Calculi for Service Oriented Computing

Roberto Bruni

Dipartimento di Informatica
Università di Pisa

SFM-WS 2009
Bertinoro, Italy
June 1–6, 2009

Tales from joint work with:

Michele Boreale, Chiara Bodei, Linda Brodo, Rocco De Nicola, Michele Loreti, Leonardo Mezzina, and several other colleagues

Outline

Service Oriented Computing (SOC)

Services

SOC is an emerging paradigm where services are understood as

- autonomous
- platform-independent

computational entities that can be:

- described
- published
- categorised
- discovered
- assembled

for developing massively distributed, interoperable, evolvable systems.

e-Expectations

Big companies put many efforts in promoting service delivery on a variety of computing platforms.

Tomorrow, there will be a plethora of new services for e-government, e-business, and e-health, and others within the rapidly evolving Information Society.

A crucial fact

Industrial consortia are developing orchestration and choreography languages, targeting the standardisation of Web Services and XML-centric technologies, but *they lack neat semantic foundations.*

Service Oriented Computing (SOC)

Services

SOC is an emerging paradigm where services are understood as

- autonomous
- platform-independent

computational entities that can be:

- described
- published
- categorised
- discovered
- assembled

for developing massively distributed, interoperable, evolvable systems.

e-Expectations

Big companies put many efforts in promoting service delivery on a variety of computing platforms.
Tomorrow, there will be a plethora of new services for e-government, e-business, and e-health, and others within the rapidly evolving Information Society.

A crucial fact

Industrial consortia are developing orchestration and choreography languages, targeting the standardisation of Web Services and XML-centric technologies, but *they lack neat semantic foundations*.

From WSDL to BPEL

Service descriptions

- Machine-processable interface
- WSDL: mere syntax + details
- Behavioural information is needed for sound interaction
- BPEL: structured workflow + links

The problem with BPEL

- One “standard semantics”: informal, textual description
- Many semantics: dozen of papers, usually dealing with BPEL fragments
- No semantics: no comparison between different formal models + ambiguity in available BPEL engines
- What is BPEL especially designed for?

From WSDL to BPEL

Service descriptions

- Machine-processable interface
- WSDL: mere syntax + details
- Behavioural information is needed for sound interaction
- BPEL: structured workflow + links

The problem with BPEL

- One “standard semantics”: informal, textual description
- Many semantics: dozen of papers, usually dealing with BPEL fragments
- No semantics: no comparison between different formal models + ambiguity in available BPEL engines
- What is BPEL especially designed for?

A Citation

From ACM Turing Award Winner Robin Milner

- In **Natural Sciences** concepts arise from the urge to understand observed phenomena
- In **Computer Science** concepts arise as distillations of our design of systems

Natural Sciences		Computer Science
Biology	Organisms	Databases, Networks
Chemistry	Molecules	Metaphors of programming
Physics	Particles	Primitives of programming

- One possibility: understand BPEL
- Another possibility: develop alternative metaphors, well-behaving by design

A Citation

From ACM Turing Award Winner Robin Milner

- In **Natural Sciences** concepts arise from the urge to understand observed phenomena
- In **Computer Science** concepts arise as distillations of our design of systems

Natural Sciences		Computer Science
Biology	Organisms	Databases, Networks
Chemistry	Molecules	Metaphors of programming
Physics	Particles	Primitives of programming

- One possibility: understand BPEL
- Another possibility: develop alternative metaphors, well-behaving by design

A Citation

From ACM Turing Award Winner Robin Milner

- In **Natural Sciences** concepts arise from the urge to understand observed phenomena
- In **Computer Science** concepts arise as distillations of our design of systems

Natural Sciences		Computer Science
Biology	Organisms	Databases, Networks
Chemistry	Molecules	Metaphors of programming
Physics	Particles	Primitives of programming

- One possibility: understand BPEL
- Another possibility: develop alternative metaphors, well-behaving by design

Sensoria (http://www.sensoria-ist.eu)

IST-FET Integrated Project funded by the EU in the GC Initiative (6th FP).

Aim

Developing a novel, comprehensive approach to the engineering of software systems for service-oriented overlay computers.

Strategy

Integration of foundational theories, techniques, methods and tools in a pragmatic software engineering approach.

The Role of Process Calculi

Coordinating and combining services

A crucial role in the project is played by formalisms for service description that can lay the mathematical basis for analysing and experimenting with components interactions, and for combining services.

Sensoria workpackage 2

We seek for a small set of primitives that might serve as a basis for formalising and programming service oriented applications over global computers.

Sensoria core calculi

- *Signal Calculus*: middleware level
- *SOCK, COWS*: service level, correlation-based
- *SCC-family (SCC, SSCC, CC,* ***CaSPiS****)*: service level, session-based
- *cc-pi, lambda-req*: SLA contract level

Service Features

Some distinguishing aspects

- Loose coupling and openness: services are developed separately
- Dynamicity: services are discovered and put together
- Stateless: long-running conversation must be tracked (correlation sets, sessions)
- Prevent misuses and locate flaws: interaction soundness should be checkable at discovery time, before binding (e.g. type safety, absence of deadlocks, client progress)
- Scalable techniques: concurrency and interaction must be inevitably addressed, causing combinatorial explosion in the analysis

Formal approaches

- Ontologies (semantic web)
- Logic-based (SRML)
- Workflow models (e.g. automata, Petri nets)
- Process calculi (abstract equivalences, type systems)

Service Features

Some distinguishing aspects

- Loose coupling and openness: services are developed separately
- Dynamicity: services are discovered and put together
- Stateless: long-running conversation must be tracked (correlation sets, sessions)
- Prevent misuses and locate flaws: interaction soundness should be checkable at discovery time, before binding (e.g. type safety, absence of deadlocks, client progress)
- Scalable techniques: concurrency and interaction must be inevitably addressed, causing combinatorial explosion in the analysis

Formal approaches

- Ontologies (semantic web)
- Logic-based (SRML)
- Workflow models (e.g. automata, Petri nets)
- Process calculi (abstract equivalences, type systems)

Process Calculi Approach

Find the right level of abstraction

Need to balance between:

- tractability (not by humans, by the machine)
- understandability (by humans)
- scalability
- flexibility
- expressiveness
- usability
- disciplined structuring

Can be used for

- Specification
- Prototyping
- Description

This Talk

Genesis of CaSPiS

- concurrent systems are difficult to handle
- interaction (CCS)
- passing references (π-calculus)
- handling sessions
- cancelling activities (Orc)
- summing up (CaSPiS)

On the side

- get used to process calculi
- labelled transition systems vs reduction
- play with simple puzzles
- type systems

Outline

Concurrency

- A sequential program has a single thread of control.
- A concurrent program has multiple threads of control (it may perform multiple computations in parallel and may control multiple external activities which occur at the same time).

Communication

The concurrent threads exchange information via

- indirect communication: the execution of concurrent processes proceeds on one or more processors all of which access a shared memory. Care is required to ensure exclusive access to shared variables
- direct communication: concurrent processes are executed by running them on separate processors, threads communicate by exchanging messages.

A Simple Problem

Let f a (computationally expensive) function from integers to integers.

- A positive zero for f is a positive integer n such that $f(n) = 0$
- A negative zero for f is a negative integer z such that $f(z) = 0$

Our Goal

We want to write a program that terminates if and only if the total function f has a positive or negative zero and proceeds indefinitely otherwise.

A Brilliant Idea

To speed up we decide to run in parallel two programs: one looking for a positive zero and the other for a negative zero

Attempt 1

We write S1 that looks for a positive zero:

```
S1=    found=false; n=0;
       while(!found) { n++; found=(f(n)==0); }
```

By cut-and-paste from S1 we write S2 that looks for a negative zero:

```
S2=    found=false; z=0;
       while(!found) { z--; found=(f(z)==0); }
```

And we run `S1` and `S2` in parallel:

```
S1 || S2
```

Let f have a positive zero and not a negative one.
If `S1` terminates before `S2` starts, the latter sets found to false and looks indefinitely for the nonexisting zero.

Attempt 1

We write S1 that looks for a positive zero:

```
S1=   found=false; n=0;
      while(!found) { n++; found=(f(n)==0); }
```

By cut-and-paste from S1 we write S2 that looks for a negative zero:

```
S2=   found=false; z=0;
      while(!found) { z--; found=(f(z)==0); }
```

And we run `S1` and `S2` in parallel:

```
S1 || S2
```

Let f have a positive zero and not a negative one.
If `S1` terminates before `S2` starts, the latter sets found to false and looks indefinitely for the nonexisting zero.

Attempt 2 (found is initialised only once)

The problem is due to the fact that found is initialised to false twice.

```
                    found=false; (R1 || R2)
where
R1=   n=0; while(!found) { n++; found=(f(n)==0); }

R2=   z=0; while(!found) { z--; found=(f(z)==0); }
```

If f has (again) only a positive zero assume that:

1. R2 is preempted when entering the while body (before z--)
2. R1 runs and finds a (positive) zero
3. R2 gets the CPU back

When R2 restarts it executes the while body and may set found to false. The program then would not terminate because it would look for a non existing negative zero.

Attempt 2 (found is initialised only once)

The problem is due to the fact that found is initialised to false twice.

```
                    found=false; (R1 || R2)
where
R1=   n=0; while(!found) { n++; found=(f(n)==0); }

R2=   z=0; while(!found) { z--; found=(f(z)==0); }
```

If f has (again) only a positive zero assume that:

1. R2 is preempted when entering the while body (before `z--`)
2. R1 runs and finds a (positive) zero
3. R2 gets the CPU back

When R2 restarts it executes the while body and may set found to false. The program then would not terminate because it would look for a non existing negative zero.

Attempt 3 (“unnecessary” assignments are removed)

The problem is due to the fact that found is set to false after it has already been assigned true.

```
                 found=false; (T1 || T2)
where

T1=   n=0; while(!found) { n++; if (f(n)==0) found=true; }

T2=   z=0; while(!found) { z--; if (f(z)==0) found=true; }
```

Let f have only a positive zero.
Assume that T2 gets the CPU to keep it until it terminates. Since this will never happen, T1 will never get the chance to find its zero.

Attempt 3 (“unnecessary” assignments are removed)

The problem is due to the fact that found is set to false after it has already been assigned true.

```
                    found=false; (T1 || T2)
where

T1=   n=0; while(!found) { n++; if (f(n)==0) found=true; }

T2=   z=0; while(!found) { z--; if (f(z)==0) found=true; }
```

Let f have only a positive zero.
Assume that T2 gets the CPU to keep it until it terminates. Since this will never happen, T1 will never get the chance to find its zero.

Attempt 4 (token passing fairness)

The problem is due to non-fair scheduling policies.

```
              turn=1; found=false; (Q1 || Q2)
where
Q1=   n=0; while(!found) {
               wait turn==1 then {
                   turn=2; n++; if (f(n)==0) found=true; } }

Q2=   z=0; while(!found) {
               wait turn==2 then {
                   turn=1; z--; if (f(z)==0) found=true; } }
```

If Q1 finds a zero and stops when Q2 has already set turn to 1, Q2 would be blocked by the wait command because the value of turn cannot be changed.

Attempt 4 (token passing fairness)

The problem is due to non-fair scheduling policies.

```
              turn=1; found=false; (Q1 || Q2)
where
Q1=   n=0; while(!found) {
                wait turn==1 then {
                    turn=2; n++; if (f(n)==0) found=true; } }

Q2=   z=0; while(!found) {
                wait turn==2 then {
                    turn=1; z--; if (f(z)==0) found=true; } }
```

If Q1 finds a zero and stops when Q2 has already set turn to 1, Q2 would be blocked by the wait command because the value of turn cannot be changed.

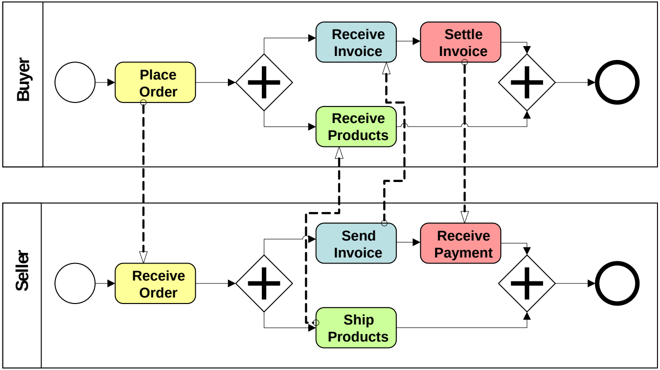
Attempt 5 (pass the token before terminating

The program may not terminate, waiting for an impossible event.

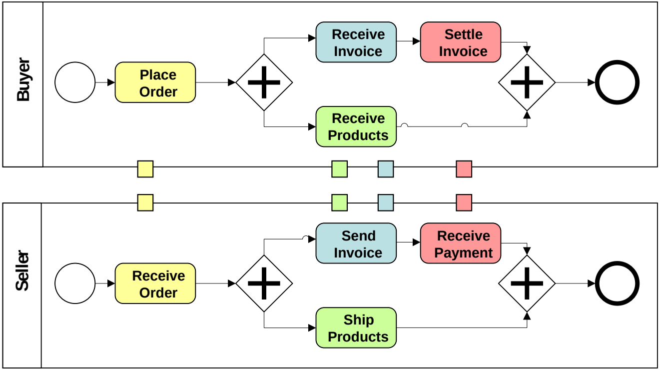
Is it a correct solution?

```
    turn=1; found=false; ( {P1; turn=2;} || {P2; turn=1;} )
where

P1=   n=0; while(!found) {
                wait turn==1 then {
                    turn=2; n++;
                    if (f(n)==0) found=true; } }

P2=   z=0; while(!found) {
                wait turn==2 then {
                    turn=1; z--;
                    if (f(z)==0) found=true; } }
```

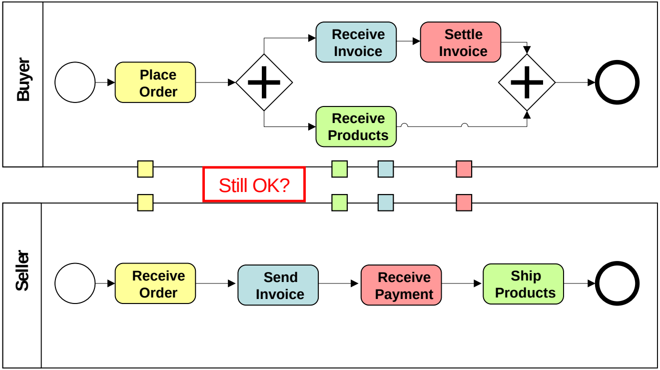
Buyer / Seller Compatibility



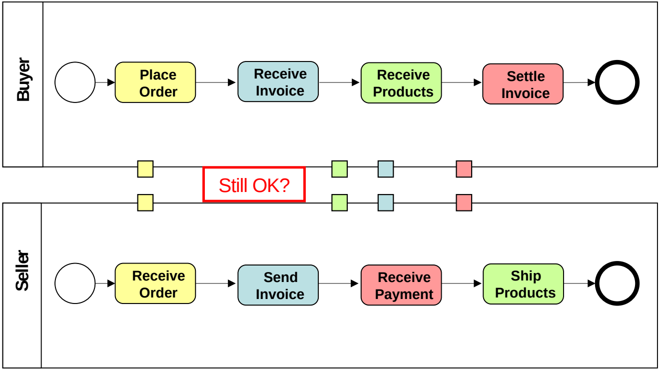
Buyer / Seller Compatibility



Buyer / Seller Compatibility



Buyer / Seller Compatibility



Outline

Activities

Elementary Action

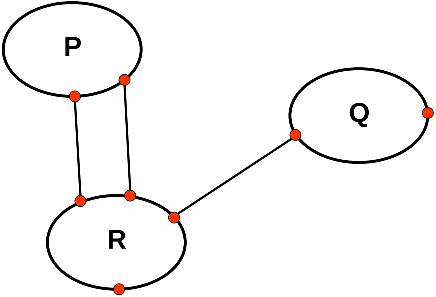
Atomic (i.e., non-interruptable at the given level of granularity) abstract step of a computation that is performed by a system to move from one state to the other

- in ordinary (sequential) models: reading from or writing on some kind of (passive) storage device or invoking a procedure with actual parameters.
- in CCS: sort of handshake between two active, autonomous processes (sending a message and receiving a message, exposing some alternatives and picking one alternative, producing a resource and consuming a resource)

Notation

- Dual actions (co-activities): a and $\overline{a}$, with $\overline{\overline{a}} = a$
- Silent action: τ

CCS View



Syntax

$$
\begin{array}{lclcl}
\lambda & ::= & a & \mid & \overline{a} \\
\alpha & ::= & \lambda & \mid & \tau \\
P & ::= & \sum_{i\in I} \alpha_i.P_i & \mid & P_1 \mid P_2 \quad \mid \quad \ldots
\end{array}
$$

Semantics (SOS style)

$$(act)\ \frac{j \in I}{\sum_{i\in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j}$$

$$(lpar)\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1' \mid P_2} \qquad (rpar)\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1 \mid P_2'}$$

$$(comm)\ \frac{P_1 \xrightarrow{\lambda} P_1' \quad P_2 \xrightarrow{\overline{\lambda}} P_2'}{P_1 | P_2 \xrightarrow{\tau} P_1' \mid P_2'}$$

Calculus of Communicating Systems

Syntax

$$\lambda \quad ::= \quad a \quad | \quad \overline{a}$$

$$\alpha \quad ::= \quad \lambda \quad | \quad \tau$$

$$P \quad ::= \quad \sum_{i\in I} \alpha_i.P_i \quad | \quad P_1 \,|\, P_2 \quad | \quad \ldots$$

Semantics (SOS style)

$$(act)\ \frac{j \in I}{\sum_{i\in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j}$$

$$(lpar)\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \,|\, P_2 \xrightarrow{\alpha} P_1' \,|\, P_2} \qquad (rpar)\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 \,|\, P_2 \xrightarrow{\alpha} P_1 \,|\, P_2'}$$

$$(comm)\ \frac{P_1 \xrightarrow{\lambda} P_1' \quad P_2 \xrightarrow{\overline{\lambda}} P_2'}{P_1 | P_2 \xrightarrow{\tau} P_1' \,|\, P_2'}$$

Calculus of Communicating Systems

Syntax

$$\lambda ::= a \mid \overline{a}$$

$$\alpha ::= \lambda \mid \tau$$

$$P ::= \sum_{i\in I} \alpha_i.P_i \mid P_1 \,|\, P_2 \mid \ldots$$

Semantics (SOS style)

$$(act)\ \frac{j \in I}{\sum_{i\in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j}$$

$$(lpar)\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \,|\, P_2 \xrightarrow{\alpha} P_1' \,|\, P_2} \qquad (rpar)\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 \,|\, P_2 \xrightarrow{\alpha} P_1 \,|\, P_2'}$$

$$(comm)\ \frac{P_1 \xrightarrow{\lambda} P_1' \quad P_2 \xrightarrow{\overline{\lambda}} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1' \,|\, P_2'}$$

CCS: An Example

Notation

The unary sum is written $\alpha.P$; the empty sum is written nil or **0** (inactive process) and the trailing of nil is often omitted.

Buyer and Seller

$$B \triangleq \overline{ord}.(prod \,|\, inv.\overline{pay})$$
$$S \triangleq ord.\overline{inv}.pay.\overline{prod}$$

$$\begin{aligned} B \,|\, S &\xrightarrow{\tau} (prod \,|\, inv.\overline{pay}) \;|\; \overline{inv}.pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \,|\, \overline{pay}) \;|\; pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \,|\, \mathbf{0}) \;|\; \overline{prod} \\ &\xrightarrow{\tau} (\mathbf{0} \,|\, \mathbf{0}) \;|\; \mathbf{0} \end{aligned}$$

CCS: An Example

Notation

The unary sum is written $\alpha.P$; the empty sum is written nil or **0** (inactive process) and the trailing of nil is often omitted.

Buyer and Seller

$$B \triangleq \overline{ord}.(prod \mid inv.\overline{pay})$$
$$S \triangleq ord.\overline{inv}.pay.\overline{prod}$$

$$\begin{aligned} B \mid S &\xrightarrow{\tau} (prod \mid inv.\overline{pay}) \mid \overline{inv}.pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \mid \overline{pay}) \mid pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \mid \mathbf{0}) \mid \overline{prod} \\ &\xrightarrow{\tau} (\mathbf{0} \mid \mathbf{0}) \mid \mathbf{0} \end{aligned}$$

CCS: An Example

Notation

The unary sum is written $\alpha.P$; the empty sum is written nil or **0** (inactive process) and the trailing of nil is often omitted.

Buyer and Seller

$$B \triangleq \overline{ord}.(prod \mid inv.\overline{pay})$$
$$S \triangleq ord.\overline{inv}.pay.\overline{prod}$$

$$\begin{aligned} B \mid S &\xrightarrow{\tau} (prod \mid inv.\overline{pay}) \mid \overline{inv}.pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \mid \overline{pay}) \mid pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \mid \mathbf{0}) \mid \overline{prod} \\ &\xrightarrow{\tau} (\mathbf{0} \mid \mathbf{0}) \mid \mathbf{0} \end{aligned}$$

CCS: An Example

Notation

The unary sum is written $\alpha.P$; the empty sum is written nil or **0** (inactive process) and the trailing of nil is often omitted.

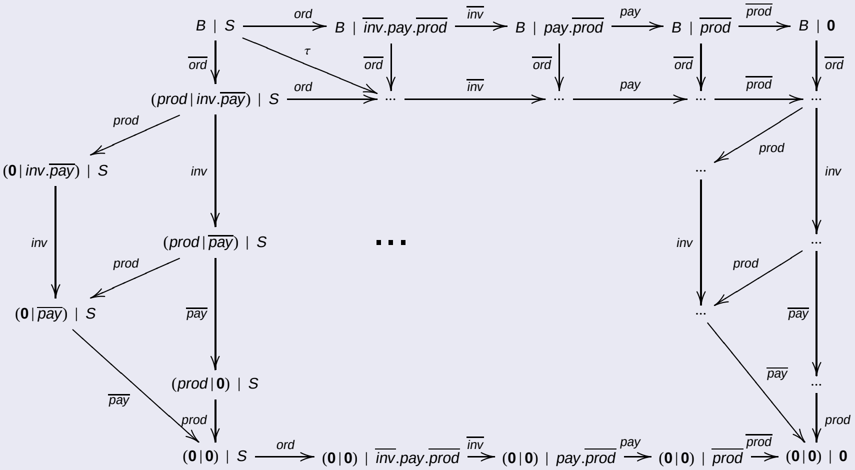
Buyer and Seller

$$B \triangleq \overline{ord}.(prod \,|\, inv.\overline{pay})$$

$$S \triangleq ord.\overline{inv}.pay.\overline{prod}$$

$$\begin{aligned} B \,|\, S &\xrightarrow{\tau} (prod \,|\, inv.\overline{pay}) \;|\; \overline{inv}.pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \,|\, \overline{pay}) \;|\; pay.\overline{prod} \\ &\xrightarrow{\tau} (prod \,|\, \mathbf{0}) \;|\; \overline{prod} \\ &\xrightarrow{\tau} (\mathbf{0} \,|\, \mathbf{0}) \;|\; \mathbf{0} \end{aligned}$$

CCS Processes as LTS



CCS: Restriction

Syntax

$$P \quad ::= \quad \sum_{i \in I} \alpha_i.P_i \quad | \quad P_1 \,|\, P_2 \quad | \quad (\nu a)P \quad | \quad \ldots$$

Semantics (SOS style)

$$(res)\ \frac{P \xrightarrow{\alpha} P' \qquad \alpha \notin \{\, a, \overline{a} \,\}}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}$$

Buyer and Seller: Revisited

$$(\nu ord)(\nu inv)(\nu pay)(\nu prod)(\ B \ |\ S\ )$$

CCS: Recursion 1

Syntax

$$P \quad ::= \quad \sum_{i\in I} \alpha_i.P_i \quad | \quad P_1 \,|\, P_2 \quad | \quad (\nu a)P \quad | \quad X \quad | \quad \mathrm{rec}\, X.\, P \quad | \quad \ldots$$

Semantics (SOS style)

$$(rec)\ \frac{P\{^{\mathrm{rec}\, X.\, P}/_{X}\} \xrightarrow{\alpha} P'}{\mathrm{rec}\, X.\, P \xrightarrow{\alpha} P'}$$

Buyer and Seller: Revisited

$$S' \quad \triangleq \quad \mathrm{rec}\, X.\, ord.\overline{inv}.pay.\overline{prod}.X$$

$$S'' \quad \triangleq \quad \mathrm{rec}\, X.\, (ord.\overline{inv}.pay.\overline{prod} \mid X)$$

$$S''' \quad \triangleq \quad \mathrm{rec}\, X.\, ord.(\overline{inv}.pay.\overline{prod} \mid X)$$

CCS: Recursion 2

Syntax

$$\Delta \quad = \quad \{\, A_d \triangleq P_d \,\}_d$$

$$P \quad ::= \quad \sum_{i\in I} \alpha_i.P_i \quad | \quad P_1 \,|\, P_2 \quad | \quad (\nu a)P \quad | \quad A_d \quad | \quad \ldots$$

Semantics (SOS style)

$$(def)\ \frac{A_d \triangleq P_d \in \Delta \qquad P_d \xrightarrow{\alpha} P'}{A_d \xrightarrow{\alpha} P'}$$

Buyer and Seller: Revisited

$$S_d \quad \triangleq \quad ord.(\overline{inv}.pay.\overline{prod} \ | \ S_d)$$

Syntax

$$P \quad ::= \quad \sum_{i \in I} \alpha_i.P_i \quad | \quad P_1 \,|\, P_2 \quad | \quad (\nu a)P \quad | \quad !P \quad | \quad \ldots$$

Semantics (SOS style, controlled)

$$(rep1)\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \,|\, !P} \qquad (rep2)\ \frac{P \xrightarrow{\lambda} P_1 \quad P \xrightarrow{\overline{\lambda}} P_2}{!P \xrightarrow{\alpha} P_1 \,|\, P_2 \,|\, !P}$$

Buyer and Seller: Revisited

$$S \quad \triangleq \quad !ord.\overline{inv}.pay.\overline{prod}$$

CCS: Structural Congruence

Equivalent Processes

- Do processes P and Q exhibit the same behaviour? (several notions are possible)
- Equivalence Relation: reflexive, symmetric and transitive
- Can we use P and Q interchangeably in any larger context? (several notions are possible)
- Congruence: equivalence preserved by composition
- Is P congruent to Q? (not necessarily decidable)
- Is P (just) an evident rephrasing of Q? (structural congruence)

$$
\begin{array}{lll}
P + \mathbf{0} \equiv P & P_1 + P_2 \equiv P_2 + P_1 & P_1 + (P_2 + P_3) \equiv (P_1 + P_2) + P_3 \\
P + P = P & !P \equiv P \,|\, !P & \\
P \,|\, \mathbf{0} \equiv P & P_1 \,|\, P_2 \equiv P_2 \,|\, P_1 & P_1 \,|\, (P_2 \,|\, P_3) \equiv (P_1 \,|\, P_2) \,|\, P_3 \\
(\nu a)\mathbf{0} \equiv \mathbf{0} & (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P & P \,|\, (\nu a)Q \equiv (\nu a)(P \,|\, Q) \ \text{ if } a \notin act(P)
\end{array}
$$

Answers these questions to proceed

1. Would it be ok to let $!(\nu a)P \equiv (\nu a)!P$?
2. Are the following Buyer and Seller ok?

$$B \triangleq \overline{ord}.inv.prod.\overline{pay}$$
$$S \triangleq\ !ord.\overline{inv}.pay.\overline{prod}$$

3. Are the following Buyer and Seller ok?

$$B \triangleq \overline{ord}.(prod \mid inv.\overline{pay})$$
$$S \triangleq\ !ord.(\overline{prod} \mid \overline{inv}.pay)$$

4. How would you encode sequential composition $P; Q$?

Outline

Extending CCS 1

Value passing

Output actions can send data and input actions carry formal parameters to be substituted with actual parameters when handshaking.

A problematic server

Let f involve some heavy scientific calculation.

$$S \triangleq !in(x).\overline{out}\langle f(x)\rangle \qquad C \triangleq \overline{in}\langle n\rangle.out(y).P$$

Some problem may arise if two or more clients are around:

$$
\begin{aligned}
& S \mid \overline{in}\langle 1\rangle.out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\
& \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\
& \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid \overline{out}\langle f(2)\rangle \mid out(y_1).P_1 \mid out(y_2).P_2 \\
& \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid P_1\{^{f(2)}/_{y_1}\} \mid out(y_2).P_2
\end{aligned}
$$

Extending CCS 1

Value passing

Output actions can send data and input actions carry formal parameters to be substituted with actual parameters when handshaking.

A problematic server

Let f involve some heavy scientific calculation.

$$S \triangleq !in(x).\overline{out}\langle f(x)\rangle \qquad C \triangleq \overline{in}\langle n\rangle.out(y).P$$

Some problem may arise if two or more clients are around:

$$\begin{array}{l} S \mid \overline{in}\langle 1\rangle.out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid \overline{out}\langle f(2)\rangle \mid out(y_1).P_1 \mid out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid P_1\{^{f(2)}/_{y_1}\} \mid out(y_2).P_2 \end{array}$$

Extending CCS 1

Value passing

Output actions can send data and input actions carry formal parameters to be substituted with actual parameters when handshaking.

A problematic server

Let f involve some heavy scientific calculation.

$$S \triangleq !in(x).\overline{out}\langle f(x)\rangle \qquad C \triangleq \overline{in}\langle n\rangle.out(y).P$$

Some problem may arise if two or more clients are around:

$$\begin{array}{l} S \mid \overline{in}\langle 1\rangle.out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid \overline{out}\langle f(2)\rangle \mid out(y_1).P_1 \mid out(y_2).P_2 \\ \quad \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid P_1\{^{f(2)}/_{y_1}\} \mid out(y_2).P_2 \end{array}$$

Extending CCS 1

Value passing

Output actions can send data and input actions carry formal parameters to be substituted with actual parameters when handshaking.

A problematic server

Let f involve some heavy scientific calculation.

$$S \triangleq !in(x).\overline{out}\langle f(x)\rangle \qquad C \triangleq \overline{in}\langle n\rangle.out(y).P$$

Some problem may arise if two or more clients are around:

$$\begin{aligned} S \mid \overline{in}\langle 1\rangle.out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid out(y_1).P_1 \mid \overline{in}\langle 2\rangle.out(y_2).P_2 \\ \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid \overline{out}\langle f(2)\rangle \mid out(y_1).P_1 \mid out(y_2).P_2 \\ \xrightarrow{\tau} S \mid \overline{out}\langle f(1)\rangle \mid P_1\{^{f(2)}/_{y_1}\} \mid out(y_2).P_2 \end{aligned}$$

Extending CCS 2

Name mobility

Ability to send and receive references to channels.

A proper server (and client)

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

$$S \triangleq !in(x,k).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle n,k\rangle.k(y).P$$

Each client gets a separate reply:

$$\begin{aligned}
& S \mid (\nu k_1)\overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid (\nu k_2)\overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2 \\
&\equiv (\nu k_1)(\nu k_2)(S \mid \overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) \\
&\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) \\
&\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid \overline{k_2}\langle f(2)\rangle \mid k_1(y_1).P_1 \mid k_2(y_2).P_2) \\
&\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_2}\langle f(2)\rangle \mid P_1\{^{f(1)}/_{y_1}\} \mid k_2(y_2).P_2) \\
&\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid P_1\{^{f(1)}/_{y_1}\} \mid P_2\{^{f(2)}/_{y_2}\})
\end{aligned}$$

Extending CCS 2

Name mobility

Ability to send and receive references to channels.

A proper server (and client)

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

$$S \triangleq !in(x,k).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle n,k\rangle.k(y).P$$

Each client gets a separate reply:

$$
\begin{aligned}
& S \mid (\nu k_1)\overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid (\nu k_2)\overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2 \\
& \quad \equiv (\nu k_1)(\nu k_2)(S \mid \overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) \\
& \quad \xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) \\
& \quad \xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid \overline{k_2}\langle f(2)\rangle \mid k_1(y_1).P_1 \mid k_2(y_2).P_2) \\
& \quad \xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_2}\langle f(2)\rangle \mid P_1\{^{f(1)}/_{y_1}\} \mid k_2(y_2).P_2) \\
& \quad \xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid P_1\{^{f(1)}/_{y_1}\} \mid P_2\{^{f(2)}/_{y_2}\}
\end{aligned}
$$

Extending CCS 2

Name mobility

Ability to send and receive references to channels.

A proper server (and client)

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

$$S \triangleq !in(x,k).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle n,k\rangle.k(y).P$$

Each client gets a separate reply:

$$\begin{aligned}
S \mid (\nu k_1)\overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid (\nu k_2)\overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2 & \\
\equiv (\nu k_1)(\nu k_2)(S \mid \overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid \overline{k_2}\langle f(2)\rangle \mid k_1(y_1).P_1 \mid k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_2}\langle f(2)\rangle \mid P_1\{^{f(1)}/_{y_1}\} \mid k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid P_1\{^{f(1)}/_{y_1}\} \mid P_2\{^{f(2)}/_{y_2}\} &
\end{aligned}$$

Extending CCS 2

Name mobility

Ability to send and receive references to channels.

A proper server (and client)

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

$$S \triangleq !in(x,k).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle n,k\rangle.k(y).P$$

Each client gets a separate reply:

$$\begin{aligned}
S \mid (\nu k_1)\overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid (\nu k_2)\overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2 & \\
\equiv (\nu k_1)(\nu k_2)(S \mid \overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid \overline{k_2}\langle f(2)\rangle \mid k_1(y_1).P_1 \mid k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_2}\langle f(2)\rangle \mid P_1\{^{f(1)}/_{y_1}\} \mid k_2(y_2).P_2) & \\
\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid P_1\{^{f(1)}/_{y_1}\} \mid P_2\{^{f(2)}/_{y_2}\} &
\end{aligned}$$

Extending CCS 2

Name mobility

Ability to send and receive references to channels.

A proper server (and client)

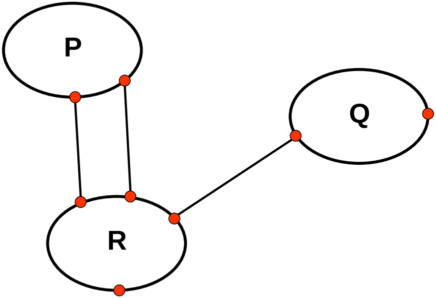
$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

$$S \triangleq !in(x,k).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)\overline{in}\langle n,k\rangle.k(y).P$$

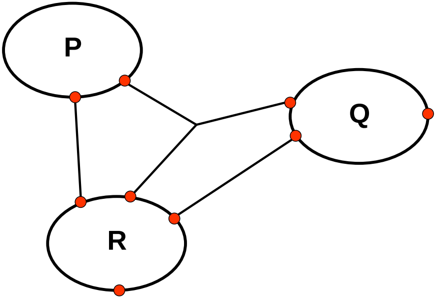
Each client gets a separate reply:

$$\begin{aligned}
&S \mid (\nu k_1)\overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid (\nu k_2)\overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2\\
&\quad\equiv (\nu k_1)(\nu k_2)(S \mid \overline{in}\langle 1,k_1\rangle.k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2)\\
&\quad\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid k_1(y_1).P_1 \mid \overline{in}\langle 2,k_2\rangle.k_2(y_2).P_2)\\
&\quad\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_1}\langle f(1)\rangle \mid \overline{k_2}\langle f(2)\rangle \mid k_1(y_1).P_1 \mid k_2(y_2).P_2)\\
&\quad\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid \overline{k_2}\langle f(2)\rangle \mid P_1\{^{f(1)}/_{y_1}\} \mid k_2(y_2).P_2)\\
&\quad\xrightarrow{\tau} (\nu k_1)(\nu k_2)(S \mid P_1\{^{f(1)}/_{y_1}\} \mid P_2\{^{f(2)}/_{y_2}\}
\end{aligned}$$

π-calculus View



π-calculus View



About Links

The π-calculus has two basic entities

1. *processes* (interacting through links)
2. *names* of links

What is a link?

π-calculus is not prescriptive on this point.

1. Hypertext links can be created, passed around, disappear.
2. Connections between cellular telephones and network bases.
3. Memory can be allocated and de-allocated, with references passed as parameters in method invocations.

Roughly, a link is determined by the sharing of names.
Action prefixes can be executed to change system connectivity over time.

About Links

The π-calculus has two basic entities

1. *processes* (interacting through links)
2. *names* of links

What is a link?

π-calculus is not prescriptive on this point.

1. Hypertext links can be created, passed around, disappear.
2. Connections between cellular telephones and network bases.
3. Memory can be allocated and de-allocated, with references passed as parameters in method invocations.

Roughly, a link is determined by the sharing of names.
Action prefixes can be executed to change system connectivity over time.

About Links

The π-calculus has two basic entities

1. *processes* (interacting through links)
2. *names* of links

What is a link?

π-calculus is not prescriptive on this point.

1. Hypertext links can be created, passed around, disappear.
2. Connections between cellular telephones and network bases.
3. Memory can be allocated and de-allocated, with references passed as parameters in method invocations.

Roughly, a link is determined by the sharing of names.
Action prefixes can be executed to change system connectivity over time.

About names

Names can be:

1. channels
2. identifiers
3. values (data)
4. objects
5. pointers
6. references
7. locations
8. encryption keys
9. ...

Names can:

1. be created and destroyed
2. sent them around to share information
3. acquired to communicate with previously unknown processes
4. used for evaluation or communication
5. be tested to take decisions based on their values
6. used as private means of communication, e.g. to share secret
7. ...

About names

Names can be:

1. channels
2. identifiers
3. values (data)
4. objects
5. pointers
6. references
7. locations
8. encryption keys
9. ...

Names can:

1. be created and destroyed
2. sent them around to share information
3. acquired to communicate with previously unknown processes
4. used for evaluation or communication
5. be tested to take decisions based on their values
6. used as private means of communication, e.g. to share secret
7. ...

π-calculus: Syntax

(Processes) P	$::=$	S	sum
	$\mid$	$P_1 \mid P_2$	parallel composition
	$\mid$	$(\nu x)P$	name restriction
	$\mid$	$!P$	replication
(Sums) S	$::=$	$\mathbf{0}$	inactive process (nil)
	$\mid$	$\pi.P$	prefix
	$\mid$	$S_1 + S_2$	choice
(Prefixes) π	$::=$	$\overline{x}\langle y \rangle$	sends y on x
	$\mid$	$x(z)$	substitutes for z the name received on x
	$\mid$	τ	internal action
	$\mid$	$[x = y]\pi$	matching: tests equality of x and y

Some Remarks

- $[x = y]\pi.P$ is known as name matching:
  it is equivalent to **if** $x = y$ **then** $\pi.P$.
- In $x(z).P$ e $(\nu z)P$, the name z is *bound* in P (i.e., P is the scope of z).
- A name that is not bound is called *free*.
- $fn(P)$ and $bn(P)$ are the sets of all free, resp. bound, names of P.
- We take processes up to *alpha-conversion*, which permits renaming of a bound name with a *fresh* one (not already in use).

$$\frac{y \notin fn(P)}{x(z).P \equiv x(y).(P\{^y/_z\})} \qquad \frac{y \notin fn(P)}{(\nu z)P \equiv (\nu y)(P\{^y/_z\})}$$

π-calculus: Structural Congruence

$S + \mathbf{0} \equiv S$ $\quad$ $S_1 + S_2 \equiv S_2 + S_1$ $\quad$ $S_1 + (S_2 + S_3) \equiv (S_1 + S_2) + S_3$

$P \mid \mathbf{0} \equiv P$ $\quad$ $P_1 \mid P_2 \equiv P_2 \mid P_1$ $\quad$ $P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$

$S + S \equiv S$ $\quad$ $!P \equiv P \mid !P$ $\quad$ $[a = a]\pi.P \equiv \pi.P$

$(\nu a)\mathbf{0} \equiv \mathbf{0}$ $\quad$ $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$ $\quad$ $\dfrac{a \notin fn(P)}{P \mid (\nu a)Q \equiv (\nu a)(P \mid Q)}$

By taking processes up to a suitable structural congruence we can:

1. Write processes in a canonical form.
2. Represent all possible interactions with few rules.

π-calculus: Structural Congruence

$$S + \mathbf{0} \equiv S \qquad S_1 + S_2 \equiv S_2 + S_1 \qquad S_1 + (S_2 + S_3) \equiv (S_1 + S_2) + S_3$$

$$P \mid \mathbf{0} \equiv P \qquad P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$$

$$S + S \equiv S \qquad !P \equiv P \mid !P \qquad [a = a]\pi.P \equiv \pi.P$$

$$(\nu a)\mathbf{0} \equiv \mathbf{0} \qquad (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad \frac{a \notin fn(P)}{P \mid (\nu a)Q \equiv (\nu a)(P \mid Q)}$$

By taking processes up to a suitable structural congruence we can:

1. Write processes in a canonical form.
2. Represent all possible interactions with few rules.

π-calculus: Structural Congruence

$$S + \mathbf{0} \equiv S \qquad S_1 + S_2 \equiv S_2 + S_1 \qquad S_1 + (S_2 + S_3) \equiv (S_1 + S_2) + S_3$$

$$P \mid \mathbf{0} \equiv P \qquad P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$$

$$S + S \equiv S \qquad !P \equiv P \mid !P \qquad [a = a]\pi.P \equiv \pi.P$$

$$(\nu a)\mathbf{0} \equiv \mathbf{0} \qquad (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad \frac{a \notin fn(P)}{P \mid (\nu a)Q \equiv (\nu a)(P \mid Q)}$$

By taking processes up to a suitable structural congruence we can:

1. Write processes in a canonical form.
2. Represent all possible interactions with few rules.

π-calculus: Reduction Semantics

Canonical Form

For each π-calculus process P there exist:

1. a finite number of names $x_1, ..., x_k$,
2. a finite number of sums $S_1, ..., S_n$, and
3. a finite number of processes $P_1, ..., P_m$ such that

$$P \equiv (\nu x_1)...(\nu x_k)(S_1|...|S_n|!P_1|...|!P_m)$$

Reduction semantics: Axioms

Reduction semantics focuses on *internal* moves $P \xrightarrow{\tau} Q$ only.

$$(Rtau)\ \frac{}{\tau.P + S \xrightarrow{\tau} P}$$

$$(Rcom)\ \frac{}{(x(y).P_1 + S_1)\,|\,(\overline{x}\langle z\rangle.P_2 + S_2) \xrightarrow{\tau} P_1\{^z/_y\}\,|\,P_2}$$

π-calculus: Reactive Contexts

Reduction semantics 1: Propagation Rules

$$(Rpar)\ \frac{P_1 \xrightarrow{\tau} P_1'}{P_1 \,|\, P_2 \xrightarrow{\tau} P_1' \,|\, P_2} \qquad (Rres)\ \frac{P \xrightarrow{\tau} P'}{(\nu x)P \xrightarrow{\tau} (\nu x)P'}$$

$$(Rstr)\ \frac{P \equiv Q \quad Q \xrightarrow{\tau} Q' \quad Q' \equiv P'}{P \xrightarrow{\tau} P'}$$

Reduction semantics 2: Reactive Contexts

$$\mathbb{C}[\![\,\cdot\,]\!] \ ::= \ [\![\,\cdot\,]\!] \quad | \quad \mathbb{C}[\![\,\cdot\,]\!] \,|\, P \quad | \quad (\nu x)\mathbb{C}[\![\,\cdot\,]\!]$$

$$(Rctx)\ \frac{P \equiv \mathbb{C}[\![\, Q \,]\!] \quad Q \xrightarrow{\tau} Q' \quad \mathbb{C}[\![\, Q' \,]\!] \equiv P'}{P \xrightarrow{\tau} P'}$$

Pi-calculus: Check Point

Answers these questions to proceed

1. Does it make sense $(\nu y)\overline{x}\langle y\rangle \equiv (\nu y)\overline{y}\langle x\rangle$?
2. Does it make sense $(\nu x)(\nu y)\overline{x}\langle y\rangle \equiv (\nu x)(\nu y)\overline{y}\langle x\rangle$?
3. Does $(\nu x)P \equiv (\nu x)P'$ imply $P \equiv P'$?
4. Are the following Server and Client ok?

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq (\nu k)(\ \overline{in}\langle k\rangle \,|\, \overline{k}\langle n\rangle \,|\, k(y).P\ )$$

5. Are the following Server and Client ok?

$$\begin{array}{rcl} S & \triangleq & !in(k).k(x).k(r).\overline{r}\langle f(x)\rangle \\ C & \triangleq & (\nu k)(\nu r)(\ \overline{in}\langle k\rangle \,|\, \overline{k}\langle n\rangle.\overline{k}\langle r\rangle \,|\, r(y).P\ ) \end{array}$$

Outline

Disciplining π-calculus

Are Names Used Properly?

π-calculus provides a rather sophisticated framework for interaction, but with quite low-level primitives: as process size increases the confidence in its design might decrease.

Type systems may help, but:

- names are used to encode many different behavioural aspects in terms of communication
- certain names require *static sorting* (e.g. all names transmitted on x must be integers, or that all names transmitted on y must be names of channels where integers can be sent, or that z can only be used for input)
- certain names require *dynamic annotations* (e.g. protocol narrations for the peers of a session, establishing that on channel z must first be sent an integer, then be received a name of a channel where integers can be sent)

Two Mugs Metaphor

More coffe in the milk or milk in the coffee?

- take a spoon of coffee (black mug), put it in the milk (white) mug and stir
- take a spoon of mixture coffee+milk, put it in the coffee mug and stir
- in proportion, is there more milk (w.r.t. to coffee) in the black mug or coffee (w.r.t. milk) in the white mug?

Names for Sessions

A common pattern of interaction

- P and Q establish a common fresh channel k to exchange data
- k represents a session between P and Q
- P assigns type T to k, which prescribes the series of actions that P wants to perform along k with Q
- Similarly, Q assigns type T' to k
- If T and T' are sort of dual to each other (modulo subtyping), then k is used in a type safe way
- Delegation can be allowed (e.g. P can pass k to R and stop using it)

$$Q \triangleq a(k).Q' \qquad P \triangleq (\nu k)\overline{a}\langle k\rangle.P'$$

Note that k can be alpha-renamed in both P and Q.
Given this analogy we write P as $\overline{a}(k).P'$.

Client Server Revisited

Remember the client server example:

$$!in(k).k(x).\overline{k}\langle f(x)\rangle \qquad (\nu k)\overline{in}\langle k\rangle.\overline{k}\langle n\rangle.k(y).P$$

Now it can be written as

$$!in(k).k(x).\overline{k}\langle f(x)\rangle \qquad \overline{in}(k).\overline{k}\langle n\rangle.k(y).P$$

- Client perspective T: k is used to send an integer *and then* to receive an integer
- Server perspective T': k is used to receive an integer *and then* to send an integer
- T and T' are syntactically dual to each other
- Channel *in*: is a channel used to transmit session keys of type T

Session Acceptance and Request

Syntax

- Session acceptance (binder for k): $a(k).P$
- Session request (binder for k): $\overline{a}(k).P$

Reduction Semantics

$$(link)\ \frac{}{a(k).P \mid \overline{a}(k).Q \xrightarrow{\tau} P \mid Q}$$

Intra-Session Communication

Syntax

- Input (binder for x): $k?(x).P$
- Output: $k!\langle y \rangle.P$

Reduction Semantics

$$(comm)\ \frac{}{k?(x).P \mid k!\langle y \rangle.Q \xrightarrow{\tau} P\{^{y}/_{x}\} \mid Q}$$

Intra-Session Selection

Syntax

- Label branching: $\sum_i k?\ell_i.P_i$
- Label selection: $k!\ell.P$

Reduction Semantics

$$(lab)\ \frac{j \in I}{\sum_{i \in I} k?\ell_i.P_i \mid k!\ell_j.Q \xrightarrow{\tau} P_j \mid Q}$$

Delegation

Syntax

- Session receiving (binder for k'): $k?((k')).P$
- Session sending: $k!\langle\langle k'\rangle\rangle.P$

Reduction Semantics

$$(pass)\ \frac{}{k?((x)).P \mid k!\langle\langle k'\rangle\rangle.Q \xrightarrow{\tau} P\{^{k'}/_{x}\} \mid Q}$$

Note that after having sent k' on k, process Q is no longer allowed to mention k'.

A Puzzle

Chess play

One young, bright computer scientists is given the possibility to pass the exam if she is able to play chess twice against the state-of-the-art computer player available on the web, without loosing both games. She has never played chess before. Which strategy can she take?

Assumptions

- We assume the game protocol consists of sending and receiving the list of moves made so far
- The AI will compute its best move by exploiting some function *next* applied on the history of moves.
- Each game runs in its own session

A Possible Solution

Computer AI

$$Chess \triangleq \text{rec } Y.\, start(k).\,(\, Y \mid k?\text{black}.k!\langle next(\epsilon)\rangle.M(k) + k?\text{white}.M(k)\,)$$

$$M(k) \triangleq \text{rec } X.\, k?(m).k!\langle m :: next(m)\rangle.X$$

Would you call it cheating?

The idea is essentially to let the computer AI play against itself.

$$Human \triangleq start(k_1).k_1!\text{black}.start(k_2).k_2!\text{white}.P(k_1, k_2)$$

$$P(k_1, k_2) \triangleq \text{rec } X.\, k_1?(m).k_2!\langle m\rangle.k_2?(n).k_1!\langle n\rangle.X$$

A Possible Solution

Computer AI

$$
\begin{aligned}
\mathit{Chess} &\triangleq \mathrm{rec}\, Y.\, \mathit{start}(k).\, (\, Y \mid k?\mathrm{black}.k!\langle \mathit{next}(\epsilon)\rangle.M(k) + k?\mathrm{white}.M(k)\,) \\
M(k) &\triangleq \mathrm{rec}\, X.\, k?(m).k!\langle m :: \mathit{next}(m)\rangle.X
\end{aligned}
$$

Would you call it cheating?

The idea is essentially to let the computer AI play against itself.

$$
\begin{aligned}
\mathit{Human} &\triangleq \mathit{start}(k_1).k_1!\mathrm{black}.\mathit{start}(k_2).k_2!\mathrm{white}.P(k_1, k_2) \\
P(k_1, k_2) &\triangleq \mathrm{rec}\, X.\, k_1?(m).k_2!\langle m\rangle.k_2?(n).k_1!\langle n\rangle.X
\end{aligned}
$$

Outline

Orchestration Calculus

Orc is an elegant language proposed by Cook and Misra as a basic programming model for structured orchestration of services:

1. The basic computational entities orchestrated by an Orc expression are not just web services but, more generally, *site* names.
2. Site names can be passed as arguments in site call, thus allowing a disciplined usage of name mobility.
3. Orc has quite original composition principles, including a form of cancellation of activities
4. Try Orc (in your browser or after download): `http://orc.csres.utexas.edu/`

Orc Sites

Orc relies on the basic notion of *site*, an abstraction amenable for:

1. being invoked
2. publishing values

Site calls

Site calls are the simplest Orc expressions:

- A site call can be a RMI, a call to a monitor procedure, to a function or to a web service.
- Each invocation to a site *s* elicits at most one response value published by *s*.
- A site computation might itself start other orchestrations, store effects locally and make (or not) such effects visible to clients.
- Sites can be composed by means of few operators to form expressions.

Orc Sites

Orc relies on the basic notion of *site*, an abstraction amenable for:

1. being invoked
2. publishing values

Site calls

Site calls are the simplest Orc expressions:

- A site call can be a RMI, a call to a monitor procedure, to a function or to a web service.
- Each invocation to a site *s* elicits at most one response value published by *s*.
- A site computation might itself start other orchestrations, store effects locally and make (or not) such effects visible to clients.
- Sites can be composed by means of few operators to form expressions.

Orc Expressions

Orc neatly separates orchestration from computation:

- Orc expressions can be considered like scripts to be invoked, e.g., within imperative programming languages
- the syntax for assigning the result of an expression e to a variable z is $z :\in e$
- Orc expressions can involve wide-area computation over multiple servers.

Contrary to site calls, an expression can, in principle, publish any number of response values

The assignment symbol $:\in$ (due to Hoare) in $z :\in e$ makes explicit that e can return zero or more results, one of which is assigned to z.

Orc Expressions

Orc neatly separates orchestration from computation:

- Orc expressions can be considered like scripts to be invoked, e.g., within imperative programming languages
- the syntax for assigning the result of an expression e to a variable z is $z :\in e$
- Orc expressions can involve wide-area computation over multiple servers.

Contrary to site calls, an expression can, in principle, publish any number of response values

The assignment symbol $:\in$ (due to Hoare) in $z :\in e$ makes explicit that e can return zero or more results, one of which is assigned to z.

Orc Composition Principles

Three ways to build expressions

1. ordinary parallel composition $f|g$, called *symmetric parallel* (e.g., the parallel of two site calls can produce zero, one or two values)
2. *sequencing* $f > x > g$: a fresh copy $g[v/x]$ of g is executed on any value v published by f (i.e., a pipeline is established from f to g).
3. *asymmetric parallel composition* f **where** $x :\in g$: f and g start in parallel, but all sub-expressions of f that depend on the value of x must wait for g to publish a value. When g produces a value it is assigned to x and that side of the orchestration is cancelled (i.e., it allows lazy evaluation, selection and pruning).

Sequencing and asymmetric parallel composition, take inspiration from universal and existential quantification, respectively.

Orc Composition Principles

Three ways to build expressions

1. ordinary parallel composition $f|g$, called *symmetric parallel* (e.g., the parallel of two site calls can produce zero, one or two values)
2. *sequencing* $f > x > g$: a fresh copy $g[v/x]$ of g is executed on any value v published by f (i.e., a pipeline is established from f to g).
3. *asymmetric parallel composition* f **where** $x :\in g$: f and g start in parallel, but all sub-expressions of f that depend on the value of x must wait for g to publish a value. When g produces a value it is assigned to x and that side of the orchestration is cancelled (i.e., it allows lazy evaluation, selection and pruning).

Sequencing and asymmetric parallel composition, take inspiration from universal and existential quantification, respectively.

Orc Composition Principles

Three ways to build expressions

1. ordinary parallel composition $f|g$, called *symmetric parallel* (e.g., the parallel of two site calls can produce zero, one or two values)
2. *sequencing* $f > x > g$: a fresh copy $g[v/x]$ of g is executed on any value v published by f (i.e., a pipeline is established from f to g).
3. *asymmetric parallel composition* f **where** $x :\in g$: f and g start in parallel, but all sub-expressions of f that depend on the value of x must wait for g to publish a value. When g produces a value it is assigned to x and that side of the orchestration is cancelled (i.e., it allows lazy evaluation, selection and pruning).

Sequencing and asymmetric parallel composition, take inspiration from universal and existential quantification, respectively.

Orc Composition Principles

Three ways to build expressions

1. ordinary parallel composition $f|g$, called *symmetric parallel* (e.g., the parallel of two site calls can produce zero, one or two values)
2. *sequencing* $f > x > g$: a fresh copy $g[v/x]$ of g is executed on any value v published by f (i.e., a pipeline is established from f to g).
3. *asymmetric parallel composition* f **where** $x :\in g$: f and g start in parallel, but all sub-expressions of f that depend on the value of x must wait for g to publish a value. When g produces a value it is assigned to x and that side of the orchestration is cancelled (i.e., it allows lazy evaluation, selection and pruning).

Sequencing and asymmetric parallel composition, take inspiration from universal and existential quantification, respectively.

Orc Syntax

(Expressions) e, f, g	$::=$	0	nil
	$\mid$	$M\langle p_1, \ldots, p_n \rangle$	site call
	$\mid$	$f > x > g$	sequencing
	$\mid$	$f \mid g$	symmetric parallel
	$\mid$	g **where** $x :\in f$	asymmetric parallel
	$\mid$	$E\langle p_1, \ldots, p_n \rangle$	expression call
(Definitions) D	$::=$	$E(x_1, \ldots, x_n) \underline{\Delta} f$	expression definition
(Parameters) p, q, r	$::=$	x	variable
	$\mid$	c	constant
	$\mid$	M	site

- x is *bound* (with scope g) in $f > x > g$ and g **where** $x :\in f$
- the free variables of an expression e are denoted by $fv(e)$
- if $x \notin fv(g)$ we abbreviate $f > x > g$ by writing $f \gg g$

Orc Syntax

(Expressions) e, f, g	$::=$	0	nil
	$\mid$	$M\langle p_1, \ldots, p_n \rangle$	site call
	$\mid$	$f > x > g$	sequencing
	$\mid$	$f \mid g$	symmetric parallel
	$\mid$	g **where** $x :\in f$	asymmetric parallel
	$\mid$	$E\langle p_1, \ldots, p_n \rangle$	expression call
(Definitions) D	$::=$	$E(x_1, \ldots, x_n) \underline{\Delta} f$	expression definition
(Parameters) p, q, r	$::=$	x	variable
	$\mid$	c	constant
	$\mid$	M	site

- x is *bound* (with scope g) in $f > x > g$ and g **where** $x :\in f$
- the free variables of an expression e are denoted by $fv(e)$
- if $x \notin fv(g)$ we abbreviate $f > x > g$ by writing $f \gg g$

Orc Syntax

(Expressions) e, f, g	$::=$	0	nil
	$\mid$	$M\langle p_1, \ldots, p_n \rangle$	site call
	$\mid$	$f > x > g$	sequencing
	$\mid$	$f \mid g$	symmetric parallel
	$\mid$	g **where** $x :\in f$	asymmetric parallel
	$\mid$	$E\langle p_1, \ldots, p_n \rangle$	expression call
(Definitions) D	$::=$	$E(x_1, \ldots, x_n) \triangleq f$	expression definition
(Parameters) p, q, r	$::=$	x	variable
	$\mid$	c	constant
	$\mid$	M	site

- x is *bound* (with scope g) in $f > x > g$ and g **where** $x :\in f$
- the free variables of an expression e are denoted by $fv(e)$
- if $x \notin fv(g)$ we abbreviate $f > x > g$ by writing $f \gg g$

Orc Syntax

(Expressions) e, f, g	$::=$	0	nil
	$\mid$	$M\langle p_1, \ldots, p_n \rangle$	site call
	$\mid$	$f > x > g$	sequencing
	$\mid$	$f \mid g$	symmetric parallel
	$\mid$	g **where** $x :\in f$	asymmetric parallel
	$\mid$	$E\langle p_1, \ldots, p_n \rangle$	expression call
(Definitions) D	$::=$	$E(x_1, \ldots, x_n) \underline{\Delta} f$	expression definition
(Parameters) p, q, r	$::=$	x	variable
	$\mid$	c	constant
	$\mid$	M	site

- x is *bound* (with scope g) in $f > x > g$ and g **where** $x :\in f$
- the free variables of an expression e are denoted by $fv(e)$
- if $x \notin fv(g)$ we abbreviate $f > x > g$ by writing $f \gg g$

Orc Semantics: Actions

The operational semantics of Orc is given by a Labelled Transition Systems defined in the SOS style

Transition Labels

- $M(\vec{c}, k)$ denotes a *site call*
- $k?c$ denotes a *site response*
- $!c$ denotes a *locally published value*
- τ denotes an *internal action*

The abstract semantics considered in the literature are trace equivalence and strong bisimilarity

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and $Signal$.

$$\frac{k \text{ globally fresh}}{M\langle\vec{c}\rangle \xrightarrow{M(\vec{c},k)} ?k}\ \text{(SiteCall)} \qquad \frac{}{let\langle c\rangle \xrightarrow{!c} 0}\ \text{(Let)}$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c\rangle}\ \text{(SiteRet)} \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0}\ \text{(Signal)}$$

Getting the latest news of date d from CNN

$$CNN\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k)}$$

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and $Signal$.

$$\frac{k \text{ globally fresh}}{M\langle \vec{c} \rangle \xrightarrow{M(\vec{c},k)} ?k} \text{(SiteCall)} \qquad \frac{}{let\langle c \rangle \xrightarrow{!c} 0} \text{(Let)}$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c \rangle} \text{(SiteRet)} \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0} \text{(Signal)}$$

Getting the latest news of date d from CNN

$$CNN\langle 3June2006 \rangle \xrightarrow{CNN(3June2006,k)} ?k \xrightarrow{k?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfricanLizardsInvadeFlorida \rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and *Signal*.

$$\frac{k \ \text{globally fresh}}{M\langle \color{red}{\vec{c}} \rangle \xrightarrow{M(\vec{c},k)} \color{red}{?k}} \ (\text{SiteCall}) \qquad \frac{}{let\langle c \rangle \xrightarrow{!c} 0} \ (\text{Let})$$

$$\frac{}{\color{red}{?k} \xrightarrow{k?c} let\langle c \rangle} \ (\text{SiteRet}) \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0} \ (\text{Signal})$$

Getting the latest news of date *d* from CNN

$$CNN\langle 3June2006 \rangle \xrightarrow{CNN(3June2006,k)} ?k \xrightarrow{k?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfricanLizardsInvadeFlorida \rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and *Signal*.

$$\frac{k \ \text{globally fresh}}{M\langle \vec{c} \rangle \xrightarrow{M(\vec{c},k)} ?k} \ \text{(SiteCall)} \qquad \frac{}{let\langle c \rangle \xrightarrow{!c} 0} \ \text{(Let)}$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c \rangle} \ \text{(SiteRet)} \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0} \ \text{(Signal)}$$

Getting the latest news of date *d* from CNN

$$CNN\langle 3June2006 \rangle \xrightarrow{CNN(3June2006,k)} ?k \xrightarrow{k?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfricanLizardsInvadeFlorida \rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and *Signal*.

$$\frac{k \text{ globally fresh}}{M\langle \vec{c} \rangle \xrightarrow{M(\vec{c},k)} ?k} \text{(SiteCall)} \qquad \frac{}{let\langle c \rangle \xrightarrow{!c} 0} \text{(Let)}$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c \rangle} \text{(SiteRet)} \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0} \text{(Signal)}$$

Getting the latest news of date *d* from CNN

$$CNN\langle 3June2006 \rangle \xrightarrow{CNN(3June2006,k)} ?k \xrightarrow{k?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfricanLizardsInvadeFlorida \rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and *Signal*.

$$\frac{k \text{ globally fresh}}{M\langle \vec{c} \rangle \xrightarrow{M(\vec{c},k)} ?k} \text{(SiteCall)} \qquad \frac{}{let\langle c \rangle \xrightarrow{!c} 0} \text{(Let)}$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c \rangle} \text{(SiteRet)} \qquad \frac{}{Signal \xrightarrow{!\langle\rangle} 0} \text{(Signal)}$$

Getting the latest news of date *d* from CNN

$$CNN\langle 3June2006 \rangle \xrightarrow{CNN(3June2006,k)} ?k \xrightarrow{k?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfricanLizardsInvadeFlorida \rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Site Call

Two special auxiliary sites are $let(x_1, \ldots, x_n)$ and *Signal*.

$$\frac{k \ \text{globally fresh}}{M\langle\vec{c}\rangle \stackrel{M(\vec{c},k)}{\longrightarrow} ?k} \ \text{(SiteCall)} \qquad \frac{}{let\langle c\rangle \stackrel{!c}{\longrightarrow} 0} \ \text{(Let)}$$

$$\frac{}{?k \stackrel{k?c}{\longrightarrow} let\langle c\rangle} \ \text{(SiteRet)} \qquad \frac{}{Signal \stackrel{!\langle\rangle}{\longrightarrow} 0} \ \text{(Signal)}$$

Getting the latest news of date *d* from CNN

$$CNN\langle 3June2006\rangle \stackrel{CNN(3June2006,k)}{\longrightarrow} ?k \stackrel{k?GiantAfricanLizardsInvadeFlorida}{\longrightarrow}$$

$$let\langle GiantAfricanLizardsInvadeFlorida\rangle \stackrel{!GiantAfricanLizardsInvadeFlorida}{\longrightarrow} 0$$

$$z :\in CNN(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \mid f \xrightarrow{\mu} g' \mid f}\ \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \mid f \xrightarrow{\mu} g \mid f'}\ \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \mid BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \mid BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \mid ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \mid let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \mid let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \dots$$

$$z :\in CNN(d) \mid BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \mid f \xrightarrow{\mu} g' \mid f} \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \mid f \xrightarrow{\mu} g \mid f'} \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \mid BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \mid BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \mid ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \mid let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \mid let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \mid BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \mid f \xrightarrow{\mu} g' \mid f} \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \mid f \xrightarrow{\mu} g \mid f'} \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \mid BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \mid BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \mid ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \mid let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \mid let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \mid BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \,|\, f \xrightarrow{\mu} g' \,|\, f} \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \,|\, f \xrightarrow{\mu} g \,|\, f'} \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \,|\, BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \,|\, BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \,|\, ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \,|\, let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \,|\, let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \,|\, BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \mid f \xrightarrow{\mu} g' \mid f} \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \mid f \xrightarrow{\mu} g \mid f'} \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \mid BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \mid BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \mid ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \mid let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \mid let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \mid BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \,|\, f \xrightarrow{\mu} g' \,|\, f} \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \,|\, f \xrightarrow{\mu} g \,|\, f'} \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \,|\, BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \,|\, BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \,|\, ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \,|\, let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \,|\, let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \,|\, BBC(d) \longrightarrow z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \,|\, f \xrightarrow{\mu} g' \,|\, f}\ \text{(SymLeft)} \qquad \frac{f \xrightarrow{\mu} f'}{g \,|\, f \xrightarrow{\mu} g \,|\, f'}\ \text{(SymRight)}$$

Getting news from CNN and BBC

$$CNN\langle 3June2006\rangle \,|\, BBC\langle 3June2006\rangle \xrightarrow{CNN(3June2006,k_{CNN})}$$

$$?k_{CNN} \,|\, BBC\langle 3June2006\rangle \xrightarrow{BBC(3June2006,k_{BBC})}$$

$$?k_{CNN} \,|\, ?k_{BBC} \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$?k_{CNN} \,|\, let\langle GiantUsaTouristsInvadeMadagascar\rangle \xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida}$$

$$let\langle GiantAfrican...\rangle \,|\, let\langle GiantUsa...\rangle \xrightarrow{!GiantAfricanLizardsInvadeFlorida} \ldots$$

$$z :\in CNN(d) \,|\, BBC(d) \ \longrightarrow\ z = GiantAfricanLizardsInvadeFlorida$$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \mid g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$$(CNN\langle d\rangle \mid BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$(?k_{CNN} \mid ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$(?k_{CNN} \mid let\langle GiantUsa\ldots\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$$

$$(?k_{CNN} \mid 0) > n > Email\langle rb@gmail.it, n\rangle \mid Email\langle rb@gmail.it, GiantUsa\ldots\rangle$$

$$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \mid 0) > n > Email\langle rb@gmail.it, n\rangle \mid$$
$$Email\langle rb@gmail.it, GiantUsa\ldots\rangle \mid Email\langle rb@gmail.it, GiantAfrican\ldots\rangle$$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \,|\, g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$$(CNN\langle d\rangle \,|\, BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$(?k_{CNN} \,|\, ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$(?k_{CNN} \,|\, let\langle GiantUsa...\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$$

$$(?k_{CNN} \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|\, Email\langle rb@gmail.it, GiantUsa...\rangle$$

$$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|$$

$$Email\langle rb@gmail.it, GiantUsa...\rangle \,|\, Email\langle rb@gmail.it, GiantAfrican...\rangle$$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \,|\, g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$$(CNN\langle d\rangle \,|\, BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$(?k_{CNN} \,|\, ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$(?k_{CNN} \,|\, let\langle GiantUsa...\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$$

$$(?k_{CNN} \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|\, Email\langle rb@gmail.it, GiantUsa...\rangle$$

$$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|$$

$$Email\langle rb@gmail.it, GiantUsa...\rangle \,|\, Email\langle rb@gmail.it, GiantAfrican...\rangle$$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \,|\, g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$$(CNN\langle d\rangle \,|\, BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$(?k_{CNN} \,|\, ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$(?k_{CNN} \,|\, let\langle GiantUsa...\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$$

$$(?k_{CNN} \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|\, Email\langle rb@gmail.it, GiantUsa...\rangle$$

$$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|$$

$$Email\langle rb@gmail.it, GiantUsa...\rangle \,|\, Email\langle rb@gmail.it, GiantAfrican...\rangle$$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \,|\, g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$(CNN\langle d\rangle \,|\, BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$

$(?k_{CNN} \,|\, ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$

$(?k_{CNN} \,|\, let\langle GiantUsa...\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$

$(?k_{CNN} \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|\, Email\langle rb@gmail.it, GiantUsa...\rangle$

$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|$

$Email\langle rb@gmail.it, GiantUsa...\rangle \,|\, Email\langle rb@gmail.it, GiantAfrican...\rangle$

Orc Semantics: Sequential Composition

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g} \text{(Seq)} \qquad \frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \,|\, g[c/x]} \text{(SeqPipe)}$$

Getting all news from CNN and BBC by email

$$(CNN\langle d\rangle \,|\, BBC\langle d\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$(?k_{CNN} \,|\, ?k_{BBC}) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{k_{BBC}?GiantUsaTouristsInvadeMadagascar}$$

$$(?k_{CNN} \,|\, let\langle GiantUsa...\rangle) > n > Email\langle rb@gmail.it, n\rangle \xrightarrow{\tau}$$

$$(?k_{CNN} \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|\, Email\langle rb@gmail.it, GiantUsa...\rangle$$

$$\xrightarrow{k_{CNN}?GiantAfricanLizardsInvadeFlorida} \xrightarrow{\tau} (0 \,|\, 0) > n > Email\langle rb@gmail.it, n\rangle \,|$$
$$Email\langle rb@gmail.it, GiantUsa...\rangle \,|\, Email\langle rb@gmail.it, GiantAfrican...\rangle$$

Orc Semantics: Asymmetric Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f} \text{ (A.L.)}$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'} \text{ (A.R.)} \qquad \frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]} \text{ (A.P.)}$$

Getting one news from CNN and BBC by email

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (CNN\langle d\rangle \mid BBC\langle d\rangle) \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \mid ?k_{BBC}) \xrightarrow{k_{BBC}?GiantUsa...}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \mid let(GiantUsa...)) \xrightarrow{\tau}$$

$$Email\langle rb@gmail.it, GiantUsaTouristsInvadeMadagascar\rangle$$

Orc Semantics: Asymmetric Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f} \text{(A.L.)}$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'} \text{(A.R.)} \qquad \frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]} \text{(A.P.)}$$

Getting one news from CNN and BBC by email

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (CNN\langle d\rangle \,|\, BBC\langle d\rangle) \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, ?k_{BBC}) \xrightarrow{k_{BBC}?GiantUsa...}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, let\langle GiantUsa...\rangle) \xrightarrow{\tau}$$

$$Email\langle rb@gmail.it, GiantUsaTouristsInvadeMadagascar\rangle$$

Orc Semantics: Asymmetric Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f} \text{ (A.L.)}$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'} \text{ (A.R.)} \qquad \frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]} \text{ (A.P.)}$$

Getting one news from CNN and BBC by email

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (CNN\langle d\rangle \mid BBC\langle d\rangle) \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \mid ?k_{BBC}) \xrightarrow{k_{BBC}?GiantUsa...}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \mid let\langle GiantUsa...\rangle) \xrightarrow{\tau}$$

$$Email\langle rb@gmail.it, GiantUsaTouristsInvadeMadagascar\rangle$$

Orc Semantics: Asymmetric Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f} \text{ (A.L.)}$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'} \text{ (A.R.)} \qquad \frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]} \text{ (A.P.)}$$

Getting one news from CNN and BBC by email

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (CNN\langle d\rangle \,|\, BBC\langle d\rangle) \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, ?k_{BBC}) \xrightarrow{k_{BBC}?GiantUsa...}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, let\langle GiantUsa...\rangle) \xrightarrow{\tau}$$

$$Email\langle rb@gmail.it, GiantUsaTouristsInvadeMadagascar\rangle$$

Orc Semantics: Asymmetric Parallel Composition

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f} \text{ (A.L.)}$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'} \text{ (A.R.)} \qquad \frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]} \text{ (A.P.)}$$

Getting one news from CNN and BBC by email

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (CNN\langle d\rangle \,|\, BBC\langle d\rangle) \xrightarrow{CNN(d,k_{CNN})} \xrightarrow{BBC(d,k_{BBC})}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, ?k_{BBC}) \xrightarrow{k_{BBC}?GiantUsa...}$$

$$Email\langle rb@gmail.it, n\rangle \textbf{ where } n :\in (?k_{CNN} \,|\, let\langle GiantUsa...\rangle) \xrightarrow{\tau}$$

$$Email\langle rb@gmail.it, GiantUsaTouristsInvadeMadagascar\rangle$$

Orc Semantics (in one slide)

$$\frac{k \text{ globally fresh}}{M\langle\vec{c}\rangle \xrightarrow{M(\vec{c},k)} ?k}\ (\text{SiteCall})$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{f > x > g \xrightarrow{\mu} f' > x > g}\ (\text{Seq})$$

$$\frac{}{?k \xrightarrow{k?c} let\langle c\rangle}\ (\text{SiteRet})$$

$$\frac{f \xrightarrow{!c} f'}{f > x > g \xrightarrow{\tau} (f' > x > g) \mid g[c/x]}\ (\text{SeqPipe})$$

$$\frac{g \xrightarrow{\mu} g'}{g \mid f \xrightarrow{\mu} g' \mid f}\ (\text{SymLeft})$$

$$\frac{g \xrightarrow{\mu} g'}{g \textbf{ where } x :\in f \xrightarrow{\mu} g' \textbf{ where } x :\in f}\ (\text{AsymLeft})$$

$$\frac{f \xrightarrow{\mu} f'}{g \mid f \xrightarrow{\mu} g \mid f'}\ (\text{SymRight})$$

$$\frac{f \xrightarrow{\mu} f' \quad \mu \neq !c}{g \textbf{ where } x :\in f \xrightarrow{\mu} g \textbf{ where } x :\in f'}\ (\text{AsymRight})$$

$$\frac{E(\vec{x}) \underline{\Delta} f}{E\langle\vec{p}\rangle \xrightarrow{\tau} f[\vec{p}/\vec{x}]}\ (\text{Def})$$

$$\frac{f \xrightarrow{!c} f'}{g \textbf{ where } x :\in f \xrightarrow{\tau} g[c/x]}\ (\text{AsymPrune})$$

$$\frac{}{let\langle c\rangle \xrightarrow{!c} 0}\ (\text{Let})$$

$$\frac{}{Signal \xrightarrow{!\langle\rangle} 0}\ (\text{Signal})$$

Fork-Join Parallelism and Synchronisation

Weather Forecast Example

$$CityDate \;\underline{\Delta}\; (\,let\langle x, y\rangle \;\textbf{where}\; x :\in GoogleLocate\,)\;\textbf{where}\; y :\in GoogleDate$$

$$WForecast \;\underline{\Delta}\; CityDate > x > CnnWeather\langle x\rangle$$

$$z :\in WForecast \;\longrightarrow\; z = 11^oC/22^oC - PartiallyCloudy$$

Generalised synchronisation

$$
\begin{aligned}
Sync(\vec{M}) \;\underline{\Delta}\;\; & let(x_1) \gg \ldots \gg let(x_n) \gg Signal \\
& \textbf{where}\; x_1 :\in M_1 \\
& \ldots \\
& \textbf{where}\; x_n :\in M_n
\end{aligned}
$$

$M_1, \ldots, M_n$ are executed in parallel, but the signal is emitted only after having the response from every M_i).

Fork-Join Parallelism and Synchronisation

Weather Forecast Example

$$\begin{array}{rcl} \mathit{CityDate} & \underline{\Delta} & (\,\mathit{let}\langle x, y\rangle \ \mathbf{where}\ x :\in \mathit{GoogleLocate}\,)\ \mathbf{where}\ y :\in \mathit{GoogleDate} \\ \mathit{WForecast} & \underline{\Delta} & \mathit{CityDate} > x > \mathit{CnnWeather}\langle x\rangle \end{array}$$

$$z :\in \mathit{WForecast} \longrightarrow z = 11^o C/22^o C - \mathit{PartiallyCloudy}$$

Generalised synchronisation

$$\begin{array}{rcl} \mathit{Sync}(\vec{M}) & \underline{\Delta} & \mathit{let}(x_1) \gg \ldots \gg \mathit{let}(x_n) \gg \mathit{Signal} \\ & & \mathbf{where}\ x_1 :\in M_1 \\ & & \ldots \\ & & \mathbf{where}\ x_n :\in M_n \end{array}$$

$M_1, \ldots, M_n$ are executed in parallel, but the signal is emitted only after having the response from every M_i). Or equivalently

$$\begin{array}{rcl} \mathit{Sync}(\vec{M}) & \underline{\Delta} & \mathit{let}(x_1, \ldots, x_n) \gg \mathit{Signal} \\ & & \mathbf{where}\ x_1 :\in M_1 \cdots \mathbf{where}\ x_n :\in M_n \end{array}$$

Fork-Join Parallelism and Synchronisation

Weather Forecast Example

$$
\begin{array}{rcl}
CityDate & \underline{\Delta} & (\,let\langle x, y\rangle\ \textbf{where}\ x :\in GoogleLocate\,)\ \textbf{where}\ y :\in GoogleDate \\
WForecast & \underline{\Delta} & CityDate > x > CnnWeather\langle x\rangle
\end{array}
$$

$$z :\in WForecast \;\longrightarrow\; z = 11^o C / 22^o C - PartiallyCloudy$$

Generalised synchronisation

$$
\begin{array}{rcl}
Sync(\vec{M}) & \underline{\Delta} & let(x_1) >> \ldots >> let(x_n) >> Signal \\
 & & \textbf{where}\ x_1 :\in M_1 \\
 & & \ldots \\
 & & \textbf{where}\ x_n :\in M_n
\end{array}
$$

$M_1, \ldots, M_n$ are executed in parallel, but the signal is emitted only after having the response from every M_i). Or equivalently:

$$
\begin{array}{rcl}
Sync(\vec{M}) & \underline{\Delta} & let(x_1, \ldots, x_n) >> Signal \\
 & & \textbf{where}\ x_1 :\in M_1 \cdots \textbf{where}\ x_n :\in M_n
\end{array}
$$

Fork-Join Parallelism and Synchronisation

Weather Forecast Example

$$\begin{array}{rcl} CityDate & \underline{\Delta} & (\,let\langle x, y\rangle \textbf{ where } x :\in GoogleLocate\,) \textbf{ where } y :\in GoogleDate \\ WForecast & \underline{\Delta} & CityDate > x > CnnWeather\langle x\rangle \end{array}$$

$$z :\in WForecast \;\longrightarrow\; z = 11^{o}C/22^{o}C - PartiallyCloudy$$

Generalised synchronisation

$$\begin{array}{rcl} Sync(\vec{M}) & \underline{\Delta} & let(x_1) \gg ... \gg let(x_n) \gg Signal \\ & & \textbf{where } x_1 :\in M_1 \\ & & ... \\ & & \textbf{where } x_n :\in M_n \end{array}$$

$M_1, ..., M_n$ are executed in parallel, but the signal is emitted only after having the response from every M_i). Or equivalently:

$$\begin{array}{rcl} Sync(\vec{M}) & \underline{\Delta} & let(x_1, ..., x_n) \gg Signal \\ & & \textbf{where } x_1 :\in M_1 \cdots \textbf{ where } x_n :\in M_n \end{array}$$

Fork-Join Parallelism and Synchronisation

Weather Forecast Example

$$
\begin{array}{rcl}
\mathit{CityDate} & \underline{\Delta} & (\,\mathit{let}\langle x, y\rangle \ \textbf{where}\ x :\in \mathit{GoogleLocate}\,)\ \textbf{where}\ y :\in \mathit{GoogleDate} \\
\mathit{WForecast} & \underline{\Delta} & \mathit{CityDate} > x > \mathit{CnnWeather}\langle x\rangle
\end{array}
$$

$$z :\in \mathit{WForecast} \ \longrightarrow\ z = 11^o C / 22^o C - \mathit{PartiallyCloudy}$$

Generalised synchronisation

$$
\begin{array}{rcl}
\mathit{Sync}(\vec{M}) & \underline{\Delta} & \mathit{let}(x_1) \gg \ldots \gg \mathit{let}(x_n) \gg \mathit{Signal} \\
 & & \textbf{where}\ x_1 :\in M_1 \\
 & & \ldots \\
 & & \textbf{where}\ x_n :\in M_n
\end{array}
$$

$M_1, ..., M_n$ are executed in parallel, but the signal is emitted only after having the response from every M_i). Or equivalently:

$$
\begin{array}{rcl}
\mathit{Sync}(\vec{M}) & \underline{\Delta} & \mathit{let}(x_1, ..., x_n) \gg \mathit{Signal} \\
 & & \textbf{where}\ x_1 :\in M_1 \cdots \textbf{where}\ x_n :\in M_n
\end{array}
$$

Conditional Expressions

Site If

$If(b)$ replies with a signal if b is true and it remains silent if b is false.

Fibonacci numbers

$$FibPair(x) \triangleq (\, If\langle x = 0\rangle \gg let(1,0)\,) \,|\, (\, If\langle x! = 0\rangle \gg FibPair(x-1) > (y,z) > let(y+z,y)\,)$$

$$Fib(x) \triangleq FibPair(x) > (y,z) > let(y)$$

Choices

$$Cond(b,S,T) \triangleq (If\langle b\rangle \gg S) \,|\, (If\langle \neg b\rangle \gg T)$$

$$A.P + B.Q \triangleq Cond\langle b,P,Q\rangle \textbf{ where } b :\in \left(\begin{array}{ll} & A \gg let(\mathrm{true}) \\ | & B \gg let(\mathrm{false}) \end{array} \right)$$

Conditional Expressions

Site If

$If(b)$ replies with a signal if b is true and it remains silent if b is false.

Fibonacci numbers

$$
\begin{array}{rcl}
FibPair(x) & \underline{\Delta} & (\, If\langle x = 0\rangle \;\gg\; let(1,0)\,)\,| \\
& & (\, If\langle x! = 0\rangle \gg FibPair(x-1) > (y,z) > let(y+z,y)\,) \\
& & \\
Fib(x) & \underline{\Delta} & FibPair(x) > (y,z) > let(y)
\end{array}
$$

Choices

$$
Cond(b,S,T) \quad \underline{\Delta} \quad (If\langle b\rangle \gg S)\,|\,(If\langle\neg b\rangle \gg T)
$$

$$
A.P + B.Q \quad \underline{\Delta} \quad Cond\langle b,P,Q\rangle \textbf{ where } b :\in \left(\begin{array}{ll} & A \gg let(\mathrm{true}) \\ | & B \gg let(\mathrm{false}) \end{array}\right)
$$

Conditional Expressions

Site If

$If(b)$ replies with a signal if b is true and it remains silent if b is false.

Fibonacci numbers

$$
\begin{array}{rcl}
FibPair(x) & \underline{\Delta} & (\,If\langle x = 0\rangle \gg let(1,0)\,)\,| \\
 & & (\,If\langle x! = 0\rangle \gg FibPair(x-1) > (y,z) > let(y+z,y)\,) \\
 & & \\
Fib(x) & \underline{\Delta} & FibPair(x) > (y,z) > let(y)
\end{array}
$$

Choices

$$Cond(b,S,T) \quad \underline{\Delta} \quad (If\langle b\rangle \gg S)\,|\,(If\langle \neg b\rangle \gg T)$$

$$A.P + B.Q \quad \underline{\Delta} \quad Cond\langle b,P,Q\rangle \ \textbf{where}\ b :\in \left(\begin{array}{ll} & A \gg let(\mathrm{true}) \\ | & B \gg let(\mathrm{false}) \end{array}\right)$$

Orc Check Point

1. Explain the difference between

$$Z1(x) \underline{\Delta} \,(\, If\langle x = 0\rangle \; \gg \; let(0)\,)$$

and

$$Z2(x) \underline{\Delta} \; let(0) \textbf{ where } y :\in If\langle x = 0\rangle$$

2. A classic problem in non-strict evaluation is the so-called *parallel-or*. Suppose there are two sites S_1 and S_2 that publish some booleans. Write an Orc expression *ParOR* that publishes the value *false* only if both sites return *false*, the value *true* as soon as either site returns *true*, and otherwise it never publishes a value. In the solution it can be assumed:
   - the existence of a site $If(b)$ that receives a boolean value and returns *true* if b is *true*, and otherwise it does not respond;
   - the existence of a site $Or(b_1, b_2)$ that return the inclusive logical disjunction of the two booleans received as arguments.

   Note that *ParOr* must publish one result, at most.

Outline

CaSPiS Genesis

Sources of inspiration

SCC [WS-FM 2006] was inspired by:

- π (names, communication): $x(y).P$, $\overline{x}y.P$, $(\nu x)P$
- Orc (pipelining and pruning of activities):
  $(EAPLS\langle 2008\rangle \,|\, EATCS\langle 2008\rangle) > cfp > Email\langle rb@gmail.it, cfp\rangle$
  $Email\langle rb@gmail.it, cfp\rangle$ **where** $cfp :\in (EAPLS\langle 2008\rangle \,|\, EATCS\langle 2008\rangle)$
- πI, session types (primitives for sessions): $a(k).P$, $\overline{a}(k).P$
  (roughly, think of $\overline{a}(k).P$ as $(\nu k)\overline{a}k.P$)

CaSPiS [FMOODS 2008] is inspired by SCC and:

- webπ, cjoin, Sagas (primitives for LRT and compensations)
- KLAIM (pattern matching)

All source were relevant to the SOC paradigm, but so far

- not available in a single calculus
- yet to be amalgamated in some disciplined way

CaSPiS Genesis

Sources of inspiration

SCC [WS-FM 2006] was inspired by:

- π (names, communication): $x(y).P$, $\overline{x}y.P$, $(\nu x)P$
- Orc (pipelining and pruning of activities):
  $(EAPLS\langle 2008\rangle \,|\, EATCS\langle 2008\rangle) > cfp > Email\langle rb@gmail.it, cfp\rangle$
  $Email\langle rb@gmail.it, cfp\rangle$ **where** $cfp :\in (EAPLS\langle 2008\rangle \,|\, EATCS\langle 2008\rangle)$
- πI, session types (primitives for sessions): $a(k).P$, $\overline{a}(k).P$
  (roughly, think of $\overline{a}(k).P$ as $(\nu k)\overline{a}k.P$)

CaSPiS [FMOODS 2008] is inspired by SCC and:

- webπ, cjoin, Sagas (primitives for LRT and compensations)
- KLAIM (pattern matching)

All source were relevant to the SOC paradigm, but so far

- not available in a single calculus
- yet to be amalgamated in some disciplined way

CaSPiS Genesis

Sources of inspiration

SCC [WS-FM 2006] was inspired by:

- π (names, communication): $x(y).P$, $\overline{x}y.P$, $(\nu x)P$
- Orc (pipelining and pruning of activities):
  $(EAPLS\langle 2008\rangle \mid EATCS\langle 2008\rangle) > cfp > Email\langle rb@gmail.it, cfp\rangle$
  $Email\langle rb@gmail.it, cfp\rangle$ **where** $cfp :\in (EAPLS\langle 2008\rangle \mid EATCS\langle 2008\rangle)$
- πI, session types (primitives for sessions): $a(k).P$, $\overline{a}(k).P$
  (roughly, think of $\overline{a}(k).P$ as $(\nu k)\overline{a}k.P$)

CaSPiS [FMOODS 2008] is inspired by SCC and:

- webπ, cjoin, Sagas (primitives for LRT and compensations)
- KLAIM (pattern matching)

All source were relevant to the SOC paradigm, but so far

- not available in a single calculus
- yet to be amalgamated in some disciplined way

CaSPiS Genesis

Sources of inspiration

SCC [WS-FM 2006] was inspired by:

- π (names, communication): $x(y).P$, $\overline{x}y.P$, $(\nu x)P$
- Orc (pipelining and pruning of activities):
  $(EAPLS\langle 2008\rangle \mid EATCS\langle 2008\rangle) > cfp > Email\langle rb@gmail.it, cfp\rangle$
  $Email\langle rb@gmail.it, cfp\rangle$ **where** $cfp :\in (EAPLS\langle 2008\rangle \mid EATCS\langle 2008\rangle)$
- πI, session types (primitives for sessions): $a(k).P$, $\overline{a}(k).P$
  (roughly, think of $\overline{a}(k).P$ as $(\nu k)\overline{a}k.P$)

CaSPiS [FMOODS 2008] is inspired by SCC and:

- webπ, cjoin, Sagas (primitives for LRT and compensations)
- KLAIM (pattern matching)

All source were relevant to the SOC paradigm, but so far

- not available in a single calculus
- yet to be amalgamated in some disciplined way

Criteria

- reduce flexibility (only disciplined way to interact)
- handle sessions in a transparent way (only as run-time syntax)
- channel names disappear (server names used instead)
- handle unexpected behaviours

Client Server Revisited

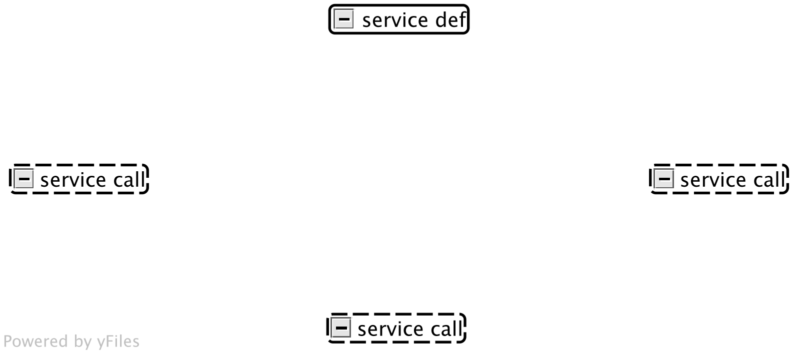
Remember the client server example:

$$S \triangleq !in(k).k(x).\overline{k}\langle f(x)\rangle \qquad C \triangleq \overline{in}(k).\overline{k}\langle n\rangle.k(y).P$$

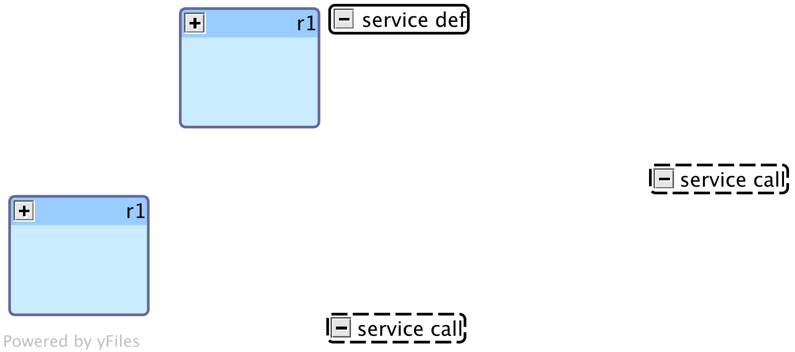
In CaSPiS it can be written

$$S \triangleq !in.(?x)\langle f(x)\rangle \qquad C \triangleq \overline{in}.\langle 1\rangle(?y)P$$

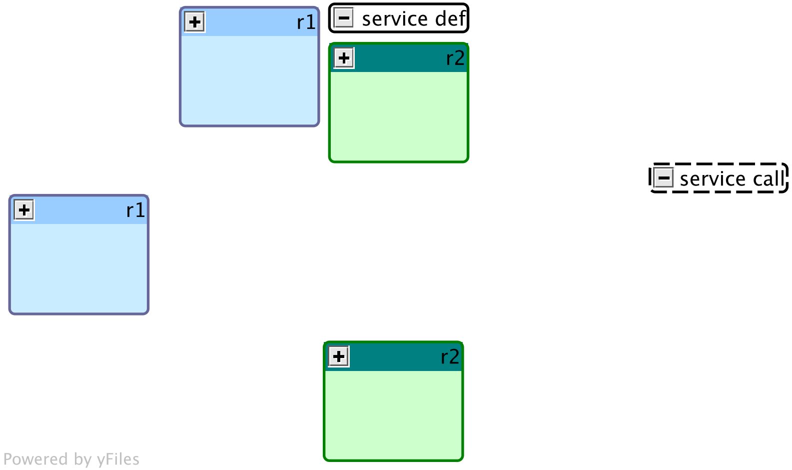
Sketch of Multiple Sessions



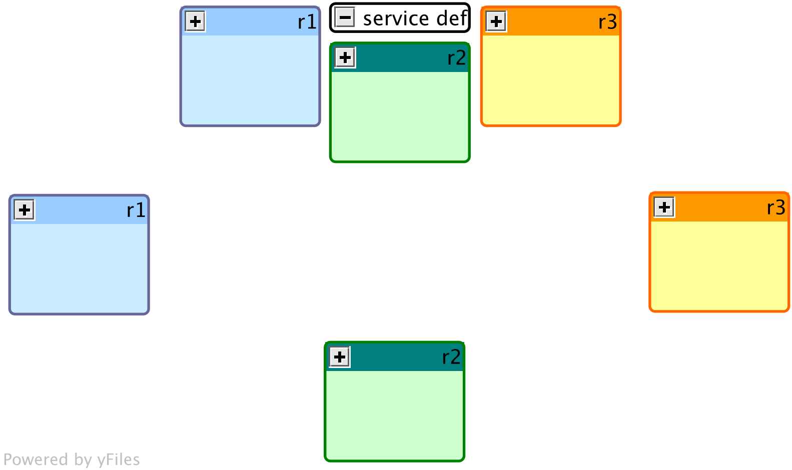
Sketch of Multiple Sessions



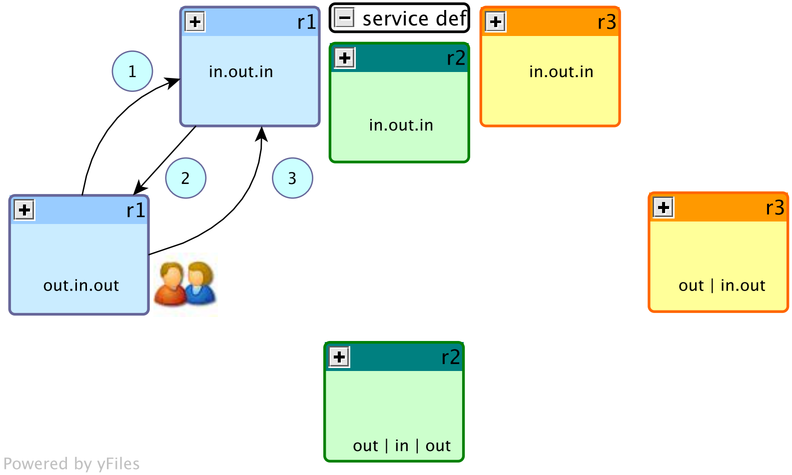
Sketch of Multiple Sessions



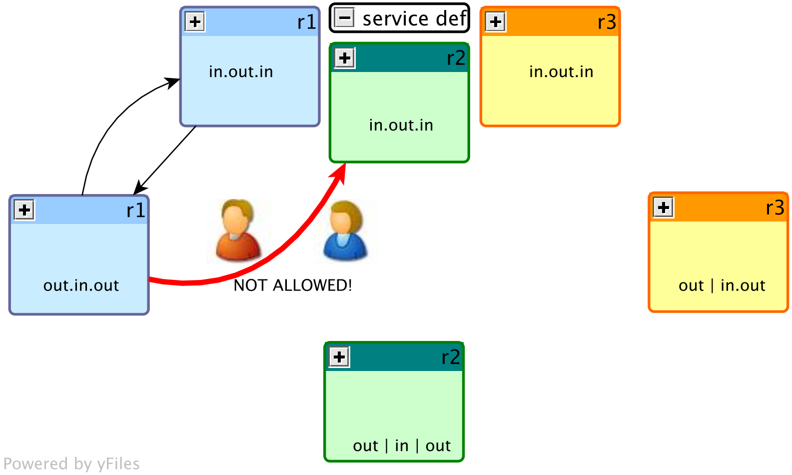
Sketch of Multiple Sessions



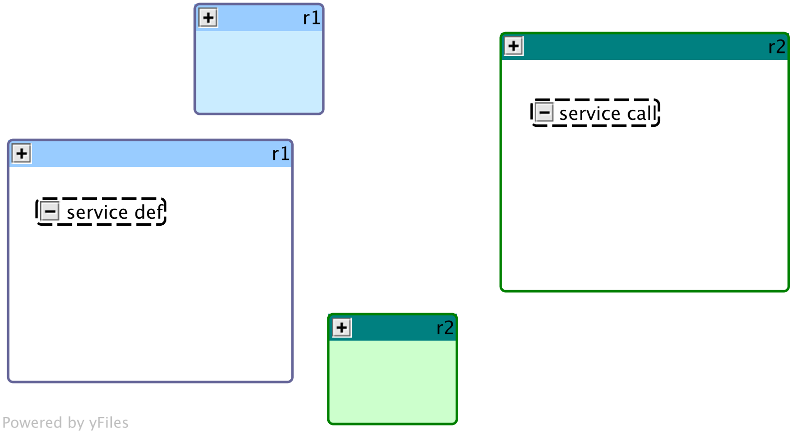
Sketch of Conversations



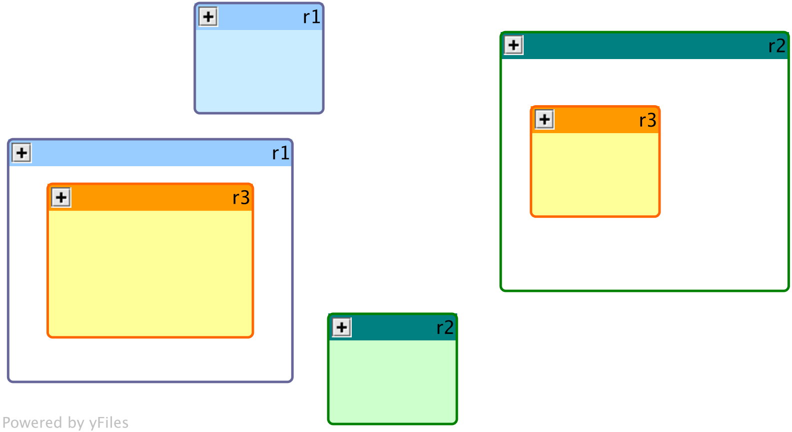
Sketch of Conversations



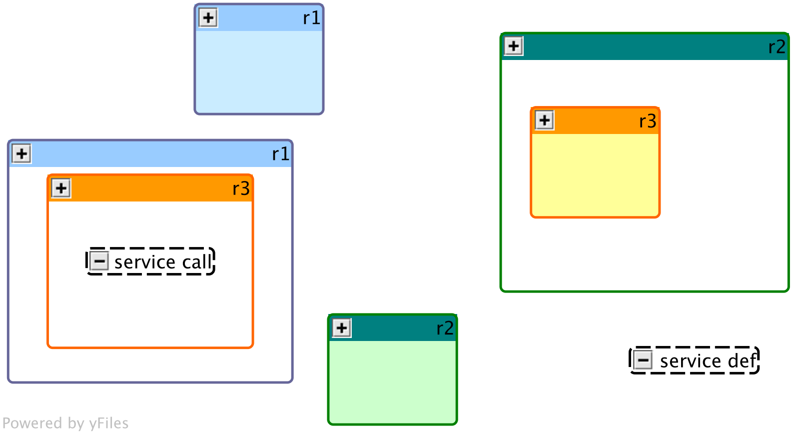
Sketch of Nested Sessions



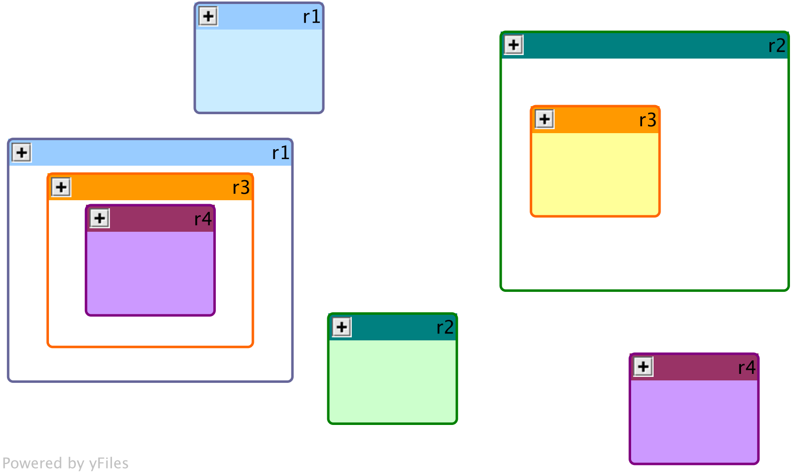
Sketch of Nested Sessions



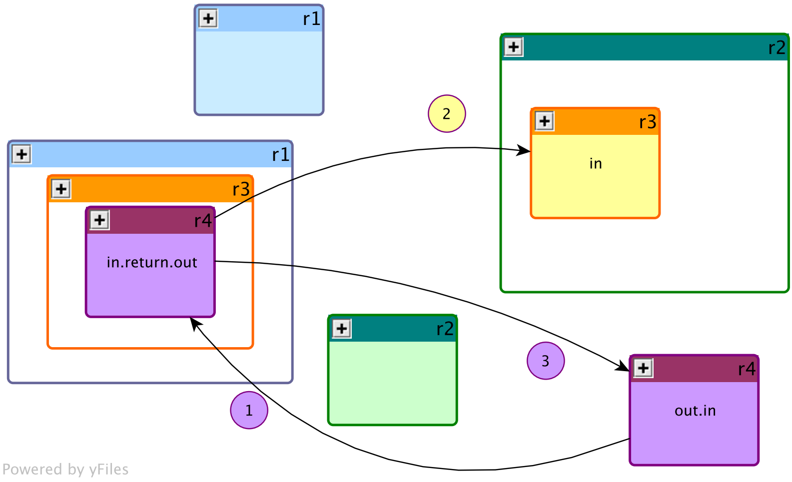
Sketch of Nested Sessions



Sketch of Nested Sessions



Sketch of Return



CaSPiS: General Principles

Service definitions: $s.P$

- services expose their protocols
- services can be deployed dynamically, shut down and updated
- services can handle multiple requests separately

Service invocations: $\overline{s}.P$

- service invocations expose their protocols
- sequential composition via pipelining (á la Orc)

Sessions: $r \triangleright P$ (run-time syntax)

- service invocation spawns fresh session parties (locally to each partner)
- sessions are: two-party (service-side + client-side) + private
- interaction between session protocols: bi-directional
- nested sessions: values can be returned outside sessions (one level up)

CaSPiS: General Principles

Service definitions: $s.P$

- services expose their protocols
- services can be deployed dynamically, shut down and updated
- services can handle multiple requests separately

Service invocations: $\overline{s}.P$

- service invocations expose their protocols
- sequential composition via pipelining (á la Orc)

Sessions: $r \triangleright P$ (run-time syntax)

- service invocation spawns fresh session parties (locally to each partner)
- sessions are: two-party (service-side + client-side) + private
- interaction between session protocols: bi-directional
- nested sessions: values can be returned outside sessions (one level up)

CaSPiS: General Principles

Service definitions: $s.P$

- services expose their protocols
- services can be deployed dynamically, shut down and updated
- services can handle multiple requests separately

Service invocations: $\overline{s}.P$

- service invocations expose their protocols
- sequential composition via pipelining (á la Orc)

Sessions: $r \triangleright P$ (run-time syntax)

- service invocation spawns fresh session parties (locally to each partner)
- sessions are: two-party (service-side + client-side) + private
- interaction between session protocols: bi-directional
- nested sessions: values can be returned outside sessions (one level up)

CaSPiS Syntax

Prefixes, Values, Patterns

π	$::=$	(F)	Abstraction
	$\mid$	$\langle V \rangle$	Concretion
	$\mid$	$\langle V \rangle^{\uparrow}$	Return
V	$::=$	$u \mid f(\tilde{V})$	Value ($f \in \Sigma$)
F	$::=$	$u \mid ?x \mid f(\tilde{F})$	Pattern ($f \in \Sigma$)

Processes

P, Q ::=	$\sum_{i \in I} \pi_i P_i$	Guarded Sum	$\mid$	$\dagger(k)$	Signal
$\mid$	$s_k.P$	Service Definition	$\mid$	$r \rhd_k P$	Session
$\mid$	$\bar{s}_k.P$	Service Invocation	$\mid$	$\blacktriangleright P$	Terminated Session
$\mid$	$P > Q$	Pipeline	$\mid$	$P \mid Q$	Parallel Composition
$\mid$	$close$	Close	$\mid$	$(\nu n)P$	Restriction
$\mid$	$k \cdot P$	Listener	$\mid$	$!P$	Replication

CaSPiS Syntax

Prefixes, Values, Patterns

π	$::=$	(F)	Abstraction
	$\mid$	$\langle V \rangle$	Concretion
	$\mid$	$\langle V \rangle^{\uparrow}$	Return
V	$::=$	$u \mid f(\tilde{V})$	Value ($f \in \Sigma$)
F	$::=$	$u \mid ?x \mid f(\tilde{F})$	Pattern ($f \in \Sigma$)

Processes

P, Q ::=	$\sum_{i \in I} \pi_i P_i$	Guarded Sum	$\mid$	$\dagger(k)$	Signal
$\mid$	$s_k.P$	Service Definition	$\mid$	$r \rhd_k P$	Session
$\mid$	$\overline{s}_k.P$	Service Invocation	$\mid$	$\blacktriangleright P$	Terminated Session
$\mid$	$P > Q$	Pipeline	$\mid$	$P \mid Q$	Parallel Composition
$\mid$	$close$	Close	$\mid$	$(\nu n)P$	Restriction
$\mid$	$k \cdot P$	Listener	$\mid$	$!P$	Replication

Structural Congruence (Close Free Fragment)

Structural axioms

$$
\begin{array}{rclrcll}
P \mid \mathbf{0} & \equiv & P & (\nu n)\mathbf{0} & \equiv & \mathbf{0} & \\
P \mid Q & \equiv & Q \mid P & (\nu n)(\nu m)P & \equiv & (\nu m)(\nu n)P & \\
(P \mid Q) \mid R & \equiv & P \mid (Q \mid R) & ((\nu n)P) > Q & \equiv & (\nu n)(P > Q) & \text{if } n \notin fn(Q) \\
!P & \equiv & P \mid !P & ((\nu n)P) \mid Q & \equiv & (\nu n)(P \mid Q) & \text{if } n \notin fn(Q) \\
 & & & r \rhd (\nu n)P & \equiv & (\nu n)(r \rhd P) & \text{if } r \neq n
\end{array}
$$

Reactive contexts

- Dynamic operators: service definition $s.[\![\cdot]\!]$ and invocation $\overline{s}.[\![\cdot]\!]$, prefix $\pi_i[\![\cdot]\!]$, left-sided pipeline $P > [\![\cdot]\!]$ and replication $![\![\cdot]\!]$
- *Static context* $\mathbb{C}[\![\cdot]\!]$: its hole does not occur under a dynamic operator
- *Session-immune* $\mathbb{S}[\![\cdot]\!]$: its hole does not occur under a session
- *Pipeline-immune* $\mathbb{P}[\![\cdot]\!]$: if its hole does not occur under a right-sided pipeline

Roughly, $\mathbb{S}[\![\cdot]\!]$ does not “intercept” abstraction and return prefixes, and $\mathbb{P}[\![\cdot]\!]$ does not “intercept” concretion prefixes.

Structural Congruence (Close Free Fragment)

Structural axioms

$$
\begin{array}{rclrcll}
P \mid \mathbf{0} & \equiv & P & (\nu n)\mathbf{0} & \equiv & \mathbf{0} & \\
P \mid Q & \equiv & Q \mid P & (\nu n)(\nu m)P & \equiv & (\nu m)(\nu n)P & \\
(P \mid Q) \mid R & \equiv & P \mid (Q \mid R) & ((\nu n)P) > Q & \equiv & (\nu n)(P > Q) & \text{if } n \notin fn(Q) \\
!P & \equiv & P \mid !P & ((\nu n)P) \mid Q & \equiv & (\nu n)(P \mid Q) & \text{if } n \notin fn(Q) \\
 & & & r \rhd (\nu n)P & \equiv & (\nu n)(r \rhd P) & \text{if } r \neq n
\end{array}
$$

Reactive contexts

- Dynamic operators: service definition $s.[\![\cdot]\!]$ and invocation $\overline{s}.[\![\cdot]\!]$, prefix $\pi_i[\![\cdot]\!]$, left-sided pipeline $P > [\![\cdot]\!]$ and replication $![\![\cdot]\!]$
- *Static context* $\mathbb{C}[\![\cdot]\!]$: its hole does not occur under a dynamic operator
- *Session-immune* $\mathbb{S}[\![\cdot]\!]$: its hole does not occur under a session
- *Pipeline-immune* $\mathbb{P}[\![\cdot]\!]$: if its hole does not occur under a right-sided pipeline

Roughly, $\mathbb{S}[\![\cdot]\!]$ does not “intercept” abstraction and return prefixes, and $\mathbb{P}[\![\cdot]\!]$ does not “intercept” concretion prefixes.

Reduction Semantics 1

Opening a session

$$(sync)\frac{r \text{ fresh for } \mathbb{C}[\![\cdot,\cdot]\!], P, Q}{\mathbb{C}[\![s.P, \overline{s}.Q]\!] \xrightarrow{\tau} (\nu r)\mathbb{C}[\![r \rhd P, r \rhd Q]\!]}$$

Intra-session communication

$$(Ssync)\frac{\sigma = match(F, V)}{\mathbb{C}_r[\![\langle V \rangle P + \sum_i \pi_i P_i, (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![P, Q\sigma]\!]}$$

where $\mathbb{C}_r[\![\cdot,\cdot]\!]$ is a context of the form $\mathbb{C}[\![r \rhd \mathbb{P}[\![\cdot]\!], r \rhd \mathbb{S}[\![\cdot]\!]]\!]$

$$(SRsync)\frac{\sigma = match(F, V)}{\mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![\langle V \rangle^\uparrow P + \sum_i \pi_i P_i]\!], (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![P]\!], Q\sigma]\!]}$$

Opening a session

$$(sync)\frac{r \text{ fresh for } \mathbb{C}[\![\cdot,\cdot]\!], P, Q}{\mathbb{C}[\![s.P, \overline{s}.Q]\!] \xrightarrow{\tau} (\nu r)\mathbb{C}[\![r \rhd P, r \rhd Q]\!]}$$

Intra-session communication

$$(Ssync)\frac{\sigma = match(F,V)}{\mathbb{C}_r[\![\langle V\rangle P + \sum_i \pi_i P_i, (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![P, Q\sigma]\!]}$$

where $\mathbb{C}_r[\![\cdot,\cdot]\!]$ is a context of the form $\mathbb{C}[\![r \rhd \mathbb{P}[\![\cdot]\!], r \rhd \mathbb{S}[\![\cdot]\!]]\!]$

$$(SRsync)\frac{\sigma = match(F,V)}{\mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![\langle V\rangle^{\uparrow} P + \sum_i \pi_i P_i]\!], (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![P]\!], Q\sigma]\!]}$$

Reduction Semantics 1

Opening a session

$$(sync)\frac{r \text{ fresh for } \mathbb{C}[\![\cdot,\cdot]\!], P, Q}{\mathbb{C}[\![s.P, \overline{s}.Q]\!] \xrightarrow{\tau} (\nu r)\mathbb{C}[\![r \rhd P, r \rhd Q]\!]}$$

Intra-session communication

$$(Ssync)\frac{\sigma = match(F, V)}{\mathbb{C}_r[\![\langle V \rangle P + \sum_i \pi_i P_i, (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![P, Q\sigma]\!]}$$

where $\mathbb{C}_r[\![\cdot,\cdot]\!]$ is a context of the form $\mathbb{C}[\![r \rhd \mathbb{P}[\![\cdot]\!], r \rhd \mathbb{S}[\![\cdot]\!]]\!]$

$$(SRsync)\frac{\sigma = match(F, V)}{\mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![\langle V \rangle^{\uparrow} P + \sum_i \pi_i P_i]\!], (F)Q + \sum_j \pi_j Q_i]\!] \xrightarrow{\tau} \mathbb{C}_r[\![r_1 \rhd \mathbb{S}_1[\![P]\!], Q\sigma]\!]}$$

Reduction Semantics 2

Pipeline orchestration

$$\frac{Q \equiv \mathbb{S}[\![(F)Q' + \sum_j \pi_j Q_i]\!] \quad \sigma = match(F, V)}{\mathbb{C}[\![\mathbb{P}[\![\langle V \rangle P + \sum_i \pi_i P_i]\!] > Q]\!] \xrightarrow{\tau} \mathbb{C}[\![\mathbb{S}[\![Q'\sigma]\!] \,|\, (\mathbb{P}[\![P]\!] > Q)]\!]}$$

$$\frac{Q \equiv \mathbb{S}[\![(F)Q' + \sum_j \pi_j Q_i]\!] \quad \sigma = match(F, V)}{\mathbb{C}[\![\mathbb{P}[\![r \triangleright \mathbb{S}_1[\![\langle V \rangle^{\uparrow} P + \sum_i \pi_i P_i]\!]]\!] > Q]\!] \xrightarrow{\tau} \mathbb{C}[\![\mathbb{S}[\![Q'\sigma]\!] \,|\, (\mathbb{P}[\![r \triangleright \mathbb{S}_1[\![P]\!]]\!] > Q)]\!]}$$

Example 1: Digital Documents

Service definition

$$!sign.(?x)(\nu t)\langle K\{x, t\}\rangle$$

- *sign* is a (replicated and thus persistent) service
- a *sign* instance waits for a digital document x, generates a fresh nonce t and then sends back both the document and the nonce signed with a key K

Service invocation

$$\overline{sign}.\langle plan\rangle(?y)\langle y\rangle^{\uparrow}$$

- a client of *sign*
- it passes the argument *plan* to the service, then waits for the signed response from the server and returns this value outside the session as a result

Example 1: Digital Documents

Service definition

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle$$

- *sign* is a (replicated and thus persistent) service
- a *sign* instance waits for a digital document x, generates a fresh nonce t and then sends back both the document and the nonce signed with a key K

Service invocation

$$\overline{sign}.\langle plan\rangle(?y)\langle y\rangle^{\uparrow}$$

- a client of *sign*
- it passes the argument *plan* to the service, then waits for the signed response from the server and returns this value outside the session as a result

Example 1: Digital Documents

A run

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \ \overline{sign}.\langle plan\rangle(?y)\langle y\rangle^{\uparrow}$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \ (\nu r)(\ r \triangleright (?x)(\nu t)\langle K\{x,t\}\rangle \quad | \ r \triangleright \langle plan\rangle(?y)\langle y\rangle^{\uparrow}\ )$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \ (\nu r,t)(\ r \triangleright \langle K\{plan,t\}\rangle \quad | \ r \triangleright (?y)\langle y\rangle^{\uparrow}\ )$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \ (\nu r,t)(\ r \triangleright \mathbf{0} \quad | \ r \triangleright \langle K\{plan,t\}\rangle^{\uparrow}\ )$$

Sessions for separation

$$(\ \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad | \quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow}\ )$$

The protocols of the two clients will run in separate sessions and will not interfere.

Pipelines for composition

$$(\ \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad | \quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow}\ ) \quad > \quad (?z)\overline{store}.\langle z\rangle$$

Example 1: Digital Documents

A run

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \; \overline{sign}.\langle plan\rangle(?y)\langle y\rangle^{\uparrow}$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \; (\nu r)(\; r \triangleright (?x)(\nu t)\langle K\{x,t\}\rangle \quad | \; r \triangleright \langle plan\rangle(?y)\langle y\rangle^{\uparrow} \;)$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \; (\nu r,t)(\; r \triangleright \langle K\{plan,t\}\rangle \quad | \; r \triangleright (?y)\langle y\rangle^{\uparrow} \;)$$

$$!sign.(?x)(\nu t)\langle K\{x,t\}\rangle \quad | \; (\nu r,t)(\; r \triangleright \mathbf{0} \quad | \; r \triangleright \langle K\{plan,t\}\rangle^{\uparrow} \;)$$

Sessions for separation

$$(\; \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad | \quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow} \;)$$

The protocols of the two clients will run in separate sessions and will not interfere.

Pipelines for composition

$$(\; \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad | \quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow} \;) \quad > \quad (?z)\overline{store}.\langle z\rangle$$

Example 1: Digital Documents

A run

$$
\begin{array}{lll}
!sign.(?x)(\nu t)\langle K\{x,t\}\rangle & & \mid \overline{sign}.\langle plan\rangle(?y)\langle y\rangle^{\uparrow} \\
!sign.(?x)(\nu t)\langle K\{x,t\}\rangle & \mid (\nu r)(\ r \rhd (?x)(\nu t)\langle K\{x,t\}\rangle & \mid r \rhd \langle plan\rangle(?y)\langle y\rangle^{\uparrow}\ ) \\
!sign.(?x)(\nu t)\langle K\{x,t\}\rangle & \mid (\nu r,t)(\ r \rhd \langle K\{plan,t\}\rangle & \mid r \rhd (?y)\langle y\rangle^{\uparrow}\ ) \\
!sign.(?x)(\nu t)\langle K\{x,t\}\rangle & \mid (\nu r,t)(\ r \rhd \mathbf{0} & \mid r \rhd \langle K\{plan,t\}\rangle^{\uparrow}\ )
\end{array}
$$

Sessions for separation

$$(\ \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad\mid\quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow}\ )$$

The protocols of the two clients will run in separate sessions and will not interfere.

Pipelines for composition

$$(\ \overline{sign}.\langle plan_1\rangle(?y)\langle y\rangle^{\uparrow} \quad\mid\quad \overline{sign}.\langle plan_2\rangle(?y)\langle y\rangle^{\uparrow}\ ) \quad > \quad (?z)\overline{store}.\langle z\rangle$$

Example 2: Common Patterns of Interaction

One way

$$s.(?x) \qquad \overline{s}.\langle V\rangle$$

Request response

$$s.(?x)\langle f(x)\rangle \qquad \overline{s}.\langle V\rangle(?r)\langle r\rangle^{\uparrow}$$

π-calculus channels

$$a(x).P \triangleq a.(?x)\langle x\rangle^{\uparrow} > (?x)P \qquad \overline{a}v.P \triangleq \overline{a}.\langle v\rangle\langle -\rangle^{\uparrow} > (-)P$$

Proxy (service name passing)

$$!proxy.(?s,?x)\overline{s}.\langle x\rangle!(?y)\langle y\rangle^{\uparrow}$$

Example 2: Common Patterns of Interaction

One way

$$s.(?x) \qquad \overline{s}.\langle V\rangle$$

Request response

$$s.(?x)\langle f(x)\rangle \qquad \overline{s}.\langle V\rangle(?r)\langle r\rangle^{\uparrow}$$

π-calculus channels

$$a(x).P \triangleq a.(?x)\langle x\rangle^{\uparrow} > (?x)P \qquad \overline{a}v.P \triangleq \overline{a}.\langle v\rangle\langle -\rangle^{\uparrow} > (-)P$$

Proxy (service name passing)

$$!proxy.(?s, ?x)\overline{s}.\langle x\rangle!(?y)\langle y\rangle^{\uparrow}$$

Example 2: Common Patterns of Interaction

One way

$$s.(?x) \qquad \overline{s}.\langle V \rangle$$

Request response

$$s.(?x)\langle f(x)\rangle \qquad \overline{s}.\langle V\rangle(?r)\langle r\rangle^{\uparrow}$$

π-calculus channels

$$a(x).P \triangleq a.(?x)\langle x\rangle^{\uparrow} > (?x)P \qquad \overline{a}v.P \triangleq \overline{a}.\langle v\rangle\langle -\rangle^{\uparrow} > (-)P$$

Proxy (service name passing)

$$!proxy.(?s, ?x)\overline{s}.\langle x\rangle!(?y)\langle y\rangle^{\uparrow}$$

Example 3: Selection

Select

$$\textbf{select } F_1, \ldots, F_n \textbf{ from } P \triangleq (\nu s)\left(s.(F_1).\ldots.(F_n)\langle F_1^{-?}, \ldots, F_n^{-?}\rangle^{\uparrow} \mid \overline{s}.P\right)$$

where $F_i^{-?}$ denotes the value V_i obtained from F_i by replacing each $?x$ with x

Select-from

$$\textbf{select } F_1, \ldots, F_n \textbf{ from } P \textbf{ in } Q \triangleq \textbf{select } F_1, \ldots, F_n \textbf{ from } P > (F_1, \ldots, F_n)Q$$

Select first two CfP

$$\textbf{select } ?x, ?y \textbf{ from } (\overline{EAPLS}^{*} \mid \overline{EATCS}^{*} \mid \overline{TYPES}^{*}) \textbf{ in } \overline{emailMe}.\langle x, y\rangle$$

where

$$\overline{s}^{*} \triangleq \overline{s}.!(?x)\langle x\rangle^{\uparrow}$$

Typed Variant

Main assumptions

Services are

- **persistent** (not consumed after invocations)
- **top-level** (not nested, not dynamically installed)
- **stateless** (no top-level return on service side)

Sessions are

- not interruptable (**close-free** fragment)
- with **non recursive** communication protocols

Interaction:

- no pattern matching
- simplified pipeline ($P > x > Q$, i.e. $P > (?x)Q$)
- conditional
- branching and selection

Example 1: Factorial

Service definition

$$\begin{aligned} fatt.(?n)if\ (n = 0) \\ then\ \langle 1 \rangle \\ else\ (\overline{fatt}.\langle n - 1 \rangle(?x).\langle x \rangle^{\uparrow}) > x > \langle n \cdot x \rangle \end{aligned}$$

A *fatt* instance waits for a natural number n: if equal to zero then sends back 1 to the client, otherwise issues a (nested) invocation to a fresh instance of *fatt* with argument $n - 1$, waits for the response and passes the result x to a pipe that sends back $n \cdot x$ to the client

Service invocation

$$\overline{fatt}.\langle 3 \rangle(?x) \quad | \quad \overline{fatt}.\langle 5 \rangle(?x)\langle x \rangle^{\uparrow}$$

The first client passes the argument *3* to the service instance, then waits for the response; the second client passes a different argument and returns the computed result to the parent session. The protocols of the two clients will run in fresh, separated sessions and will not interfere.

Example 1: Factorial

Service definition

$$\begin{array}{l} \mathit{fatt}.(?n)\mathit{if}\ (n = 0) \\ \qquad \mathit{then}\ \langle 1 \rangle \\ \qquad \mathit{else}\ (\overline{\mathit{fatt}}.\langle n - 1 \rangle(?x).\langle x \rangle^{\uparrow}) > x > \langle n \cdot x \rangle \end{array}$$

A *fatt* instance waits for a natural number n: if equal to zero then sends back 1 to the client, otherwise issues a (nested) invocation to a fresh instance of *fatt* with argument $n - 1$, waits for the response and passes the result x to a pipe that sends back $n \cdot x$ to the client

Service invocation

$$\overline{\mathit{fatt}}.\langle 3 \rangle(?x) \quad | \quad \overline{\mathit{fatt}}.\langle 5 \rangle(?x)\langle x \rangle^{\uparrow}$$

The first client passes the argument *3* to the service instance, then waits for the response; the second client passes a different argument and returns the computed result to the parent session. The protocols of the two clients will run in fresh, separated sessions and will not interfere.

Example 2: Room reservation

Service definition (with branching)

$$reserve.\Big(\; (single)(?x)\langle code(x, \text{""})\rangle \\ + (double)(?x, ?y).\langle code(x, y)\rangle \Big)$$

(where $code : str \times str \to int$ is a function only available on service side)

Service invocations (with selection)

$$\overline{reserve}.\langle single\rangle\langle \text{"Bob"}\rangle(?x)\langle x\rangle^{\uparrow}$$

$$\overline{reserve}.\langle double\rangle\langle \text{"Bob"}, \text{"Leo"}\rangle(?y)\langle y\rangle^{\uparrow}$$

$$\overline{reserve}.if\ (...) \\ then\ \langle single\rangle\langle \text{"Bob"}\rangle(?x).\langle x\rangle^{\uparrow} \\ else\ \langle double\rangle\langle \text{"Bob"}, \text{"Leo"}\rangle(?y)\langle y\rangle^{\uparrow}$$

Example 2: Room reservation

Service definition (with branching)

$$reserve.\Big(\ (single)(?x)\langle code(x,\text{""})\rangle \\ +\ (double)(?x,?y).\langle code(x,y)\rangle\ \Big)$$

(where $code : str \times str \to int$ is a function only available on service side)

Service invocations (with selection)

$$\overline{reserve}.\langle single\rangle\langle\text{"Bob"}\rangle(?x)\langle x\rangle^{\uparrow}$$

$$\overline{reserve}.\langle double\rangle\langle\text{"Bob"},\text{"Leo"}\rangle(?y)\langle y\rangle^{\uparrow}$$

$$\overline{reserve}.if\ (...) \\ then\ \langle single\rangle\langle\text{"Bob"}\rangle(?x).\langle x\rangle^{\uparrow} \\ else\ \langle double\rangle\langle\text{"Bob"},\text{"Leo"}\rangle(?y)\langle y\rangle^{\uparrow}$$

Example 3: Proxy service for load balancing

Service definition (with name passing and extrusion)

$$(\nu a, b)\Big(\; a.P$$
$$\mid b.P$$
$$\mid \textit{loadbalance}.\textit{if}\ (\textit{choose}(a, b) = 1)\ \textit{then}\ \langle a \rangle\ \textit{else}\ \langle b \rangle \Big)$$

Service invocation

$$(\overline{\textit{loadbalance}}(?z)\langle z \rangle^{\uparrow}) > x > \overline{z}.Q$$

Example 3: Proxy service for load balancing

Service definition (with name passing and extrusion)

$$(\nu a, b)\big(\ a.P \mid b.P \mid \mathit{loadbalance}.\mathit{if}\ (\mathit{choose}(a, b) = 1)\ \mathit{then}\ \langle a\rangle\ \mathit{else}\ \langle b\rangle\ \big)$$

Service invocation

$$(\overline{\mathit{loadbalance}}(?z)\langle z\rangle^{\uparrow}) > x > \overline{z}.Q$$

Type judgements

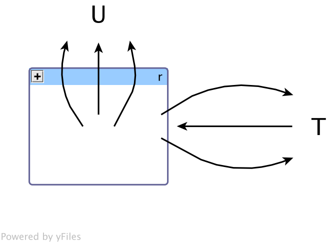
Overall idea

- Type values: $\Gamma \vdash v : S$
- Type a process as if part of a current session:

$$\Gamma \vdash P : U[T]$$

  separating intra-session interaction T from upward interaction U
- The type T of the protocol on one side of a session should be compatible w.r.t. the type T' of its partner's protocol
- In case of nested sessions, the U typed upward interaction will contribute to the type of its "father" session

Sketch of Typing



Some issues and limitations

- Some flexibility required w.r.t. branching and selection
- Some care needed in parallel composition of protocols
- Some care needed in dealing with the replication due to pipelines
- Recursive invocation of services is possible
- No form of delegation allowed
- Mobility of service names

Type system basics

Syntax of types

$$
\begin{array}{rcll}
S & ::= & [T] & \text{(session)} \\
 & | & \mathcal{B} & \text{(basic data types)} \\
T & ::= & end & \text{(no action)} \\
 & | & ?(S_1, \ldots, S_n).T & \text{(input of a tuple)} \\
 & | & !(S_1, \ldots, S_n).T & \text{(output of a tuple)} \\
 & | & \&\{l_1 : T_1, \ldots, l_n : T_n\} & \text{(external choice)} \\
 & | & \oplus\{l_1 : T_1, \ldots, l_n : T_n\} & \text{(internal choice)} \\
U & ::= & !(\tilde{S})^k.end & \text{(upward interaction)}
\end{array}
$$

Dual types

$$
\overline{end} = end \qquad
\begin{array}{rcl}
\overline{?(\tilde{S}).T} & = & !(\tilde{S}).\overline{T} \\
\overline{!(\tilde{S}).T'} & = & ?(\tilde{S}).\overline{T'}
\end{array}
\qquad
\begin{array}{rcl}
\overline{\&\{l_i : T_i\}_i} & = & \oplus\{l_i : \overline{T_i}\}_i \\
\overline{\oplus\{l_i : T_i\}_i} & = & \&\{l_i : \overline{T_i}\}_i
\end{array}
$$

Type System Highlights: Services and Sessions

Services

(Service)
$$\Gamma, s : S \vdash s : S$$

$$\frac{\Gamma \vdash P : end[T] \quad \Gamma \vdash s : [T]}{\Gamma \vdash s.P : end[end]} \text{(Tdef)}$$

$$\frac{\Gamma \vdash Q : U[\overline{T}] \quad \Gamma \vdash s : [T]}{\Gamma \vdash \overline{s}.Q : end[U]} \text{(Tinv)}$$

Sessions

$$\frac{\Gamma \vdash P : U[T]}{\Gamma, r : [T] \vdash r^{+} \triangleright P : end[U]} \text{(Tses)}$$

$$\frac{\Gamma \vdash Q : U[\overline{T}]}{\Gamma, r : [T] \vdash r^{-} \triangleright Q : end[U]} \text{(Tsesl)}$$

Type System Highlights: Services and Sessions

Services

(Service)

$$\Gamma, s : S \vdash s : S$$

$$\frac{\Gamma \vdash P : end[T] \quad \Gamma \vdash s : [T]}{\Gamma \vdash s.P : end[end]} \text{(Tdef)} \qquad \frac{\Gamma \vdash Q : U[\overline{T}] \quad \Gamma \vdash s : [T]}{\Gamma \vdash \overline{s}.Q : end[U]} \text{(Tinv)}$$

Sessions

$$\frac{\Gamma \vdash P : U[T]}{\Gamma, r : [T] \vdash r^{+} \triangleright P : end[U]} \text{(Tses)} \qquad \frac{\Gamma \vdash Q : U[\overline{T}]}{\Gamma, r : [T] \vdash r^{-} \triangleright Q : end[U]} \text{(TsesI)}$$

Type System Highlights: Protocols

Input, output, and return

$$\frac{\Gamma, \tilde{x} : \tilde{S} \vdash P : U[T]}{\Gamma \vdash (?\tilde{x})P : U[?(\tilde{S}).T]} \text{(Tin)} \qquad \frac{\Gamma \vdash P : U[T] \quad \Gamma \vdash \tilde{v} : \tilde{S}}{\Gamma \vdash \langle \tilde{v} \rangle P : U[!(\tilde{S}).T]} \text{(Tout)}$$

$$\frac{\Gamma \vdash P : U[T] \quad \Gamma \vdash \tilde{v} : \tilde{S}}{\Gamma \vdash \langle \tilde{v} \rangle^{\uparrow} P : !(\tilde{S}).U[T]} \text{(Tret)}$$

Branching and Selection

$$\frac{I \subseteq \{1, \ldots, n\} \; \forall i \in I. \Gamma \vdash P_i : U[T_i]}{\Gamma \vdash \Sigma_{i=0}^{n} (\ell_i) P_i : U[\&\{\ell_i : T_i\}]_{i \in I}} \text{(Tbranch)} \qquad \frac{k \in I \quad \Gamma \vdash P : U[T_k]}{\Gamma \vdash \langle \ell_k \rangle P : U[\oplus\{\ell_i : T_i\}_{i \in I}]} \text{(TChoice)}$$

Type System Highlights: Protocols

Input, output, and return

$$\frac{\Gamma, \tilde{x} : \tilde{S} \vdash P : U[T]}{\Gamma \vdash (?\tilde{x})P : U[?(\tilde{S}).T]} \text{(Tin)} \qquad \frac{\Gamma \vdash P : U[T] \quad \Gamma \vdash \tilde{v} : \tilde{S}}{\Gamma \vdash \langle \tilde{v} \rangle P : U[!(\tilde{S}).T]} \text{(Tout)}$$

$$\frac{\Gamma \vdash P : U[T] \quad \Gamma \vdash \tilde{v} : \tilde{S}}{\Gamma \vdash \langle \tilde{v} \rangle^{\uparrow} P : !(\tilde{S}).U[T]} \text{(Tret)}$$

Branching and Selection

$$\frac{I \subseteq \{1, \ldots, n\} \; \forall i \in I. \Gamma \vdash P_i : U[T_i]}{\Gamma \vdash \Sigma_{i=0}^{n} (\ell_i) P_i : U[\&\{\ell_i : T_i\}]_{i \in I}} \text{(Tbranch)} \qquad \frac{k \in I \quad \Gamma \vdash P : U[T_k]}{\Gamma \vdash \langle \ell_k \rangle P : U[\oplus\{\ell_i : T_i\}_{i \in I}]} \text{(TChoice)}$$

CaSPiS Check Point

A honest customer

$$HC \triangleq \overline{buy}.\langle item_k \rangle (ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, x_{price_k}, name, cc)\rangle$$

e-shop server and database

$$\begin{array}{rcl} ESHOP & \triangleq & (\nu price)(D \mid S) \\ D & \triangleq & !price.\sum_i (item_i)\langle price_i \rangle) \\ S & \triangleq & !buy.\sum_i (item_i)(\nu code)(OF_i \mid PF_i) \\ OF_i & \triangleq & \overline{price}.\langle item_i \rangle\ (?x_{price_i})\langle ord(code, item_i, x_{price_i}))\rangle^{\uparrow} \\ PF_i & \triangleq & (cancel)\mathbf{0} + (pay(code, item_i, ?y_{price_i}, ?y_{name}, ?y_{cc}))PAY \end{array}$$

Malicious user: how to redesign ESHOP?

$$MC \triangleq \overline{buy}.\langle item_k \rangle (ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, 5cents, name, cc)\rangle$$

CaSPiS Check Point

A honest customer

$$HC \triangleq \overline{buy}.\langle item_k\rangle(ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, x_{price_k}, name, cc)\rangle$$

e-shop server and database

$$\begin{aligned}
ESHOP &\triangleq (\nu price)(D \mid S)\\
D &\triangleq\ !price.\textstyle\sum_i (item_i)\langle price_i\rangle)\\
S &\triangleq\ !buy.\textstyle\sum_i (item_i)(\nu code)(OF_i \mid PF_i)\\
OF_i &\triangleq \overline{price}.\langle item_i\rangle\ (?x_{price_i})\langle ord(code, item_i, x_{price_i})\rangle^{\uparrow}\\
PF_i &\triangleq (cancel)\mathbf{0} + (pay(code, item_i, ?y_{price_i}, ?y_{name}, ?y_{cc}))PAY
\end{aligned}$$

Malicious user: how to redesign ESHOP?

$$MC \triangleq \overline{buy}.\langle item_k\rangle(ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, 5cents, name, cc)\rangle$$

CaSPiS Check Point

A honest customer

$$HC \triangleq \overline{buy}.\langle item_k \rangle (ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, x_{price_k}, name, cc)\rangle$$

e-shop server and database

$$\begin{array}{rcl}
ESHOP & \triangleq & (\nu price)(D \mid S) \\
D & \triangleq & !price. \sum_i (item_i)\langle price_i \rangle) \\
S & \triangleq & !buy. \sum_i (item_i)(\nu code)(OF_i \mid PF_i) \\
OF_i & \triangleq & \overline{price}.\langle item_i \rangle\ (?x_{price_i})\langle ord(code, item_i, x_{price_i}))\rangle^{\uparrow} \\
PF_i & \triangleq & (cancel)\mathbf{0} + (pay(code, item_i, ?y_{price_i}, ?y_{name}, ?y_{cc}))PAY
\end{array}$$

Malicious user: how to redesign ESHOP?

$$MC \triangleq \overline{buy}.\langle item_k \rangle (ord(?x_{code}, item_k, ?x_{price_k}))\langle pay(x_{code}, item_k, 5cents, name, cc)\rangle$$

CaSPiS: Advanced Principles

Service definitions: $s_k.P$, $k \cdot P$

- services expose their protocols + *generic* termination handlers
- services can be deployed dynamically, shut down and updated
- services can handle multiple requests separately

Service invocations: $\overline{s}_k.P$, $k \cdot P$

- service invocations expose their protocols + *specific* termination handlers
- sequential composition via pipelining (á la Orc)

Session termination: $r \rhd_k P$, $close$, $\blacktriangleright P$, $\dagger(k)$

- local session termination: autonomous + on partner's request
- the local closure of a session activates partner's handler (if any)
- session termination cancels all locally nested processes (including service definitions) + informs their partners

Termination Handlers

Step 1: Exchanging information about handlers

$\overline{s}_{k_1}.Q|s_{k_2}.P$ can evolve to $(\nu r)(r \rhd_{k_2} Q|r \rhd_{k_1} P)$

Step 2: Closing own session

$r \rhd_k (close \mid P)$ can evolve to $\dagger(k)| \blacktriangleright P$

Step 3: Propagate closure to nested sessions

for example: $\blacktriangleright P|Q \equiv \blacktriangleright P| \blacktriangleright Q$ and $\blacktriangleright (r \rhd_k P) \xrightarrow{\tau} \blacktriangleright P \mid \dagger(k)$

Step 4: Inform handlers

$\dagger(k) \mid k \cdot P$ can evolve to P

Default closing policy

$(\nu k_1)\overline{s}_{k_1}.(P_1|k_1 \cdot close)$ and $(\nu k_2)s_{k_2}.(P_2|k_2 \cdot close)$

Termination Handlers

Step 1: Exchanging information about handlers

$\overline{s}_{k_1}.Q|s_{k_2}.P$ can evolve to $(\nu r)(r \rhd_{k_2} Q|r \rhd_{k_1} P)$

Step 2: Closing own session

$r \rhd_k (close \mid P)$ can evolve to $\dagger(k)| \blacktriangleright P$

Step 3: Propagate closure to nested sessions

for example: $\blacktriangleright P|Q \equiv \blacktriangleright P| \blacktriangleright Q$ and $\blacktriangleright (r \rhd_k P) \xrightarrow{\tau} \blacktriangleright P \mid \dagger(k)$

Step 4: Inform handlers

$\dagger(k) \mid k \cdot P$ can evolve to P

Default closing policy

$(\nu k_1)\overline{s}_{k_1}.(P_1|k_1 \cdot close)$ and $(\nu k_2)s_{k_2}.(P_2|k_2 \cdot close)$

CaSPiS Semantics Revisited

Structural Congruence

$$\begin{array}{rcl} r \rhd_{k'} (\dagger(k)|P) & \equiv & \dagger(k)|r \rhd_{k'} P \\ (\dagger(k)|P) > Q & \equiv & \dagger(k)|(P > Q) \\ \blacktriangleright (\nu x)P & \equiv & (\nu x) \blacktriangleright P \end{array} \quad \begin{array}{rcl} \blacktriangleright r \rhd_k P & \equiv & \blacktriangleright r \rhd_k \blacktriangleright P \\ \blacktriangleright (P > Q) & \equiv & (\blacktriangleright P) > Q \\ \blacktriangleright P|Q & \equiv & \blacktriangleright P| \blacktriangleright Q \end{array} \quad \begin{array}{rcl} \blacktriangleright\blacktriangleright P & \equiv & \blacktriangleright P \\ \blacktriangleright \mathbf{0} & \equiv & \mathbf{0} \\ \blacktriangleright \dagger(k) & \equiv & \dagger(k) \end{array}$$

Reduction Semantics

$$(sync)\frac{r \text{ fresh for } \mathbb{C}[\![\cdot,\cdot]\!], P, Q}{\mathbb{C}[\![s_{k_1}.P, \overline{s}_{k_2}.Q]\!] \xrightarrow{\tau} (\nu r)\mathbb{C}[\![r \rhd_{k_2} P, r \rhd_{k_1} Q]\!]}$$

$$(Send)\frac{}{\mathbb{C}[\![r \rhd_k \mathbb{S}[\![close]\!]]\!] \xrightarrow{\tau} \mathbb{C}[\![\dagger(k) | \blacktriangleright \mathbb{S}[\![\mathbf{0}]\!]]\!]}$$

$$(Tend)\frac{}{\mathbb{C}[\![\blacktriangleright (r \rhd_k P)]\!] \xrightarrow{\tau} \mathbb{C}[\![\blacktriangleright P | \dagger(k)]\!]}$$

$$(Tsync)\frac{}{\mathbb{C}[\![\dagger(k) | k \cdot P]\!] \xrightarrow{\tau} \mathbb{C}[\![P]\!]}$$

Graceful Termination Property

Balanced process

A process where session-sides that balance with each other in pairs.

Any session-free process is balanced, and in the close-free fragment it reduces only to balanced processes

Unbalanced processes

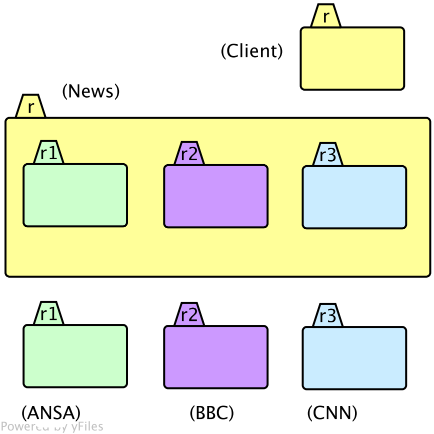
Termination of one side may lead to unbalanced terms.

Graceful termination (of session-sides)

Any possibly unbalanced term reachable from a balanced term can get balanced in a finite number of reductions.

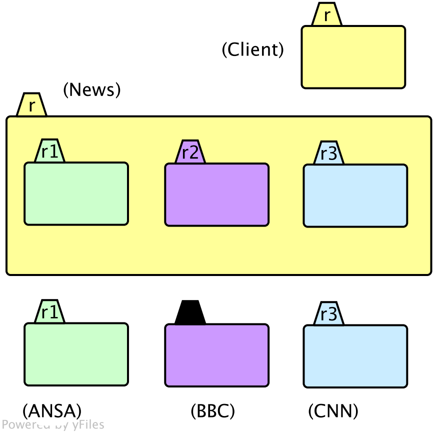
A Last Example: All Sides are Active

$$
\begin{array}{lllllll}
News & \triangleq & !(\nu k)collect_k. & (& k \cdot close & | & (\nu k_1)\overline{ANSA}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_1 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_2)\overline{BBC}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_2 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_3)\overline{CNN}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_3 \cdot (close \,|\, \dagger(k)))\ )
\end{array}
$$



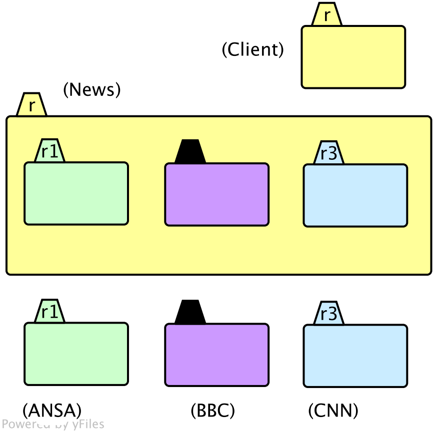
A Last Example: BBC-side Terminates

$$
\begin{array}{llllllll}
\mathit{News} & \triangleq & !(\nu k)\mathit{collect}_k. & (& k \cdot \mathit{close} & | & (\nu k_1)\overline{\mathit{ANSA}}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} & | & k_1 \cdot (\mathit{close}\,|\dagger(k))) \\
 & & & & & | & (\nu k_2)\overline{\mathit{BBC}}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} & | & k_2 \cdot (\mathit{close}\,|\dagger(k))) \\
 & & & & & | & (\nu k_3)\overline{\mathit{CNN}}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} & | & k_3 \cdot (\mathit{close}\,|\dagger(k)))\ )
\end{array}
$$



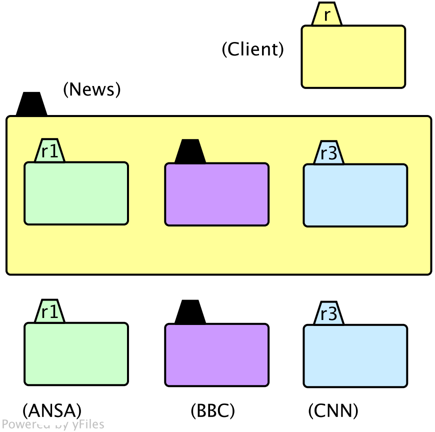
A Last Example: BBC-partner-side Terminates

$$
\begin{array}{lllllll}
News & \triangleq & !(\nu k)collect_k. & (& k \cdot close & | & (\nu k_1)\overline{ANSA}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_1 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_2)\overline{BBC}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_2 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_3)\overline{CNN}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} \quad | \quad k_3 \cdot (close \,|\, \dagger(k))) \;)
\end{array}
$$



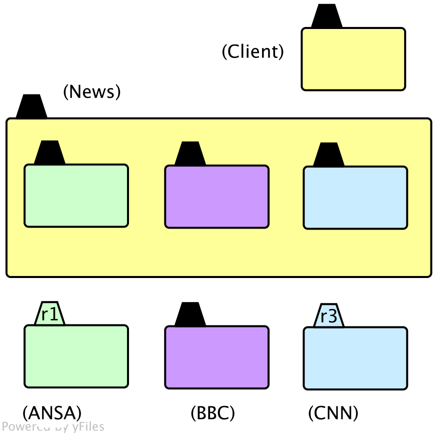
A Last Example: News-side is Triggered to Terminate

$$
\begin{array}{llllllll}
News & \triangleq & !(\nu k)collect_k. & (& k \cdot close & | & (\nu k_1)\overline{ANSA}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} & | & k_1 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_2)\overline{BBC}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} & | & k_2 \cdot (close \,|\, \dagger(k))) \\
 & & & & & | & (\nu k_3)\overline{CNN}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} & | & k_3 \cdot (close \,|\, \dagger(k)))\ )
\end{array}
$$



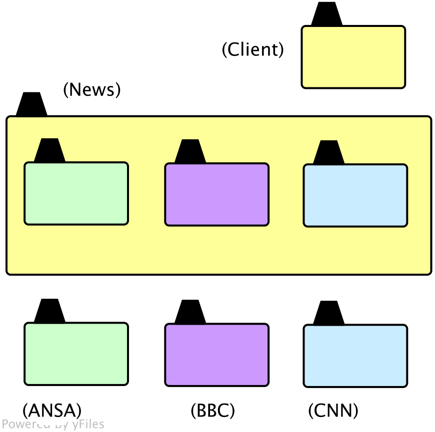
A Last Example: Client- and Nested-sides Terminate

$$\begin{array}{llllllll} News & \triangleq & !(\nu k)collect_k . & (& k \cdot close & | & (\nu k_1)\overline{ANSA}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} & | & k_1 \cdot (close\,|\dagger(k))) \\ & & & & & | & (\nu k_2)\overline{BBC}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} & | & k_2 \cdot (close\,|\dagger(k))) \\ & & & & & | & (\nu k_3)\overline{CNN}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} & | & k_3 \cdot (close\,|\dagger(k)))\ ) \end{array}$$



A Last Example: ANSA/CNN-sides Terminate

$$\begin{array}{llllllll} News & \triangleq & !(\nu k)collect_k. & (& k \cdot close & | & (\nu k_1)\overline{ANSA}_{k_1}.(!(?x)\langle x\rangle^{\uparrow} & | & k_1 \cdot (close \,|\, \dagger(k))) \\ & & & & & | & (\nu k_2)\overline{BBC}_{k_2}.(!(?x)\langle x\rangle^{\uparrow} & | & k_2 \cdot (close \,|\, \dagger(k))) \\ & & & & & | & (\nu k_3)\overline{CNN}_{k_3}.(!(?x)\langle x\rangle^{\uparrow} & | & k_3 \cdot (close \,|\, \dagger(k)))\ ) \end{array}$$



Outline

Conclusion and Future Work

CaSPiS

- Original mix of several ingredients
- Flexible and expressive
- Sound operational properties and type systems
- Only proposal, up to our knowledge, able to guarantee a disciplined termination of nested sessions.

Ongoing and future work

- Prototype implementations
- Type inference (see Leonardo Mezzina's PhD Thesis)
- Hierarchical graph models
- Abstract equivalences
- Delegation
- Multiparty sessions

THANKS FOR THE ATTENTION!

Conclusion and Future Work

CaSPiS

- Original mix of several ingredients
- Flexible and expressive
- Sound operational properties and type systems
- Only proposal, up to our knowledge, able to guarantee a disciplined termination of nested sessions.

Ongoing and future work

- Prototype implementations
- Type inference (see Leonardo Mezzina's PhD Thesis)
- Hierarchical graph models
- Abstract equivalences
- Delegation
- Multiparty sessions

THANKS FOR THE ATTENTION!