Sequential Function Chart Analysis

Introduction

Sequential Function Charts can inherit all theory developed for Petri-Net. For a detailed study there are many textbooks available. Not all of the theory can be directly applied to plant control. In the sequel you will find the theory which is necessary and helpful in the context of industrial plant control. Many of the properties of an SFC can be shown in an array representation of the net. The goals of the chapter are

- that you are able to map an SFC-graph to an array representation.
- that you can explain and proof liveness of a graph
- that you can explain the compiler message, that the SFC is not secure, and that you are able to correct the SFC

Example

Alarms with manual reset are used whenever the process is not allowed to continue automatically, i.e. without a manual reset of the error. This is always necessary, when the control action resulting from an error condition makes the error cause to disappear, but as soon as action is stopped, the error will reappears again. In Figure 1, a SFC for an alarm with manual reset is drawn. The alarm logic is explicitly modeled in the two steps 'no alarm' and 'alarm'.

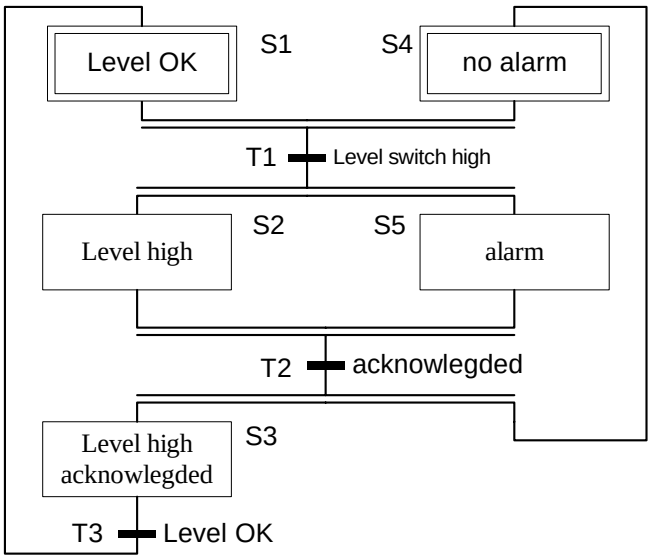
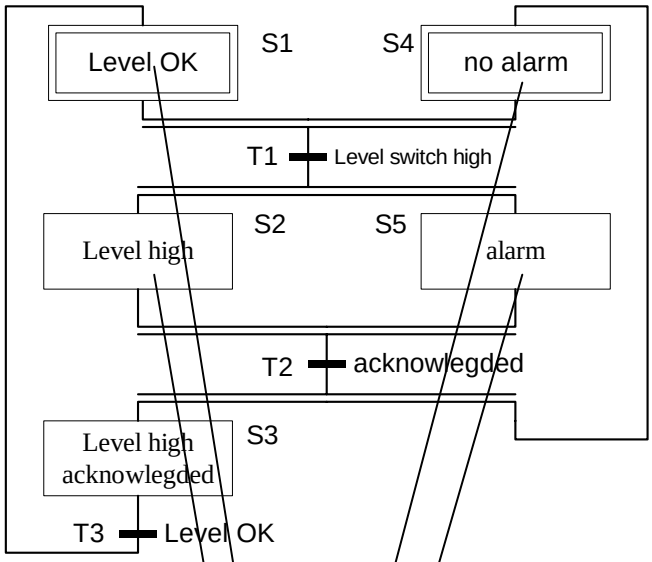


Figure 1: Alarm with Reset

Array representation of a SFC

The array representation of a sequential function chart describes the flow of tokens. The value of the array elements indicates the change of the number of tokens in a step, when a transition is fired. The array rows are assigned to steps, the columns to the fired transition. The row headers are therefore the step names, the column headers the transition names. Similarly, step and transitions can be enumerated so that the step number is the row index and the transition number the column index. The entries in the array are {-1, 0, 1}. A value of -1 in the i-th row and the j-th column indicates that the i-th step will loose one token if the transition j is fired. The array representation of the alarm example is shown in Figure 2.



Step\ Transition	T1	T2	T3
S1	-1	0	1
S2	1	-1	0
S3	0	1	-1
S4	-1	1	0
S5	1	-1	0

Figure 2: Array representation

Analysis

Components of a SFC

When simulating an SFC the number of tokens or the number of simultaneously marked steps may change. In the example show in Figure 3 there will never be more than one token, but in Figure 1 there

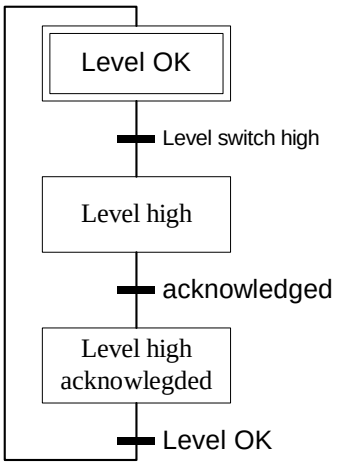


Figure 3: Component

are always two tokens. When there is only one step marked, the SFC is a state automata. The state of an automata describes, what the automata is actually doing. When several steps are simultaneously marked, the state can be described by the union of the step names and the automata is doing all actions in the marked steps. If there is never more than one step marked, the step names are equal to state names and it is reasonable to call this SFC a state automata.

The states and steps of the SFC in are therefore:

- Level OK
- Level high
- Level high acknowledged

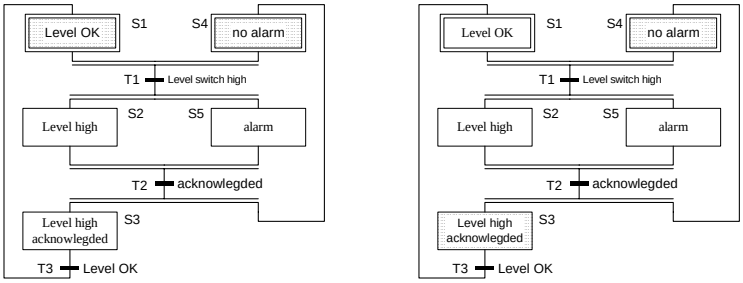


Figure 4: States of the alarm example

The SFC of the alarm example in

Figure 1 always has two tokens and is therefore not a state automata. In Figure 4 two of the three possible states are drawn. The state on the left can be described as 'Level OK' and 'no Alarm' and for the marking on the right 'Level high acknowledged' and 'no alarm'

Forward conflict

In a SFC it is not allowed, that a step can have more than one token. A step is either marked or not marked. This can lead to a conflict as shown in Figure 5. If T1 and T2 fire, the step C should be marked with two tokens. If a SFC is properly drawn this conflict situation does not occur. This topic is discussed in detail in a following lesson.

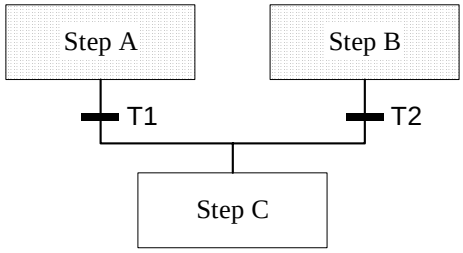


Figure 5: Forward conflict

Backward conflict

If a divergence is combined with two synchronisation conditions as shown in Figure 6 there results a problem.

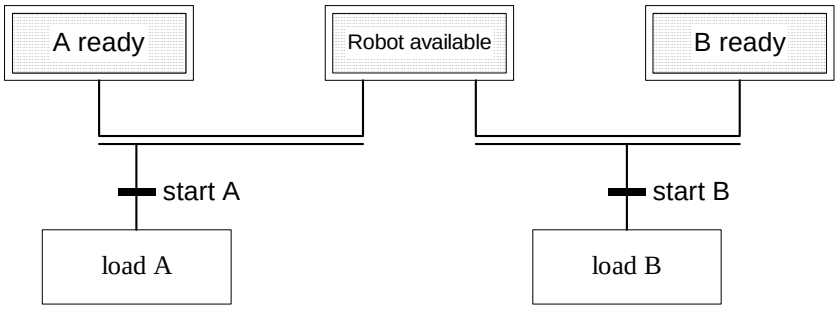


Figure 6: Backward conflict

If both transition conditions 'start A' and 'start B' become simultaneously true, which one should fire? In many cases this problem can not be solved in the SFC, because there are no general roules for priority. The most sensible way to solve the problem is to modify the transition conditions so that it is impossible that both can be true at the same time. If the transition condition on the right is changed to 'start B and (not start A)', the path on the left has priority.

If that solution is to static another type of arbitration can be solved within the SFC. In Figure *7* a solution leading to an alternating arbitration of the robot is drawn

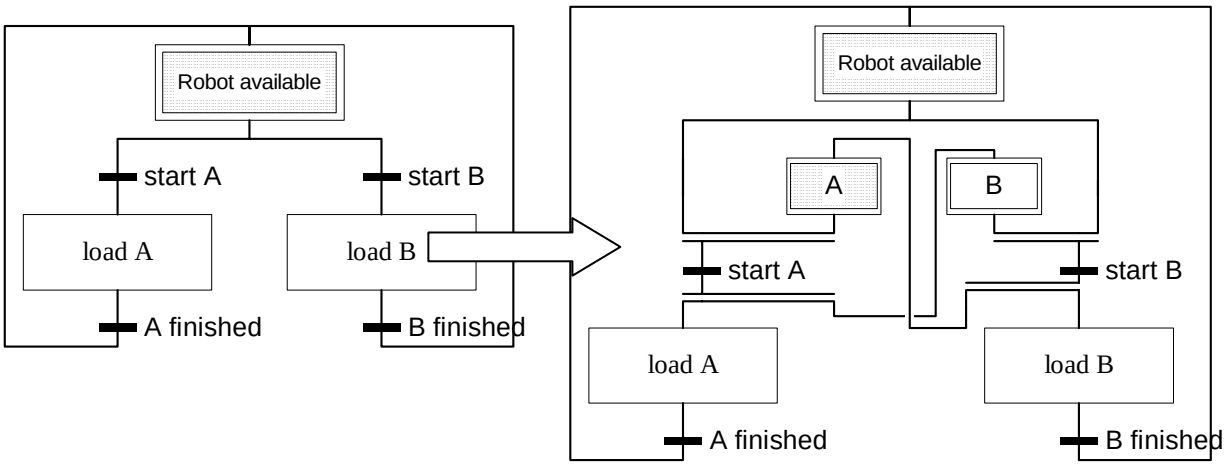


Figure *7: Alternating arbitration*

Analysis of conflict situation in SFC is very useful to find unsolved decision problems!