Communication and Concurrency: Introduction

Colin Stirling (cps)

School of Informatics

16th September 2013

Goals

- Modelling: a notation for describing concurrent systems (CCS)

Goals

- Modelling: a notation for describing concurrent systems (CCS)
- Equivalence: when two descriptions are the same system

Goals

- Modelling: a notation for describing concurrent systems (CCS)
- Equivalence: when two descriptions are the same system
- Properties: modal and temporal properties of systems.

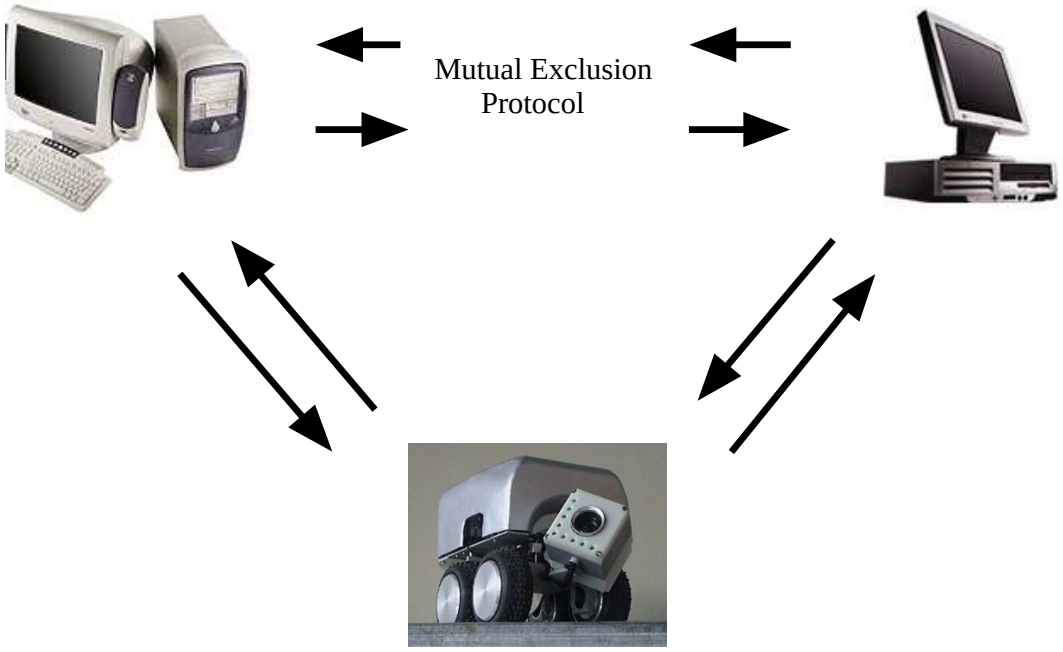
Goals

- Modelling: a notation for describing concurrent systems (CCS)
- Equivalence: when two descriptions are the same system
- Properties: modal and temporal properties of systems.
- Model checking: algorithmic techniques for checking equivalence and properties.

Goals

- Modelling: a notation for describing concurrent systems (CCS)
- Equivalence: when two descriptions are the same system
- Properties: modal and temporal properties of systems.
- Model checking: algorithmic techniques for checking equivalence and properties.
- Software tools: automatically checks properties and equivalence

An Example: Mutual Exclusion



Specification: Temporal Properties

- Mutual exclusion

Specification: Temporal Properties

- Mutual exclusion
- Absence of deadlock

Specification: Temporal Properties

- Mutual exclusion
- Absence of deadlock
- Absence of starvation

CCS model of Peterson's solution

$$
\begin{array}{rcl}
\text{B1f} & = & \overline{\text{b1rf}}.\text{B1f} + \text{b1wf}.\text{B1f} + \text{b1wt}.\text{B1t} \\
\text{B1t} & = & \overline{\text{b1rt}}.\text{B1t} + \text{b1wt}.\text{B1t} + \text{b1wf}.\text{B1f} \\
 & & \\
\text{B2f} & = & \overline{\text{b2rf}}.\text{B2f} + \text{b2wf}.\text{B2f} + \text{b2wt}.\text{B2t} \\
\text{B2t} & = & \overline{\text{b2rt}}.\text{B2t} + \text{b2wt}.\text{B2t} + \text{b2wf}.\text{B2f} \\
 & & \\
\text{K1} & = & \overline{\text{kr1}}.\text{K1} + \text{kw1}.\text{K1} + \text{kw2}.\text{K2} \\
\text{K2} & = & \overline{\text{kr2}}.\text{K2} + \text{kw2}.\text{K2} + \text{kw1}.\text{K1} \\
 & & \\
\text{P1} & = & \overline{\text{b1wt}}.\text{req1}.\overline{\text{kw2}}.\text{P11} \\
\text{P11} & = & \text{b2rt}.\text{P11} + \text{b2rf}.\text{P12} + \text{kr2}.\text{P11} + \\
 & & \text{kr1}.\text{P12} \\
\text{P12} & = & \text{enter1}.\text{exit1}.\overline{\text{b1wf}}.\text{P1} \\
 & & \\
\text{P2} & = & \overline{\text{b2wt}}.\text{req2}.\overline{\text{kw1}}.\text{P21} \\
\text{P21} & = & \text{b1rf}.\text{P22} + \text{b1rt}.\text{P21} + \text{kr1}.\text{P21} + \\
 & & \text{kr2}.\text{P22} \\
\text{P22} & = & \text{enter2}.\text{exit2}.\overline{\text{b2wf}}.\text{P2} \\
 & & \\
\text{Peterson} & = & (\text{P1} \mid \text{P2} \mid \text{K1} \mid \text{B1f} \mid \text{B2f}) \setminus L
\end{array}
$$

Formalising Temporal Properties

$$
\begin{array}{rcl}
\text{Mutex} & = & \text{AG}\ ([\text{exit1}]\text{ff} \vee [\text{exit2}]\ \text{ff}) \\
\text{NoDeadlock} & = & \text{AG}\ \langle - \rangle\ \text{tt} \\
\text{NoStarvation} & = & \text{AG}([\text{req1}]\ \text{AF}\ \langle \text{exit1} \rangle\ \text{tt}) \wedge \\
 & & \text{AG}([\text{req2}]\ \text{AF}\ \langle \text{exit2} \rangle\ \text{tt})
\end{array}
$$

Model checking

- The Edinburgh Concurrency Workbench
  - A tool for simulating and verifying CCS agents
  - http://homepages.inf.ed.ac.uk/perdita/cwb/

Model checking

- The Edinburgh Concurrency Workbench
  - A tool for simulating and verifying CCS agents
  - http://homepages.inf.ed.ac.uk/perdita/cwb/
- Proving Peterson's solution correct

Model checking

- The Edinburgh Concurrency Workbench
  - A tool for simulating and verifying CCS agents
  - http://homepages.inf.ed.ac.uk/perdita/cwb/
- Proving Peterson's solution correct
  - `Command: checkprop(Peterson,Mutex);`
  - `Command: checkprop(Peterson,NoDeadlock);`
  - `Command: checkprop(Peterson,NoStarvation);`

Model checking

- The Edinburgh Concurrency Workbench
  - A tool for simulating and verifying CCS agents
  - http://homepages.inf.ed.ac.uk/perdita/cwb/
- Proving Peterson's solution correct
  - `Command: checkprop(Peterson,Mutex);`
  - `true`
  - `Command: checkprop(Peterson,NoDeadlock);`
  - `true`
  - `Command: checkprop(Peterson,NoStarvation);`
  - `true`

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking
- Software: Microsoft prototype software model checking

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking
- Software: Microsoft prototype software model checking
- ⋮

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking
- Software: Microsoft prototype software model checking
- ⋮
- Life: cells and pathways (Systems biology: huge new area)

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking
- Software: Microsoft prototype software model checking
- ⋮
- Life: cells and pathways (Systems biology: huge new area)

Paper on hardware verification and one on BLAST tool for software verification on course web page

In Reality . . .

Modelling and model checking large (and infinite state) systems

- Circuits: since Pentium-bug Intel uses model checking
- Software: Microsoft prototype software model checking
- ⋮
- Life: cells and pathways (Systems biology: huge new area)

Paper on hardware verification and one on BLAST tool for software verification on course web page
Look up "model checking" in Wikipedia, Google, . . .