

# On Locality and the Exchange Law for Concurrent Processes

C.A.R. Hoare<sup>1</sup>   A. Hussain<sup>2</sup>   B. Möller<sup>3</sup>   P.W. O'Hearn<sup>2</sup>  
R.L. Petersen<sup>2</sup>   G. Struth<sup>4</sup>

1. Microsoft Research Cambridge
2. Queen Mary University of London
3. Universität Augsburg
4. University of Sheffield

September 7 2011

# On Locality and the Exchange Law for Concurrent Processes

C.A.R. Hoare<sup>1</sup>   A. Hussain<sup>2</sup>   B. Möller<sup>3</sup>   P.W. O'Hearn<sup>2</sup>  
R.L. Petersen<sup>2</sup>   G. Struth<sup>4</sup>

1. Microsoft Research Cambridge
2. Queen Mary University of London
  3. Universität Augsburg
  4. University of Sheffield

September 7 2011

# On Locality and the Exchange Law for Concurrent Processes

C.A.R. Hoare<sup>1</sup>   **A. Hussain**<sup>2</sup>   B. Möller<sup>3</sup>   **P.W. O'Hearn**<sup>2</sup>  
**R.L. Petersen**<sup>2</sup>   G. Struth<sup>4</sup>

1. Microsoft Research Cambridge
2. Queen Mary University of London
  3. Universität Augsburg
  4. University of Sheffield

September 7 2011

# Motivation

*We are giving an algebraic treatment of concurrency and linking it to standard models.*

# Motivation

YACC  
(Yet Another Concurrency Conceptualization)

# Motivation

YACC  
(Yet Another Concurrency Conceptualization)

How can this possibly be justified?

# Motivation

1. Easy proofs of basic laws of Concurrent Separation Logic  
(so easy that we will see them in this talk)
2. Uniform treatment of very disparate models for concurrent computation  
(message passing, shared memory, weak memory, etc. . . )

# Motivation

Peter O'Hearn



“What is going on here!?”

# Motivation

Peter O'Hearn



“What is going on here!?”

Tony Hoare



“We are cheating :-)”

# Motivation

*“The validity of Hoare logic in this weak model is entirely due to a cheat: that we use the same model for our assertions as for our programs.”*

Concurrent Kleene Algebra, CONCUR 2009

# Motivation

*“The validity of Hoare logic in this weak model is entirely due to a cheat: that we use the same model for our assertions as for our programs.”*

Concurrent Kleene Algebra, CONCUR 2009

But did they cheat... ?

# Motivation

Two instances of the algebra:

## Trace based model

in which the Exchange Law will be obvious  
(Simplified version of CONCUR'09)

## State based model

where we know what locality and separation logic  
should mean.

# Outline

## The Algebra

# Outline

The Algebra

The Trace Model

# Outline

The Algebra

The Trace Model

The Resource Model

# Outline

The Algebra

The Trace Model

The Resource Model

Does it mean what it says?

# Locality Bimonoid

In the paper, the algebra is built up in stages, but the end result looks as follows:

## Definition (Locality bimonoid)

An *locality bimonoid* is a structure  $(M, \sqsubseteq, *, \text{nothing}, ;, \text{skip})$  such that  $(M, \sqsubseteq, *, \text{nothing})$  is a commutative ordered monoid and  $(M, \sqsubseteq, ;, \text{skip})$  is an ordered monoid and the following holds:

# Locality Bimonoid

In the paper, the algebra is built up in stages, but the end result looks as follows:

## Definition (Locality bimonoid)

An *locality bimonoid* is a structure  $(M, \sqsubseteq, *, \text{nothing}, ;, \text{skip})$  such that  $(M, \sqsubseteq, *, \text{nothing})$  is a commutative ordered monoid and  $(M, \sqsubseteq, ;, \text{skip})$  is an ordered monoid and the following holds:

# Locality Bimonoid

In the paper, the algebra is built up in stages, but the end result looks as follows:

## Definition (Locality bimonoid)

An *locality bimonoid* is a structure  $(M, \sqsubseteq, *, \text{nothing}, ;, \text{skip})$  such that  $(M, \sqsubseteq, *, \text{nothing})$  is a commutative ordered monoid and  $(M, \sqsubseteq, ;, \text{skip})$  is an ordered monoid and the following holds:

# Locality Bimonoid

In the paper, the algebra is built up in stages, but the end result looks as follows:

## Definition (Locality bimonoid)

An *locality bimonoid* is a structure  $(M, \sqsubseteq, *, \text{nothing}, ;, \text{skip})$  such that  $(M, \sqsubseteq, *, \text{nothing})$  is a commutative ordered monoid and  $(M, \sqsubseteq, ;, \text{skip})$  is an ordered monoid and the following holds:

- $(p * r); (q * s) \sqsubseteq (p; q) * (r; s)$  (Exchange Law)

# Locality Bimonoid

In the paper, the algebra is built up in stages, but the end result looks as follows:

## Definition (Locality bimonoid)

An *locality bimonoid* is a structure  $(M, \sqsubseteq, *, \text{nothing}, ;, \text{skip})$  such that  $(M, \sqsubseteq, *, \text{nothing})$  is a commutative ordered monoid and  $(M, \sqsubseteq, ;, \text{skip})$  is an ordered monoid and the following holds:

- $(p * r) ; (q * s) \sqsubseteq (p ; q) * (r ; s)$  (Exchange Law)
- $\text{skip} = \text{skip} * \text{skip}$  (skip is local)

# Locality Bimonoid

## Definition

An element  $p$  in a locality bimonoid is *local* if  $p * \text{skip} = p$ .

The fact that  $\text{skip}$  is local has two consequences:

- The local elements form a sub locality bimonoid
- The map  $(-)*\text{skip}$  acts as a localizer.

# Algebraic Triples

We postulate that this is a reasonable definition of a Hoare triple:

$$\{p\} c \{q\} \Leftrightarrow p; c \sqsubseteq q$$

Intuition: Being  $p$  and then running  $c$  is described by  $q$ .

## Algebraic Triples

If we accept this for a moment, there are nice consequences:

Rule of Consequence:

$$\forall p, c, q, p', q'. \{p'\} c \{q'\} \wedge p \sqsubseteq p' \wedge q' \sqsubseteq q \Rightarrow \{p\} c \{q\}$$

Sequencing Rule:

$$\forall p, c, r, c', q. \{p\} c \{r\} \wedge \{r\} c' \{q\} \Rightarrow \{p\} c; c' \{q\}$$

Skip Rule:

$$\forall p. \{p\} \text{skip} \{p\}$$

Concurrency Rule:

$$\forall p, c, q, p', c', q'. \{p\} c \{q\} \wedge \{p'\} c' \{q'\} \Rightarrow \{p * p'\} c * c' \{q * q'\}$$

Frame Rule (for  $c$  local):

$$\forall p, q, r. \{p\} c \{q\} \Rightarrow \{p * r\} c \{q * r\}$$

# Easy Proofs

## Concurrency Rule

$$\{p\} c \{q\} \wedge \{p'\} c' \{q'\} \Rightarrow \{p * p'\} c * c' \{q * q'\}$$

$$(p * p'); (c * c') \sqsubseteq (p; c) * (p'; c') \sqsubseteq q * q'$$

## Frame Rule (for $c$ local)

$$\{p\} c \{q\} \Rightarrow \{p * r\} c \{q * r\}$$

$$(p * r); c = (p * r); (c * \text{skip}) \sqsubseteq (p; c) * (r; \text{skip}) = (p; c) * r \sqsubseteq q * r$$

# Locality is the same as local reasoning!

## Theorem

*For all  $c$ :*

$$\forall p, q, r. \{p\} c \{q\} \Rightarrow \{p * r\} c \{q * r\}$$

iff

$c$  is local

# An obvious model

## Definition (The Trace Model)

Let  $A$  be a set. Then *Traces* is the set of finite sequences over  $A$ . We have the following two binary operations on  $\mathcal{P}(\textit{Traces})$

1.  $T_1 * T_2$  is the set of interleavings of traces in  $T_1$  with traces in  $T_2$ ;
2.  $T_1 ; T_2$  is the set of concatenations of traces in  $T_1$  with traces in  $T_2$ .

$*$  and  $;$  have a shared unit  $\{\epsilon\}$ , where  $\epsilon$  is the empty sequence. The model is  $(\mathcal{P}(\textit{Traces}), \subseteq, *, \{\epsilon\}, ;, \{\epsilon\})$ .

# Exchange in the Trace Model

$$(p * r); (q * s) \subseteq (p; q) * (r; s)$$

Let  $t_1$  be an interleaving of  $p$  and  $r$  and  $t_2$  be an interleaving of  $q$  and  $s$ . Then  $t_1 ; t_2$  is certainly an interleaving of  $p; q$  and  $r; s$ .

# Exchange in the Trace Model

$$(p * r); (q * s) \subseteq (p; q) * (r; s)$$

Let  $t_1$  be an interleaving of  $p$  and  $r$  and  $t_2$  be an interleaving of  $q$  and  $s$ . Then  $t_1 ; t_2$  is certainly an interleaving of  $p; q$  and  $r; s$ .

$p_1 p_2 r_1 p_3 r_2 \dots p_n$

# Exchange in the Trace Model

$$(p * r); (q * s) \subseteq (p; q) * (r; s)$$

Let  $t_1$  be an interleaving of  $p$  and  $r$  and  $t_2$  be an interleaving of  $q$  and  $s$ . Then  $t_1 ; t_2$  is certainly an interleaving of  $p; q$  and  $r; s$ .

$p_1 p_2 r_1 p_3 r_2 \dots p_n \quad q_1 s_1 s_2 q_2 q_3 \dots s_n$

## Exchange in the Trace Model

$$(p * r); (q * s) \subseteq (p; q) * (r; s)$$

Let  $t_1$  be an interleaving of  $p$  and  $r$  and  $t_2$  be an interleaving of  $q$  and  $s$ . Then  $t_1; t_2$  is certainly an interleaving of  $p; q$  and  $r; s$ .

$p_1 p_2 r_1 p_3 r_2 \dots p_n q_1 s_1 s_2 q_2 q_3 \dots s_n$

# A desirable model

## States

Let  $(\Sigma, \bullet, u)$  be a partial, commutative monoid:

$$\begin{aligned} \bullet : \Sigma \times \Sigma &\rightarrow \Sigma, \quad \text{commutative and associative} \\ \forall \sigma \in \Sigma. \sigma \bullet u &= u \bullet \sigma = \sigma \end{aligned}$$

## Predicates

$(\mathcal{P}(\Sigma), \subseteq, *, \text{emp})$  is an ordered total commutative monoid with

$$\begin{aligned} p * q &= \{\sigma_0 \bullet \sigma_1 \mid \sigma_0 \# \sigma_1 \wedge \sigma_0 \in p \wedge \sigma_1 \in q\} \\ \text{emp} &= \{u\} \end{aligned}$$

where  $\sigma_0 \# \sigma_1$  means that  $\sigma_0 \bullet \sigma_1$  is defined.

## A desirable model

### Programs

Elements of the monotone function space  $[\mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)]$  represent (backwards) predicate transformers.

We use the reverse pointwise order  $\sqsubseteq$ .

The algebra has the following operators:

$$(F_1 * F_2)Y = \bigcup \{F_1 Y_1 * F_2 Y_2 \mid Y_1 * Y_2 \subseteq Y\}$$

$$\text{nothing } Y = Y \cap \text{emp}$$

$$(F_1 ; F_2)Y = F_1(F_2(Y))$$

$$\text{skip } Y = Y$$

The model is  $([\mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)], \sqsubseteq, *, \text{nothing}, ;, \text{skip})$ .

# Locality in the Ressource Model

Locality is supposed to mean that processes mind their own business.

# Locality in the Ressource Model

Locality is supposed to mean that processes mind their own business.

Now we say it means  $F = F * \text{skip}$ ?

## Locality in the Resource Model

Locality is supposed to mean that processes mind their own business.

Now we say it means  $F = F * \text{skip}$ ?

### Lemma

*A predicate transformer  $F$  satisfies  $F = F * \text{skip}$ . iff*

$$\forall Y, R. FY * R \subseteq F(Y * R)$$

So the states that lead to  $Y$  can be extended by  $R$  to obtain states that lead to  $Y * R$ .

# Turning predicates into programs

## Definition (Dijkstra Triple)

For  $X$  and  $Y$  predicates and  $F$  a predicate transformer, we define

$$\langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle \Leftrightarrow X \subseteq FY$$

This captures that all states in  $X$  will lead to  $Y$  when  $F$  is run.

Comparing Hoare- and Dijkstra triples requires that we can turn the predicates  $X$  and  $Y$  in the relationship  $X \subseteq FY$  defining the Dijkstra triple into predicate transformers.

# Turning predicates into programs

$$bpt[X, Y] = \lambda P. \text{if } Y \subseteq P \text{ then } X \text{ else } \emptyset$$

It is easily seen that  $bpt[X, Y]$  is the largest monotone transformer  $F$  such that  $\langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle$ .

## Turning predicates into programs

$$bpt[X, Y] = \lambda P. \text{if } Y \subseteq P \text{ then } X \text{ else } \emptyset$$

It is easily seen that  $bpt[X, Y]$  is the largest monotone transformer  $F$  such that  $\langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle$ .

$$bpt[X, Y] * bpt[X', Y'] = bpt[X * X', Y * Y']$$

So if we define  $create[Y] = bpt[\text{emp}, Y]$ , we obtain

- $create[-]$  preserves  $*$
- $create[-]$  preserves unit ( $create[\text{emp}] = \text{nothing}$ )

# Correspondence of triples

## Theorem

$$\langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle \Leftrightarrow \{ \mathit{create}[X] \} F \{ \mathit{create}[Y] \}$$

# Correspondence of triples

## Theorem

$$\langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle \Leftrightarrow \{ \text{create}[X] \} F \{ \text{create}[Y] \}$$

So for instance

$$\begin{aligned} & \langle\langle X \rangle\rangle F \langle\langle Y \rangle\rangle \\ \Leftrightarrow & \{ \text{create}[X] \} F \{ \text{create}[Y] \} \\ \Rightarrow & \{ \text{create}[X] * \text{create}[R] \} F \{ \text{create}[Y] * \text{create}[R] \} \\ \Leftrightarrow & \{ \text{create}[X * R] \} F \{ \text{create}[Y * R] \} \\ \Leftrightarrow & \langle\langle X * R \rangle\rangle F \langle\langle Y * R \rangle\rangle \end{aligned}$$

# Summary

We have presented an abstract algebraic treatment of concurrency and linked it to standard concrete models.

In the process we obtained an algebraic characterization of locality.