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

agent Process = a
Process
Process
agent Process1 = b
Process
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
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

\[
\text{agent Process} = a.0 + \text{'}a.\text{Process};
\]
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

```
agent Process = a.0 + '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

```
agent Process = a.0 + '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

\[
\text{agent } \text{Process} = \text{a.}0 + \text{'}a.\text{Process};
\]

- Process names begin with upper case and action names are written in lower case
- ‘a denotes the action \( \overline{\text{a}} \)
- Process can be used in other definitions

\[
\text{agent } \text{Process1} = \text{b.}\text{Process} \mid \text{a.}\text{Process};
\]
Modal logic in the workbench

► $\texttt{tt}$ is written $T$
Modal logic in the workbench

- \( \top \) is written \( T \)
- \( \bot \) is written \( F \)
Modal logic in the workbench

- \( \top \) is written \( T \)
- \( \bot \) is written \( F \)
- \( \land \) is written \&
Modal logic in the workbench

- $\top$ is written $\top$
- $\bot$ is written $\bot$
- $\land$ is written $\&$
- $\lor$ is written $|$ 
- Defining properties

$$\text{prop Property} = \langle a \rangle T;$$
Modal logic in the workbench

- $\top$ is written $T$
- $\bot$ is written $F$
- $\land$ is written $\&$
- $\lor$ is written $|$
- Defining properties

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

- Property can be used in other definitions

$$\text{prop Property1} = [b]\text{Property} \mid [a]F;$$
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 Φ 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";```
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 Φ is written AG (Φ)
CTL$^-$ in the workbench

- The workbench’s specification language is not CTL$^-$, but the \textit{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 $P$
The workbench’s specification language is notCTL−, 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 Φ is written AG (Φ)

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”.