Traces, Pomsets, Fairness, and Full Abstraction for Communicating Processes

Stephen Brookes

Department of Computer Science
Carnegie Mellon University

Slides adapted from talk given at

CONCUR 2002

PARALLEL PARADIGMS

- **Shared memory**
  - concurrent reads and writes

- **Asynchronous communication**
  - output always enabled
  - input waits until data is available
  - channels behave like queues

- **Synchronous communication**
  - output waits until matching input
  - input waits until matching output
  - synchronized handshake

SEMANTIC MODELS

- **State-based**
  - sequences of state changes

    $$(s_0, s_0')(s_1, s_1') \ldots (s_n, s_n') \ldots$$

  - "transition traces"

- **Communication-based**
  - communication traces + book-keeping

    $$(\lambda_1 \lambda_2 \ldots \lambda_n \ldots, X)$$

  - "failures"

FAIRNESS

For shared-memory or asynchronous i/o

$$P \| Q \xrightarrow{\lambda} \quad \text{if} \quad P \xrightarrow{\lambda} \text{ or } Q \xrightarrow{\lambda}$$

- Assume *no process left behind*
- Weak (process) fairness
- Satisfied by round-robin scheduler
- Can be modelled by transition traces
- Ensures that

  local *stop*=**false in**
  (*stop*:=**true** $\|$ **while** $\neg$*stop* **do** *go*)
  always terminates

FAIRNESS

For synchronous i/o

$$P\|Q \xrightarrow{\lambda} \quad \text{if} \quad P \xrightarrow{\lambda} \text{ or } Q \xrightarrow{\lambda}$$
$$P\|Q \xrightarrow{\delta} \quad \text{if} \quad P \xrightarrow{h!v} \ \& \ Q \xrightarrow{h?v}$$

- Assume *no process left behind*, and *no synchronization ignored*
- Weak (synchronizing) fairness
- Satisfied by variant of round-robin
- Not modelled by failures
- Ensures validity of fair liveness laws, e.g.

$$\textbf{local } h \textbf{ in } P\|(h!0;Q)\|(h?x;R)$$
$$= \textbf{local } h \textbf{ in } P\|Q\|(x{:=}0;R)$$

if $h \notin chans(P)$

THIS TALK

- **A fair trace semantics for CSP**
  - synchronous communication
  - avoids complex book-keeping
  - *state* handled implicitly
  - generalization of failures
  - fully abstract
- **A partial-order semantics**
  - synchronous pomsets
  - recovering the fair traces
- **Adaptability**
  - asynchronous communication
  - shared memory

SYNTAX

- **Processes**

$$\begin{aligned} P ::= \ & \textbf{skip} \mid x{:=}e \mid h?x \mid h!e \\ \mid \ & P_1; P_2 \mid P_1 \| P_2 \mid P_1 \sqcap P_2 \\ \mid \ & \textbf{if } G \textbf{ fi} \mid \textbf{do } G \textbf{ od} \\ \mid \ & \textbf{if } b \textbf{ then } P_1 \textbf{ else } P_2 \mid \textbf{while } b \textbf{ do } P \\ \mid \ & \textbf{local } h \textbf{ in } P \mid \textbf{local } x{=}e \textbf{ in } P \end{aligned}$$

- **Guarded commands**

$$G ::= (h?x \rightarrow P) \mid G_1 \Box G_2$$

NOTES

- $P_1 \sqcap P_2$ is *internal choice*
- $G_1 \Box G_2$ is *external choice*

ACTIONS

$$\begin{array}{rcll} \lambda & ::= & x{=}v & \text{read} \\ & | & x{:=}v & \text{write} \\ & | & h?v & \text{input} \\ & | & h!v & \text{output} \\ & | & \delta_X & \text{wait} \end{array}$$

where $X \subseteq \mathbf{Dir} = \{h?, h! \mid h \in \mathbf{Chan}\}$

TRACES

Finite or infinite sequences of actions

$$\alpha \in \Lambda^\infty = \Lambda^+ \cup \Lambda^\omega$$

$$\delta\lambda = \lambda\delta = \lambda$$

STATES

Characterized implicitly by enabling relation

$$s \xrightarrow{\lambda} s'$$

NOTATION

- Λ is the set of actions
- $\Delta = \{\delta_X \mid X \subseteq_{\text{fin}} \mathbf{Dir}\}$ is the set of waiting actions
- δ abbreviates $\delta_{\{\}}$
- δ_λ abbreviates $\delta_{\{\lambda\}}$

MATCHING

- $match(\lambda_1, \lambda_2)$ iff $\{\lambda_1, \lambda_2\} = \{h?v, h!v\}$
- $match(d_1, d_2)$ iff $\{d_1, d_2\} = \{h?, h!\}$
- $match(X_1, X_2)$ iff $match(d_1, d_2)$ for some $d_1 \in X_1, d_2 \in X_2$

OPERATIONAL SEMANTICS

Sketch

- **State**

$$S = \mathbf{Ide} \rightharpoonup_{fin} Z$$

- **Transitions**

$$P, s \xrightarrow{\lambda} P', s'$$

$$G, s \xrightarrow{\lambda} P', s'$$

- **Termination**

$$P, s \ \mathbf{term}$$

- **Fair executions**

$$P, s \xrightarrow{\alpha}$$

TRANSITION RULES

for guarded commands

$$\frac{}{(h?x \to P), s \xrightarrow{h?v} x{:=}v; P, s}$$

$$\frac{}{(h?x \to P), s \xrightarrow{\delta_{h?}} (h?x \to P), s}$$

$$\frac{G_1, s \xrightarrow{\lambda} P_1, s'}{G_1 \Box G_2, s \xrightarrow{\lambda} P_1, s'} \quad \lambda \notin \Delta$$

$$\frac{G_2, s \xrightarrow{\lambda} P_2, s'}{G_1 \Box G_2, s \xrightarrow{\lambda} P_2, s'} \quad \lambda \notin \Delta$$

$$\frac{G_1, s \xrightarrow{\delta_X} G_1, s \quad G_2, s \xrightarrow{\delta_Y} G_2, s}{G_1 \Box G_2, s \xrightarrow{\delta_{X \cup Y}} G_1 \Box G_2, s}$$

TRANSITION RULES

for parallel composition

$$\frac{P_1, s \xrightarrow{\lambda} P_1', s'}{P_1 \| P_2, s \xrightarrow{\lambda} P_1' \| P_2, s'}$$

$$\frac{P_2, s \xrightarrow{\lambda} P_2', s'}{P_1 \| P_2, s \xrightarrow{\lambda} P_1 \| P_2', s'}$$

$$\frac{P_1, s \xrightarrow{\lambda_1} P_1', s \quad P_2, s \xrightarrow{\lambda_2} P_2', s}{P_1 \| P_2, s \xrightarrow{\delta} P_1' \| P_2', s} \text{ if } match(\lambda_1, \lambda_2)$$

interleaving,
possible synchronization

TRANSITION RULES

for local channels

$$\frac{P, s \xrightarrow{\lambda} P', s' \quad chan(\lambda) \neq h}{\textbf{local } h \textbf{ in } P, s \xrightarrow{\lambda} \textbf{local } h \textbf{ in } P', s'}$$

$$\frac{P, s \xrightarrow{\delta_X} P', s}{\textbf{local } h \textbf{ in } P, s \xrightarrow{\delta_{X \setminus h}} \textbf{local } h \textbf{ in } P', s}$$

only non-local actions
remain visible

TERMINATION

$$\frac{}{\textbf{skip}, s \textbf{ term}}$$

$$\frac{P_1, s \textbf{ term} \quad P_2, s \textbf{ term}}{P_1 \| P_2, s \textbf{ term}}$$

$$\frac{P, s \textbf{ term}}{\textbf{local } h \textbf{ in } P, s \textbf{ term}}$$

all parallel components must terminate

FAIRNESS

. . . operationally

- Can give syntax-directed definition of the fair transition sequences of P
- A sequence for $P_1 \| P_2$, s is *fair* iff
  - (i) the subsequences for P_1 and P_2 are each maximal (and fair)
  - (ii) the processes do not become *blocked* on a matching pair of directions
- Maximal = finite and terminal, or infinite
- A *computation* is a maximal sequence of *consecutive* transitions

EXAMPLES

- $h!0, s \xrightarrow{\delta_{h!}} h!0, s \xrightarrow{\delta_{h!}} \cdots \mathit{fair}$
- $h?x, s \xrightarrow{\delta_{h?}} h?x, s \xrightarrow{\delta_{h?}} \cdots \mathit{fair}$
- But the computation

  $(h!0 \| h?x), s \xrightarrow{\delta_{h!}} (h!0 \| h?x), s \xrightarrow{\delta_{h?}} \cdots$

  is *not fair*
- Every fair computation of $h!0 \| h?x$ has an output $h!0$, or an input $h?v$, or a write $x{:=}0$
- Every fair computation of **local** h **in** $(h!0 \| h?x)$ has a write $x{:=}0$

TECHNICALITIES

- The operational definition of fairness involves *interactive computations*, that allow for state change by the *environment* between actions by the *process*
- The *fair interactive computations* of P can be defined compositionally
- The *fair interference-free* computations of P can't be defined compositionally...
- ...but form a natural subset of the fair interactive computations
- Write $P \xrightarrow{\alpha} \cdot \; fair$ when there is a fair interactive computation of P in which P performs action sequence α
  NOTE: we elide the state!

DENOTATIONAL SEMANTICS

- Define trace sets

$$\mathcal{T}(P) \subseteq \Lambda^\infty$$

with

$$\mathcal{T}(e) \subseteq \Lambda^* \times Z$$
$$\mathcal{T}(b) \subseteq \Lambda^* \times \{\mathbf{true}, \mathbf{false}\}$$
$$\mathcal{T}(G) \subseteq \Lambda^\infty$$

by structural induction

- Based on same operational model

- Just traces for *fair interactive computations*

- State is implicit!

SEMANTIC DEFINITIONS

$$\mathcal{T}(\mathbf{skip}) = \{\delta\}$$

$$\mathcal{T}(x{:=}e) = \{\alpha\, x{:=}v \mid (\alpha, v) \in \mathcal{T}(e)\}$$

$$\mathcal{T}(h?x) = \{h?v\, x{:=}v \mid v \in Z\} \cup \{(\delta_{h?})^\omega\}$$

$$\mathcal{T}(h!e) = \{\alpha\, h!v,\ \alpha(\delta_{h!})^\omega \mid (\alpha, v) \in \mathcal{T}(e)\}$$

$$\mathcal{T}(P_1; P_2) = \{\alpha_1\alpha_2 \mid \alpha_1 \in \mathcal{T}(P_1)\ \&\ \alpha_2 \in \mathcal{T}(P_2)\}$$

$$\begin{aligned}\mathcal{T}(P_1 \| P_2) = \{\alpha \in \alpha_1 \| \alpha_2 \mid\ & \\ & \alpha_1 \in \mathcal{T}(P_1)\ \&\ \alpha_2 \in \mathcal{T}(P_2)\ \& \\ & \neg match(blocks\, \alpha_1,\ blocks\, \alpha_2)\}\end{aligned}$$

$$\mathcal{T}(P_1 \sqcap P_2) = \mathcal{T}(P_1) \cup \mathcal{T}(P_2)$$

SEMANTIC DEFINITIONS

$$\mathcal{T}(h?x \to P) = \{h?v\, x{:=}v\, \alpha \mid \alpha \in \mathcal{T}(P)\}$$

$$\begin{aligned}\mathcal{T}(G_1 \Box G_2) = \\ &\{\alpha \in \mathcal{T}(G_1) \cup \mathcal{T}(G_2) \mid \alpha \notin \Delta^\omega\} \\ \cup\ &\{(\delta_{X\cup Y})^\omega \mid (\delta_X)^\omega \in \mathcal{T}(G_1)\ \&\ (\delta_Y)^\omega \in \mathcal{T}(G_2)\}\end{aligned}$$

$$\mathcal{T}(\textbf{if } G \textbf{ fi}) = \mathcal{T}(G)$$

$$\mathcal{T}(\textbf{do } G \textbf{ od}) = (\mathcal{T}(G))^\omega$$

$$\begin{aligned}\mathcal{T}(\textbf{if } b \textbf{ then } P_1 \textbf{ else } P_2) = \\ &\mathcal{T}(b)_{\textbf{true}}\, \mathcal{T}(P_1)\ \cup\ \mathcal{T}(b)_{\textbf{false}}\, \mathcal{T}(P_2)\end{aligned}$$

$$\begin{aligned}\mathcal{T}(\textbf{while } b \textbf{ do } P) = \\ &(\mathcal{T}(b)_{\textbf{true}}\, \mathcal{T}(P))^*\, \mathcal{T}(b)_{\textbf{false}}\ \cup\ (\mathcal{T}(b)_{\textbf{true}}\, \mathcal{T}(P))^\omega\end{aligned}$$

SEMANTIC DEFINITIONS

$$\mathcal{T}(\textbf{local } h \textbf{ in } P) = \{\alpha \backslash h \mid \alpha \in \mathcal{T}(P) \;\&\; h \notin chans(\alpha)\}$$

$$\mathcal{T}(\textbf{local } x = e \textbf{ in } P) = \{\alpha \backslash x \mid \alpha \in \mathcal{T}(P)_{x=v} \;\&\; (\alpha, v) \in \mathcal{T}(e)\}$$

DETAILS

- $\delta\lambda = \lambda = \lambda\delta$
- $(\delta_{h?})h?v = h?v$
- $d \in \mathit{blocks}(\alpha)$ iff $\alpha\lceil d$ has suffix $(\delta_d)^\omega$
- $\alpha \| \beta$ is set of fairmerges, defined coinductively, as a *greatest fixed point*

  – interleave, allow synchronization
- **local** h **in** P "forces" synchronization on h

  ⋆ $\alpha \backslash h$ when α has no $h?v, h!v$ actions

  ⋆ replaces δ_X with $\delta_{X-\{h?,h!\}}$
- **local** $x{=}e$ **in** P "localizes" x

  ⋆ $\alpha \in \mathcal{T}(P)_{x=v}$ iff $\alpha \in \mathcal{T}(P)$ and no external interference on x

EXAMPLES

- $\mathcal{T}(h!0) = \{h!0\} \ \cup \ \{(\delta_{h!})^\omega\}$
- $\mathcal{T}(h?x) = \{h?v\, x{:=}v \mid v \in Z\} \ \cup \ \{(\delta_{h?})^\omega\}$
- $\mathcal{T}(h!0 \| h?x)$ given by

  $\{\alpha \in \alpha_1 \| \alpha_2 \mid$
  $\alpha_1 \in \mathcal{T}(h!0)$ & $\alpha_2 \in \mathcal{T}(h?x)$ &
  $\neg match(blocks\, \alpha_1,\ blocks\, \alpha_2)\}$

- $h!0 \| h?0 = \{h!0\, h?0,\ h?0\, h!0,\ \delta\}$
- $\mathcal{T}(\mathbf{local}\ h\ \mathbf{in}\ (h!0 \| h?x)) = \{x{:=}0\}$

EXAMPLE

- Let $buff_1(in, out)$ be

  local x **in**
  while true do $(in?x; out!x)$

- Behaves like a 1-place buffer

  $\mathcal{T}(buff_1(in, out)) = copy^* wait \cup copy^\omega$

  where

  $copy = (\delta_{in?})^* \{in?v\ out!v \mid v \in Z\}$
  $wait = \{(\delta_{in})^\omega\}$

- Greatest (and unique!) solution to the recursive process definition

  $buff_1 = \mathbf{local}\, x\, \mathbf{in}\, (in?x; out!x; buff_1)$

EXAMPLE

- Let $buff_2(in, out)$ be

  local mid **in**

  $buff_1(in, mid) \| buff_1(mid, out)$

- Behaves like a 2-place buffer

$$\mathcal{T}(buff_2(in, out)) = B_2(\epsilon)$$

$$\begin{aligned} B_2(\epsilon) &= (\delta_{in?})^{\infty}\,\{in?v\,\beta \mid \beta \in B_2(v),\ v \in Z\} \\ B_2(v) &= (\delta_{in?})^{*}\,out!v\,B_2(\epsilon) \\ &\cup\ (\delta_{in?})^{*}\{in?v'\,\gamma \mid \gamma \in B_2(vv'), v' \in Z\} \\ B_2(vv') &= out!v\,B_2(v') \end{aligned}$$

- For every prefix β of a trace in $\mathcal{T}(buff_2(in, out))$

  (i) $\beta\lceil out!$ is a prefix of $\beta\lceil in?$

  (ii) $0 \leq \mathtt{len}(\beta\lceil in?) - \mathtt{len}(\beta\lceil out!) \leq 2$

SEMANTIC PROPERTIES

Standard CSP laws

$$P_1 \| P_2 = P_2 \| P_1$$
$$P_1 \| (P_2 \| P_3) = (P_1 \| P_2) \| P_3$$
$$P \| \mathbf{skip} = P$$

$$P \sqcap P = P$$
$$P_1 \sqcap P_2 = P_2 \sqcap P_1$$
$$P_1 \sqcap (P_2 \sqcap P_3) = (P_1 \sqcap P_2) \sqcap P_3$$

$$G \Box G = G$$
$$G_1 \Box G_2 = G_2 \Box G_1$$
$$G_1 \Box (G_2 \Box G_3) = (G_1 \Box G_2) \Box G_3$$

$$(h?x \to P_1) \Box (h?x \to P_2) = h?x \to (P_1 \sqcap P_2)$$

Handshake law

$$\mathbf{local}\ h\ \mathbf{in}\ (h!e \| h?x) = x{:=}e$$

SEMANTIC PROPERTIES

Scope motion

$$\textbf{local } h \textbf{ in } (P_1 \| P_2) = P_1 \| (\textbf{local } h \textbf{ in } P_2)$$
$$\text{if } h \notin chans(P_1)$$

$$\textbf{local } h \textbf{ in } (P_1; P_2) = P_1; (\textbf{local } h \textbf{ in } P_2)$$
$$\text{if } h \notin chans(P_1)$$

$$\textbf{local } h \textbf{ in } (P_1; P_2) = (\textbf{local } h \textbf{ in } P_1); P_2$$
$$\text{if } h \notin chans(P_2)$$

SEMANTIC PROPERTIES

Nesting

$$\textbf{local } h_1 \textbf{ in local } h_2 \textbf{ in } P$$
$$= \textbf{local } h_2 \textbf{ in local } h_1 \textbf{ in } P$$

$$\textbf{local } h \textbf{ in local } h \textbf{ in } P$$
$$= \textbf{local } h \textbf{ in } P$$

Useful abbreviation:

$$\mathcal{T}(\textbf{local } h_1, h_2 \textbf{ in } P) =$$
$$\{\alpha \backslash \{h_1, h_2\} \mid \alpha \in \mathcal{T}(P) \ \& \ h_1, h_2 \notin chans(\alpha)\}$$

FAIR LAWS

Inevitable Synchronization

$$\begin{aligned}&\textbf{local } h \textbf{ in } [P \| (h!0; Q) \| (h?x; R)] \\ &\quad = \textbf{ local } h \textbf{ in } [P \| (x{:=}0; (Q \| R))] \\ &\qquad \text{if } h \notin \mathtt{chans}(P)\end{aligned}$$

Non-local Promotion

$$\begin{aligned}&\textbf{local } h \textbf{ in } [(h?x; P) \| (Q_1; Q_2)] \\ &\quad = Q_1; \textbf{local } h \textbf{ in } [(h?x; P) \| Q_2] \\ &\qquad \text{if } h \notin \mathtt{chans}(Q_1)\end{aligned}$$

$$\begin{aligned}&\textbf{local } h \textbf{ in } [(h!0; P) \| (Q_1; Q_2)] \\ &\quad = Q_1; \textbf{local } h \textbf{ in } [(h!0; P) \| Q_2] \\ &\qquad \text{if } h \notin \mathtt{chans}(Q_1)\end{aligned}$$

Not valid in unfair semantics

COMPOSITIONALITY

- Semantic equivalence is *compositional*
- For all P_1, P_2, if $\mathcal{T}(P_1) = \mathcal{T}(P_2)$ then for all program contexts $C[-]$, $\mathcal{T}(C[P_1]) = \mathcal{T}(C[P_2])$.
- A trivial consequence of the way we defined the semantics!

SEMANTIC PROPERTIES

Fixed point laws

while b **do** P
$= \textbf{if } b \textbf{ then } (P;\ \textbf{while } b \textbf{ do } P) \textbf{ else skip}$

do G **od**
$= \textbf{if } G \textbf{ fi};\ \textbf{do } G \textbf{ od}$

Recursive process definitions

. . . behave like greatest fixed points

$$(\nu p.F(p)) = F(\nu p.F(p))$$
$$Q = F(Q) \ \Rightarrow\ Q \subseteq \nu p.F(p)$$

RESULTS

- **Denotational matches operational**

$$\mathcal{T}(P) = \{\alpha \mid P \xrightarrow{\alpha} \cdot \mathit{fair}\}$$

PROOF

Structural induction

RESULTS

- **Traces are sensitive to deadlock**

$$(a?x \to P_1) \Box (b?y \to P_2)$$

has $(\delta_{\{a?,b?\}})^\omega$

$$(a?x \to P_1) \sqcap (b?y \to P_2)$$

has $(\delta_{a?})^\omega$ and $(\delta_{b?})^\omega$

RESULTS

- **Full abstraction**

  $\mathcal{T}(P) = \mathcal{T}(Q) \Leftrightarrow \forall C.\mathcal{B}(C[P]) = \mathcal{B}(C[Q])$

  where $\mathcal{B}$ observes communications, state changes, and deadlock

- A *program* is a process executed without interference
- $\mathcal{B}(P) =_{def} \{\alpha \in \mathcal{T}(P) \mid \alpha \textit{ interference-free}\}$

SUMMARY

- Trace semantics given *denotationally*, so supports *compositional* reasoning
- Validates natural laws based on fairness, allowing *calculational* reasoning about safety and liveness
- Recursion and fixed-point reasoning

AN EXAMPLE

- Dining Cryptographers
- Trace semantics used to establish that a protocol ensures *anonymity*
- Expressed as equational properties

$$P[Q_1] = P[Q_2]$$

RELATED WORK

- **Traditional CSP models**
  - used finite traces and prefix-closure
  - cannot model fairness
  - treat divergence as catastrophic

- **Traces subsume (stable) failures**

$$(\alpha, R) \in \mathcal{F}(P) \;\Leftrightarrow\; \alpha(\delta_X)^\omega \in \mathcal{T}(P)$$

  for some X such that $\neg match(X, R)$

- **Older's models**
  - traces + book-keeping
  - different fairness notions
  - introduced *fairness mod X*
  - α is *fair mod X* iff $blocks(\alpha) \subseteq X$

PARTIAL-ORDER SEMANTICS

- Process denotes set $\mathcal{P}(P)$ of pomsets
- A pomset $(T, <)$ consists of:
  - a multiset T of actions
  - a partial (pre-)order $<$ on T
  - *synchronization* when $h?v <> h!v$
- A pomset determines a trace set $\mathcal{L}(T, <)$
  - linearizations consistent with $<$
- $T_1 \| T_2$ is disjoint union
  - *fair* only if there are no concurrent matching blocks
- $T \preceq_h T'$ if T' arises by choosing a *synchronizing schedule* for T on channel h

SEMANTIC DEFINITIONS

$$\mathcal{T}(\textbf{skip}) = \{\delta\}$$

$$\mathcal{P}(h?x) = \{\{h?v\, x{:=}v\} \mid v \in Z\} \cup \{\{{\delta_{h?}}^{\omega}\}\}$$

$$\mathcal{P}(h!e) = \{T;\{h!v\}, T;\{(\delta_{h!})^{\omega}\} \mid (T, v) \in \mathcal{P}(e)\}$$

$$\mathcal{P}(P_1 \sqcap P_2) = \mathcal{P}(P_1) \cup \mathcal{P}(P_2)$$

$$\mathcal{P}(P_1; P_2) = \{T_1; T_2 \mid T_1 \in \mathcal{P}(P_1)\ \&\ T_2 \in \mathcal{P}(P_2)\}$$

$$\begin{aligned}&\mathcal{P}(G_1 \Box G_2) = \\ &\quad \{T \in \mathcal{P}(G_1) \cup \mathcal{P}(G_2) \mid T \cap \Delta^{\omega} = \{\}\} \cup \\ &\quad \{\{(\delta_{X \cup Y})^{\omega}\} \mid \{(\delta_X)^{\omega}\} \in \mathcal{P}(G_1), \{(\delta_Y)^{\omega}\} \in \mathcal{P}(G_2)\}\end{aligned}$$

$$\begin{aligned}&\mathcal{P}(P_1 \| P_2) = \\ &\quad \{T_1 \| T_2 \mid T_1 \in \mathcal{P}(P_1)\ \&\ T_2 \in \mathcal{P}(P_2)\ \& \\ &\qquad\qquad \neg match(blocks\, T_1,\ \ blocks\, T_2)\}\end{aligned}$$

$$\begin{aligned}&\mathcal{P}(\textbf{local}\ h\ \textbf{in}\ P) = \\ &\qquad\qquad \{T' \backslash h \mid T \in \mathcal{P}(P)\ \&\ T \preceq_h T'\}\end{aligned}$$

RESULTS

- Recovering traces:

$$\mathcal{T}(P) = \bigcup \{ \mathcal{L}(T, <) \mid (T, <) \in \mathcal{P}(P) \}$$

- Transfer Principle:

$$\mathcal{P}(P_1) = \mathcal{P}(P_2) \;\Rightarrow\; \mathcal{T}(P_1) = \mathcal{T}(P_2)$$

TRUE CONCURRENCY

The processes

$$a?x \| b?y$$
$$(a?x \rightarrow b?y) \Box (b?y \rightarrow a?x)$$
$$(a?x \rightarrow b?y) \sqcap (b?y \rightarrow a?x)$$

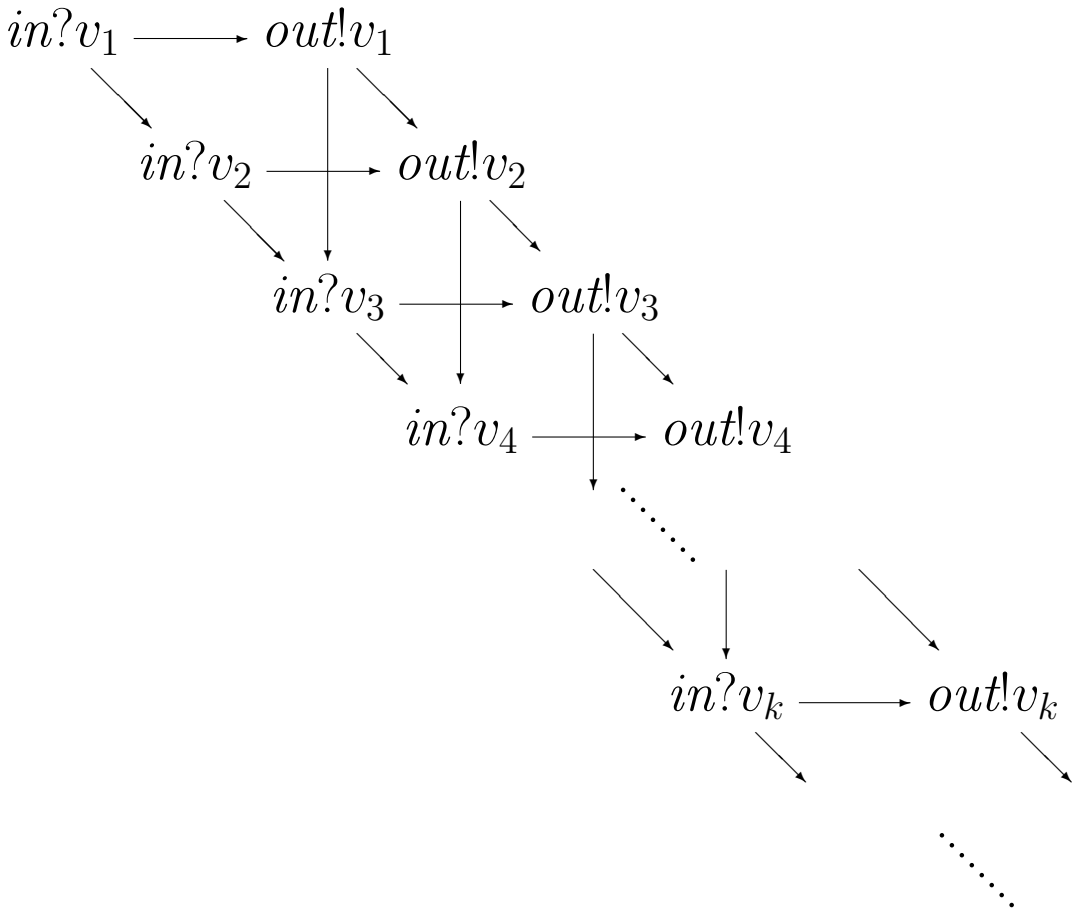
are not trace- or pomset-equivalent

EXAMPLE

local mid **in**

$$buff_1(in, mid) \| buff_1(mid, out)$$

Typical unblocked pomset:



. . . behaves like a 2-place buffer

ADAPTABILITY

Can handle other parallel paradigms by making minor changes

- Choose appropriate set of actions Λ
- Adjust relevant semantic definitions
  - parallel composition
  - input/output
  - local channels

In each case:

- Processes denote trace sets
- Full abstraction for safety and liveness
- Can give partial-order semantics

ASYNCHRONOUS COMMUNICATION

$$\lambda ::= x{=}v \mid x{:=}v \mid h?v \mid h!v \mid \delta_X$$

$$\text{where } X \subseteq \{h? \mid h \in \mathbf{Chan}\}$$

$$\mathcal{T}(h!e) = \{\alpha\, h!v \mid (\alpha, v) \in \mathcal{T}(e)\}$$

$$\mathcal{T}(P_1 \| P_2) = \{\alpha \in \alpha_1 \| \alpha_2 \mid \alpha_1 \in \mathcal{T}(P_1) \;\&\; \alpha_2 \in \mathcal{T}(P_2)\}$$

$$\mathcal{T}(\mathbf{local}\ h\ \mathbf{in}\ P) = \{\alpha \backslash h \mid \alpha \in \mathcal{T}(P) \;\&\; \alpha \lceil h \text{ is FIFO}\}$$

- $\alpha \| \beta$ interleaves, without synchronization
- $\alpha \lceil h$ is FIFO if every input is *justified* by earlier output

ASYNCHRONOUS LAWS

Inevitable Synchronization

$$\begin{aligned}&\textbf{local } h \textbf{ in } P \| (h!0; Q) \| (h?x; R)\\ &\quad = \textbf{ local } h \textbf{ in } P \| Q \| (x{:=}0; R)\\ &\qquad \text{if } h \notin \texttt{chans}(P)\end{aligned}$$

Non-local Promotion

$$\begin{aligned}&\textbf{local } h \textbf{ in } (h?x; P) \| (Q_1; Q_2)\\ &\quad = Q_1; \textbf{local } h \textbf{ in } (h?x; P) \| Q_2\\ &\qquad \text{if } h \notin \texttt{chans}(Q_1)\end{aligned}$$

Not valid in unfair semantics

SHARED MEMORY

$$\lambda ::= x{=}v \mid x{:=}v \mid \langle\alpha\rangle$$
(α finite, sequential)

$$\mathcal{T}(P_1 \| P_2) = \{\alpha \in \alpha_1 \| \alpha_2 \mid \alpha_1 \in \mathcal{T}(P_1) \;\&\; \alpha_2 \in \mathcal{T}(P_2)\}$$

$$\mathcal{T}(\textbf{local } x \textbf{ in } P) = \{\alpha \backslash x \mid \alpha \in \mathcal{T}(P) \;\&\; \alpha \lceil x \text{ sequential}\}$$

- $\alpha \lceil x$ sequential iff every read of x is *justified* by the previous write

COMMON THEME

- Programs denote sets of traces
- Fully abstract for safety and liveness
- Can extract traditional semantics
- Trace sets form complete lattice
- Program constructs denote *monotone functions* on trace sets

$$T_1 \subseteq T_2 \;\Rightarrow\; F(T_1) \subseteq F(T_2)$$

- Recursive constructs denote fixed points
  - least fixed point = finite traces
  - greatest fixed point = finite + infinite traces

UNIFICATION

Action traces can be used to model

- shared-memory
- asynchronous communication
- synchronous communication

Can extract traditional semantics

- transition traces
- failures

FUTURE RESEARCH

- **Low-level traces**
  - pointers, stores, heaps
  - Concurrent Separation Logic
- **Footstep traces**
  - abstract away from granularity
- **Probabilistic traces**
  - measurable trace sets
  - "fairly likely" correctness
- **Procedures**
  - possible worlds, parametricity
- **Other fairness notions**
  - strong, weak / process, channel