Modelling with Automata

- Real-life systems are modelled using several automata.
- The system behaviour is represented by several automata running together.

*How can automata interact?*

Transitions of the Product

Example: Three Counters

Restricting Combined Actions

1. Strict Synchronisation

   Either all automata take a step, or none. Only allow actions like

   \[ inc = (inc, inc, inc) \]
Result with Strict Synchronisation

\[ \begin{align*}
0,0,0 & \rightarrow 0,0,1 \\
0,0,1 & \rightarrow 0,0,2 \\
1,0,0 & \rightarrow 1,0,1 \\
1,0,1 & \rightarrow 1,0,2 \\
1,1,0 & \rightarrow 1,1,2 \\
1,1,2 & \rightarrow 1,2,0 \\
0,2,0 & \rightarrow 0,2,1 \\
0,2,1 & \rightarrow 0,2,2 \\
1,1,0 & \rightarrow 1,1,1
\end{align*} \]

Restricting Combined Actions

2. Loose Synchronisation
Automata operate independently.
Only allow actions like
\[ \begin{align*}
ic_1 & = (inc, -, -) \\
ic_2 & = (\neg, inc, -) \\
ic_3 & = (\neg, \neg, inc)
\end{align*} \]

Definition of Strict Synchronisation

Given
\[ A_1 = (Q_1, E, T_1, q_{01}, Q_{m1}) \]
\[ A_2 = (Q_2, E, T_2, q_{02}, Q_{m2}) \]
define
\[ A_1 \parallel A_2 = (Q_1 \times Q_2, E, T, (q_{01}, q_{02}), Q_{m1} \times Q_{m2}) \]
where
\[ T = \{ (q_1, q_2, e, r_1, r_2) | (q_1, e, r_1) \in T_1 \text{ and } (q_2, e, r_2) \in T_2 \} \]

Kinds of Synchronisation

1. Strict synchronisation \( \rightarrow \) VALID
2. Loose synchronisation \( \rightarrow \) SMV
3. Intermediate forms are possible

Using the VALID Toolset

Example: Small Factory

Machine 1
Buffer
Machine 2

Capacity: 1 work piece
Model of Small Factory

Safety Requirement

Required property:
Machine 1 must never be in a position to finish operation when the buffer is full. The system must never reach a state in which machine 1 is in state Working and the buffer is in state Full.

Introducing Selfloops

“Events that are not mentioned in an automaton can happen at any time.”

A Problem ...

Machine1: Idle
 Machine2: Idle
 Buffer: Empty

Synchronous Product Algorithm

To build the synchronous product of $A_1, \ldots, A_n$:
Create initial state $q_0 = (q_{01}, \ldots, q_{0n})$
Add $q_0$ to state set $Q$
While there are unvisited states $q = (q_1, \ldots, q_n) \in Q$:

For each event $e$ that can be executed by each automaton $A_i$ in state $q_i$:
Compute successor state $r = (r_1, \ldots, r_n)$
Add $r$ to state set $Q$ if not yet present
Create transition from $q$ to $r$ labelled $e$

How to Fix It?

Assumptions

- Machine1 and Machine2 cannot be changed.
- Events finish1 and finish2 cannot be prevented from happening.

Problem

- How can the Buffer automaton be changed to make sure that finish1 never happens when the buffer is full?