The essence of
PARALLEL ALGOL

Stephen Brookes

Department of Computer Science
Carnegie Mellon University

Slides based on paper
presented at LICS '96

ESSENTIALS

- PARALLEL ALGOL =

  shared-variable parallel programs

  \+ call-by-name λ-calculus

- simply typed

  $\theta ::= \mathbf{exp}[\tau] \mid \mathbf{var}[\tau] \mid \mathbf{comm}$
  $\mid (\theta \to \theta') \mid \theta \times \theta'$ *phrase types*

  $\tau ::= \mathbf{int} \mid \mathbf{bool}$ *data types*

- recursion and conditional at each type

cf. Reynolds: *The essence of* ALGOL

RATIONALE

- Can write parallel programs that *cooperate* by reading and writing shared memory
- Procedures can *encapsulate* parallel idioms (e.g. mutual exclusion, readers–writers)
- Local variable declarations can be used to limit the scope of *interference*

INTUITION

Procedures and parallelism are *orthogonal*

- should combine smoothly
- semantics should be "modular"
- should obtain a *conservative* extension

MUTUAL EXCLUSION

procedure $mutex(n_1, c_1, n_2, c_2)$;
new[**bool**] s **in**
  s:=**true**;
  (**while true do**
    (n_1; **await** s **then** s:=**false**;
     c_1; s:=**true**)
  || **while true do**
    (n_2; **await** s **then** s:=**false**;
     c_2; s:=**true**)
)

- $mutex$: $\mathbf{comm}^4 \rightarrow \mathbf{comm}$
- Encapsulates common use of a *semaphore*
- Correctness relies on *locality* of s
- Independent of n_i and c_i

OUTLINE of SEMANTICS

- Traditional *global state* models fail to validate natural equivalences, e.g.

$$\textbf{new}[\tau]\ \iota\ \textbf{in}\ P = P \quad \text{if } \iota \notin \mathit{free}(P)$$

- Adapt *possible worlds* model of sequential ALGOL to the parallel setting...
- ...and simultaneously extend *transition trace* semantics of shared memory parallel programs to include procedures and recursion.

- Adapt a *relationally parametric* model of sequential ALGOL to the parallel setting...
- ...and introduce a form of parametric reasoning for shared-variable programs.

cf. Reynolds, Oles

cf. O'Hearn, Tennent

CATEGORY of WORLDS

- Objects are countable sets (of "allowed states")
- Morphisms are "expansions"

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

where

  - f is a *function* from X to W
  - Q is an *equivalence relation* on X
  - f puts each Q-class in bijection with W

INTUITION

- X is a set of "large" states extending the "small" states of W
- f extracts the "small" part of a state
- Q identifies states with the same extra parts

cf. Frank Oles' *Ph.D. thesis*

DETAILS

A morphism $(f, Q) : W \to X$ satisfies:

- $f : X \to W$ is a (total, onto) function
- Q is a (total) equivalence relation on X
  - reflexive, symmetric, transitive
- Let $[x]_Q$ be the equivalence class of x
  - $[x]_Q = \{x' \in X \mid (x, x') \in Q\}$
- For each $x \in X$, $f : [x]_Q \to W$ is a bijection

IDENTITY

- For any W the *identity morphism on W* is

$$\mathtt{id}_W = (id_W, W \times W)$$

  where id_W is the identity function on W
- The equivalence relation of $\mathtt{id}_W$ has a single equivalence class: for all $w \in W$, $[w] = W$.

BIJECTIONS

More generally...

- A bijection $f : X \to W$ gives rise to a *morphism* from W to X given by

$$(f, X \times X)$$

COMPOSITION

- When $(f, Q) : W \to X$ and $(g, R) : X \to Y$ are morphisms, define the *composite* to be

$$(f \circ g, S) : W \to Y,$$

  where

$$S = \{(y_1, y_2) \mid (gy_1, gy_2) \in Q\}.$$

- This is a morphism from W to Y, because:
  - $f \circ g : Y \to W$ is a function (total, onto)
  - S is a (total) equivalence relation on Y
  - For all $y \in Y$, $f \circ g : [y]_S \to W$ is a bijection

PROPERTIES

- Composition is associative
- When $(f, Q) : W \to X$ is a morphism,

$$\mathtt{id}_W; (f, Q) = (f, Q) \quad (f, Q); \mathtt{id}_X = (f, Q)$$

ISOMORPHISMS

- A morphism $(f, Q) : W \rightarrow X$ is an *isomorphism* iff there is a morphism $(g, R) : X \rightarrow W$ such that $(f, Q); (g, R) = \mathtt{id}_W$
- This happens iff f is a bijection from X to W and Q is $X \times X$
  - g will be f^{-1}
  - R will be $W \times W$

EXPANSIONS

- For each pair of objects W and V there is a canonical *expansion* morphism

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

given by

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

where

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

so that Q is $(=_V) \circ \text{snd}$

- An expansion $- \times V_\tau$ models the introduction of a new variable of datatype τ.

THEOREM

Every morphism is an expansion composed with an isomorphism. (Oles)

CATEGORIES

- Let $\mathbf{W}$ be the category of *worlds*, with morphisms representing expansions

- Let $\mathbf{D}$ be the category of *domains*, with continuous functions as morphisms
  - A *domain* $(D, \sqsubseteq)$ is a complete partial order with a least element $\bot_D$
  - A function $F : (D, \sqsubseteq) \to (D', \sqsubseteq')$ is *continuous* iff for all chains $\langle d_n \rangle_{n=0}^{\infty}$ in D, the sequence $\langle F(d_n) \rangle_{n=0}^{\infty}$ is also a chain, and

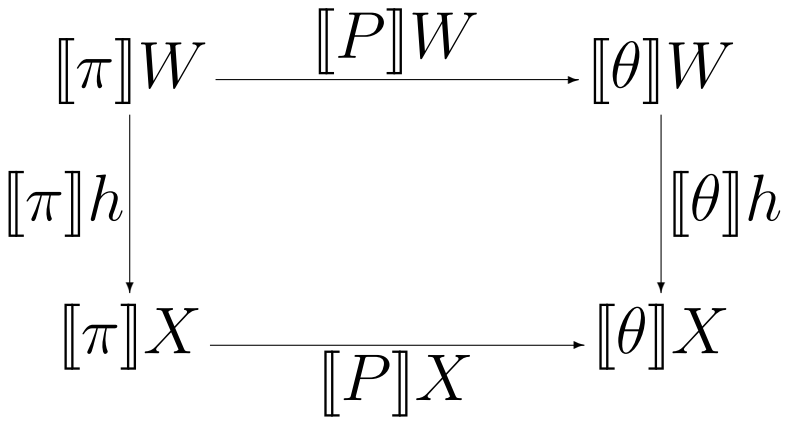
$$\bigsqcup_{n=0}^{\infty}{}' F(d_n) = F(\bigsqcup_{n=0}^{\infty} d_n)$$

SEMANTICS

- Types denote *functors* from worlds to domains

$$[\![\theta]\!] : \mathbf{W} \to \mathbf{D}$$

- Phrases denote *natural transformations*, i.e. when $\pi \vdash P : \theta$, $[\![P]\!] : [\![\pi]\!] \dot{\to} [\![\theta]\!]$
- This means that when $h : W \to X$,



commutes.

When h is an expansion, *naturality* enforces locality.

CARTESIAN CLOSURE

- The functor category $\mathbf{D}^{\mathbf{W}}$ is cartesian closed.
- Can use ccc structure to interpret arrow types.

Procedures of type $\theta \to \theta'$ denote, at world W, natural families of functions $p(-)$:

- When $h : W \to X$ and $h' : X \to Y$,

$$\begin{array}{ccc} [\![\theta]\!]X & \xrightarrow{\quad p(h) \quad} & [\![\theta']\!]X \\ {\scriptstyle [\![\theta]\!]h'}\downarrow & & \downarrow{\scriptstyle [\![\theta']\!]h'} \\ [\![\theta]\!]Y & \xrightarrow[\quad p(h;h') \quad]{} & [\![\theta']\!]Y \end{array}$$

commutes.

INTUITION

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

COMMANDS

- Commands denote sets of *traces*

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

- Trace sets are *closed*, e.g.
  - $\alpha\beta \in c \;\&\; w \in W \;\Rightarrow\; \alpha(w,w)\beta \in c$
  - $\alpha(w,w')(w',w'')\beta \in c \;\Rightarrow\; \alpha(w,w'')\beta \in c$

- When $h : W \to X$, $[\![\textbf{comm}]\!]h$ converts a trace set over W to a trace set over X

$$[\![\textbf{comm}]\!](f,Q)c = \{\beta \mid \mathrm{map}(f \times f)\beta \in c \;\&\; \mathrm{map}(Q)\beta\}$$

INTUITION

- A trace $(w_0,w_0')(w_1,w_1')\ldots(w_n,w_n')$ represents a fair interactive computation.
- Each step (w_i,w_i') represents a finite sequence of atomic actions.
- $[\![\textbf{comm}]\!]hc$ behaves like c on the *W*-component of state and has no effect on the "extra" component.

SPECIAL CASE

- Let h be an expansion from W to $W \times V$
- Let $T \in [\![\mathbf{comm}]\!]W$ be a trace set over W
- $[\![\mathbf{comm}]\!]hT$ is the trace set T' over $W \times V$ such that, for $\alpha \in ((W \times V) \times (W \times V))^{\infty}$, $\alpha \in T'$ iff

  $\mathtt{map\,fst}\,\alpha \in T \;\&\; \forall((w, v), (w', v')) \in \alpha.\, v = v'$

EXPRESSIONS

Expressions denote trace sets

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

$$[\![\mathbf{exp}[\tau]]\!](f, Q)e = \{(\rho', v) \mid (\mathrm{map} f \rho', v) \in e\} \cup \{\rho' \mid \mathrm{map} f \rho' \in e \cap W^{\omega}\}$$

VARIABLES

"Object-oriented" interpretation *à la* Reynolds:

variable = acceptor + expression

$$[\![\mathbf{var}[\tau]]\!]W = (V_{\tau} \to [\![\mathbf{comm}]\!]W) \times [\![\mathbf{exp}[\tau]]\!]W$$

RECURSION

Requires a careful use of *greatest fixed points*

- Embed $[\![\theta]\!]W$ in a complete lattice $[\theta]W$ (like $[\![\theta]\!]W$ but without closure and naturality)
- Generalize semantic definitions to $[P]W$.
- Introduce natural transformations

$$\mathrm{stut}_\theta : [\theta] \dot{\longrightarrow} [\theta] \qquad \mathrm{clos}_\theta : [\theta] \dot{\longrightarrow} [\![\theta]\!]$$

- Can then define $[\![\mathbf{rec}\ \iota.P]\!]Wu$ to be

$$\mathrm{clos}_\theta W(\nu x.\mathrm{stut}_\theta W([P]W(u \mid \iota : x)))$$

EXAMPLE

- Divergence = infinite stuttering:

$$\begin{aligned}[\![\mathbf{rec}\ \iota.\iota]\!]Wu &= (\nu c.\{(w,w)\alpha \mid \alpha \in c\})^\dagger \\ &= \{(w,w) \mid w \in W\}^\omega\end{aligned}$$

AGREEMENT THEOREM

Suppose $\pi \vdash P : \theta$ is valid.
Then for all W, all $u, u' \in [\![\pi]\!]W$,
if u and u' agree on $\mathtt{free}(P)$ then

$$[\![P]\!]Wu = [\![P]\!]Wu'$$

LAWS

- This semantics validates:

$$\begin{aligned}&\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P' = P'\\ &\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P \| P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P) \| P'\\ &\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P; P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P); P'\end{aligned}$$

  when ι does not occur free in P'.

- Also (still) validates:

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

- Orthogonal combination of laws of shared-variable programming with laws of λ-calculus.

LAWS

For all suitably typed P,

$$\textbf{new}[\tau_1]\ \iota_1\ \textbf{in}\ \textbf{new}[\tau_2]\ \iota_2\ \textbf{in}\ P = \textbf{new}[\tau_2]\ \iota_2\ \textbf{in}\ \textbf{new}[\tau_1]\ \iota_1\ \textbf{in}\ P$$

$$\textbf{new}[\tau]\ \iota_1\ \textbf{in}\ \textbf{new}[\tau]\ \iota_2\ \textbf{in}\ P(\iota_1, \iota_2) = \textbf{new}[\tau]\ \iota_1\ \textbf{in}\ \textbf{new}[\tau]\ \iota_2\ \textbf{in}\ P(\iota_2, \iota_1).$$

Reason: naturality of $[\![P]\!]$ wrt the obvious isomorphism between

$$(W \times V_{\tau_1}) \times V_{\tau_2}$$

and

$$(W \times V_{\tau_2}) \times V_{\tau_1}$$

EXAMPLE

Semantics validates some program equivalences based on *representation independence*...
e.g.

$$\begin{array}{l}\textbf{new}[\textbf{bool}]\ \iota{:=}\mathbf{true}\\ \textbf{procedure}\ \mathit{flip};\ \iota{:=}\textbf{not}\ \iota;\\ \textbf{procedure}\ \mathit{test};\ \mathbf{return}\ \iota;\\ \textbf{in}\\ P(\mathit{flip}, \mathit{test})\end{array}$$

and

$$\begin{array}{l}\textbf{new}[\textbf{bool}]\ \iota{:=}\mathbf{false}\\ \textbf{procedure}\ \mathit{flip};\ \iota{:=}\textbf{not}\ \iota;\\ \textbf{procedure}\ \mathit{test};\ \mathbf{return}\ (\textbf{not}\ \iota);\\ \textbf{in}\\ P(\mathit{flip}, \mathit{test})\end{array}$$

are equivalent
(Reason: naturality wrt the obvious isomorphism)

PROBLEM

Semantics fails to validate

$$\textbf{new}[\textbf{int}]\ \iota{:=}0\ \textbf{in}\ P(\iota{:=}\iota+1) = P(\textbf{skip}),$$

where P is a free identifier of type **comm** $\to$ **comm**.

REASON

- Equivalence relies on *relational reasoning*.
- Naturality does not enforce enough constraints on procedure meanings.

SOLUTION

- Same problem arose in sequential setting.
- Develop a relationally parametric semantics...

cf. O'Hearn and Tennent

PARAMETRIC MODEL

- Category of relations $R : W_0 \leftrightarrow W_1$
- A morphism from R to S is a pair (h_0, h_1) of morphisms in **W** such that

$$\begin{array}{ccc} W_0 & \xrightarrow{h_0} & X_0 \\ R \updownarrow & & \updownarrow S \\ W_1 & \xrightarrow[h_1]{} & X_1 \end{array}$$

- Types denote *parametric* functors, e.g.
  - if $R : W_0 \leftrightarrow W_1$, $[\![\theta]\!]R : [\![\theta]\!]W_0 \leftrightarrow [\![\theta]\!]W_1$
  - $(d_0, d_1) \in [\![\theta]\!]R \Rightarrow ([\![\theta]\!]h_0 d_0, [\![\theta]\!]h_1 d_1) \in [\![\theta]\!]S$
- Phrases denote *parametric* natural transformations

  $(u_0, u_1) \in [\![\pi]\!]R \;\Rightarrow\; ([\![P]\!]W_0 u_0, [\![P]\!]W_1 u_1) \in [\![\theta]\!]R$
- The *parametric functor* category is cartesian closed.

COMMANDS

When $R : W_0 \leftrightarrow W_1$ define:

$$(c_0, c_1) \in [\![\textbf{comm}]\!] R \iff$$

$$\begin{array}{l}
\forall(\rho_0, \rho_1) \in \text{map}(R). \\
\quad [\forall \alpha_0 \in c_0.\ \text{map fst}\ \alpha_0 = \rho_0 \ \Rightarrow \\
\quad\ \exists \alpha_1 \in c_1.\ \text{map fst}\ \alpha_1 = \rho_1 \ \& \\
\qquad\quad (\text{map snd}\ \alpha_0,\ \text{map snd}\ \alpha_1) \in \text{map}(R)] \\
\ \& \\
\quad [\forall \alpha_1 \in c_1.\ \text{map fst}\ \alpha_1 = \rho_1 \ \Rightarrow \\
\quad\ \exists \alpha_0 \in c_0.\ \text{map fst}\ \alpha_0 = \rho_0 \ \& \\
\qquad\quad (\text{map snd}\ \alpha_0, \text{map snd}\ \alpha_1) \in \text{map}(R)].
\end{array}$$

This is parametric!

INTUITION

When related commands are started and interrupted in related states their responses are related.

LAWS

- As before,

$$\begin{aligned}&\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P' = P'\\ &\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P \| P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P) \| P'\\ &\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P; P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P); P'\end{aligned}$$

  when ι does not occur free in P'.

- As before,

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

- In addition,

$$\begin{aligned}&\mathbf{new}[\mathbf{int}]\ \iota\ \mathbf{in}\ (\iota{:=}1; P(\iota)) \ = \ P(1)\\ &\mathbf{new}[\mathbf{int}]\ \iota\ \mathbf{in}\ (\iota{:=}0;\ P(\iota{:=}\iota+1)) \ = \ P(\mathbf{skip}),\end{aligned}$$

  relying crucially on parametricity.

EXAMPLE

$$\textbf{new}[\textbf{int}]\ x\ \textbf{in}$$
$$(x{:=}0;\ P(x{:=}x+1; x{:=}x+1);$$
$$\textbf{if}\ even(x)\ \textbf{then diverge else skip})$$

and

$$\textbf{new}[\textbf{int}]\ x\ \textbf{in}$$
$$(x{:=}0;\ P(x{:=}x+2);$$
$$\textbf{if}\ even(x)\ \textbf{then diverge else skip})$$

are equivalent in sequential ALGOL
but not equivalent in PARALLEL ALGOL.

The relation

$$(w,(w',z)) \in R \iff w = w' \,\&\, even(z)$$

works for sequential model but not for parallel.

CONCLUSIONS

- Can combine parallelism and procedures smoothly:
  - faithful to the essence of ALGOL
  - allows formalization of parallel idioms
  - retains laws of component languages
- Semantics by "modular" combination:
  - traces + possible worlds
  - traces + relational parametricity
- Advantages:
  - full abstraction at ground types
  - supports common reasoning principles:
    - representation independence
    - global invariants
    - assumption–commitment
- Limitations:
  - does not build in irreversibility of state change

SEMANTICS of skip

Finite stuttering:

$$\begin{aligned}[\![\textbf{skip}]\!]Wu &= \{(w,w) \mid w \in W\}^\dagger \\ &= \{(w,w) \mid w \in W\}^+\end{aligned}$$

ASSIGNMENT

Non-atomic; source expression evaluated first:

$$\begin{aligned}&[\![I{:=}E]\!]Wu = \\ &\quad \{(\mathrm{map}\Delta_W\rho)\beta \mid (\rho,v) \in [\![E]\!]Wu \\ &\qquad\qquad \& \ \beta \in \mathrm{fst}([\![I]\!]Wu)v\}^\dagger \\ &\quad \cup \{\mathrm{map}\Delta_W\rho \mid \rho \in [\![E]\!]Wu \cap W^\omega\}^\dagger.\end{aligned}$$

PARALLEL COMPOSITION

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

where

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

LOCAL VARIABLES

$$[\![\textbf{new}[\tau]\ \iota\ \textbf{in}\ P]\!] W u = \{\mathrm{map}(\mathrm{fst} \times \mathrm{fst})\alpha \mid \mathrm{map}(\mathrm{snd} \times \mathrm{snd})\alpha \text{ interference-free } \&\ \alpha \in [\![P]\!](W \times V_\tau)([\![\pi]\!](- \times V_\tau)u \mid \iota : (a, e))\}$$

- No external changes to local variable
- $(a, e) \in [\![\textbf{var}[\tau]]\!](W \times V_\tau)$ is a "fresh variable" corresponding to the V_τ component of the state

AWAIT

$$\begin{aligned}[\![\textbf{await } B \textbf{ then } P]\!]Wu = \\ \{(w, w') \in [\![P]\!]Wu \mid (w, \mathtt{tt}) \in [\![B]\!]Wu\}^\dagger \\ \cup \{(w, w) \mid (w, \mathtt{ff}) \in [\![B]\!]Wu\}^\omega \\ \cup \{\mathrm{map}\Delta_W \rho \mid \rho \in [\![B]\!]Wu \cap W^\omega\}^\dagger.\end{aligned}$$

- P is atomic, enabled only when B true.
- Busy wait when B false.

λ-CALCULUS

$$[\![\iota]\!]Wu = u\iota$$

$$[\![\lambda\iota : \theta.P]\!]Wuha = [\![P]\!]W'([\![\pi]\!]hu \mid \iota : a)$$

$$[\![P(Q)]\!]Wu = [\![P]\!]Wu(\mathrm{id}_W)([\![Q]\!]Wu),$$

- This is the standard interpretation, based on the ccc structure of the functor category.