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

Mutual Exclusion Protocol
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

\[
B_{1f} = b_{1rf}B_{1f} + b_{1wf}B_{1f} + b_{1wt}B_{1t}
\]

\[
B_{1t} = b_{1rt}B_{1t} + b_{1wt}B_{1t} + b_{1wf}B_{1f}
\]

\[
B_{2f} = b_{2rf}B_{2f} + b_{2wf}B_{2f} + b_{2wt}B_{2t}
\]

\[
B_{2t} = b_{2rt}B_{2t} + b_{2wt}B_{2t} + b_{2wf}B_{2f}
\]

\[
K_{1} = k_{r1}K_{1} + k_{w1}K_{1} + k_{w2}K_{2}
\]

\[
K_{2} = k_{r2}K_{2} + k_{w2}K_{2} + k_{w1}K_{1}
\]

\[
P_{1} = b_{1wt}req_{1}k_{w2}P_{11}
\]

\[
P_{11} = b_{2rt}P_{11} + b_{2rf}P_{12} + k_{r2}P_{11} + k_{r1}P_{12}
\]

\[
P_{12} = enter_{1}exit_{1}b_{1wf}P_{1}
\]

\[
P_{2} = b_{2wt}req_{2}k_{w1}P_{21}
\]

\[
P_{21} = b_{1rf}P_{22} + b_{1rt}P_{21} + k_{r1}P_{21} + k_{r2}P_{22}
\]

\[
P_{22} = enter_{2}exit_{2}b_{2wf}P_{2}
\]

Peterson = (P_{1} | P_{2} | K_{1} | B_{1f} | B_{2f}) \backslash L
Formalising Temporal Properties

\[
\begin{align*}
\text{Mutex} & = \text{AG} ([\text{exit1}] \text{ff} \lor [\text{exit2}] \text{ff}) \\
\text{NoDeadlock} & = \text{AG} \langle - \rangle \text{tt} \\
\text{NoStarvation} & = \text{AG}([\text{req1}] \text{AF} \langle \text{exit1} \rangle \text{tt}) \land \\
& \quad \text{AG}([\text{req2}] \text{AF} \langle \text{exit2} \rangle \text{tt})
\end{align*}
\]
Model checking

- The Edinburgh Concurrency Workbench
  - A tool for simulating and verifying CCS agents
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/permita/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
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
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, . . .