# Resume

- Started with looking at a particular process calculus, Milner's CCS (Calculus of Communicating Systems)

- Some operators and rules for transitions: . prefix, $\sum$, $|$, $\backslash K$

$$\mathrm{R}(|\,\mathrm{com}) \quad \frac{E \mid F \xrightarrow{\tau} E' \mid F'}{E \xrightarrow{a} E' \qquad F \xrightarrow{\overline{a}} F'}$$

$$\mathrm{R}(|) \quad \frac{E \mid F \xrightarrow{a} E' \mid F}{E \xrightarrow{a} E'} \qquad \frac{E \mid F \xrightarrow{a} E \mid F'}{F \xrightarrow{a} F'}$$

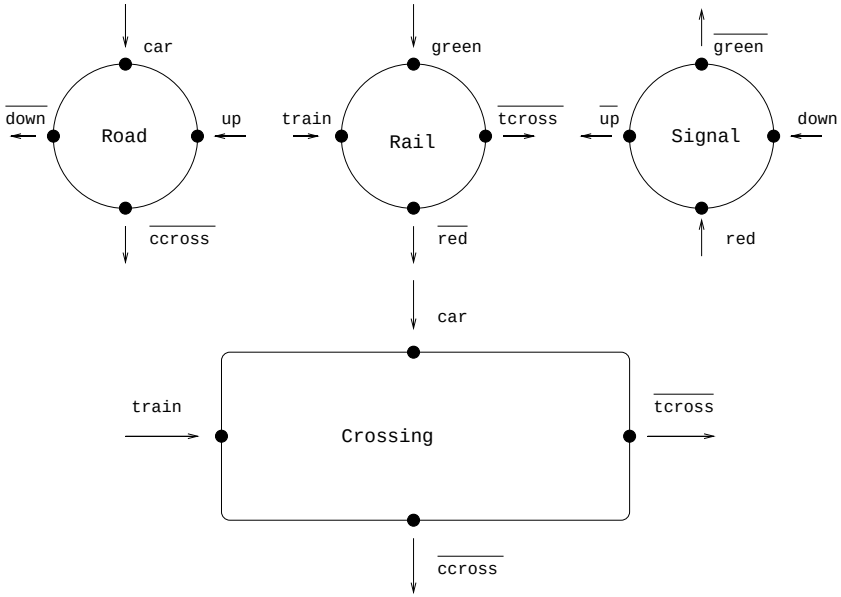
- Transition Graphs and Flow Graphs

# Example

$P \equiv F$ means that $P$ abbreviates $F$.

$$
\begin{array}{lcl}
\texttt{Road} & \stackrel{\text{def}}{=} & \texttt{car.up.}\overline{\texttt{ccross}}\texttt{.}\overline{\texttt{down}}\texttt{.Road} \\
\texttt{Rail} & \stackrel{\text{def}}{=} & \texttt{train.green.}\overline{\texttt{tcross}}\texttt{.}\overline{\texttt{red}}\texttt{.Rail} \\
\texttt{Signal} & \stackrel{\text{def}}{=} & \overline{\texttt{green}}\texttt{.red.Signal} + \overline{\texttt{up}}\texttt{.down.Signal} \\
 & & \\
\texttt{Crossing} & \equiv & (\texttt{Road} \mid \texttt{Rail} \mid \texttt{Signal}) \backslash K \\
 & & \\
K & = & \{\texttt{green}, \texttt{red}, \texttt{up}, \texttt{down}\}
\end{array}
$$

# Flow Graphs


car
down
Road
up
ccross
green
train
Rail
tcross
red
green
up
Signal
down
red
car
train
Crossing
tcross
ccross

# Protocol that may lose messages

$$
\begin{array}{lcl}
\texttt{Sender} & \stackrel{\text{def}}{=} & \texttt{in}(x).\overline{\texttt{sm}}(x).\texttt{Send1}(x) \\
\texttt{Send1}(x) & \stackrel{\text{def}}{=} & \texttt{ms}.\overline{\texttt{sm}}(x).\texttt{Send1}(x) + \texttt{ok}.\texttt{Sender} \\
\texttt{Medium} & \stackrel{\text{def}}{=} & \texttt{sm}(y).\texttt{Med1}(y) \\
\texttt{Med1}(y) & \stackrel{\text{def}}{=} & \overline{\texttt{mr}}(y).\texttt{Medium} + \tau.\overline{\texttt{ms}}.\texttt{Medium} \\
\texttt{Receiver} & \stackrel{\text{def}}{=} & \texttt{mr}(x).\overline{\texttt{out}}(x).\overline{\texttt{ok}}.\texttt{Receiver} \\
 & & \\
\texttt{Protocol} & \equiv & (\texttt{Sender} \mid \texttt{Medium} \mid \texttt{Receiver})\backslash\{\texttt{sm}, \texttt{ms}, \texttt{mr}, \texttt{ok}\}
\end{array}
$$

# Abstracting from silent activity

Difference between $\tau$ and "observable" actions.
Assume $E$ may at some time perform $\mathtt{ok}$

$$(E \mid \overline{\mathtt{ok}}.\mathtt{Resource}) \backslash \{\mathtt{ok}\}$$

Access to $\mathtt{Resource}$ is triggered by $\mathtt{ok}$ by $E$

Observation of $\mathtt{ok}$ = release of $\mathtt{Resource}$

$\tau$ cannot be observed in this way

# Observable transitions

$E \stackrel{\varepsilon}{\Longrightarrow} F$ or $E \stackrel{a}{\Longrightarrow} F$ where $a \neq \tau$

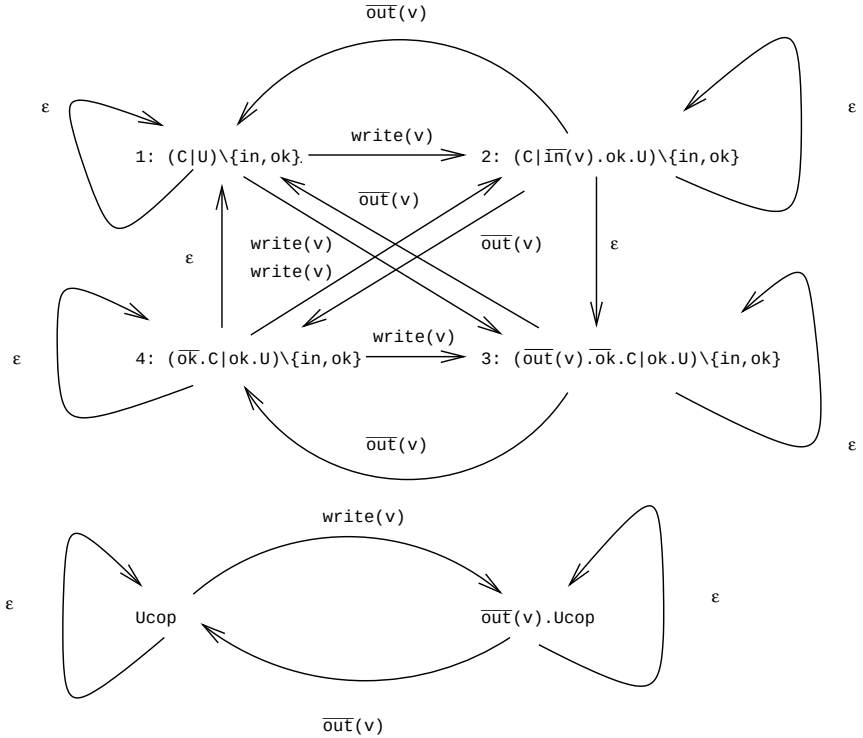
$$\mathrm{R}(\stackrel{\varepsilon}{\Longrightarrow}) \quad E \stackrel{\varepsilon}{\Longrightarrow} E \qquad \frac{E \stackrel{\varepsilon}{\Longrightarrow} F}{E \stackrel{\tau}{\longrightarrow} E' \quad E' \stackrel{\varepsilon}{\Longrightarrow} F}$$

$$\mathrm{R}(\stackrel{a}{\Longrightarrow}) \qquad \frac{E \stackrel{a}{\Longrightarrow} F}{E \stackrel{\varepsilon}{\Longrightarrow} E' \quad E' \stackrel{a}{\longrightarrow} F' \quad F' \stackrel{\varepsilon}{\Longrightarrow} F}$$

# Observable Transition Graphs

$$
\begin{array}{lcl}
\mathtt{C} & \stackrel{\text{def}}{=} & \mathtt{in}(x).\overline{\mathtt{out}}(x).\overline{\mathtt{ok}}.\mathtt{C} \\
\mathtt{U} & \stackrel{\text{def}}{=} & \mathtt{write}(x).\overline{\mathtt{in}}(x).\mathtt{ok}.\mathtt{U} \\
\mathtt{Ucop} & \stackrel{\text{def}}{=} & \mathtt{write}(x).\overline{\mathtt{out}}(x).\mathtt{Ucop}
\end{array}
$$

# Observable Transition Graphs



# Summary

1. Syntax of CCS: prefix, sum, parallel composition, restriction (but not renaming)
2. Two types of transition, $\xrightarrow{a}$ $\stackrel{a}{\Longrightarrow}$
3. Two types of transition graph that abstracts from derivation of transitions
4. Flow Graphs

# Process Calculi

1. Lots of different process calculi (ACP, CSP, ...)

2. Even "formats" for defining behavioural rules

3. Lots of added extras: time, probability, location, ...

4. Consider the restricted process language where $I$ is finite

$$E ::= P \mid \sum\{a_i.E_i : i \in I\} \mid E_1 \mid E_2 \mid E\backslash\{a\}$$

A (closed) process, a finite family $\{P_i \stackrel{\text{def}}{=} E_i : 1 \leq i \leq n\}$ of definitions, where all the process names in each $E_i$ belong to the set $\{P_1, \ldots, P_n\}$.

"Turing powerful" (simulate Turing machines)

# Doing a counter

$$
\begin{array}{lcl}
\texttt{Count} & \stackrel{\text{def}}{=} & \texttt{round.Count} + \texttt{up.}(\texttt{Count}_1 \mid a.\texttt{Count})\backslash\{a\} \\
\texttt{Count}_1 & \stackrel{\text{def}}{=} & \texttt{down.}\overline{a}.0 + \texttt{up.}(\texttt{Count}_2 \mid b.\texttt{Count}_1)\backslash\{b\} \\
\texttt{Count}_2 & \stackrel{\text{def}}{=} & \texttt{down.}\overline{b}.0 + \texttt{up.}(\texttt{Count}_1 \mid a.\texttt{Count}_2)\backslash\{a\}
\end{array}
$$

But so what?

1. unclear what the significance of this is

2. is there a concurrent version of Church-Turing thesis for sequential programs?

# Reasoning about processes

Are two descriptions equivalent?

Is `Protocol` equivalent to `Cop`?

Do descriptions have important (temporal) properties?

`Crossing` $\models$ never has crashes?

`Crossing` $\models$ whenever a car approaches eventually it crosses?

# Exercise

Is this pair equivalent?

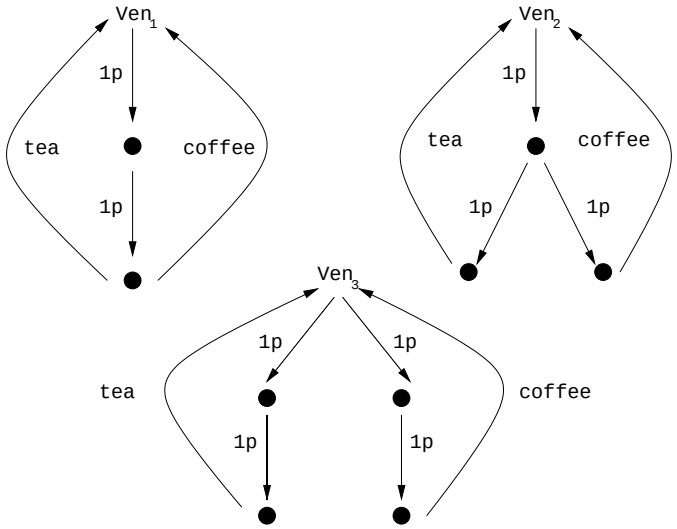
$$\texttt{Cl} \stackrel{\text{def}}{=} \texttt{tick.Cl}$$
$$\texttt{Cl}' \stackrel{\text{def}}{=} \texttt{tick.tick.Cl}'$$

# Exercise

Are any of these equivalent?

$$
\begin{array}{lcl}
\mathtt{Ven}_1 & \stackrel{\text{def}}{=} & \mathtt{1p.1p.(tea.Ven}_1 + \mathtt{coffee.Ven}_1\mathtt{)} \\
\mathtt{Ven}_2 & \stackrel{\text{def}}{=} & \mathtt{1p.(1p.tea.Ven}_2 + \mathtt{1p.coffee.Ven}_2\mathtt{)} \\
\mathtt{Ven}_3 & \stackrel{\text{def}}{=} & \mathtt{1p.1p.tea.Ven}_3 + \mathtt{1p.1p.coffee.Ven}_3
\end{array}
$$

# Pictorially



# Equivalences

Unlimited choice of candidates for equivalence

Some criteria

1. Should be a congruence w.r.t. process combinators (Language equivalence excluded)

2. Should preserve "crucial properties" (Trace equivalence excluded)

3. Should have a nice mathematical theory (. . . excluded)

# Approaches to equivalence I

"Semantical approach" (compare $\lambda$-calculus)

1. $E \equiv F$ if they have same "basic features"

2. Extend to a congruence: largest $\equiv^c \subseteq \equiv$ such that

for all process contexts $C[\ ]$, $C[E] \equiv C[F]$

Sensitive to

what are basic features?

what are the process combinators?

is $\equiv^c$ definable independently?

# Approaches to equivalence II

"Logical approach"

1. Give a logic $L$ for properties of processes

2. $E \equiv F$ iff for all properties $\Phi \in L$. $E \models \Phi \quad \text{iff} \quad F \models \Phi$

Sensitive to

1. what is an appropriate logic?

2. is $\equiv$ a congruence?

3. is $\equiv$ definable independently?

# General logical approach

Ehrenfeucht-Fraisse Games

1. Two structures (e.g., processes)

2. How alike are they?

3. Play games to distinguish them

# Ingredients

1. Two players V (verifier) R (refuter)

   R wants to show structures are distinguishable

   V wants to show they are not

2. What is a move?

3. What is it to win?

# Distinguishable Processes

A pair of processes $E$ and $F$ is distinguishable if one has a transition the other doesnt

$$E \xrightarrow{a} E' \quad \text{and} \quad \text{not}(F \xrightarrow{a}) \quad \text{or} \quad F \xrightarrow{a} F' \quad \text{and} \quad \text{not}(E \xrightarrow{a})$$

Alternatively, w.r.t. *observable* disitinguishability

$$E \stackrel{a}{\Longrightarrow} E' \quad \text{and} \quad \text{not}(F \stackrel{a}{\Longrightarrow}) \quad \text{or} \quad F \stackrel{a}{\Longrightarrow} F' \quad \text{and} \quad \text{not}(E \stackrel{a}{\Longrightarrow})$$

# Bisimulation Game $\mathsf{G}(E_0, F_0)$

Play of $\mathsf{G}(E_0, F_0)$ is a finite/infinite sequence $(E_0, F_0) \ldots (E_i, F_i) \ldots$

If $(E_0, F_0) \ldots (E_j, F_j)$ then $(E_{j+1}, F_{j+1})$ is determined by move

- Player R chooses a transition $E_j \xrightarrow{a} E_{j+1}$, then player V chooses a transition with the same label $F_j \xrightarrow{a} F_{j+1}$
- Player R chooses a transition $F_j \xrightarrow{a} F_{j+1}$, then player V chooses a transition with the same label $E_j \xrightarrow{a} E_{j+1}$

Winning a play

R wins if reach a distinguishable pair

V wins otherwise (play is infinite or becomes stuck)

# Examples

V wins every play of $\mathsf{G}(\mathtt{Cl}, \mathtt{Cl}')$

$$\begin{array}{lcl} \mathtt{Cl} & \stackrel{\text{def}}{=} & \mathtt{tick}.\mathtt{Cl} \\ \mathtt{Cl}' & \stackrel{\text{def}}{=} & \mathtt{tick}.\mathtt{tick}.\mathtt{Cl}' \end{array}$$

V and R both win plays of $\mathsf{G}(\mathtt{Ven}_1, \mathtt{Ven}_2)$

$$\begin{array}{lcl} \mathtt{Ven}_1 & \stackrel{\text{def}}{=} & \mathtt{1p}.\mathtt{1p}.(\mathtt{tea}.\mathtt{Ven}_1 + \mathtt{coffee}.\mathtt{Ven}_1) \\ \mathtt{Ven}_2 & \stackrel{\text{def}}{=} & \mathtt{1p}.(\mathtt{1p}.\mathtt{tea}.\mathtt{Ven}_2 + \mathtt{1p}.\mathtt{coffee}.\mathtt{Ven}_2) \\ \mathtt{Ven}_3 & \stackrel{\text{def}}{=} & \mathtt{1p}.\mathtt{1p}.\mathtt{tea}.\mathtt{Ven}_3 + \mathtt{1p}.\mathtt{1p}.\mathtt{coffee}.\mathtt{Ven}_3 \end{array}$$

However, V is able always to win $\mathsf{G}(\mathtt{Ven}_1, \mathtt{Ven}_2)$