus, p_i will eventually enters the critical section of C—a contradiction.

[1] Recall that we assume that every process is correct and no process stays in the crtitical sectiuon forever. Otherwise, starvation-freedom is trivially satisfied.

3 Safety

Safety of an implementation

The set of runs of an implementation I is trivially *prefix-closed*: every prefix of a run of I is also a run if I.

Suppose that all *finite* runs of I are *safe* (with respect to some safety property P). We want to show that even *infinite* runs of I are also in P.

Let σ be any infinite run of I. Let σ_1, ..., σ_k, ... be prefixes of σ, where σ_i, $i = 1, 2, \ldots$, has length i. By our assumption, every σ_i is in P. Since P is limit-closed, $\sigma = \lim i \to \infty \sigma_i$ is also in P.

Checking safety

We want to argue that to check that a safety property P is violated, we can look for a *finite* run.

Indeed, consider a run $\sigma \notin P$. If σ is finite we are done: for every extension σ' of σ, we have $\sigma' \notin P$ (otherwise, P is not prefix-closed).

Let σ be infinite. Suppose, by contradiction, that σ has no unsafe prefixes, Then, by limit-closedness of P, we get that σ (as the infinite limit of these safe prefixes) is safe—a contradiction.

Determining safety

Given a property P, we want to construct S, a safety property, and L, a liveness property, such that $P = S \cap L$.

S can be constructed as a *prefix-* and *limit-closure* of P, defined as P plus all prefixes and limits of runs in P:

$$\begin{aligned} S = \quad & \{\sigma : \exists \sigma' \in P,\ \sigma \textit{ is a prefix of } \sigma'\} \cup \\ & \{\sigma : \exists \sigma_1, \sigma_2, \ldots \in P,\ \forall i, \sigma_i \textit{ is a prefix of } \sigma_{i+1},\ \sigma = \lim_{i \to \infty} \sigma_i\} \end{aligned}$$

By construction, S is prefix- *and* limit-closed.

We define L as the *largest possible set* that gives P under intersection with S:

$$L = P \cup \neg S$$

> Recall that a liveness property must contain extensions of *all* possible runs: something good should always be able to happen eventually. In this sense, it is better to make L as large as possible.

By construction, $S \cap L = P$.

It remains to show that L is indeed a liveness property, i.e., for every finite σ, there exists $\sigma' \in L$, an extension of σ.

Consider any $\sigma \notin L$. By the definiton of L, $\sigma \in S - P$, and, by the definition of S, σ is either a finite prefix of a run in P or an infinite limit of a sequence of runs in P. Since, σ is finite, we derive that an extension of σ is in P.

4 Liveness

First of all, we observe that wait-freedom (WF) is a subset of every other property in the table, i.e., WF is the strongest liveness property in the set.

Consider obstruction-freedom (OF) and lock-freedom (LF) and take any run $\sigma \in LF$. LF is an independent property, so it guarantees progress to some process in all runs, while OF only guarantees progress if some process runs in isolation (for sufficiently long). Therefore, every run in LF is also in OF. Further, any run in which no process ever runs in isolation, e.g., in which processes run one-by-one in the round-robin order, but no process makes progress is, *trivially*, in OF, not in LF. Thus, $LF \subsetneq OF$.

> Here we use standard logical reasoning. Consider a set of runs defined as follows:
>
> $$P = \{\sigma : A\sigma \Rightarrow B\sigma\},$$
>
> i.e., P consists of all runs σ, such that if σ satisfies A, then it satisfies B. Then any run that *does not* satisfy A is *trivially* in P.
>
> For example, consider the property: "I like all fruits, but if it is an apple, then I only like red ones." Then if you give me an orange, I should like it.
>
> Similarly, when deadlock-freedom says: "if every process is correct, then some process makes progress", a run in which not every process is correct, is trivially deadlock-free.

The remaining relations can be established analogously.

5 Linearizability

Left as an exercise.