# Edinburgh Concurrency Workbench
# Getting Started

Colin Stirling (cps)

School of Informatics

10th October 2013

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline
- Type quit; to finish a session

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline
- Type quit; to finish a session
- Defining processes

$$\texttt{agent Process} = \texttt{a.0} + \texttt{'a.Process;}$$

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline
- Type quit; to finish a session
- Defining processes

$$\texttt{agent Process} = \texttt{a.0} + \texttt{'a.Process;}$$

- Process names begin with upper case and action names are written in lower case

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline
- Type quit; to finish a session
- Defining processes

$$\texttt{agent Process} = \texttt{a.0} + \texttt{'a.Process;}$$

- Process names begin with upper case and action names are written in lower case

- 'a denotes the action $\bar{a}$

## The Edinburgh Concurrency Workbench

- Type cwb on a dice machine
- All commands end with a semicolon ; and a newline
- Type quit; to finish a session
- Defining processes

$$\texttt{agent Process} = \texttt{a.0} + \texttt{'a.Process;}$$

- Process names begin with upper case and action names are written in lower case
- 'a denotes the action $\bar{a}$

- Process can be used in other definitions

$$\texttt{agent Process1} = \texttt{b.Process} \mid \texttt{a.Process;}$$

## Modal logic in the workbench

- `tt` is written `T`

## Modal logic in the workbench

- `tt` is written `T`
- `ff` is written `F`

## Modal logic in the workbench

- `tt` is written `T`
- `ff` is written `F`
- $\wedge$ is written `&`

## Modal logic in the workbench

- `tt` is written `T`
- `ff` is written `F`
- $\wedge$ is written `&`
- $\vee$ is written `|`
- Defining properties

$$\texttt{prop Property} = \langle a \rangle \texttt{T};$$

## Modal logic in the workbench

- `tt` is written `T`
- `ff` is written `F`
- $\wedge$ is written &
- $\vee$ is written |
- Defining properties

  `prop Property` $= \langle a \rangle$`T;`
- Property can be used in other definitions

  `prop Property1` $= [b]$`Property` | $[a]$`F;`

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.
- CTL$^-$ can be encoded into the mu-calculus. The encoding is contained in a file that can be downloaded from the modules homepage. After giving it a name, say ctl.cwb, type the command

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.
- CTL$^-$ can be encoded into the mu-calculus. The encoding is contained in a file that can be downloaded from the modules homepage. After giving it a name, say ctl.cwb, type the command
- `input "ctl.cwb";`

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.
- CTL$^-$ can be encoded into the mu-calculus. The encoding is contained in a file that can be downloaded from the modules homepage. After giving it a name, say ctl.cwb, type the command
- `input "ctl.cwb";`
- `AG` $\Phi$ is written `AG` $(\Phi)$

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.
- CTL$^-$ can be encoded into the mu-calculus. The encoding is contained in a file that can be downloaded from the modules homepage. After giving it a name, say ctl.cwb, type the command
- `input "ctl.cwb";`
- `AG` $\Phi$ is written `AG` $(\Phi)$
- Given a process $E$ and a property $P$, the command

  `checkprop(E,P);`

  checks if $E$ satisfies $P$

## CTL$^-$ in the workbench

- The workbench's specification language is not CTL$^-$, but the *modal mu-calculus*.
- CTL$^-$ can be encoded into the mu-calculus. The encoding is contained in a file that can be downloaded from the modules homepage. After giving it a name, say ctl.cwb, type the command
- `input "ctl.cwb";`
- `AG` $\Phi$ is written `AG` $(\Phi)$
- Given a process $E$ and a property $P$, the command

  `checkprop(E,P);`

  checks if $E$ satisfies $P$
- Answer the questions about playing games with "no".