

**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 λ -calculus

- simply typed

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

phrase types

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

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[ $\tau$ ] x in  
    while true do (in?x; out!x)
```

```
procedure buff(in, out) =  
  newchan[ $\tau$ ] 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 ≠ 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**

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

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

- **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 channel declaration**

$$\frac{\pi, \iota : \mathbf{chan}[\tau] \vdash P : \mathbf{comm}}{\pi \vdash \mathbf{newchan}[\tau] \iota \mathbf{in} P : \mathbf{comm}}$$

CATEGORY of WORLDS

- Objects are countable posets of “allowed states”

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

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
 - f respects prefix order

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

$$\begin{aligned} - \times V &= (\text{fst} : W \times V \rightarrow W, Q) \\ (w_0, v_0) Q (w_1, v_1) &\iff v_0 = v_1 \end{aligned}$$

- 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, $\llbracket \theta \rrbracket : \mathbf{W} \rightarrow \mathbf{D}$
- Judgements $\pi \vdash P : \theta$ denote natural transformations

$$\llbracket P \rrbracket : \llbracket \pi \rrbracket \dot{\rightarrow} \llbracket \theta \rrbracket$$

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

$$\begin{array}{ccc} \llbracket \pi \rrbracket W & \xrightarrow{\llbracket P \rrbracket W} & \llbracket \theta \rrbracket W \\ \llbracket \pi \rrbracket h \downarrow & & \downarrow \llbracket \theta \rrbracket h \\ \llbracket \pi \rrbracket X & \xrightarrow{\llbracket P \rrbracket X} & \llbracket \theta \rrbracket X \end{array}$$

commutes.

Naturality enforces locality

CARTESIAN CLOSURE

- The functor category $\mathbf{D}^{\mathbf{W}}$ is cartesian closed
- Use ccc structure to interpret arrow and product types

$$\begin{aligned} \llbracket \theta \times \theta' \rrbracket &= \llbracket \theta \rrbracket \times \llbracket \theta' \rrbracket \\ \llbracket \theta \rightarrow \theta' \rrbracket &= \llbracket \theta \rrbracket \Rightarrow \llbracket \theta' \rrbracket \end{aligned}$$

- Procedures are natural and therefore respect locality

PROCEDURES

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

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

$$\begin{array}{ccc} \llbracket \theta \rrbracket X & \xrightarrow{p(h)} & \llbracket \theta' \rrbracket X \\ \llbracket \theta \rrbracket h' \downarrow & & \downarrow \llbracket \theta' \rrbracket h' \\ \llbracket \theta \rrbracket Y & \xrightarrow{p(h; h')} & \llbracket \theta' \rrbracket Y \end{array}$$

commutes.

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

COMMANDS

$$\llbracket \text{comm} \rrbracket W = \wp^\dagger((W \times W)^\infty)$$

... 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

$$\langle w_0, w'_0 \rangle \langle w_1, w'_1 \rangle \dots \langle w_n, w'_n \rangle \dots$$

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 $c \in \llbracket \mathbf{comm} \rrbracket W$

$$\llbracket \mathbf{comm} \rrbracket (f, Q)c$$

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

$$\llbracket \mathbf{chan}[\tau] \rrbracket W = (V_\tau \rightarrow \llbracket \mathbf{comm} \rrbracket W) \times \llbracket \mathbf{exp}[\tau] \rrbracket W$$

$$\llbracket \mathbf{exp}[\tau] \rrbracket W = \wp((W \times W)^+ \times V_\tau \cup (W \times W)^\omega)$$

cf. Reynolds, Oles

PARALLEL COMPOSITION

Fair merge of traces

$$\begin{aligned} \llbracket P_1 \parallel P_2 \rrbracket W u = \\ \{ \alpha \mid \exists \alpha_1 \in \llbracket P_1 \rrbracket W u, \alpha_2 \in \llbracket P_2 \rrbracket W u. \\ (\alpha_1, \alpha_2, \alpha) \in \text{fairmerge}_{W \times W} \}^\dagger \end{aligned}$$

where

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

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

$$\langle (v_0, \epsilon), (v_0, \epsilon) \rangle \dots \langle (v_k, \epsilon), (v_k, \epsilon) \rangle \dots$$
- $h!(x + 1)$ has traces

$$\langle (m, \sigma), (m, \sigma(m+1)) \rangle \quad (m \in V_{int}, \sigma \in V_{int}^*)$$
- $h?x || h!(x + 1)$ 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

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

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

newchan $[\tau]$ h **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

- **newchan** $[\tau]$ h **in** $(h!e || h?x) = x := e$
- **newchan** $[\tau]$ h **in** $(h!0; P) = P$
if h does not occur free in P
- **newchan** $[\tau]$ h **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

$\langle(0, \epsilon), (\epsilon, \epsilon)\rangle$	input 0
$\langle(1, \epsilon), (1, 0)\rangle$	output 0
$\langle(1, 0), (\epsilon, 0)\rangle$	input 1

...

Similarly

$buff(left, right)$

has trace

$\langle(0, \epsilon), (\epsilon, \epsilon)\rangle$	input 0
$\langle(1, \epsilon), (\epsilon, \epsilon)\rangle$	input 1
$\langle(2, \epsilon), (2, 0)\rangle$	output 0

...

LAWS

- **Symmetry**

$$\begin{aligned} & \text{newchan}[\tau_1] h_1 \text{ in} \\ & \quad \text{newchan}[\tau_2] h_2 \text{ in } P \\ & = \text{newchan}[\tau_2] h_2 \text{ in} \\ & \quad \text{newchan}[\tau_1] h_1 \text{ in } P \end{aligned}$$

- **Frobenius**

$$\begin{aligned} & \text{newchan}[\tau] h \text{ in } (P_1 \parallel P_2) = \\ & \quad (\text{newchan}[\tau] h \text{ in } P_1) \parallel P_2 \end{aligned}$$

provided h does not occur free in P_2

LAWS

- **Local variables**

$$\begin{aligned} \mathbf{newvar}[\tau] \ \iota \ \mathbf{in} \ P' &= P' \\ \mathbf{newvar}[\tau] \ \iota \ \mathbf{in} \ (P \parallel P') &= \\ &(\mathbf{newvar}[\tau] \ \iota \ \mathbf{in} \ P) \parallel P' \end{aligned}$$

when ι does not occur free in P'

- **Functional laws**

$$\begin{aligned} (\lambda \iota : \theta.P)(Q) &= P[Q/\iota] \\ \mathbf{rec} \ \iota.P &= P[\mathbf{rec} \ \iota.P/\iota] \end{aligned}$$

LOCAL LAWS

- **Local output**

$$\begin{aligned} \text{newchan}[\tau] \ h = \rho \ \mathbf{in} \ P_1 \parallel h!v; P_2 \\ = \text{newchan}[\tau] \ h = \rho v \ \mathbf{in} \ P_1 \parallel P_2 \end{aligned}$$

if $h!$ not in P_1

- **Local input**

$$\begin{aligned} \text{newchan}[\tau] \ h = v\rho \ \mathbf{in} \ P_1 \parallel h?v; P_2 \\ = \text{newchan}[\tau] \ h = \rho \ \mathbf{in} \ P_1 \parallel P_2 \end{aligned}$$

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

$$\mathbf{newchan}[\tau] h = \rho \mathbf{in} P$$

diverges, then

$$\begin{aligned} \mathbf{newchan}[\tau] h = \rho \mathbf{in} P \parallel (P_1; P_2) \\ = P_1; \mathbf{newchan}[\tau] h = \rho \mathbf{in} P \parallel P_2 \end{aligned}$$

- **Unrequited input**

If h not free in P_1 then

$$\begin{aligned} \mathbf{newchan}[\tau] h \mathbf{in} (h?x; P) \parallel (P_1; P_2) \\ = P_1; \mathbf{newchan}[\tau] h \mathbf{in} (h?x; P) \parallel P_2 \end{aligned}$$

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

CONCURRENT OBJECTS

These are semantically equivalent...

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

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