Verification of System Properties Using Place Invariants

Problem description:

Suppose that n processes in an operating system are each allowed to access a buffer in reading or writing mode. To guarantee reliability, reading and writing access is restricted in the following way: when no process is writing to the buffer then up to k<=n processes are allowed to read it. But writing access to the buffer is only permitted as long as no other process is reading or writing the buffer.

Desired system properties:

Property #1: The number of processes, n, remains constant and each process is in one of the states only.

Property #2: There is at most one process that is in writing mode.

Property #3: When no process is writing then up to k processes can be in the reading mode.

System of reader and writer processes is shown as P/T net. Each process is in one of five states, represented by separate places s_0, s_1, s_2, s_3, s_4. In the initial state, all n processes are passive. Hence, state s_0 contains n tokes, each representing one process. This represents an initial state of the system. The processes are not distinguished among themselves. The place s_5 contains k tokens in the initial marking, where $k<=n$. This corresponds to the number of processes that are allowed to read the buffer concurrently.

Two invariants can be defined for the system: the first contains places s_0, s_1, s_2, s_3, s_4. The second invariant contains places: s_2, s_4, s_5.

From the *first invariant*, for each marking M that can be reached from initial marking M_0 we can prove the following property:

$$M(s_0)+M(s_1)+M(s_2)+M(s_3)+M(s_4) = M_0(s_0)=n$$

This means that the number of processes, n, remains constant and each process is in precisely one of the states represented by places s_0, s_1, s_2, s_3, s_4.

From the *second invariant*, for each marking M that can be reached from initial marking M_0 we can prove the following property:

$$M(s_2)+k*M(s_4)+M(s_5) = M_0(s_2)+k*M_0(s_4)+M_0(s_5) =k$$

We find that s_4 contains at most one token under any marking M, i.e. there exists only one writing process. When s_4 carries a token then s_2 and s_5 are empty. So, while some process is writing, no other process reads the buffer.

Place s_2 carries at most k tokens, i.e. there are at most k processes reading concurrently. When no process is writing, i.e. $M(s_4)=0$, then s_2 may in fact obtain k tokens. Then, the synchronization place s_5 is empty.

Proving liveness of the P/T net that models the CREW operating system

Proposition. P/T net modeling the Readers and Writers problem is live, i.e. that each reachable marking enables at least one transition.

Proof. Places s_0, s_1, s_3 have capacity n, i.e. there is at most n tokens in these places at any point in time assuming initial marking as described above. Place s_4 has capacity 1 and places s_2 and s_5 have capacity k with the same assumption.

One can notice that capacity of places will never hinder any firing of transitions.

In case of $M(s_0)+M(s_2)+M(s_4)>0$, from the net structure we see that at least one of the transitions t_0, t_3, t_2, or t_5 is enabled.

If $M(s_0)+M(s_2)+M(s_4)=0$ we get from invariant i_1 that $M(s_1)+M(s_3)=n$ and from invariant i_2 that $M(s_5)=k$. Then t_1 or t_4 is enabled.

Now, if s_0 is empty for some marking M that can be reached from $[M_0>$, it may be marked by some succession of firings. This implies the liveness of t_0 and t_3. The liveness of other transitions follows immediately.