Trace semantics: towards a unification of parallel paradigms

Stephen Brookes

Department of Computer Science
Carnegie Mellon University

MFCSIT 2002

PARALLEL PARADIGMS

State-based

- **Shared-memory**
  - global state
  - concurrent read and write

- **Concurrent constraint**
  - global state
  - concurrent ask and tell

Focus on state change

PARALLEL PARADIGMS

Communication-based

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

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

Focus on communications

SEMANTIC MODELS

- **State-based**
  - sequences of state changes
    $$(s_0, s_0')(s_1, s_1') \dots (s_n, s_n') \dots$$
  - "transition traces"

- **Communication-based**
  - communication traces + book-keeping
    $$(\lambda_1 \lambda_2 \dots \lambda_n \dots, X)$$
  - "failures"

PROGRAM BEHAVIOR

- **Partial correctness**

$$\{pre\}\ P\ \{post\}$$

- **Total correctness**

$$[pre]\ P\ [post]$$

- **Safety properties**

$$pre \;\Rightarrow\; \Box \neg bad$$

- **Liveness properties**

$$pre \;\Rightarrow\; \Diamond\, good$$

Need to assume fair execution

FAIRNESS

For shared-memory or asynchronous i/o

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

- Reasonable to assume that

no process is ignored

- Weak (process) fairness
- Ensures that

  $$\mathit{stop}{:=}\mathbf{true} \,\|\, \mathbf{while}\ \neg \mathit{stop}\ \mathbf{do}\ \mathit{go}$$

  always terminates
- Satisfied by round-robin scheduler
- Can be modelled using transition traces

FAIRNESS

For synchronous i/o

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

- Reasonable to assume that
  - **no process is ignored**
  - **no synchronization is ignored**
- Weak (synchronizing) fairness

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

- Satisfied by variant of round-robin
- Not modelled by failures…

PROBLEMS

- Different models for different paradigms
  - no cross-platform analysis
  - hides underlying similarities
  - replication of effort
- Lack of uniformity
  - some models fair, some not
  - some models use state, some don't
- Lack of robustness
  - Failures aren't fair
  - Communication traces ignore state

Need a unifying framework

THIS TALK

Action traces

- **A fair semantics for CSP**
  - synchronous communication
  - avoids complex book-keeping
  - *state* handled implicitly
  - generalization of failures
  - fully abstract
- **Adaptability**
  - asynchronous communication
  - shared memory
- **A unifying framework**
  - state-based
  - communication-based

CSP

- **Processes**

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

- **Guarded commands**

$$G \ ::= \ (g \rightarrow P) \mid G_1 \Box G_2$$

- **Guards**

$$g \ ::= \ b \mid b \wedge h?x \mid b \wedge h!e$$

ACTIONS

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

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

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
- **Dir** is the set of directions

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

- Δ is the set of waiting actions

$$\Delta = \{\delta_X \mid X \subseteq_{\text{fin}} \mathbf{Dir}\}$$

- δ abbreviates $\delta_{\{\}}$
- δ_λ abbreviates $\delta_{\{\lambda\}}$
- $match(\lambda_1, \lambda_2)$ iff $\{\lambda_1, \lambda_2\} = \{h?v, h!v\}$

ENABLING

$$s \xrightarrow{x=v} s \quad \text{iff} \quad s(x) = v$$

$$s \xrightarrow{x:=v} s' \quad \text{iff} \quad s' = [s \mid x : v]$$

$$s \xrightarrow{h!v} s' \quad \text{iff} \quad s(h) = \epsilon \;\&\; s' = [s \mid h : v]$$

$$s \xrightarrow{h?v} s' \quad \text{iff} \quad s(h) = v \;\&\; s' = [s \mid h : \epsilon]$$

$$s \xrightarrow{\delta_X} s \quad \text{iff} \quad \begin{array}{l} \forall h? \in X.\ s(h) = \epsilon \;\& \\ \forall h! \in X.\ s(h) \neq \epsilon \end{array}$$

OPERATIONAL SEMANTICS

State is implicit

- **Transitions**

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

$$G \xrightarrow{\lambda} G'$$

- **Termination**

$$P \ \mathbf{term}$$

- **Fair execution**

$$P \xrightarrow{\alpha}$$

TRANSITION RULES FOR GUARDED COMMANDS

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

$$\frac{}{(h?x \longrightarrow P) \xrightarrow{\delta_{h?}} (h?x \longrightarrow P)}$$

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

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

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

TRANSITION RULES FOR PROCESSES

$$\frac{P_1 \xrightarrow{\lambda} P_1'}{P_1 \| P_2 \xrightarrow{\lambda} P_1' \| P_2} \qquad \frac{P_2 \xrightarrow{\lambda} P_2'}{P_1 \| P_2 \xrightarrow{\lambda} P_1 \| P_2'}$$

$$\frac{P_1 \xrightarrow{\lambda_1} P_1' \quad P_2 \xrightarrow{\lambda_2} P_2'}{P_1 \| P_2 \xrightarrow{\delta} P_1' \| P_2'}$$

$$\text{if } \mathit{match}(\lambda_1, \lambda_2)$$

TERMINATION

$$\frac{}{\textbf{skip term}} \qquad \frac{P_1 \textbf{ term} \quad P_2 \textbf{ term}}{P_1 \| P_2 \textbf{ term}}$$

FAIR EXECUTIONS

Parallel composition

$P \| Q \xrightarrow{\gamma}$ iff

$$P \xrightarrow{\alpha} \ \& \ Q \xrightarrow{\beta} \ \&$$
$$\gamma \in merges(\alpha, \beta) \ \&$$
$$\neg match(blocks(\alpha), blocks(\beta))$$

- $merges(\alpha, \beta)$ allows synchronization
- $blocks(\alpha)$ is set of directions occurring infinitely often in Δ steps of α

Local channels

local h **in** $P \xrightarrow{\alpha}$ iff $P \xrightarrow{\alpha} \ \& \ h \notin chans(\alpha)$

- forces synchronization on h

DENOTATIONAL SEMANTICS

- Define trace sets

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

with

$$\begin{aligned}\mathcal{T}(e) &\subseteq \Lambda^{*} \times V_{int}\\ \mathcal{T}(g) &\subseteq \Lambda^{*} \times V_{bool}\\ \mathcal{T}(G) &\subseteq \Lambda^{\infty}\end{aligned}$$

by structural induction

- Designed to match operational semantics

- $\mathcal{T}(P)$ only includes fair traces

SEMANTIC DEFINITIONS

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

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

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

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

$$\begin{aligned}&\mathcal{T}(\mathbf{local}\ h\ \mathbf{in}\ P) = \\ &\qquad \{\alpha \backslash h \mid \alpha \in \mathcal{T}(P)\ \&\ h \notin \mathit{chans}(\alpha)\}\end{aligned}$$

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

RESULTS

- **Denotational matches operational**

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

- **Traces are sensitive to deadlock**

$$\mathbf{if}\ (a?x \rightarrow P)\Box(b?y \rightarrow Q)\ \mathbf{fi}$$

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

$$\mathbf{if}\ (\mathbf{true} \rightarrow a?x; P)\Box(\mathbf{true} \rightarrow b?y; Q)\ \mathbf{fi}$$

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

- **Full abstraction**

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

where $\mathcal{B}$ observes sequence of states

SEMANTIC LAWS

synchronous

Fairness properties

$$\begin{array}{l} \textbf{local } h \textbf{ in } (h?x;P)\|(h!v;Q)\|R \\ \quad = \ \textbf{local } h \textbf{ in } (x{:=}v;(P\|Q))\|R \\ \qquad\qquad \text{if } h \notin \mathtt{chans}(R) \end{array}$$

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

$$\begin{array}{l} \textbf{local } h \textbf{ in } (h!v;P)\|(Q_1;Q_2) \\ \quad = \ Q_1;\textbf{local } h \textbf{ in } (h!v;P)\|Q_2 \\ \qquad\qquad \text{if } h \notin \mathtt{chans}(Q_1) \end{array}$$

Not valid in unfair semantics

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 if $blocks(\alpha) \subseteq X$

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

ASYNCHRONOUS COMMUNICATION

$$\lambda ::= x{=}v \mid x{:=}v \mid h?v \mid h!v \mid \delta_X$$
$$\text{where } X \subseteq \{h? \mid h \in \mathbf{Chan}\}$$

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

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

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

- $\mathit{merges}(\alpha, \beta)$ without synchronization
- $\alpha\lceil h$ is FIFO if every input is *justified* by earlier output

SEMANTIC LAWS

asynchronous

Fairness properties

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

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

Not valid in unfair semantics

SHARED MEMORY

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

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

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

$$\mathcal{T}(\textbf{await } b \textbf{ then } a) = wait^{*} go \ \cup \ wait^{\omega}$$

$$wait = \{\langle \alpha \rangle \mid (\alpha, \textbf{false}) \in \mathcal{A}(b)\}$$

$$go = \{\langle \alpha\beta \rangle \mid (\alpha, \textbf{true}) \in \mathcal{A}(b) \ \&\ \beta \in \mathcal{A}(a)\}$$

- $\alpha\lceil x$ sequential iff every (non-initial) read is *justified* by previous writes

COMMON THEME

- Programs denote sets of traces
  - built from action set Λ
- 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 = countable traces

UNIFICATION

Action traces can be used to model

- shared-memory
- asynchronous communication
- synchronous communication

Can extract traditional semantics

- transition traces
- failures

FUTURE RESEARCH

- **Other fairness notions**
  - strong, weak / process, channel
- **Partial order semantics**
  - "truly fair" concurrency
- **Low-level traces**
  - pointers, stores, heaps
- **Procedures**
  - possible worlds, parametricity
- **Intensional traces**
  - abstract runtime
- **Probabilistic traces**
  - "fairly true" correctness

REFERENCES

- *Full abstraction for a shared-variable parallel language*, S. Brookes, LICS'93
- *On the Kahn Principle and Fair Networks*, S. Brookes, MFPS 14 (1998)
- *Communicating Sequential Processes*, C. A. R. Hoare, CACM (1978)
- *A Framework for Fair Communicating Processes*, S. Older, MFPS 13 (1997)
- *On the semantics of fair parallelism*, D. Park, Springer LNCS 86 (1979)
- *The Theory and Practice of Concurrency*, A. W. Roscoe, Prentice-Hall (1998)

PARTIAL ORDER SEMANTICS

- Process denotes set $\mathcal{P}(P)$ of pomsets
- Pomset $(T, <)$
  - multiset T of actions
  - partial order $<$ on T
- A pomset determines a trace set $\mathcal{L}(T, <)$
  - traces built from T consistent with $<$
- Recovering traces:

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

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

BUFFER PROCESSES

- $\mathit{buff}_1(in, out) =_{\mathrm{def}}$

  $$\textbf{while true do } (in?x;\ out!x)$$

- $\mathit{buff}_*(in, out) =_{\mathrm{def}}$

  $$\begin{array}{l} \textbf{local } mid \textbf{ in} \\ \qquad \mathit{buff}_1(l, mid) \| \mathit{buff}_1(mid, r) \end{array}$$

- $\mathit{buff}_2(in, out) =_{\mathrm{def}}$

  $$\begin{array}{l} \textbf{local } mid,\ ack \textbf{ in} \\ \qquad \textbf{while true do } (in?x;\ mid!x;\ ack?\_) \\ \quad \| \textbf{ while true do } (mid?y;\ ack!\_;\ out!y) \end{array}$$

BUFFER BEHAVIOR

- Safety: FIFO order
- Liveness: Every input is output

CAPACITY

- buff_1
  - synchronous: 1-place
  - asynchronous: 1-place
- buff_*
  - synchronous: 2-place
  - asynchronous: unbounded
- buff_2
  - synchronous: 2-place
  - asynchronous: 2-place

Asynchronous traces of buff_1

Typical unblocked case:

$$\begin{tikzcd}
in?v_1 \arrow[r] & out!v_1 \arrow[d] & & & & \\
 & in?v_2 \arrow[r] & out!v_2 \arrow[d] & & & \\
 & & in?v_3 \arrow[r] & out!v_3 \arrow[d] & & \\
 & & & \ddots & & \\
 & & & & in?v_k \arrow[r] & out!v_k \arrow[d] \\
 & & & & & \ddots
\end{tikzcd}$$

where each $v_i \in V$

$$(in?v\ out!v \mid v \in V)^\omega$$

Asynchronous traces of buff_1

Typical blocked case:

$$
\begin{tikzcd}[column sep=small]
in?v_1 \arrow[rr] & & out!v_1 \arrow[dl] & & & & & \\
& in?v_2 \arrow[rr] & & out!v_2 \arrow[dl] & & & & \\
& & in?v_3 \arrow[rr] & & out!v_3 \arrow[dl] & & & \\
& & & \ddots & & & & \\
& & & & in?v_k \arrow[rr] & & out!v_k \arrow[dl] & \\
& & & & & \delta_{in?} \arrow[dr] & & \\
& & & & & & \delta_{in?} & \ddots
\end{tikzcd}
$$

where $k \geq 0$ and each $v_i \in V$

$$(in?v\; out?v \mid v \in V)^* \, {\delta_{in?}}^{\omega}$$

Asynchronous traces of buff_*

Typical unblocked case:

$$\begin{tikzcd}
in?v_1 \arrow[r] \arrow[dr] & out!v_1 \arrow[dr] & & & & & \\
& in?v_2 \arrow[r] \arrow[dr] & out!v_2 \arrow[dr] & & & & \\
& & in?v_3 \arrow[r] \arrow[dr] & out!v_3 \arrow[dr] & & & \\
& & & \ddots \arrow[dr] & \ddots \arrow[dr] & & \\
& & & & in?v_k \arrow[r] \arrow[dr] & out!v_k \arrow[dr] & \\
& & & & & \ddots & \ddots
\end{tikzcd}$$

where each $v_i \in V$

any trace consistent with this

Asynchronous traces of buff_*

Typical blocked case:

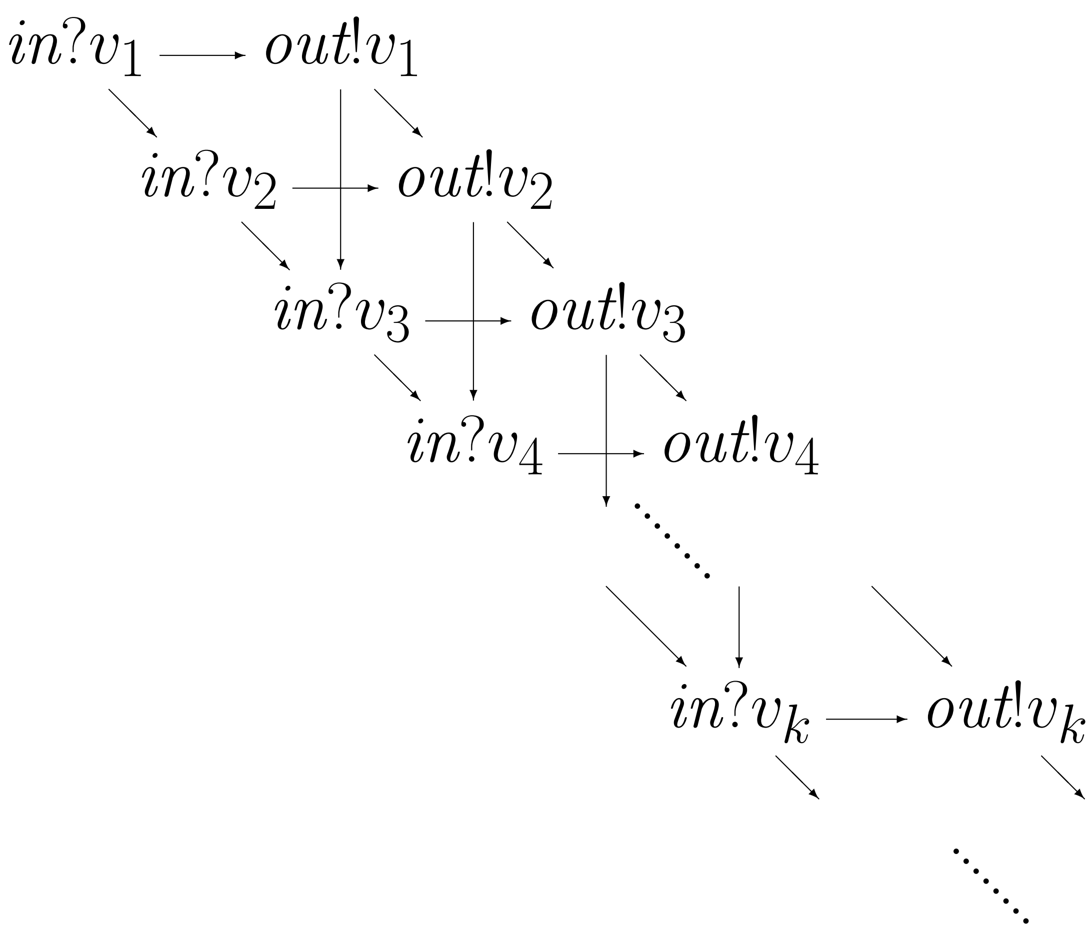
$$
\begin{tikzcd}
in?v_1 \arrow[r] \arrow[dr] & out!v_1 \arrow[dr] & & & & & \\
& in?v_2 \arrow[r] \arrow[dr] & out!v_2 \arrow[dr] & & & & \\
& & in?v_3 \arrow[r] \arrow[dr] & out!v_3 \arrow[dr] & & & \\
& & & \ddots \arrow[dr] & \ddots \arrow[dr] & & \\
& & & & in?v_k \arrow[r] \arrow[dr] & out!v_k & \\
& & & & & \delta_{in?} \arrow[dr] & \\
& & & & & & \delta_{in?} \ddots
\end{tikzcd}
$$

where $k \geq 0$ and each $v_i \in V$

any interleaving consistent with this

Asynchronous traces of $buff_2$

Typical unblocked case:

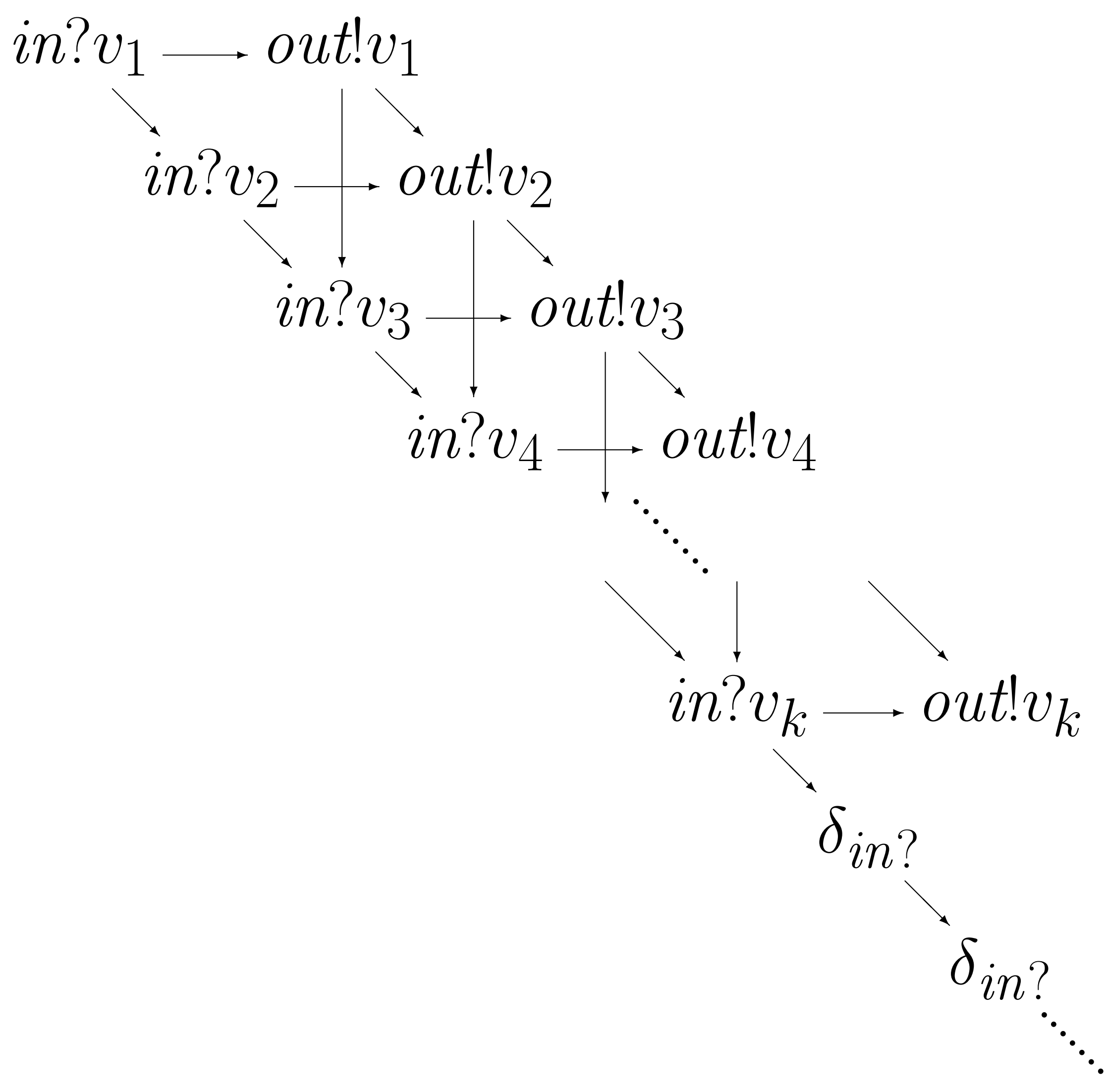


where each $v_i \in V$

any trace consistent with this

Asynchronous traces of buff_2

Typical blocked case:



where $k \geq 0$ and each $v_i \in V$

Synchronous traces of buff_1

Typical unblocked case:

$$\begin{tikzcd}
in?v_1 \arrow[r] & out!v_1 \arrow[d] & & & \\
& in?v_2 \arrow[r] & out!v_2 \arrow[d] & & \\
& & in?v_3 \arrow[r] & out!v_3 \arrow[d] & \\
& & & \ddots & \\
& & & in?v_k \arrow[r] & out!v_k \arrow[d] \\
& & & & \ddots
\end{tikzcd}$$

where each $v_i \in V$

$$(in?v\ out!v \mid v \in V)^{\omega}$$

Synchronous traces of buff_1

Blocked on input:

$$\begin{tikzcd}[column sep=small]
in?v_1 \arrow[r] & out!v_1 \arrow[d] & & & & & \\
& in?v_2 \arrow[r] & out!v_2 \arrow[d] & & & & \\
& & in?v_3 \arrow[r] & out!v_3 \arrow[d] & & & \\
& & & \ddots & & & \\
& & & & in?v_k \arrow[r] & out!v_k \arrow[d] & \\
& & & & & \delta_{in?} \arrow[dr] & \\
& & & & & & \delta_{in?} \ddots
\end{tikzcd}$$

where $k \geq 0$ and each $v_i \in V$

$$(in?v\; out?v \mid v \in V)^* \, {\delta_{in?}}^{\omega}$$

Synchronous traces of buff_1

Blocked on output:

$$
\begin{tikzcd}
in?v_1 \arrow[r] & out!v_1 \arrow[d] & & & & & \\
& in?v_2 \arrow[r] & out!v_2 \arrow[d] & & & & \\
& & in?v_3 \arrow[r] & out!v_3 \arrow[d] & & & \\
& & & \ddots \arrow[dr, dotted, no head] & & & \\
& & & & in?v_k \arrow[r] & \delta_{out!} \arrow[dr] & \\
& & & & & & \delta_{out!} \arrow[d] \\
& & & & & & \delta_{out!} \\
& & & & & & \vdots
\end{tikzcd}
$$

where $k \geq 0$ and each $v_i \in V$

$$(in?v\ out?v \mid v \in V)^*\ (in?v \mid v \in V)\delta_{out!}{}^{\omega}$$

Synchronous traces of buff_*

Typical unblocked case:

$$
\begin{tikzcd}
in?v_1 \arrow[rr] \arrow[dr] & & out!v_1 \arrow[dd] \arrow[dr] & & & & & & \\
& in?v_2 \arrow[rr] \arrow[dr] & & out!v_2 \arrow[dd] \arrow[dr] & & & & & \\
& & in?v_3 \arrow[rr] \arrow[dr] & & out!v_3 \arrow[dd] \arrow[dr] & & & & \\
& & & in?v_4 \arrow[rr] & & out!v_4 & & & \\
& & & & & \ddots & & & \\
& & & & \arrow[r] & in?v_k \arrow[rr] \arrow[dr] & & out!v_k \arrow[dr] & \\
& & & & & & & \ddots &
\end{tikzcd}
$$

where each $v_i \in V$

behaves like a 2-place buffer