CONCURRENT OBJECTS IN IDEALIZED CSP

Stephen Brookes

Department of Computer Science
Carnegie Mellon University

July 1998

IDEALIZED CSP

communicating processes

$$+$$

call-by-name λ-calculus

- **simply typed**

$$\begin{array}{rcl} \theta & ::= & \mathbf{var}[\tau] \;\mid\; \mathbf{chan}[\tau] \\ & \mid & \mathbf{exp}[\tau] \;\mid\; \mathbf{comm} \\ & \mid & \theta \rightarrow \theta' \;\mid\; \theta \times \theta' \end{array}$$

$$\tau \;::=\; \mathbf{int} \;\mid\; \mathbf{bool} \;\mid\; \mathbf{unit}$$

- **asynchronous communication**

  channels as unbounded buffers

- **fair parallel execution**

  abstracts from network details

CONNECTIONS

- **generalizes CSP**
  - fairness
  - nested parallelism
  - dynamic process creation
  - asynchronous communication
- **generalizes Idealized Algol**
  - typed channels
  - communicating processes
- **generalizes Kahn networks**
  - non-determinism and fairness
- **supports concurrent objects**
  - parallel methods
  - shared or private state

SYNTAX

- **Input**

$$\frac{\pi \vdash h : \mathbf{chan}[\tau] \quad \pi \vdash X : \mathbf{var}[\tau]}{\pi \vdash h?X : \mathbf{comm}}$$

- **Output**

$$\frac{\pi \vdash h : \mathbf{chan}[\tau] \quad \pi \vdash E : \mathbf{exp}[\tau]}{\pi \vdash h!E : \mathbf{comm}}$$

- **Parallel composition**

$$\frac{\pi \vdash P_1 : \mathbf{comm} \quad \pi \vdash P_2 : \mathbf{comm}}{\pi \vdash P_1 \| P_2 : \mathbf{comm}}$$

- **Local declaration**

$$\frac{\pi \vdash D : \pi' \quad \pi, \pi' \vdash P : \mathbf{comm}}{\pi \vdash \mathbf{local}\ D\ \mathbf{in}\ P : \mathbf{comm}}$$

CATEGORY of WORLDS

Oles, Reynolds

- **Objects: countable sets of states**

$$V_1 \times \cdots \times V_k \times H_1^* \times \cdots H_n^*$$

- **Morphisms:**

$$(f, Q) : W \to X$$

  - **function** f **from** X **to** W
  - **equivalence relation** Q **on** X
  - **each** Q**-class isomorphic to** W

ADAPTATION

- **channels as components of state**
- **communication as state change**

EXPANSIONS

- **The expansion morphism**

$$- \times V : W \to W \times V$$

is given by

$$- \times V = (\mathbf{fst} : W \times V \to W,\ Q)$$
$$(w_0, v_0) Q (w_1, v_1) \iff v_0 = v_1$$

- **Used to model local variables and local channels**

- **Every morphism is an expansion, modulo isomorphism**

SEMANTICS

- **Types denote functors from worlds to domains, $[\![\theta]\!] : \mathbf{W} \to \mathbf{D}$**
- **Judgements $\pi \vdash P : \theta$ denote natural transformations**

$$[\![P]\!] : [\![\pi]\!] \stackrel{\cdot}{\to} [\![\theta]\!]$$

i.e. when $h : W \to X$,

$$\begin{tikzcd}
[\![\pi]\!]W \arrow[r, "{[\![P]\!]W}"] \arrow[d, "{[\![\pi]\!]h}"'] & [\![\theta]\!]W \arrow[d, "{[\![\theta]\!]h}"] \\
[\![\pi]\!]X \arrow[r, "{[\![P]\!]X}"'] & [\![\theta]\!]X
\end{tikzcd}$$

commutes.

Naturality enforces locality

COMMANDS

$$[\![\mathbf{comm}]\!]W = \wp^{\dagger}((W \times W)^{\infty})$$

- **Commands denote closed trace sets**

  $\alpha\beta \in t \;\&\; w \in W \;\Rightarrow \alpha\langle w, w\rangle\beta \in t$
  $\alpha\langle w, w'\rangle\langle w', w''\rangle\beta \in t \;\Rightarrow\; \alpha\langle w, w''\rangle\beta \in t$

- **A trace** $\langle w_0, w_0'\rangle\langle w_1, w_1'\rangle \ldots \langle w_n, w_n'\rangle \ldots$ **models a fair interaction**

- **A step** $\langle w_i, w_i'\rangle$ **represents a finite sequence of atomic actions**

CHANNELS

An "object-oriented" semantics:

- **sender**

$$give : W \to (W \times V_{\mathcal{T}})\,option$$

- **receiver**

$$take : V_{\mathcal{T}} \to (W \to W)$$

satisfying

$$
\begin{array}{l}
give(take\ v\ w) \ = \\
\qquad \textbf{case}\ give\ w\ \textbf{of} \\
\qquad\quad \begin{array}{ll}
\textbf{none} : & \textbf{some}(w, v) \\
\textbf{some}(w', v') : & \textbf{some}(take\ v\ w', v')
\end{array}
\end{array}
$$

PARALLEL COMPOSITION

Fair merge of traces

$$
\begin{aligned}
&[\![P_1 \| P_2]\!]W u = \\
&\qquad \{\alpha \mid \exists \alpha_1 \in [\![P_1]\!]W u,\ \ \alpha_2 \in [\![P_2]\!]W u. \\
&\qquad\qquad (\alpha_1, \alpha_2, \alpha) \in \mathit{fairmerge}_{W \times W}\}^\dagger
\end{aligned}
$$

where

$$
\begin{aligned}
&\mathit{fairmerge}_A = \mathit{both}_A^* \cdot \mathit{one}_A \cup \mathit{both}_A^\omega \\
&\mathit{both}_A = \{(\alpha, \beta, \alpha\beta), (\alpha, \beta, \beta\alpha) \mid \alpha, \beta \in A^+\} \\
&\mathit{one}_A = \{(\alpha, \epsilon, \alpha), (\epsilon, \alpha, \alpha) \mid \alpha \in A^\infty\}
\end{aligned}
$$

fairmerge is natural

LOCAL CHANNELS

The traces of

$$\mathbf{local}\ h : \mathbf{chan}[\tau]\ \mathbf{in}\ P$$

at W **are projected from the traces of** P **at** $W \times V_\tau^*$ **in which**

- **initially** $h = \epsilon$
- **contents of** h **never change across step boundaries**

EXAMPLES

- **local** h **in** $(h!0; P) = P$
  if h not free in P

- **local** h **in** $(h?x; P) =$ **while true do skip**

LAWS

- **Symmetry**

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

- **Scope contraction**

$$\textbf{local } h \textbf{ in } (P_1 \| P_2)$$
$$= (\textbf{local } h \textbf{ in } P_1) \| P_2$$

if h not free in P_2

... justifies graphical notation for networks of processes

LOCAL LAWS

- **Local output**

$$\textbf{local } h = \rho \textbf{ in } P_1 \| (h!v; P_2) = \textbf{local } h = \rho v \textbf{ in } P_1 \| P_2$$

if $h!$ not free in P_1

- **Local input**

$$\textbf{local } h = v\rho \textbf{ in } P_1 \| (h?x; P_2) = \textbf{local } h = \rho \textbf{ in } P_1 \| (x{:=}v; P_2)$$

if $h?$ not free in P_1

...help when channels are uni-directional

FAIRNESS LAWS

- **Fair prefix**

$$\begin{aligned}&\textbf{local } h \textbf{ in } (h?x; P)\|(Q_1; Q_2)\\ &\quad = Q_1; \textbf{local } h \textbf{ in } (h?x; P)\|Q_2\end{aligned}$$

if h not free in Q_1

- **Cyclic synchronization**

$$\begin{array}{ll}\textbf{local } h_1, h_2 \textbf{ in} & = (P_1\|P_2);\\ \quad (P_1;\ h_1!\star;\ h_2?\star;\ Q_1) & \quad \textbf{local } h_1, h_2\\ \ \| \ (P_2;\ h_2!\star;\ h_1?\star;\ Q_2) & \qquad \textbf{in } (Q_1\|Q_2)\end{array}$$

if h_1, h_2 not free in P_1, P_2

... *require and reflect fair semantics*

CLASSES AND OBJECTS

- **Declarations as first-class citizens**

$$\pi \vdash D : \pi'$$

- **Class is template for declaration:**

$$\begin{aligned}&\textbf{class } C = \\ &\qquad \textbf{private } \pi_1 \\ &\qquad \textbf{public } \pi_2\end{aligned}$$

- **Object instantiates template:**

$$\begin{aligned}\textbf{object } X : C \;=\; &\textbf{private } D_1 \\ &\textbf{public } D_2\end{aligned}$$

translates to

$$\textbf{local } X.D_1 \textbf{ in } X.D_2$$

BUFFER CLASSES

$$\textbf{class } \mathit{Buffer}_1 \ = \\ \quad \textbf{public} \\ \qquad \mathit{put} : \textbf{exp}[\tau] \rightarrow \textbf{comm} \\ \qquad \mathit{get} : \textbf{var}[\tau] \rightarrow \textbf{comm}$$

$$\textbf{class } \mathit{Buffer}_2 \ = \\ \quad \mathit{Buffer}_1 \textbf{ with private } \mathit{data} : \textbf{chan}[\tau]$$

$$\textbf{class } \mathit{Buffer}_3 \ = \\ \quad \mathit{Buffer}_1 \textbf{ with private } \mathit{data} : \textbf{var}[\tau]$$

SUBCLASSES

$$\mathit{Buffer}_2 \leq \mathit{Buffer}_1$$
$$\mathit{Buffer}_3 \leq \mathit{Buffer}_1$$

A BUFFER OBJECT

$$
\begin{array}{l}
\textbf{object } B_1 : \mathit{Buffer}_2 \ = \\
\quad \textbf{private} \\
\qquad \mathit{empty} : \textbf{chan}[\textbf{unit}] = [*]; \\
\qquad \mathit{data} : \textbf{chan}[\textbf{int}] \\
\quad \textbf{public} \\
\qquad \mathit{put}(e) = (\mathit{empty}?*;\ \ \mathit{data}!e); \\
\qquad \mathit{get}(z) = (\mathit{data}?z;\ \ \mathit{empty}!*)
\end{array}
$$

PROPERTIES

- B_1 has class Buffer_2
- $\mathit{Buffer}_2 \leq \mathit{Buffer}_1$
- B_1 also has class Buffer_1
- B_1 behaves like a 1-place buffer

ANOTHER BUFFER

$$
\begin{array}{l}
\textbf{object } B_2 : \mathit{Buffer}_2 \;=\\
\quad \textbf{private}\\
\qquad\quad \mathit{empty} : \textbf{chan}[\textbf{unit}] = [*];\\
\qquad\quad \mathit{data} : \textbf{chan}[\textbf{int}]\\
\quad \textbf{public}\\
\qquad\quad \mathit{put}(e) = (\mathit{empty}?*;\;\; \mathit{data}!(-e));\\
\qquad\quad \mathit{get}(z) = \;\textbf{local } x : \textbf{var}[\textbf{int}] \textbf{ in}\\
\qquad\qquad\qquad\quad (\mathit{data}?x;\;\; z{:=}(-x);\;\; \mathit{empty}!*)
\end{array}
$$

PROPERTIES

- Codes and decodes data
- Still behaves like 1-place buffer

YET ANOTHER BUFFER

$$
\begin{array}{l}
\textbf{object } B_3 : \mathit{Buffer}_3 \;=\\
\quad \textbf{private}\\
\qquad \mathit{empty} : \textbf{var}[\textbf{bool}] = \textbf{true};\\
\qquad \mathit{full} : \textbf{var}[\textbf{bool}] = \textbf{false};\\
\qquad \mathit{data} : \textbf{var}[\tau]\\
\quad \textbf{public}\\
\qquad \mathit{put}(e) \;=\\
\qquad\quad (\textbf{await } \mathit{empty} \textbf{ then } \mathit{empty}{:=}\textbf{false};\\
\qquad\quad\;\; \mathit{data}{:=}e;\\
\qquad\quad\;\; \mathit{full}{:=}\textbf{false});\\
\qquad \mathit{get}(x) \;=\\
\qquad\quad (\textbf{await } \mathit{full} \textbf{ then } \mathit{full}{:=}\textbf{true};\\
\qquad\quad\;\; x{:=}\mathit{data};\\
\qquad\quad\;\; \mathit{empty}{:=}\textbf{true})
\end{array}
$$

EQUIVALENCES

- **All three implementations of buffers are "equivalent"**
  - no way to tell them apart
- **Need to compare across paradigms**
  - communicating processes
  - shared-variable
- **Trace semantics can be used in both cases**
  - all three buffer objects have same trace semantics
  - closure blurs granularity

CONCLUSIONS

- **Idealized CSP supports a form of concurrent objects**
- **Trace semantics validates natural laws of equivalence**
  - **locality**
  - **fairness**
  - **synchronization patterns**
- **Can compare across paradigms**
- **Can abstract from granularity**

SPECIFICATIONS

spec $BUFFER$ =
  interface
    $empty,\ full : \mathbf{exp}[\mathbf{bool}]$
  with

$$\{empty\}put(v)\{full\}$$

$$\{full\}get(x)\{empty\}$$

$$put(v_1)\|put(v_2) \ = \ (put(v_1); put(v_2)) \ \mathbf{or} \ (put(v_2); put(v_1))$$

$$\{empty\}(get(x_1)\|get(x_2)) \ = \ (get(x_1); get(x_2)) \ \mathbf{or} \ (get(x_2); get(x_1))$$

$$\{empty\}(put(v)\|get(x)) \ = \ put(v); get(x) \ = \ x{:=}v$$