COMP 424/524-06A
Topics in Software Engineering

Part I – Finite State Machines
6. Controller Synthesis
Robi Malik

Application-Specific Properties

Question:
Does the transfer line model ensure that Buffer 1 never has an overflow or underflow?

Answer:
Use property automata and Language Inclusion Check.

Universal Properties

Controllability and Nonblocking are
- Universal properties
- General consistency checks
- Application-independent

Language Inclusion Check

Does it Really Work?

How to check that the system really does what we want it to do?
- Simulation
- Application-specific properties

Language Inclusion Check

Definition:
Let A and B be automata. The behaviour of A is included in the behaviour of B if
\[ \mathcal{L}(A) \subseteq \mathcal{L}(B) . \]

VALID checks whether:
\[ \mathcal{L}(A_1 \ldots \ldots A_n) \subseteq \mathcal{L}(B_1 \ldots \ldots B_m) . \]
Language Inclusion Check

Does the system behaviour always remain within the constraints given by a property automaton?

Can start2 happen in this state?
Can finish1 or reject happen in this state?

A Maze for Cat and Mouse

Plants for Cat and Mouse

Specification for Cat and Mouse

Environment
- 5 rooms, 13 doors
- Each door passed only by cat or mouse
- Some doors can be shut by controller

Desired behaviour
- Cat and mouse never in the same room
- Can always return to their starting point

The Easy Way…
**Supervisor Synthesis**

Automatically finds a new specification which
- ... is controllable,
- ... is nonconflicting,
- ... restricts all given specifications,
- ... is as general as possible.

**Events for Tic-Tac-Toe**

- \( \text{black}.x.y \)
  Black moves on field \((x,y)\).
  (controllable)
- \( \text{white}.x.y \)
  White moves on field \((x,y)\).
  (uncontrollable)

**Playing Tic-Tac-Toe**

```
O  O  X
X  O  
```

**Modelling the Game**

- \( \text{field}.0.0 \)
- \( \text{field}.1.0 \)
- \( \text{field}.2.0 \)
- \( \text{field}.1.1 \)
- \( \text{field}.2.1 \)
- \( \text{field}.0.1 \)
- \( \text{field}.2.2 \)
- \( \text{field}.0.2 \)
- \( \text{move} \)

**Model of Tic-Tac-Toe**

**Model**
- Two players
  - white (O) to move first
  - black (X) to move second
- Nine fields indexed by \( x = 0..2, y = 0..2 \)

**Control Objective**
- Play for black so that you will never lose.

**Making a Row**

New event:
“White wins by completing diagonal 1”
When the Game Ends

Marked state: To be nonblocking, the controller must allow the game to end in some way.

And Finally the Control Objective

These events must never happen.

Solving Games

We have used supervisor synthesis to...

- generate a strategy for Tic-Tac-Toe,
- show that Tic-Tac-Toe is a “fair” game.

Challenge

- Can you do it for more complex games?