Limited/bounded or safe Sequential Function Charts

A SFC is limited or bounded, if the multiplicity of tokens in any step is limited to be less than or equal to k, a number less than infinity. The SFC is called safe, if k=1, i.e. the multiplicity of tokens in a step is never larger than 1. For plant control only safe SFC are used.

What are possible reasons?

For plant control it is useful to have a tool to check, whether the SFC is limited. Boundedness can be investigated with the cover property. This is explained in the next paragraph,

Covering:

When compiling a SFC-program in an advanced engineering tool, you might get an error message, that the SFC is not bounded. It is useful to know the reason for unboundedness and how to modify to get a bounded SFC. How can we check for boundedness? If there is an initial marking and we can find a sequence of firing transitions in such a way, that all the steps of the initial marking are marked again and there are some additional steps marked, then the SFC is obviously unbounded. Since all step of the initial Firing the transitions 'start & no error' and 'error' result in the marking M2. In M2 the step 'ready' is marked as in the initial marking M1 but additional the step 'Temp. control' is marked. When the same sequence of transitions is fired, the step 'Temp. control' gets a second token.

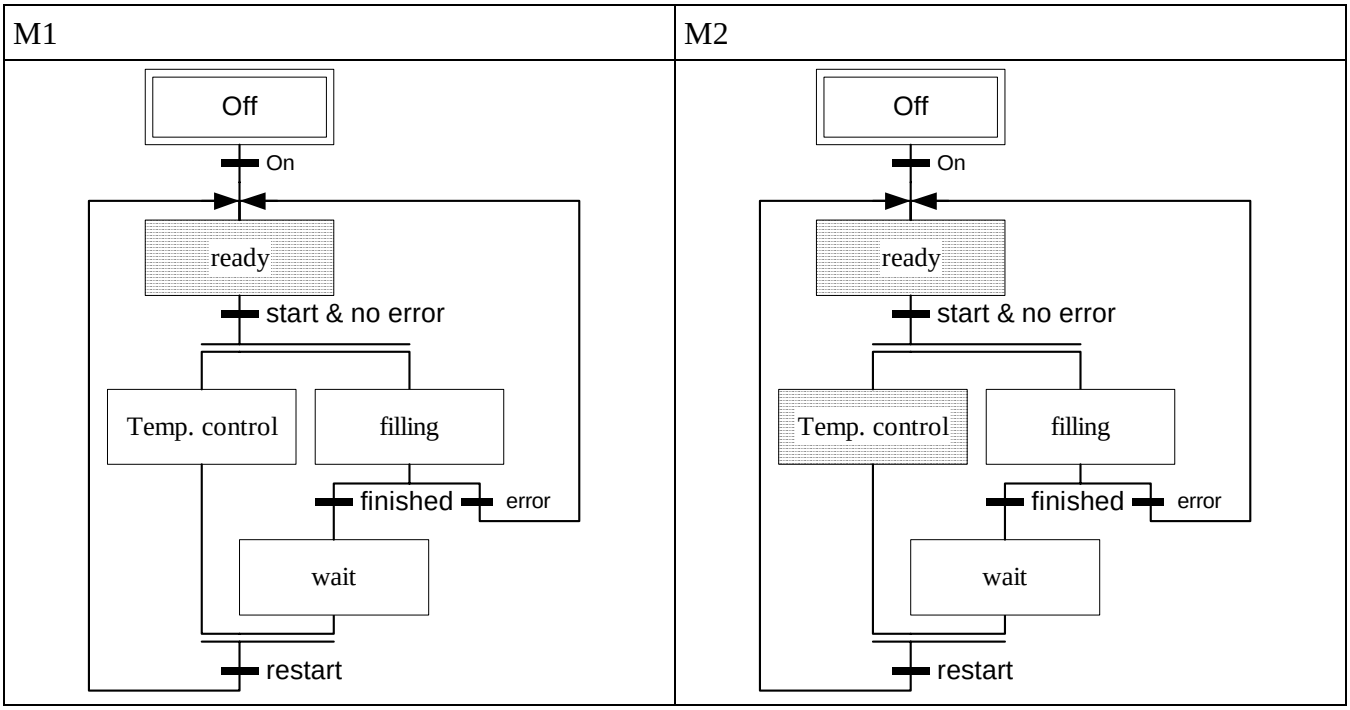

Figure 1: covering marking

Visually the marking M2 covers up the marking M1.

Generally: a marking M2 covers up a marking M1 if the markings M1 and M2 are different and all steps marked in M1 are also marked in M2.

In the example in Figure 1 M2 covers up M1 because

- the markings are different, i.e. 'Temp. Control' is only marked in M2
- every step in M1, in this case 'ready' is also marked in M2

It can be proven that a SFC is bounded, if and only if there are no covering markings, when the SFC is completely simulated. As soon as one marking covers up a previous marking simulation can be stopped, because the SFC is not bounded. Simulation analysis can reveal the transition leading to such a marking. Usually, unbounded SFC results from incorrect handling of concurrent processes. In the preceding example the 'start & no error' transition splits the process into two processes. In one branch the 'error' transition leads to a jump back to the step 'ready' which is ahead of the splitting transition. With this mechanism an indefinite number of processes are created resulting in an unbounded SFC.

In Figure 2 a solution is proposed, which shows a correct error handling within the filling process.

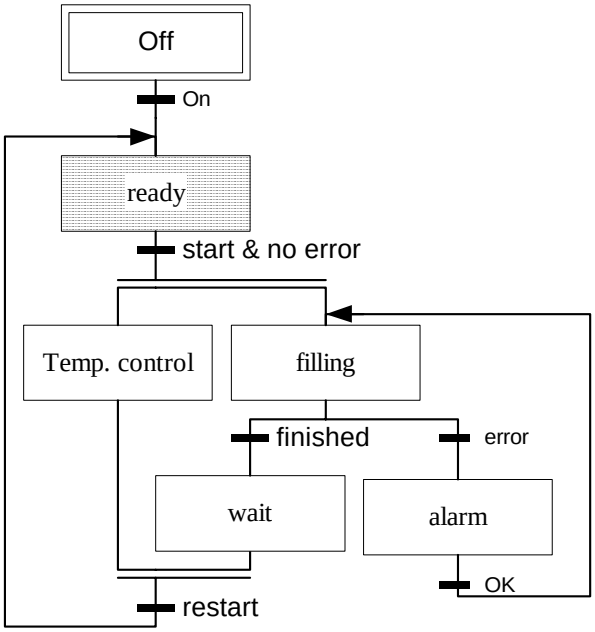

Figure 2: SFC without covering

Another reason for unbounded SFC is that concurrent processes are not terminated correctly. Figure 3 show an incorrect synchronisation of concurrent processes with an 'either Or' convergence. The correct solution is shown in Figure 4.

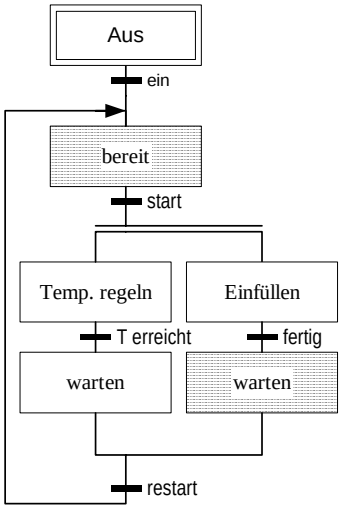

Figure 3: Incorrect synchronisation of concurrent processes

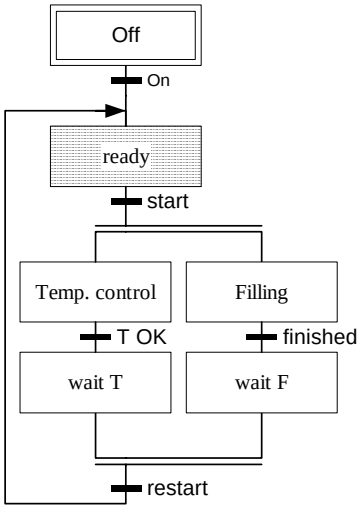

Figure 4: Correct synchronisation of concurrent processes