

# Supervisory Control vs. Game Automata GASICS meeting, Bruxelles

Mikkel Larsen Pedersen and Line Juhl

Department of Computer Science, Aalborg University, Denmark

{mikkelp,linej}@cs.aau.dk



March 8, 2009

## Thought behind this work

- To make a common standpoint between control theoreticians and computer scientists
- Supervisory control of discrete event systems (DES) may be expressed as a game

*A DES is a dynamic system that evolves in accordance with the abrupt occurrence, at possible unknown irregular intervals, of physical events.*

*[Ramadge and Wonham, 1989]*

# In this talk

The control theory for DESs was initiated by Ramadge and Wonham in 1987. We will try to connect some of these concepts with game theory.

We translate a supervisory control problem given in [Ramadge and Wonham, 1989] to a problem that can be solved by UPPAAL-TIGA.

- No time – i.e. logical DES models
- Only finite DESs are considered

1 The Ramadge-Wonham Framework

2 Game Automata

3 An Example Using UPPAAL-TIGA

# Generators

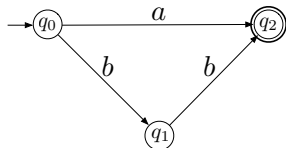
We can model a DES as an DFA-like automaton

$G = (Q, \Sigma, \delta, q_0, Q_m)$ , called a generator.

- Deterministic
- Transition function  $\delta : \Sigma \times Q \rightarrow Q$
- $Q_m$ , the marker states

$$\Sigma = \{a, b\}$$

$$Q = \{q_0, q_1, q_2\} \quad Q_m = \{q_2\}$$



# Languages

- $L(G) = \{s \in \Sigma^* \mid \delta(s, q_0) \text{ is defined}\}$ , i.e. all possible paths in  $G$ 
  - Consider all states to be accepting
- We know that all prefixes of event sequences in  $L(G)$  are in  $L(G)$ ,  $\overline{L(G)} = L(G)$  (that is  $L(G)$  is prefix closed)
- $L_m(G) = \{s \in \Sigma^* \mid s \in L(G) \text{ and } \delta(s, q_0) \in Q_m\}$
- $G$  is called nonblocking if  $\overline{L_m(G)} = L(G)$ , that is if all valid event sequences in  $G$  can be extended to a sequence ending in a marker state.

# Supervision

Let us assume that some events,  $\Sigma_c$ , are controllable and that some are uncontrollable,  $\Sigma_u$ , such that  $\Sigma = \Sigma_c \cup \Sigma_u$ .

## Control input

A control input for  $G$  is a subset  $\gamma \subseteq \Sigma$ , such that  $\Sigma_u \subseteq \gamma$ .

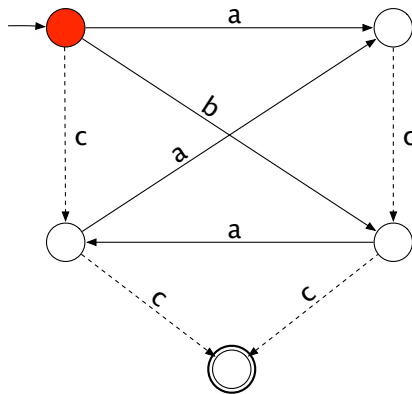
The events in  $\gamma$  are *enabled* and all other events are *disabled*.

## Supervisor

A supervisor for  $G$  is a mapping  $f : L(G) \rightarrow 2^\Sigma$ .

## Supervision – example

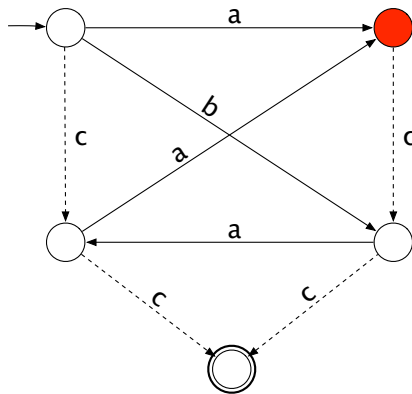
$$\Sigma_c = \{a, b\} \quad \Sigma_u = \{c\}$$



Observation $w$	Control input $f(w)$
$\varepsilon$	$\{a, c\}$

## Supervision – example

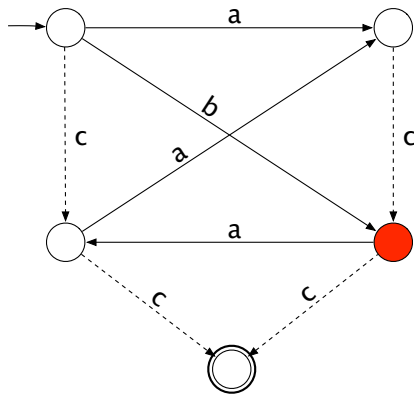
$$\Sigma_c = \{a, b\} \quad \Sigma_u = \{c\}$$



Observation $w$	Control input $f(w)$
$\varepsilon$	$\{a, c\}$
$a$	$\{c\}$

## Supervision – example

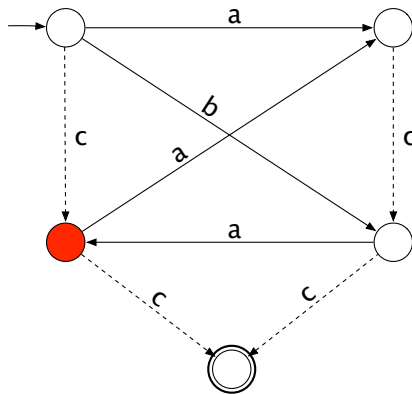
$$\Sigma_c = \{a, b\} \quad \Sigma_u = \{c\}$$



Observation $w$	Control input $f(w)$
$\varepsilon$	$\{a, c\}$
$a$	$\{c\}$
$ac$	$\{a, c\}$

## Supervision – example

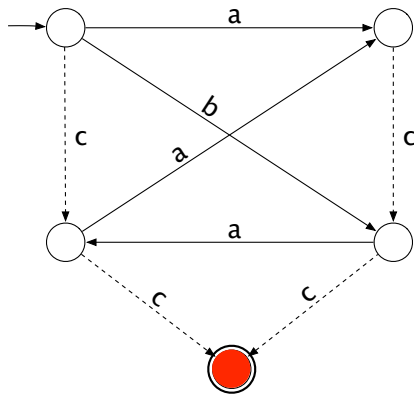
$$\Sigma_c = \{a, b\} \quad \Sigma_u = \{c\}$$



Observation $w$	Control input $f(w)$
$\varepsilon$	$\{a, c\}$
$a$	$\{c\}$
$ac$	$\{a, c\}$
$aca$	$\{c\}$

## Supervision – example

$$\Sigma_c = \{a, b\} \quad \Sigma_u = \{c\}$$



Observation $w$	Control input $f(w)$
$\varepsilon$	$\{a, c\}$
$a$	$\{c\}$
$ac$	$\{a, c\}$
$aca$	$\{c\}$

## Language under supervision

### $L_f(G)$

The language of  $G$  under supervision by  $f$  is defined by:

- $\varepsilon \in L_f(G)$  and
- $va \in L_f(G) \Leftrightarrow v \in L_f(G), a \in f(v)$  and  $va \in L(G)$ .

### $L_{mf}(G)$

The marked language under supervision by  $f$  is defined by  $L_{mf}(G) = L_m(G) \cap L_f(G)$ .

# The language to supervise

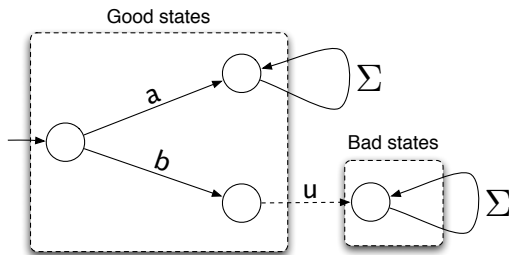
How do we specify what to supervise? By giving a language  $K \subseteq L(G)$ . But can  $K$  be controlled?

## Controllable

A language  $K \subseteq \Sigma^*$  is controllable if

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}.$$

# Controllability – example



- $K := a\Sigma^* \cup b$  is not controllable
- $K := a\Sigma^*$  is controllable

# Existence of supervisor

## Proposition

Given a non-blocking discrete-event system  $G$ :

- For  $K \subseteq L(G)$ ,  $K \neq \emptyset$ , there exists a supervisor  $f$  such that  $L_f(G) = K$  if and only if  $K = \overline{K}$  and  $K$  is controllable.
- For  $K \subseteq L_m(G)$ ,  $K \neq \emptyset$ , there exists a supervisor  $f$  such that  $L_{mf}(G) = K$  if and only if  $\overline{K} \cap L_m(G) = K$  and  $K$  is controllable.

[Ramadge and Wonham, 1989]

# An uncontrollable $K$

What to do if  $K$  is uncontrollable?

- There exist a unique largest controllable language  $K^\uparrow$  such that  $K^\uparrow \subseteq K$ .
- $K^\uparrow$  can be found using a fixpoint equation:
  - $\Omega(J) = K \cap \sup\{T \mid T \subseteq \Sigma^*, T = \overline{T}, T\Sigma_u \cap L \subseteq J\}$
  - $K_0 = K$
  - $K_{j+1} = \Omega(K_j)$
  - Find the largest fixpoint
- Minimally restrictive solution.

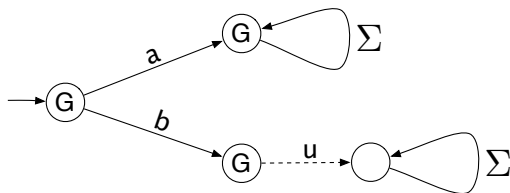
## A corresponding game automata

- We would like to transform this into a UPPAAL-TIGA game without time
- Player 1 chooses a transition and player 2 can choose to overrule this by taking an uncontrollable transition

# Strategy

- A strategy specifies what exact transition to take at each state
- The supervisor is a strategy
- We can restrict the supervisor to always output exactly one controllable event
- $L_f$  corresponds to the usual outcome of a strategy

# The language to supervise

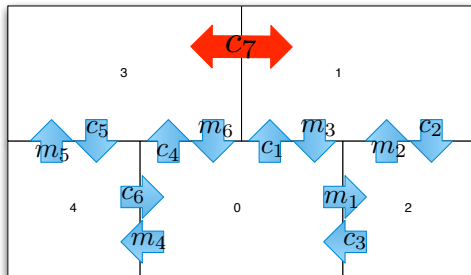


- Event sequences  $a\Sigma^*$  and  $b$  will keep us in G-states, but  $K = a\Sigma^* \cup b$  is not controllable
- The query `control: A[] G` can be verified to be true
- The strategy corresponds to the event sequence  $a\Sigma^*$

# Dictionary

Game setting	RW framework
Final states	Marker states
Automaton	Generator
Strategy	Supervisor
Environment	Uncontrollable events
Player	Controllable events
$L(G)$	$L_m(G)$
Outcome	$L_f$

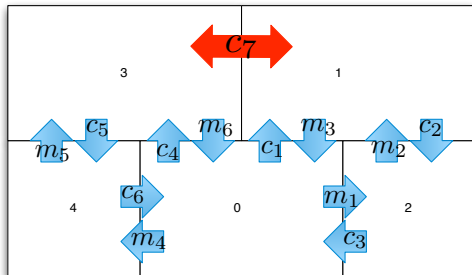
# Rules and objectives



## Rules:

- The cat start in room 2, the mouse in room 4
- The cat can move only through  $c$ -doors, the mouse only through  $m$ -doors.
- Only one animal moves at a time
- All doors except  $c_7$  are directed and can be closed at any time

## Rules and objectives



Objectives:

- The cat and the mouse must not occupy the same room at any time
- It should always be possible for the cat and the mouse to return to the initial state

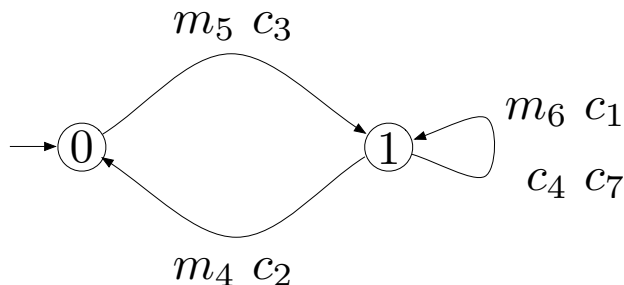
## The supervisory approach

The cat and mouse each has a generator. We can then produce the product generator,  $G$ .

- $Q = \{(i, j) : i, j \in \{0, 1, 2, 3, 4\}\}$
- $\Sigma = \{c_i, m_j : 1 \leq i \leq 7, 1 \leq j \leq 6\}$
- $\delta$  is derived from the doors in the maze
- $\{q_0\} = Q_m = \{(2, 4)\}$
- $\Sigma_u = \{c_7\}$

## A supervisor

By using the fixpoint equation the largest controllable sublanguage which satisfies the objectives is found. The supervisor can then be depicted as a automaton.



If the cat moves out, the mouse stays put and vice versa.

## Game approach

A game with two players:

- The environment – the cat and the mouse
- The controller – the gatekeeper

Winning condition for the controller:

- Always avoiding the states where the cat and mouse are in the same room – a safety condition

The problem now is how to synthesize this controller?



Ramadge, P. and Wonham, W. (1989).  
The control of discrete event systems.  
*Proceedings of the IEEE*, 77(1):81–98.