IDEALIZED CSP:
Procedures + Processes

Stephen Brookes
Department of Computer Science
Carnegie Mellon University

July 1997
THE ESSENCE OF CSP

- Idealized CSP =
  communicating processes
  + call-by-name $\lambda$-calculus

- simply typed

  $\theta ::= \exp[\tau] \mid \var[\tau] \mid \chan[\tau]$
  $\mid \text{comm} \mid (\theta \rightarrow \theta') \mid \theta \times \theta'$

  phrase types

  $\tau ::= \text{int} \mid \text{bool}$

  data types

cf. Reynolds: Idealized Algol

cf. Brookes: Parallel Algol
CONNECTIONS

• generalizes Hoare’s CSP
  – fairness
  – nested parallel
  – dynamic process creation
  – channel-based communication
  – asynchronous output, synchronous input

• generalizes Idealized Algol
  – typed channels
  – communicating processes
  – local channel declarations

• generalizes Kahn networks
RATIONALE

• Programs can cooperate
  – message-passing
  – access to shared memory

• Procedures can encapsulate parallel idioms
  – bounded buffer
  – communication protocols

• Local variables and channels can limit scope of interference

INTUITION

Procedures and parallelism should be orthogonal
BUFFERS

procedure buff1(in, out) =
    new[τ] x in
    while true do (in?x; out!x)

procedure buff(in, out) =
    newchan[τ] h in
    buff1(in, h) || buff1(h, out)

• Encapsulates common way to build buffers

• Relies on locality of x and h
SIEVE

procedure filter(p, in, out) =
    new[int] x in
    while true do
        (in?x; if x mod p \neq 0 then out!x);

procedure sift(in, out) =
    newchan[int] h in
    new[int] p in
    begin
        in?p; out!p;
        filter(p, in, h) || sift(h, out)
    end
SYNTAX

• Input and Output

\[ \pi \vdash h : \text{chan}[\tau] \quad \pi \vdash E : \text{exp}[\tau] \]
\[ \pi \vdash h!E : \text{comm} \]
\[ \pi \vdash h : \text{chan}[\tau] \quad \pi \vdash X : \text{var}[\tau] \]
\[ \pi \vdash h?X : \text{comm} \]

• Parallel composition

\[ \pi \vdash P_1 : \text{comm} \quad \pi \vdash P_2 : \text{comm} \]
\[ \pi \vdash P_1 \parallel P_2 : \text{comm} \]

• Local channel declaration

\[ \pi, \iota : \text{chan}[\tau] \vdash P : \text{comm} \]
\[ \pi \vdash \text{newchan}[\tau] \iota \text{ in } P : \text{comm} \]
CATEGORY of WORLDS

• Objects are countable posets of “allowed states”
  \[ V_1 \times \cdots \times V_k \times H_1^* \times \cdots H_n^* \]
  ordered by prefix, componentwise

• Morphisms \((f, Q) : W \rightarrow X\)
  – function \(f\) from \(X\) to \(W\)
  – equivalence relation \(Q\) on \(X\)
  – \(f\) puts each \(Q\)-class into order-isomorphism with \(W\)

Generalizes Oles’s category:

• channels as components of state
• morphisms respect queue structure
EXPANSIONS

• For each pair of objects $W$ and $V$ there is an expansion

$$- \times V : W \rightarrow W \times V$$

given by

$$- \times V = (\text{fst} : W \times V \rightarrow W, \ Q)$$

$$(w_0, v_0)Q(w_1, v_1) \iff v_0 = v_1$$

• Use $- \times V^*_\tau$ to model local channel declaration

• Each morphism is an expansion, up to order-isomorphism

• Some order-isomorphisms:

\[\text{swap} : W \times V \rightarrow V \times W\]
\[\text{assoc} : W \times (V \times U) \rightarrow (W \times V) \times U\]

but not $\text{rev} : V^*_\tau \rightarrow V^*_\tau$
SEMANTICS

• Types denote functors from worlds to domains, $[\theta] : W \to D$

• Judgements $\pi \vdash P : \theta$ denote natural transformations

$$[P] : [\pi] \to [\theta]$$

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

\[
\begin{array}{c}
:\[\pi\]W & \xrightarrow{[P]W} & \[\theta\]W \\
\downarrow{[\pi]h} & & \downarrow{[\theta]h} \\
:\[\pi\]X & \xrightarrow{[P]X} & \[\theta\]X
\end{array}
\]

commutes.

\textit{Naturality enforces locality}
The functor category $\mathbf{D}^W$ is cartesian closed.

Use ccc structure to interpret arrow and product types:

\[
\begin{align*}
[\theta \times \theta'] &= [\theta] \times [\theta'] \\
[\theta \rightarrow \theta'] &= [\theta] \Rightarrow [\theta']
\end{align*}
\]

Procedures are natural and therefore respect locality.

CARTESIAN CLOSURE
A procedure of type $\theta \rightarrow \theta'$ at world $W$ denotes a natural family of functions $p(-)$:

if $h : W \rightarrow X$ and $h' : X \rightarrow Y$,

\[
\begin{array}{c}
\theta]X \xrightarrow{p(h)} [\theta']X \\
\theta]h' \downarrow \quad \quad \quad \quad \downarrow [\theta']h' \\
\theta]Y \xrightarrow{p(h; h')} [\theta']Y \\
\end{array}
\]

commutes.

Procedures can be called at expanded worlds, and naturality enforces locality constraints.
COMMANDS

\[ [\text{comm}]W = \varnothing^\dagger((W \times W)^\infty) \]

... as for shared-variable programs

- commands denote trace sets
- closed under stuttering and mumbling

\[
\alpha\beta \in t \& w \in W \Rightarrow \alpha\langle w, w\rangle\beta \in t
\]
\[
\alpha\langle w, w'\rangle\langle w', w''\rangle\beta \in t \Rightarrow \alpha\langle w, w''\rangle\beta \in t
\]

... with certain modifications

- message-passing as state change
- interference thus models communication by environment
- unrequited input = busy wait
INTUITION

• 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

• Each step \( \langle w_i, w'_i \rangle \) represents a finite sequence of atomic actions

• If \((f, Q) : W \rightarrow X\) and \(c \in \llbracket \text{comm} \rrbracket W\)

\[
\llbracket \text{comm} \rrbracket (f, Q)c
\]
behaves like \(c\) on the \(W\)-component of state, has no effect elsewhere.
CHANNELS

An “object-oriented” semantics:

• output = acceptor
• input = expression with side-effect

\[
\begin{align*}
[\text{chan}[\tau]]W &= \(V_\tau \rightarrow [\text{comm}]W\) \times [\text{exp}[\tau]]W \\
[\text{exp}[\tau]]W &= \wp((W \times W)^+ \times V_\tau \cup (W \times W)^\omega)
\end{align*}
\]

cf. Reynolds, Oles
PARALLEL COMPOSITION

Fair merge of traces

\[
[P_1 \parallel P_2]Wu = \\
\{ \alpha \mid \exists \alpha_1 \in [P_1]Wu, \ \alpha_2 \in [P_2]Wu. \ \\
(\alpha_1, \alpha_2, \alpha) \in \text{fairmerge}_W \times W \}^\dagger
\]

where

\[
\text{fairmerge}_A = \text{both}_A^* \cdot \text{one}_A \cup \text{both}_A^\omega
\]

\[
\text{both}_A = \{ (\alpha, \beta, \alpha\beta), (\alpha, \beta, \beta\alpha) \mid \alpha, \beta \in A^+ \}
\]

\[
\text{one}_A = \{ (\alpha, \epsilon, \alpha), (\epsilon, \alpha, \alpha) \mid \alpha \in A^\infty \}
\]

fairmerge is natural
INPUT and OUTPUT

In world $V_{int} \times V_{int}^*$ and a suitable environment

• $h?x$ has traces
  \[
  \langle (v, n\rho), (n, \rho) \rangle \quad (v, n \in V_{int}, \rho \in V_{int}^*)
  \]
  and
  \[
  \langle (v_0, \epsilon), (v_0, \epsilon) \rangle \ldots \langle (v_k, \epsilon), (v_k, \epsilon) \rangle \ldots
  \]

• $h!(x + 1)$ has traces
  \[
  \langle (m, \sigma), (m, \sigma(m+1)) \rangle \quad (m \in V_{int}, \sigma \in V_{int}^*)
  \]

• $h?x \parallel h!(x + 1)$ has traces
  \[
  \langle (v, n\rho), (n, \rho) \rangle \langle (m, \sigma), (m, \sigma(m + 1)) \rangle
  \]
  and
  \[
  \langle (m, \sigma), (m, \sigma(m + 1)) \rangle \langle (v, n\rho), (n, \rho) \rangle
  \]
  and
  \[
  \langle (m, \epsilon), (m + 1, \epsilon) \rangle
  \]
CHOICE

An external choice

$$(a?x \rightarrow P_1) \lozenge (b?x \rightarrow P_2)$$

can

• input on $a$ and behave like $P_1$
• input on $b$ and behave like $P_2$
• busy-wait while $a$ and $b$ are both empty

However, an internal choice

$$(a?x \rightarrow P_1) \sqcap (b?x \rightarrow P_2)$$

can busy-wait if either $a$ or $b$ is empty
LOCAL CHANNELS

The traces of

\texttt{newchan}[\tau] \ h \ \texttt{in} \ P

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

- initially \( h \) is empty
- contents of \( h \) never change across step boundaries

EXAMPLES

- \texttt{newchan}[\tau] \ h \ \texttt{in} \ (h!e \parallel h?x) = x:=e

- \texttt{newchan}[\tau] \ h \ \texttt{in} \ (h!0; P) = P
  if \( h \) does not occur free in \( P \)

- \texttt{newchan}[\tau] \ h \ \texttt{in} \ (h?x; P)
  has only infinite stuttering traces, because of unrequited input
BUFFERS

In world $V_{int}^* \times V_{int}^*$ and a suitable environment

$$buff1(left, right)$$

has trace

\[ \langle (0, \epsilon), (\epsilon, \epsilon) \rangle \]  \hspace{1cm} \text{input 0}

\[ \langle (1, \epsilon), (1, 0) \rangle \]  \hspace{1cm} \text{output 0}

\[ \langle (1, 0), (\epsilon, 0) \rangle \]  \hspace{1cm} \text{input 1}

\[ \ldots \]

Similarly

$$buff(left, right)$$

has trace

\[ \langle (0, \epsilon), (\epsilon, \epsilon) \rangle \]  \hspace{1cm} \text{input 0}

\[ \langle (1, \epsilon), (\epsilon, \epsilon) \rangle \]  \hspace{1cm} \text{input 1}

\[ \langle (2, \epsilon), (2, 0) \rangle \]  \hspace{1cm} \text{output 0}

\[ \ldots \]
LAWS

• Symmetry

\[
\text{newchan}[\tau_1] \ h_1 \ \text{in} \\
\text{newchan}[\tau_2] \ h_2 \ \text{in} \ P
\]
\[
= \text{newchan}[\tau_2] \ h_2 \ \text{in} \\
\text{newchan}[\tau_1] \ h_1 \ \text{in} \ P
\]

• Frobenius

\[
\text{newchan}[\tau] \ h \ \text{in} \ (P_1\|P_2) = \\
(\text{newchan}[\tau] \ h \ \text{in} \ P_1)\|P_2
\]

provided \(h\) does not occur free in \(P_2\)
LAWS

• Local variables

\[
\text{newvar}[^\tau] \; \iota \; \text{in} \; P' = P' \\
\text{newvar}[^\tau] \; \iota \; \text{in} \; (P \parallel P') = \\
(\text{newvar}[^\tau] \; \iota \; \text{in} \; P) \parallel P'
\]

when \( \iota \) does not occur free in \( P' \)

• Functional laws

\[
(\lambda \iota : \theta. P)(Q) = P[Q/\iota] \\
\text{rec } \iota. P = P[\text{rec } \iota. P/\iota]
\]
LOCAL LAWS

• Local output

\[
\text{newchan}[\tau] \ h = \rho \ \text{in} \ P_1 \parallel h!v; P_2
\]
\[
= \ \text{newchan}[\tau] \ h = \rho v \ \text{in} \ P_1 \parallel P_2
\]

if \( h! \) not in \( P_1 \)

• Local input

\[
\text{newchan}[\tau] \ h = \nu \rho \ \text{in} \ P_1 \parallel h?v; P_2
\]
\[
= \ \text{newchan}[\tau] \ h = \rho \ \text{in} \ P_1 \parallel P_2
\]

if \( h? \) not in \( P_1 \)

...help when channels are used in
at most one direction by each process
FAIRNESS LAWS

- **Fair prefix**
  If $h$ not free in $P_1$ and
  \[
  \text{newchan}[\tau] h = \rho \text{ in } P
  \]
diverges, then
  \[
  \text{newchan}[\tau] h = \rho \text{ in } P \parallel (P_1; P_2) = P_1; \text{newchan}[\tau] h = \rho \text{ in } P \parallel P_2
  \]

- **Unrequited input**
  If $h$ not free in $P_1$ then
  \[
  \text{newchan}[\tau] h \text{ in } (h?x; P) \parallel (P_1; P_2) = P_1; \text{newchan}[\tau] h \text{ in } (h?x; P) \parallel P_2
  \]
CONCLUSIONS

• Transition traces are fundamental, general and unifying
  – shared-variable
  – communicating process

• Fairness incorporated smoothly

• Deadlock = busy-waiting
  – avoids need for failure sets

• Implicit treatment of channels
  – no channel names in traces
  – object-oriented model
  – channels kept separate
FURTHER WORK

• Relational parametricity
  – representation independence
  – concurrent objects

• Full abstraction at ground types
  – observing sequence of states

• Disjoint processes
  \[
  \llbracket P_1 \parallel P_2 \rrbracket(W_1 \times W_2, H) = \llbracket P_1 \rrbracket(W_1, H) \parallel \llbracket P_2 \rrbracket(W_2, H)
  \]

• Unreliable communication
  – lossy channels
  – bounded channels

• Synchronous communication
newchan[τ] h in
  procedure put(y) = h!y;
  procedure get(z) =
    new[τ] x in (h?x; z:=x);
  begin
    P(put, get)
  end

newchan[τ] h in
  procedure put(y) = h!(-y);
  procedure get(z) =
    new[τ] x in (h?x; z:=(-x));
  begin
    P(put, get)
  end