

COMP424/524-06A Topics in Software Engineering

Part I – Finite State Machines

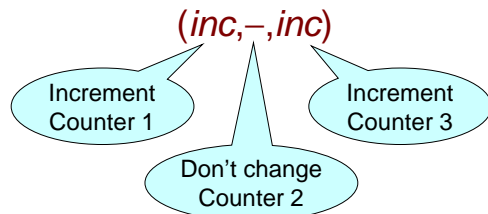
3. Synchronous Product

Robi Malik

THE UNIVERSITY OF
WAIKATO DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Combined Actions

Actions in the synchronous product:



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 4

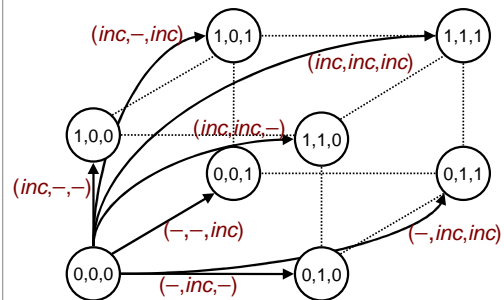
Modelling with Automata

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

How can automata interact?

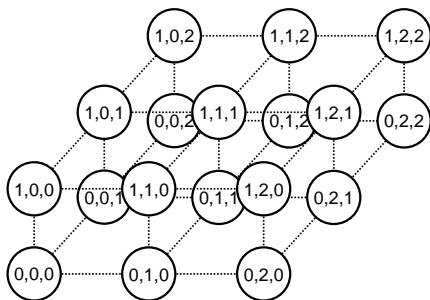
© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 2

Transitions of the Product



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 5

Example: Three Counters



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 3

Restricting Combined Actions

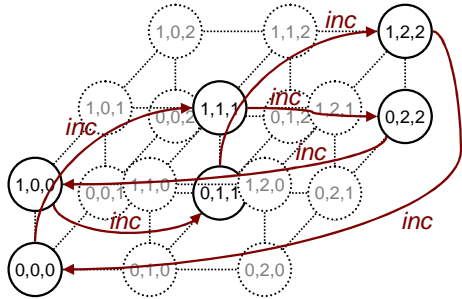
1. Strict Synchronisation

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

$$inc = (inc, inc, inc)$$

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 6

Result with Strict Synchronisation



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 7

Restricting Combined Actions

2. Loose Synchronisation

Automata operate independently.
Only allow actions like

$$inc_1 = (inc, -, -)$$

$$inc_2 = (-, inc, -)$$

$$inc_3 = (-, -, inc)$$

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 10

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

$$T = \{ (q_1, q_2, e, r_1, r_2) \mid (q_1, e, r_1) \in T_1 \text{ and } (q_2, e, r_2) \in T_2 \}$$

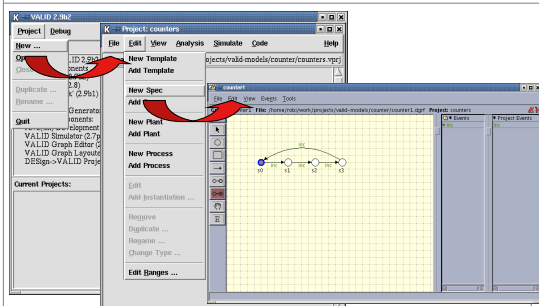
© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 8

Kinds of Synchronisation

1. Strict synchronisation
→ VALID
2. Loose synchronisation
→ SMV
3. Intermediate forms are possible

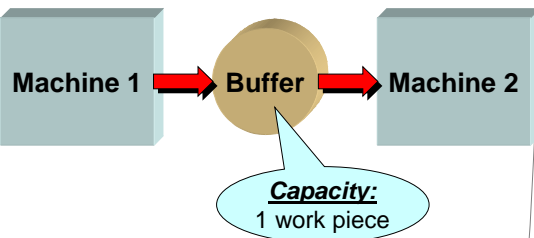
© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 11

Using the VALID Toolset

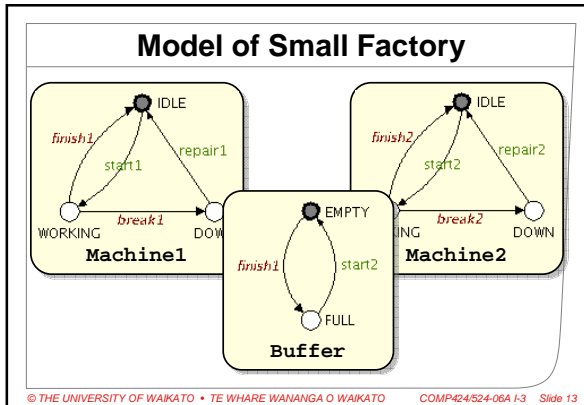


© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 9

Example: Small Factory



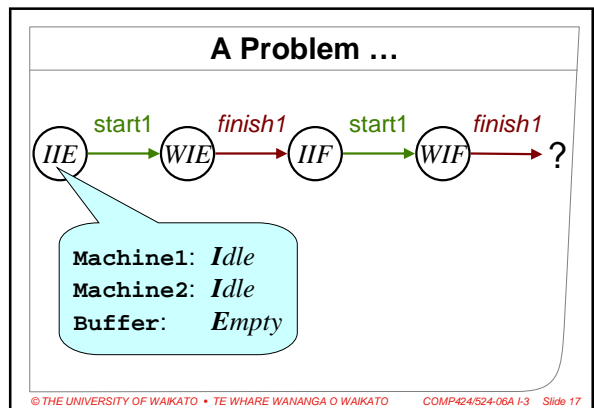
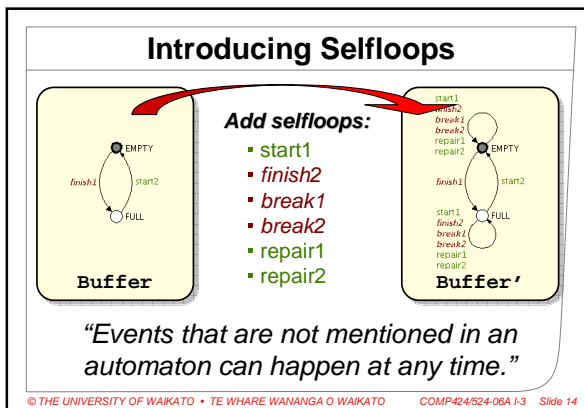
© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 12



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*.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 16



Synchronous Product Algorithm

To build the synchronous product of A_1, \dots, A_n :

Create initial state $q_0 = (q_{01}, \dots, q_{0n})$
 Add q_0 to state set Q
While there are unvisited states $q = (q_1, \dots, q_n) \in Q$:
 For each event e that can be executed by each automaton A_i in state q_i :
 Compute successor state $r = (r_1, \dots, r_n)$
 Add r to state set Q if not yet present
 Create transition from q to r labelled e

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 15

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?

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-3 Slide 18