

# Reasoning about Resource Management for Message Passing Concurrency

Adrian Francalanza, Edsko de Vries<sup>1</sup> and Matthew Hennessy<sup>1</sup>

Dublin Concurrency Workshop  
April 14-15, 2011

---

<sup>1</sup>The financial support of SFI is gratefully acknowledged.

# Motivation

## 3 Clients

```
CLIENT1  $\triangleq$  rec  $X$ .  
    alloc  $x_1$ . alloc  $x_2$ .  
    getTime!  $x_1$ .  $x_1$ ?y.  
    getDate!  $x_2$ .  $x_2$ ?z.  
    report!(y, z).  $X$ 
```

# Motivation

## 3 Clients

```
CLIENT2  $\triangleq$  rec X.  
    alloc x.  
    getTime !x. x?y.  
    getDate !x. x?z.  
    report!(y, z). X
```

# Motivation

## 3 Clients

```
CLIENT3  $\triangleq$  rec X.  
    alloc x.  
    getTime!x. x?y.  
    getDate!x. x?z.  
    free x. report!(y, z). X
```

# Motivation

## Typical Servers

$$\text{TIME SERVER} \triangleq \text{rec } Y. \text{getTime?}x. \text{ } x!\text{time}. \text{ } Y$$
$$\text{DATE SERVER} \triangleq \text{rec } Z. \text{getDate?}x. \text{ } x!\text{date}. \text{ } Z$$

# Agenda

1. We want to show that the clients are behaviourally equivalent wrt. well-behaved servers.

# Agenda

1. We want to show that the clients are behaviourally equivalent *wrt.* well-behaved servers.
2. We want to **order** these clients, based on their channel usage.

# Agenda

1. We want to show that the clients are behaviourally equivalent *wrt.* well-behaved servers.
2. We want to order these clients, based on their channel usage.
3. We want to determine these relations **compositionally** (without concerns of the specific server implementations.)

# Towards a Behavioural Equivalence

CLIENT<sub>1</sub>       $\not\approx_{\text{ctxt}}$       CLIENT<sub>2</sub>

# Towards a Behavioural Equivalence

```
rec X.alloc x1. alloc x2.  
  getTime!x1. x1?y.  
  getDate!x2. x2?z.  
  report!(y, z). X
```

$\not\approx_{\text{ctxt}}$

```
rec X.alloc x.  
  getTime!x. x?y.  
  getDate!x. x?z.  
  report!(y, z). X
```

# Towards a Behavioural Equivalence

```
rec X.alloc x1. alloc x2.  
  getTime!x1. x1?y.  
  getDate!x2. x2?z.  
  report!(y, z). X
```

$\not\approx_{\text{ctxt}}$

```
rec X.alloc x.  
  getTime!x. x?y.  
  getDate!x. x?z.  
  report!(y, z). X
```

## Generic Servers

```
GIDDYTIME SERVER  $\triangleq$  rec Y. getTime?x.(x!time. Y || x!time)  
DATE SERVER  $\triangleq$  rec Z. getDate?x. x!date. Z
```

# Towards a Behavioural Equivalence

## First Requirement

We need to express our assumptions about a well-behaved server.

# Towards a Behavioural Equivalence

## First Requirement

We need to express our assumptions about a **well-behaved** server.

## Implicit Assumption

We want to reason about **well-behaved** clients.

$$\text{RAVINGCLIENT} \triangleq \begin{array}{l} \text{rec } X.\text{alloc } x. \\ \quad \text{getTime!}x. \text{free } x. x?y. \\ \quad \text{getDate!}x. x?z. \\ \quad \text{report!}(y, z). X \end{array}$$

# A Substructural Type System

## Type language

$\mathbf{T} ::= [\vec{\mathbf{T}}]^a$  (channel type)

$a ::= \omega$  (unrestricted)  
|  $1$  (affine)  
|  $(\bullet, i)$  (unique after  $i$  steps,  $i \in \mathbb{N}$ )

# A Substructural Type System

## Type language

$\mathbf{T} ::= [\vec{\mathbf{T}}]^a$  (channel type)

$a ::= \omega$  (unrestricted)  
|  $1$  (affine)  
|  $(\bullet, i)$  (unique after  $i$  steps,  $i \in \mathbb{N}$ )

# A Substructural Type System

## Type language

$$\begin{array}{lcl} a & ::= & \omega \quad (\text{unrestricted}) \\ & | & 1 \quad (\text{affine}) \\ & | & \dots \end{array}$$

## Describing Well-Behaved Servers

$$\Gamma \triangleq \text{getTime} : [[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate} : [[\mathbf{T}_{\text{date}}]^1]^\omega$$

# A Substructural Type System

## Type language

$$\begin{array}{lcl} a & ::= & \omega \quad (\text{unrestricted}) \\ & | & 1 \quad (\text{affine}) \\ & | & \dots \end{array}$$

## Describing Well-Behaved Servers

$$\Gamma \triangleq \text{getTime}:[[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate}:[[\mathbf{T}_{\text{date}}]^1]^\omega$$

$$\Gamma \vdash \text{TIME SERVER} \parallel \text{DATE SERVER}$$

# A Substructural Type System

## Type language

$$\begin{array}{lcl} a & ::= & \omega \quad (\text{unrestricted}) \\ & | & 1 \quad (\text{affine}) \\ & | & \dots \end{array}$$

## Describing Well-Behaved Servers

$$\Gamma \triangleq \text{getTime}:[[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate}:[[\mathbf{T}_{\text{date}}]^1]^\omega$$

$$\Gamma \not\vdash \text{GIDDYTIME SERVER} \parallel \text{DATE SERVER}$$

# A Substructural Type System

## Type language

$$\begin{array}{lcl} a & ::= & \omega \quad (\text{unrestricted}) \\ & | & 1 \quad (\text{affine}) \\ & | & \dots \end{array}$$

## Describing Well-Behaved Servers

$$\Gamma \triangleq \text{getTime}:[[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate}:[[\mathbf{T}_{\text{date}}]^1]^\omega$$

$$\Gamma \models \text{CLIENT}_1 \approx_{\text{ctxt}} \text{CLIENT}_2$$

# A Substructural Type System

## Type language

$\mathbf{T} ::= [\vec{\mathbf{T}}]^a$  (channel type)

$a ::= \omega$  (unrestricted)  
|  $1$  (affine)  
|  $(\bullet, i)$  (unique after  $i$  steps,  $i \in \mathbb{N}$ )

# A Substructural Type System

## Type language

$$\begin{array}{lcl} a & ::= & \omega \quad (\text{unrestricted}) \\ & | & 1 \quad (\text{affine}) \\ & | & (\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N}) \end{array}$$

## Typing Clients

$$\begin{aligned} \text{CLIENT}_3 &\triangleq \text{rec } X. \\ &\quad \text{alloc } x. \\ &\quad \text{getTime! } x. \ x?y. \\ &\quad \text{getDate! } x. \ x?z. \\ &\quad \text{free } x. \ \text{report!}(y, z). \ X \end{aligned}$$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
 $x : [\mathbf{T}_{\text{time}}]^{(\bullet, 0)}$       alloc  $x.$   
     $\searrow \text{getTime}!x. x?y.$   
     $\text{getDate}!x. x?z.$   
    free  $x.$  report!( $y, z$ ).  $X$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
 $x : [\mathbf{T}_{\text{time}}]^{(\bullet, 1)}$        $\text{alloc } x.$   
                                   $\text{getTime!}x. \searrow x?y.$   
                                   $\text{getDate!}x. x?z.$   
                                   $\text{free } x. \text{report!}(y, z). X$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
   $\text{alloc } x.$   
   $\text{getTime!}x. x?y.$   
     $\searrow \text{getDate!}x. x?z.$   
   $\text{free } x. \text{report!}(y, z). X$

$x : [\mathbf{T}_{\text{time}}]^{(\bullet, 0)}$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
   $\text{alloc } x.$   
   $\text{getTime! } x. \, x?y.$   
     $\searrow \text{getDate! } x. \, x?z.$   
   $\text{free } x. \, \text{report!} (y, z). \, X$

$x : [\mathbf{T}_{\text{date}}]^{(\bullet, 0)}$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
 $x : [\mathbf{T}_{\text{date}}]^{(\bullet, 1)}$   
alloc  $x.$   
getTime!  $x.$   $x?y.$   
getDate!  $x.$   $\searrow x?z.$   
free  $x.$  report!  $(y, z).$   $X$

# A Substructural Type System

## Type language

$a ::= \omega \quad (\text{unrestricted})$   
|  $1 \quad (\text{affine})$   
|  $(\bullet, i) \quad (\text{unique after } i \text{ steps, } i \in \mathbb{N})$

## Typing Clients

$\text{CLIENT}_3 \triangleq \text{rec } X.$   
alloc  $x$ .  
getTime ! $x$ .  $x?y$ .  
getDate ! $x$ .  $x?z$ .  
 $x : [\mathbf{T}_{\text{date}}]^{(\bullet, 0)}$        $\searrow \text{free } x. \text{report!}(y, z). X$

# A Substructural Type System

## Typing Rules

$$\frac{\Gamma, x : [\vec{T}]^{(\bullet, 0)} \vdash P}{\Gamma \vdash \text{alloc } x.P} \text{ TALL}$$

$$\frac{\Gamma \vdash P}{\Gamma, u : [\vec{T}]^{(\bullet, 0)} \vdash \text{free } u.P} \text{ TFREE}$$

# A Substructural Type System

## Typing Rules

$$\frac{\Gamma, x : [\vec{T}]^{(\bullet, 0)} \vdash P}{\Gamma \vdash \text{alloc } x.P} \text{ TALL}$$

$$\frac{\Gamma \vdash P}{\Gamma, u : [\vec{T}]^{(\bullet, 0)} \vdash \text{free } u.P} \text{ TFREE}$$

$$\frac{\Gamma, u : [\vec{T}_2]^{(\bullet, 0)} \vdash P}{\Gamma, u : [\vec{T}_1]^{(\bullet, 0)} \vdash P} \text{ TREV}$$

# A Substructural Type System

## Structural Rules

$$\frac{\mathbf{T} = \mathbf{T}_1 \circ \mathbf{T}_2 \quad \Gamma, u:\mathbf{T}_1, u:\mathbf{T}_2 \vdash P}{\Gamma, u:\mathbf{T} \vdash P} \text{TSPL}$$

# A Substructural Type System

## Structural Rules

$$\frac{\mathbf{T} = \mathbf{T}_1 \circ \mathbf{T}_2 \quad \Gamma, u:\mathbf{T}_1, u:\mathbf{T}_2 \vdash P}{\Gamma, u:\mathbf{T} \vdash P} \text{TSPL}$$

$$\frac{}{[\vec{\mathbf{T}}]^\omega = [\vec{\mathbf{T}}]^\omega \circ [\vec{\mathbf{T}}]^\omega} \text{PUNR}$$

# A Substructural Type System

## Structural Rules

$$\frac{\mathbf{T} = \mathbf{T}_1 \circ \mathbf{T}_2 \quad \Gamma, u : \mathbf{T}_1, u : \mathbf{T}_2 \vdash P}{\Gamma, u : \mathbf{T} \vdash P} \text{TSPL}$$

$$\overline{[\vec{\mathbf{T}}]^\omega} = [\vec{\mathbf{T}}]^\omega \circ [\vec{\mathbf{T}}]^\omega \text{ PUNR}$$

$$\overline{[\vec{\mathbf{T}}]^{(\bullet, i)}} = [\vec{\mathbf{T}}]^1 \circ [\vec{\mathbf{T}}]^{(\bullet, i+1)} \text{ PUNQ}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$$\frac{\Gamma \vdash P}{\Gamma, u:[\vec{T}]^1, \overrightarrow{v:\vec{T}} \vdash u!\vec{v}.P} \text{TOA}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$$\frac{\Gamma \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^1, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOA}$$

$$\frac{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet,i)} \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet,i+1)}, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOU}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$$\frac{\Gamma \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^1, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOA}$$

$$\frac{\Gamma, \overrightarrow{x:\mathbf{T}} \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^1 \vdash u?\vec{x}.P} \text{TINA}$$

$$\frac{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet, i)} \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet, i+1)}, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOU}$$

$$[\vec{\mathbf{T}}]^{(\bullet, i)} = [\vec{\mathbf{T}}]^1 \circ [\vec{\mathbf{T}}]^{(\bullet, i+1)}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$\Gamma_1 \quad \Gamma_2$

$$\frac{\Gamma \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^1, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOA}$$

$$\frac{\Gamma, \overrightarrow{x:\mathbf{T}} \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^1 \vdash u?\vec{x}.P} \text{TINA}$$

$$\frac{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet, i)} \vdash P}{\Gamma, u:[\vec{\mathbf{T}}]^{(\bullet, i+1)}, \overrightarrow{v:\mathbf{T}} \vdash u!\vec{v}.P} \text{TOU}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR} \quad c : [\vec{\mathbf{T}}]^{(\bullet, 1)}, \quad \begin{matrix} \Gamma_1 \\ c : [\vec{\mathbf{T}}]^{(\bullet, 1)} \end{matrix}, \quad \begin{matrix} \Gamma_2 \\ c : [\vec{\mathbf{T}}]^1 \end{matrix} \text{ consistent}$$

$$\frac{\Gamma \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOA}$$

$$\frac{\Gamma, \overrightarrow{x : \mathbf{T}} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1 \vdash u?x.P} \text{TINA}$$

$$\frac{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet, i)} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet, i+1)}, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOU}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$c : [\vec{\mathbf{T}}]^{(\bullet,1)}, \quad \Gamma_1$   
 $c : [\vec{\mathbf{T}}]^{(\bullet,0)}, \quad \Gamma_2$

consistent  
not consistent!!

$$\frac{\Gamma \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOA}$$

$$\frac{\Gamma, \overrightarrow{x : \mathbf{T}} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1 \vdash u?x.P} \text{TINA}$$

$$\frac{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet,i)} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet,i+1)}, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOU}$$

# A Substructural Type System

## Typing Rules for Channel Usage

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1, \Gamma_2 \vdash P \parallel Q} \text{TPAR}$$

$c : [\vec{\mathbf{T}}]^{(\bullet, 1)}$ ,	$\Gamma_1$	$\Gamma_2$
$c : [\vec{\mathbf{T}}]^{(\bullet, 0)}$ ,	$c : [\vec{\mathbf{T}}]^1$	consistent
$c : [\vec{\mathbf{T}}]^{(\bullet, 0)}$ ,	$c : [\vec{\mathbf{T}}]^1$	not consistent!!
	$c : [\vec{\mathbf{T}}]^\omega$	not consistent!!

$$\frac{\Gamma \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOA}$$

$$\frac{\Gamma, \overrightarrow{x : \mathbf{T}} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^1 \vdash u?x.P} \text{TINA}$$

$$\frac{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet, i)} \vdash P}{\Gamma, u : [\vec{\mathbf{T}}]^{(\bullet, i+1)}, \overrightarrow{v : \mathbf{T}} \vdash u!v.P} \text{TOU}$$

## Configurations

$\Gamma \triangleleft P$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Transitions

$$\Gamma, c:[\vec{\mathbf{T}}]^a \triangleleft c!\vec{d}.Q \quad \xrightarrow{c!\vec{d}}$$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Transitions

$$\Gamma, c:[\vec{\mathbf{T}}]^a \triangleleft c!\vec{d}.Q \quad \xrightarrow{c!\vec{d}} \quad \Gamma, \quad \vec{d}:\vec{\mathbf{T}} \triangleleft Q$$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Transitions

$$\Gamma, c:[\vec{\mathbf{T}}]^\omega \triangleleft c!\vec{d}.Q \xrightarrow{c!\vec{d}} \Gamma, \textcolor{red}{c}:[\vec{\mathbf{T}}]^\omega, \vec{d}:\vec{\mathbf{T}} \triangleleft Q$$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Transitions

$$\Gamma, c:[\vec{\mathbf{T}}]^{\textcolor{red}{1}} \triangleleft c!\vec{d}.Q \quad \xrightarrow{c!\vec{d}} \quad \Gamma, \quad \vec{d}:\vec{\mathbf{T}} \triangleleft Q$$

## Configurations

$\Gamma \triangleleft P$  implies  $\exists \Gamma'$  such that:

- ▶  $\Gamma' \vdash P$
- ▶  $\Gamma$  is consistent with  $\Gamma'$

## Transitions

$$\Gamma, c:[\vec{\mathbf{T}}]^{(\bullet, i+1)} \triangleleft c!\vec{d}.Q \xrightarrow{c!\vec{d}} \Gamma, c:[\vec{\mathbf{T}}]^{(\bullet, i)}, \vec{d}:\vec{\mathbf{T}} \triangleleft Q$$

# Bisimulation Equivalence

## First Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

# Bisimulation Equivalence

## First Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.

# Bisimulation Equivalence

## First Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft Q'$  such that  
 $\Gamma' \models P' \approx_{\text{bis}} Q'$

# Bisimulation Equivalence

## First Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft Q'$  such that  
 $\Gamma' \models P' \approx_{\text{bis}} Q'$
- ▶ the dual case.

# Bisimulation Equivalence

## First Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft Q'$  such that  
 $\Gamma' \models P' \approx_{\text{bis}} Q'$
- ▶ the dual case.

## Theorem (Subject Reduction)

$\Gamma \triangleleft P$  is a configuration and  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  then  $\Gamma' \triangleleft P'$  is a configuration

# Bisimulation Equivalence

Example (Comparing the two basic Clients)

For  $\Gamma = \text{getTime} : [[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate} : [[\mathbf{T}_{\text{date}}]^1]^\omega$

$$\left\{ \begin{array}{c} \langle \Gamma \\ \langle \\ \langle \\ \langle \\ , \text{CLIENT}_1 \\ , \text{CLIENT}_2 \\ \rangle \\ \rangle \\ \rangle \\ \rangle \end{array} \right\}$$

# Bisimulation Equivalence

Example (Comparing the two basic Clients)

For  $\Gamma = \text{getTime}:[[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate}:[[\mathbf{T}_{\text{date}}]^1]^\omega$

$$\left\{ \begin{array}{c} \langle \Gamma, \text{CLIENT}_1, \text{CLIENT}_2 \rangle \\ \langle \Gamma, \text{getTime!}c_1. c_1?y.P\{c_1, c_2/x_1, x_2\}, \text{getTime!}c_1. c_1?y.Q\{c_1/x_1\} \rangle \\ \langle \langle \rangle \rangle \end{array} \right\}$$

# Bisimulation Equivalence

Example (Comparing the two basic Clients)

For  $\Gamma = \text{getTime}:[[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate}:[[\mathbf{T}_{\text{date}}]^1]^\omega$

$$\left\{ \begin{array}{c} \langle \Gamma, \text{CLIENT}_1, \text{CLIENT}_2 \rangle \\ \langle \Gamma, \text{getTime}!c_1. c_1?y.P\{c_1, c_2/x_1, x_2\}, \text{getTime}!c_1. c_1?y.Q\{c_1/x_1\} \rangle \\ \langle \Gamma, c_1:[\mathbf{T}_{\text{time}}]^1, c_1?y.P\{c_1, c_2/x_1, x_2\}, c_1?y.Q\{c_1/x_1\} \rangle \\ \langle \rangle \end{array} \right\}$$

# Bisimulation Equivalence

Example (Comparing the two basic Clients)

For  $\Gamma = \text{getTime} : [[\mathbf{T}_{\text{time}}]^1]^\omega, \text{getDate} : [[\mathbf{T}_{\text{date}}]^1]^\omega$

$$\left\{ \begin{array}{lll} \langle \Gamma & , \text{CLIENT}_1 & , \text{CLIENT}_2 \\ \langle \Gamma & , \text{getTime}!c_1. c_1?y.P\{c_1, c_2/x_1, x_2\}, \text{getTime}!c_1. c_1?y.Q\{c_1/x_1\} \rangle \\ \langle \Gamma, c_1 : [\mathbf{T}_{\text{time}}]^1, c_1?y.P\{c_1, c_2/x_1, x_2\} & , c_1?y.Q\{c_1/x_1\} \rangle \\ \langle \Gamma & , \text{getDate}!c_2.P'\{c_1, c_2/x_1, x_2\} & , \text{getDate}!c_1.Q\{c_1/x_1\} \rangle \\ \langle \dots & , \dots & , \dots \end{array} \right\}$$

# Bisimulation Equivalence

## Renaming Modulo $\Gamma$

Let  $\sigma_\Gamma : \text{NAME} \mapsto \text{NAME}$  range over bijective name substitutions satisfying the constraint that

$$c \in \text{dom}(\Gamma) \text{ implies } c\sigma_\Gamma = c\sigma_\Gamma^{-1} = c$$

# Bisimulation Equivalence

## Second Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \sigma_\Gamma \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft Q'$  such that  $\Gamma' \models P' \approx_{\text{bis}} Q'$
- ▶  $\Gamma \triangleleft Q \xrightarrow{\alpha} \Gamma' \triangleleft Q'$  implies  $\Gamma \triangleleft P \sigma_\Gamma \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft P'$  such that  $\Gamma' \models P' \approx_{\text{bis}} Q'$

# Bisimulation Equivalence

## Second Attempt

$\Gamma \models P \approx_{\text{bis}} Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\alpha} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \sigma_\Gamma \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft Q'$  such that  $\Gamma' \models P' \approx_{\text{bis}} Q'$
- ▶  $\Gamma \triangleleft Q \xrightarrow{\alpha} \Gamma' \triangleleft Q'$  implies  $\Gamma \triangleleft P \sigma_\Gamma \xrightarrow{\hat{\alpha}} \Gamma' \triangleleft P'$  such that  $\Gamma' \models P' \approx_{\text{bis}} Q'$

## Equality

$\Gamma \models \text{CLIENT}_1 \approx_{\text{bis}} \text{CLIENT}_2 \approx_{\text{bis}} \text{CLIENT}_3$

# Partial Order

## Costed Actions

$$\frac{\text{c not allocated}}{\Gamma \triangleleft \text{alloc } x.P \xrightarrow[\textcolor{red}{+1}]{\tau} \Gamma \triangleleft P\{c/x\}} \text{LALL}$$

$$\frac{}{\Gamma \triangleleft \text{free } c.P \xrightarrow[\textcolor{red}{-1}]{\tau} \Gamma \triangleleft P} \text{LFREE}$$

$$\frac{}{\Gamma, c:[\vec{\mathbf{T}}]^{(\bullet, i+1)} \triangleleft c!\vec{d}.P \xrightarrow[0]{c!\vec{d}} c:[\vec{\mathbf{T}}]^{(\bullet, i)}, \vec{d}:\vec{\mathbf{T}} \triangleleft P} \text{LOUTU}$$

# Partial Order

## Amortised Typed Bisimulation

$\Gamma \models P \sqsubseteq_{\text{bis}}^n Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\mu} \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q\sigma_\Gamma \xrightarrow{\hat{\mu}}_I \Gamma' \triangleleft Q'$  where  
 $\Gamma' \models P' \sqsubseteq_{\text{bis}}^{n+l-k} Q'$
- ▶  $\Gamma \triangleleft Q \xrightarrow{\mu} _I \Gamma' \triangleleft Q'$  implies  $\Gamma \triangleleft P\sigma_\Gamma \xrightarrow{\hat{\mu}}_k \Gamma' \triangleleft P'$  where  
 $\Gamma' \models P' \sqsubseteq_{\text{bis}}^{n+l-k} Q'$

# Partial Order

## Amortised Typed Bisimulation

$\Gamma \models P \sqsubseteq_{\text{bis}}^n Q$  if

- ▶  $\Gamma \triangleleft P$  and  $\Gamma \triangleleft Q$  are configurations.
- ▶  $\Gamma \triangleleft P \xrightarrow{\mu} k \Gamma' \triangleleft P'$  implies  $\Gamma \triangleleft Q \sigma_\Gamma \xrightarrow{\hat{\mu}} l \Gamma' \triangleleft Q'$  where  
 $\Gamma' \models P' \sqsubseteq_{\text{bis}}^{n+l-k} Q'$
- ▶  $\Gamma \triangleleft Q \xrightarrow{\mu} l \Gamma' \triangleleft Q'$  implies  $\Gamma \triangleleft P \sigma_\Gamma \xrightarrow{\hat{\mu}} k \Gamma' \triangleleft P'$  where  
 $\Gamma' \models P' \sqsubseteq_{\text{bis}}^{n+l-k} Q'$

## Ordering

$\Gamma \models \text{CLIENT}_3 \sqsubseteq_{\text{bis}} \text{CLIENT}_2 \sqsubseteq_{\text{bis}} \text{CLIENT}_1$

# Compositionality

## Theorem

$\Gamma, \Gamma' \models P \sqsubset_{bis}^n Q$  and  $\Gamma' \vdash R$  implies:

- ▶  $\Gamma \models P \parallel R \sqsubset_{bis}^n Q \parallel R$
- ▶  $\Gamma \models R \parallel P \sqsubset_{bis}^n R \parallel Q$

# Conclusions

## Main contributions

- ▶ A compositional framework for reasoning about resource usage in a concurrent setting.
- ▶ A justification of such framework in terms of an independent behavioural contextual preorder on processes.

## Future work

- ▶ Apply theory to analyse protocol refactoring.
- ▶ Extend the theory to a Higher-Order Resource  $\pi$ -calculus.

## References

- [1] E. DeVries, A. Francalanza, M. Hennessy  
Uniqueness Typing for Resource Management in Message Passing Concurrency,  
*Linerity 2009.* (Journal version submitted for publication)
- [2] E. DeVries, A. Francalanza, M. Hennessy  
Reasoning about Resource Management for Message Passing Concurrency,  
*Places 2011.*