On Separation, Session Types and Algebra

Akbar Hussain Peter W. O'Hearn Rasmus L. Petersen

Department of Computer Science
Queen Mary University of London

Dublin Concurrency Workshop, 2011

Formalisms

Modularity

Formalisms

Modularity

Session Types

- Process Calculi
- Message Passing

Formalisms

Modularity

Session Types

- Process Calculi
- Message Passing

Concurrent Separation Logic

- Imperative Programs
- Shared Resource

Formalisms

Modularity

Session Types

- Process Calculi
- Message Passing

Concurrent Separation Logic

- Imperative Programs
- Shared Resource

Algebra

Outline

Outline

Baby Session Types

Programs

$$P \quad ::= \quad k?j.P \mid k!j.P \mid P \parallel P \mid \texttt{inact}$$

Baby Session Types

Programs

$$P \quad ::= \quad k?j.P \mid k!j.P \mid P \parallel P \mid \texttt{inact}$$

Types

$$\alpha, \beta \ ::= !{[\alpha]};\beta \mid ?{[\alpha]};\beta \mid \texttt{end}$$

Baby Session Types

Programs

$$P \quad ::= \quad k?j.P \mid k!j.P \mid P \parallel P \mid \texttt{inact}$$

Types

$$\alpha, \beta \ ::= ![\alpha];\beta \mid ?[\alpha];\beta \mid \texttt{end}$$

Co-Types

$$\overline{![\alpha];\beta} = ?[\alpha];\overline{\beta} \quad \overline{?[\alpha];\beta} = ![\alpha];\overline{\beta} \quad \overline{\texttt{end}} = \texttt{end}$$

Baby Session Types

Typing Context

- Δ ranges over finite multisets of variable/type pairs

Baby Session Types

Typing Context

- Δ ranges over finite multisets of variable/type pairs
- Δ is consistent when channels occur at most twice and are co-types of each other

Baby Session Types

Typing Context

- Δ ranges over finite multisets of variable/type pairs
- Δ is consistent when channels occur at most twice and are co-types of each other
- Δ is said to be complete if `end` is the only type that appears in it and it is denoted by Φ.

Baby Session Types

Typing Context

- Δ ranges over finite multisets of variable/type pairs
- Δ is consistent when channels occur at most twice and are co-types of each other
- Δ is said to be complete if `end` is the only type that appears in it and it is denoted by Φ.
- $\Delta \circ \Delta'$ is multiset union, where we write $\Delta \asymp \Delta'$ to mean that $\Delta \circ \Delta'$ is consistent.

Baby Session Types

Typing Context

- Δ ranges over finite multisets of variable/type pairs
- Δ is consistent when channels occur at most twice and are co-types of each other
- Δ is said to be complete if $\mathtt{end}$ is the only type that appears in it and it is denoted by Φ.
- $\Delta \circ \Delta'$ is multiset union, where we write $\Delta \asymp \Delta'$ to mean that $\Delta \circ \Delta'$ is consistent.

Typing

$$P \triangleright \Delta$$

Proof Rules for BST

$$[\text{Consequence}]\ \frac{\Delta_1 \vdash \Delta_2 \quad P \triangleright \Delta_2}{P \triangleright \Delta_1}$$

$$[\text{Inact}]\ \frac{}{\texttt{inact} \triangleright \emptyset} \qquad [\text{Par}]\ \frac{P_1 \triangleright \Delta_1 \quad P_2 \triangleright \Delta_2}{P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2}$$

$$[\text{Receive}]\ \frac{P \triangleright \Delta \circ k : \beta \circ j : \alpha}{k?j.P \triangleright \Delta \circ k : ?[\alpha]; \beta} \qquad [\text{Send}]\ \frac{P \triangleright \Delta \circ k : \beta}{k!j.P \triangleright \Delta \circ k : ![\alpha]; \beta \circ j : \alpha}$$

Proof Rules for BST

$$[\text{Consequence}]\ \frac{\Delta_1 \vdash \Delta_2 \quad P \triangleright \Delta_2}{P \triangleright \Delta_1}$$

$$[\text{Inact}]\ \frac{}{\texttt{inact} \triangleright \emptyset} \qquad [\text{Par}]\ \frac{P_1 \triangleright \Delta_1 \quad P_2 \triangleright \Delta_2}{P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2}$$

$$[\text{Receive}]\ \frac{P \triangleright \Delta \circ k : \beta \circ j : \alpha}{k?j.P \triangleright \Delta \circ k : ?[\alpha]; \beta} \qquad [\text{Send}]\ \frac{P \triangleright \Delta \circ k : \beta}{k!j.P \triangleright \Delta \circ k : ![\alpha]; \beta \circ j : \alpha}$$

Examples

Example (1 - Racey programs do not type check)

$$(k!x.\texttt{inact}) \parallel (k!y.\texttt{inact}) \triangleright \Delta$$

Examples

Example (1 - Racey programs do not type check)

$$(k!x.\texttt{inact}) \parallel (k!y.\texttt{inact}) \triangleright \Delta$$

$$\Rightarrow \Delta \text{ is not consistent}$$

Examples

Example (1 - Racey programs do not type check)

$$(k!x.\texttt{inact}) \parallel (k!y.\texttt{inact}) \triangleright \Delta$$

$$\Rightarrow \Delta \text{ is not consistent}$$

Example (2 - Ownership Transfer)

$$\texttt{Let } H = ![\alpha];\texttt{end}$$

Process 1	Process 2	Process 3
$\{h : H,\ h' : \overline{H},\ k :![H];\texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{k :?[H];\texttt{end},\ w : \texttt{end}\}$	$j :?[\overline{H}];\texttt{end}\}$
$k!h$	$k?(x)$	$j?(y)$
$\{h' : \overline{H},\ k : \texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{x : H,\ k : \texttt{end},\ w : \texttt{end}\}$	$\{y : \overline{H},\ j : \texttt{end}\}$
$j!h'$	$x!w$	$y?(z)$
$\{k : \texttt{end},\ j : \texttt{end}\}$	$\{k : \texttt{end},\ x : \texttt{end}\}$	$\{j : \texttt{end},\ y : \texttt{end},\ z : \texttt{end}\}$

Examples

Example (1 - Racey programs do not type check)

$$(k!x.\texttt{inact}) \parallel (k!y.\texttt{inact}) \triangleright \Delta$$

$$\Rightarrow \Delta \text{ is not consistent}$$

Example (2 - Ownership Transfer)

$$\texttt{Let } H = ![\alpha];\texttt{end}$$

Process 1	Process 2	Process 3
$\{h : H,\ h' : \overline{H},\ k :![H];\texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{k :?[H];\texttt{end},\ w : \texttt{end}\}$	$\{j :?[\overline{H}];\texttt{end}\}$
$k!h$	$k?(x)$	$j?(y)$
$\{h' : \overline{H},\ k : \texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{x : H,\ k : \texttt{end},\ w : \texttt{end}\}$	$\{y : \overline{H},\ j : \texttt{end}\}$
$j!h'$	$x!w$	$y?(z)$
$\{k : \texttt{end},\ j : \texttt{end}\}$	$\{k : \texttt{end},\ x : \texttt{end}\}$	$\{j : \texttt{end},\ y : \texttt{end},\ z : \texttt{end}\}$

Examples

Example (1 - Racey programs do not type check)

$$(k!x.\texttt{inact}) \parallel (k!y.\texttt{inact}) \triangleright \Delta$$

$$\Rightarrow \Delta \text{ is not consistent}$$

Example (2 - Ownership Transfer)

$$\texttt{Let } H = ![\alpha];\texttt{end}$$

$\{h : H,\ h' : \overline{H},\ k :![H];\texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{k :?[H];\texttt{end},\ w : \texttt{end}\}$	$j :?[\overline{H}];\texttt{end}\}$
$k!h$	$k?(x)$	$j?(y)$
$\{h' : \overline{H},\ k : \texttt{end},\ j :![\overline{H}];\texttt{end}\}$	$\{x : H,\ k : \texttt{end},\ w : \texttt{end}\}$	$\{y : \overline{H},\ j : \texttt{end}\}$
$j!h'$	$x!w$	$y?(z)$
$\{k : \texttt{end},\ j : \texttt{end}\}$	$\{k : \texttt{end},\ x : \texttt{end}\}$	$\{j : \texttt{end},\ y : \texttt{end},\ z : \texttt{end}\}$
Process 1	*Process* 2	*Process* 3

Outline

Basic Concurrent Separation Logic

A preordered commutative monoid of propositions

$$(\mathit{Props}, \vdash, *, \mathit{emp})$$

A set of commands (*Com*)

Equipped with total binary operations $c \parallel c'$ and $c; c'$ with $\mathit{skip} \in \mathit{Com}$

Proof Rules for BCSL

$$[\text{Skip}]\ \frac{}{\{X\}\ \texttt{skip}\ \{X\}} \qquad [\text{Frame}]\ \frac{\{X\}\ c\ \{Y\}}{\{X * F\}\ c\ \{Y * F\}}$$

$$[\text{Seq}]\ \frac{\{X\}\ c_1\ \{Y\} \quad \{Y\}\ c_2\ \{Z\}}{\{X\}\ c_1 ; c_2\ \{Z\}} \qquad [\text{Par}]\ \frac{\{X_1\}\ c_1\ \{Y_1\} \quad \{X_2\}\ c_2\ \{Y_2\}}{\{X_1 * X_2\}\ c_1 \parallel c_2\ \{Y_1 * Y_2\}}$$

$$[\text{Consequence}]\ \frac{X' \vdash X \quad \{X\}\ c\ \{Y\} \quad Y \vdash Y'}{\{X'\}\ c\ \{Y'\}}$$

Parallel Rules

$$[\text{BST}]\quad \frac{P_1 \triangleright \Delta_1 \quad P_2 \triangleright \Delta_2}{P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2}$$

$$[\text{BCSL}]\quad \frac{\{X_1\}\, c_1\, \{Y_1\} \quad \{X_2\}\, c_2\, \{Y_2\}}{\{X_1 * X_2\}\, c_1 \parallel c_2\, \{Y_1 * Y_2\}}$$

Heap Model Instantiation

Structure of propositions

$$(\mathit{Props}, \vdash, *, \mathit{emp}) = (P(\mathit{Heaps}), \subseteq, *, \{u\})$$

- Heaps: $\mathbb{N} \rightharpoonup_f \mathbb{N}$
- $P(\mathit{Heaps})$: Powerset
- u: Empty partial function.
- $X * Y = \{h_X \bullet h_Y \mid h_X \in X \wedge h_Y \in Y \wedge h_X \bullet h_Y \downarrow\}$ where $h \bullet h'$ denotes the union of disjoint heap.

Heap Model Instantiation

Structure of propositions

$$(\mathit{Props}, \vdash, *, \mathit{emp}) = (P(\mathit{Heaps}), \subseteq, *, \{u\})$$

- Heaps: $\mathbb{N} \rightharpoonup_f \mathbb{N}$
- $P(\mathit{Heaps})$: Powerset
- u: Empty partial function.
- $X * Y = \{h_X \bullet h_Y \mid h_X \in X \wedge h_Y \in Y \wedge h_X \bullet h_Y \downarrow\}$ where $h \bullet h'$ denotes the union of disjoint heap.

Mutation statement $[n] := m$ where $m, n \in \mathbb{N}$.

Heap Model Instantiation

Structure of propositions

$$(\mathit{Props}, \vdash, *, \mathit{emp}) = (P(\mathit{Heaps}), \subseteq, *, \{u\})$$

- Heaps: $\mathbb{N} \rightharpoonup_f \mathbb{N}$
- $P(\mathit{Heaps})$: Powerset
- u: Empty partial function.
- $X * Y = \{h_X \bullet h_Y \mid h_X \in X \wedge h_Y \in Y \wedge h_X \bullet h_Y \downarrow\}$ where $h \bullet h'$ denotes the union of disjoint heap.

Mutation statement $[n] := m$ where $m, n \in \mathbb{N}$.

$$\frac{}{\{n \mapsto -\}[n] := m\{n \mapsto m\}}$$

Examples

Example (1 - Racey programs)

$$[10] := 23 \parallel [10] := 44$$

Examples

Example (1 - Racey programs)

$$[10] := 23 \parallel [10] := 44$$

$$10 \mapsto - \ * \ 10 \mapsto - \text{ is } \textit{false}$$

Examples

Example (1 - Racey programs)

$$[10] := 23 \parallel [10] := 44$$

$$10 \mapsto - \ * \ 10 \mapsto -$$ is *false*

Example (2 - Ownership Transfer via Shared Buffer)

$$\begin{array}{c}
\{emp\} \\
\{emp * emp\} \\
\begin{array}{c|c}
\{emp\} & \{emp\} \\
x := \texttt{cons(a,b)}; & \mathit{getWhenFull}(y); \\
\{x \mapsto -, -\} & \{y \mapsto -, -\} \\
\mathit{putWhenEmpty}(x); & \mathit{use}(y); \\
\{emp\} & \{y \mapsto -, -\} \\
 & \mathit{dispose}(y) \\
\{emp\} & \{emp\}
\end{array} \\
\{emp * emp\} \\
\{emp\}
\end{array}$$

Outline

Algebra for Concurrency (Hoare et al 2009)

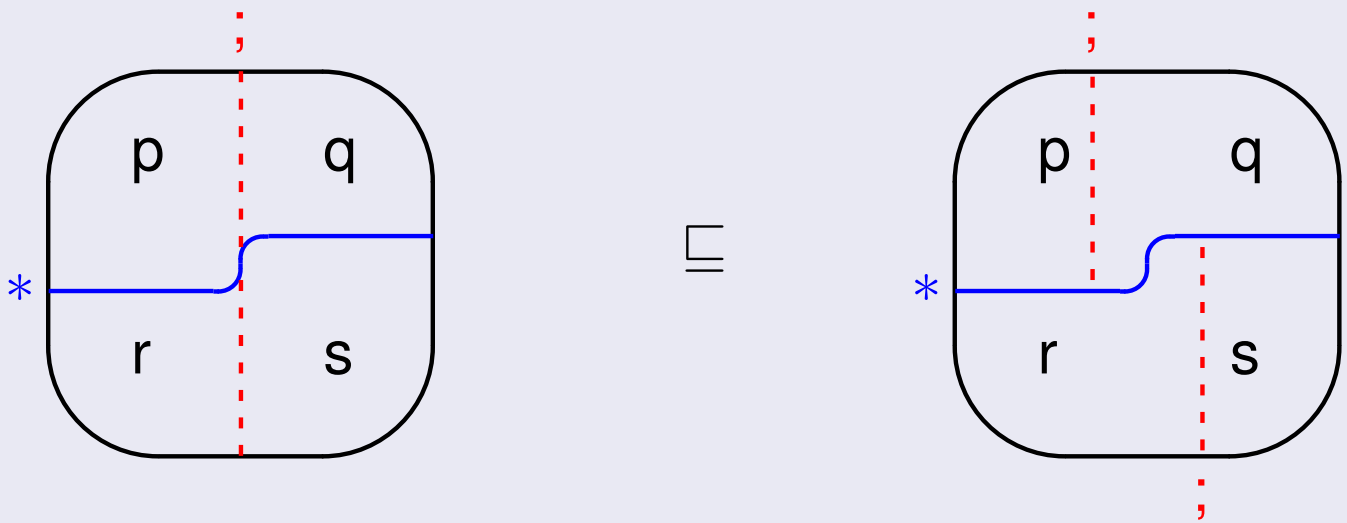
- Two ordered monoids $(S, \sqsubseteq, *, u)$ and $(S, \sqsubseteq, ;, \mathit{skip})$ representing parallel and sequential composition, where $*, ;$ are montone and $*$ is commutative.
- Parallel and Sequencing are related by the Exchange Law

$$(p * r); (q * s) \sqsubseteq (p; q) * (r; s) \quad p, q, r, s \in S$$

Algebra for Concurrency (Hoare et al 2009)

- Two ordered monoids $(S, \sqsubseteq, *, u)$ and $(S, \sqsubseteq, ;, skip)$ representing parallel and sequential composition, where $*, ;$ are montone and $*$ is commutative.
- Parallel and Sequencing are related by the Exchange Law

$$(p * r); (q * s) \sqsubseteq (p; q) * (r; s) \quad p, q, r, s \in S$$



Exchange Law

Validates Plotkin Triple (to come)

- Concurrency Rule
- Frame Rule (when $P * \mathit{skip} = P$)

Exchange Law

Validates Plotkin Triple (to come)

- Concurrency Rule
- Frame Rule (when $P * skip = P$)

$\{P\}\, C\, \{Q\} \Leftrightarrow P \sqsupseteq C; Q$

Proof:

$$
\begin{array}{lll}
 & P \sqsupseteq C; Q \ \wedge \ P' \sqsupseteq C'; Q' & \\
\Rightarrow & P * P' \sqsupseteq (C; Q) * (C'; Q') & \texttt{monotonicity of} * \\
\Rightarrow & P * P' \sqsupseteq (C * C'); (Q * Q') & \texttt{exchange Law}
\end{array}
$$

■

Outline

Structure

$(Props, \vdash, *, emp)$

- *Props* to be the set of session typing contexts Δ
- $\Delta * \Delta'$ to be $\Delta \circ \Delta'$
- *emp* is the empty context $\emptyset$
- $X \vdash Y$ where $X \vdash Y$ *iff* X is inconsistent or $\exists \Phi.\ X = Y \circ \Phi$

Structure

$(Props, \vdash, *, emp)$

- *Props* to be the set of session typing contexts Δ
- $\Delta * \Delta'$ to be $\Delta \circ \Delta'$
- *emp* is the empty context $\emptyset$
- $X \vdash Y$ where $X \vdash Y$ *iff* X is inconsistent or $\exists \Phi.\ X = Y \circ \Phi$

Commands

$$C ::= k?j.C \mid k!j \mid C \parallel C \mid C; C \mid \texttt{skip}$$

Specialised Rules for Session Instantiation

$$[\text{Send}] \quad \frac{}{\{k : ![\alpha]; \beta * j : \alpha\}\ k!j\ \{k : \beta\}}$$

$$[\text{Receive}] \quad \frac{\{A * k : \beta * j : \alpha\}\ P\ \{B\}}{\{A * k : ?[\alpha]; \beta\}\ k?j.P\ \{B\}}$$

Outline

Translation

BST to BSCL

$$
\begin{aligned}
\langle\!\langle \texttt{inact} \rangle\!\rangle &= \texttt{skip} \\
\langle\!\langle P \parallel Q \rangle\!\rangle &= \langle\!\langle P \rangle\!\rangle \parallel \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle k?j.P \rangle\!\rangle &= k?j.\langle\!\langle P \rangle\!\rangle \\
\langle\!\langle k!j.\, P \rangle\!\rangle &= (k!j)\,;\, \langle\!\langle P \rangle\!\rangle
\end{aligned}
$$

Translation

BST to BSCL

$$
\begin{aligned}
\langle\!\langle \texttt{inact} \rangle\!\rangle &= \texttt{skip} \\
\langle\!\langle P \parallel Q \rangle\!\rangle &= \langle\!\langle P \rangle\!\rangle \parallel \langle\!\langle Q \rangle\!\rangle \\
\langle\!\langle k?j.P \rangle\!\rangle &= k?j.\langle\!\langle P \rangle\!\rangle \\
\langle\!\langle k!j.\, P \rangle\!\rangle &= (k!j)\ ;\ \langle\!\langle P \rangle\!\rangle
\end{aligned}
$$

Theorem 1 - Soundness & Completeness

$P \triangleright \Delta$ is provable in **BST** if and only if $\{\Delta\}\ \langle\!\langle P \rangle\!\rangle\ \{emp\}$ is provable in **BCSL/ST**

Outline

Structure

Propositions

Suppose we have an ordered total commutative monoid $(\mathit{Props}, \vdash, *, \mathit{emp})$ with a least element $\bot$

Structure

Propositions

Suppose we have an ordered total commutative monoid $(\mathit{Props}, \vdash, *, \mathit{emp})$ with a least element $\bot$

Predicates

- Model built from predicate transformers on non-empty down-wards closed subsets of *Props* (*Preds*).
- $(\mathit{Preds}, \subseteq)$ has a total commutative monoid structure $(\mathit{Preds}, \subseteq, \otimes, I)$

$$\begin{array}{lcl} X \otimes Y & = & \{p \mid p \vdash x * y \wedge x \in X \wedge y \in Y\} \\ I & = & \{p \mid p \vdash \mathit{emp}\} \end{array}$$

Structure

Commands

Montone functions space $\mathit{Preds} \to \mathit{Preds}$

$$
\begin{aligned}
(F \parallel G)X &= \bigcup\{FX_1 \otimes GX_2 \mid X_1 \otimes X_2 \subseteq X\} \\
\texttt{nothing}X &= \texttt{if } X \supseteq I \texttt{ then } I \texttt{ else false} \\
(F; G)X &= F(G(X)) \\
\texttt{skip}X &= X
\end{aligned}
$$

$X \in \mathit{Preds}$

Structure

Commands

Montone functions space $\mathit{Preds} \to \mathit{Preds}$

$$
\begin{aligned}
(F \parallel G)X &= \bigcup\{FX_1 \otimes GX_2 \mid X_1 \otimes X_2 \subseteq X\} \\
\texttt{nothing}X &= \texttt{if } X \supseteq I \texttt{ then } I \texttt{ else } \mathsf{false} \\
(F; G)X &= F(G(X)) \\
\texttt{skip}X &= X
\end{aligned}
$$

$X \in \mathit{Preds}$

Order

$$F \sqsubseteq G \iff \forall X.\, FX \supseteq GX.$$

Algebraic Structure

Monoids

$(\mathit{Preds}, \sqsubseteq, \|, \texttt{nothing})$ and $(\mathit{Preds}, \sqsubseteq, ;, \mathit{skip})$ form monoids where $\|, ;$ are monotone and $\|$ is commutative.

Algebraic Structure

Monoids

$(\mathit{Preds}, \sqsubseteq, \|, \texttt{nothing})$ and $(\mathit{Preds}, \sqsubseteq, ;, \mathit{skip})$ form monoids where $\|, ;$ are monotone and $\|$ is commutative.

Exchange Law

The predicates transformers satisfy

$$(F_1 \parallel F_2); (G_1 \parallel G_2) \sqsubseteq (F_1; G_1) \parallel (F_2; G_2)$$

Outline

Futuristic pre/post spec in Algebra

Plotkin Triple

$$\{P\}\, C\, \{Q\} \quad \Longleftrightarrow \quad P \sqsupseteq C; Q$$

Futuristic pre/post spec in Algebra

Plotkin Triple

$$\{P\}\, C\, \{Q\} \quad \Longleftrightarrow \quad P \sqsupseteq C; Q$$

Session Types

$$\{\Delta\}\, C\, \{\Delta'\} \quad \Longleftrightarrow \quad [\![\Delta]\!] \sqsupseteq [\![C]\!]; [\![\Delta']\!]$$

Futuristic pre/post spec in Algebra

Plotkin Triple

$$\{P\}\, C\, \{Q\} \quad \Longleftrightarrow \quad P \sqsupseteq C; Q$$

Session Types

$$\{\Delta\}\, C\, \{\Delta'\} \quad \Longleftrightarrow \quad [\![\Delta]\!] \sqsupseteq [\![C]\!]; [\![\Delta']\!]$$

Predicate Transformer Dijkstra Triple

$$\langle Y \rangle F \langle Z \rangle \quad \Longleftrightarrow \quad Y \subseteq FZ$$

Futuristic pre/post spec in Algebra

Plotkin Triple

$$\{P\}\, C\, \{Q\} \quad \Longleftrightarrow \quad P \sqsupseteq C; Q$$

Session Types

$$\{\Delta\}\, C\, \{\Delta'\} \quad \Longleftrightarrow \quad [\![\Delta]\!] \sqsupseteq [\![C]\!]; [\![\Delta']\!]$$

Predicate Transformer Dijkstra Triple

$$\langle Y \rangle F \langle Z \rangle \quad \Longleftrightarrow \quad Y \subseteq FZ$$

Predicate Transformer Plotkin Triple

$$trans[Y] \sqsupseteq F; trans[Z]$$

Dijkstra & Plotkin Triples

Theorem 2: Predicate Transformers and Proof Theory Agree

Assuming that a local and monotone predicate transformer $[\![c_{prim}]\!]$ is given for a collection of primitive commands, then

$$p \in [\![c]\!]X \iff \exists q \in X.\ \{p\}\, c\, \{q\}$$

holds for all c, as long as it holds for primitive commands.

Dijkstra & Plotkin Triples

Theorem 2: Predicate Transformers and Proof Theory Agree

Assuming that a local and monotone predicate transformer $[\![c_{prim}]\!]$ is given for a collection of primitive commands, then

$$p \in [\![c]\!]X \iff \exists q \in X.\ \{p\}\, c\, \{q\}$$

holds for all c, as long as it holds for primitive commands.

Theorem 3: Predicate Transformers and Algebra Agree

For all $Y, Z \in \mathit{Preds}$ and monotone $F : \mathit{Preds} \to \mathit{Preds}$,

$$Y \subseteq FZ \iff \mathit{trans}[Y] \sqsupseteq F; \mathit{trans}[Z]$$

Modularity

Session Types

- Process Calculi
- Message Passing

Concurrent Separation Logic

- Imperative Programs
- Shared Memory

Algebra

Modularity

Session Types

- Process Calculi
- Message Passing

Concurrent Separation Logic

- Imperative Programs
- Shared Memory

Algebra

Questions...

www.eecs.qmul.ac.uk/∼akbar/OnSeparationSessionTypesAlgebra.pdf

Exchange Law Proof

$$(F_1 \parallel F_2);(G_1 \parallel G_2) \sqsubseteq (F_1;G_1) \parallel (F_2;G_2)$$

The validity of the exchange law can be seen from the following calculation.

$$\begin{array}{rl} & ((F_1 \parallel F_2);(G_1 \parallel G_2))X \\ = & \bigcup\{F_1 Y_1 \otimes F_2 Y_2 \mid Y_1 \otimes Y_2 \subseteq (G_1 \parallel G_2)X\} \\ = & \bigcup\{F_1 Y_1 \otimes F_2 Y_2 \mid Y_1 \otimes Y_2 \subseteq \bigcup\{G_1 X_1 \otimes G_2 X_2 \mid X_1 \otimes X_2 \subseteq X\}\} \\ \supseteq & \bigcup\{F_1(G_1 X_1) \otimes F_2(G_2 X_2) \mid X_1 \otimes X_2 \subseteq X\} \\ = & \bigcup\{(F_1;G_1)X_1 \otimes (F_2;G_2)X_2 \mid X_1 \otimes X_2 \subseteq X\} \\ = & ((F_1;G_1) \parallel (F_2;G_2))X \end{array}$$

In the $\supseteq$ step we take $Y_1 = G_1 X_1$, $Y_2 = G_2 X_2$. This step uses that $X_1 \otimes X_2 \subseteq X \Rightarrow G_1 X_1 \otimes G_2 X_2 \subseteq \bigcup\{G_1 X_1 \otimes G_2 X_2 \mid X_1 \otimes X_2 \subseteq X\}$.

Predicate Converter (Trans)

$$\mathit{do\text{-}after}[Y]X = \texttt{if}\ X = \mathit{Props}\ \texttt{then}\ Y\ \texttt{else}\ \mathsf{false}$$