

# A Concurrent Object Calculus

Andy Gordon, Microsoft Research

Paul Hankin, Cambridge University

$$\mathbf{conc}\zeta = \mathbf{imp}\zeta + \pi$$

## Why is a solution of $\mathbf{conc}\zeta = \mathbf{imp}\zeta + \pi$ interesting?

- A solution  $\mathbf{conc}\zeta$  could help describe aspects of the vast amount of software coded using objects and threads.
- The  $\mathbf{imp}\zeta$ -calculus enjoys lots of expressive type systems (supporting functions, classes, subtyping, inheritance, ... ).  
The  $\pi$ -calculus enjoys lots of powerful proof tools.  
Therefore, a  $\mathbf{conc}\zeta$ -calculus would enjoy both!

## Abadi and Cardelli's $\text{imp}\zeta$ -calculus

### Syntax of (one form of) the $\text{imp}\zeta$ -calculus

|  |                             |
|--|-----------------------------|
| $u, v ::=$                               | results                     |
| $x$                                      | variable                    |
| $p$                                      | name (pointer to an object) |
| $a, b, c ::=$                            | terms                       |
| $u$                                      | result                      |
| $[l_i = \zeta(x_i)b_i \quad i \in 1..n]$ | object                      |
| $a.l$                                    | select method               |
| $a.l \leftarrow \zeta(x)b$               | update method               |
| $\text{clone}(a)$                        | clone object                |
| $\text{let } x=a \text{ in } b$          | let                         |

## An Example Computation in $\text{imp}\zeta$

Let the object  $d = [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x]$ .

Running the expression  $d.\ell$  goes as follows:

- $d.\ell$  stores the object  $d$  at a fresh location  $p$ , and runs  $p.\ell$ ,
- which fetches the method  $\zeta(x)x.\ell \Leftarrow \zeta(y)x$ ,  
and runs  $p.\ell \Leftarrow \zeta(y)p$ ,
- which updates location  $p$  with the method  $\zeta(y)p$ ,  
and returns the result  $p$ .

## Semantics of $\text{imp}\zeta$ in More Detail

- Abadi and Cardelli's relation  $\sigma \bullet S \vdash a \rightsquigarrow v \bullet \sigma'$  is a big-step environment-based semantics.
- Gordon, Hankin, and Lassen's relation  $\langle \sigma \parallel a \rangle \rightarrow \langle \sigma' \parallel a' \rangle$  is an equivalent small-step substitution-based semantics, where
  - $\langle \sigma \parallel a \rangle$  and  $\langle \sigma' \parallel a' \rangle$  are **configurations**, and
  - $\sigma$  and  $\sigma'$  are **stores** of the form  $p_1 \mapsto d_1, \dots, p_n \mapsto d_n$  where each  $d_i$  is an object.

## Small-step Reductions of the Earlier Example

To illustrate Gordon, Hankin, and Lassen's semantics:

$$\langle \emptyset \parallel [\ell = \zeta(x)x.l \Leftarrow \zeta(y)x].l \rangle$$

$$\rightarrow \langle p \mapsto [\ell = \zeta(x)x.l \Leftarrow \zeta(y)x] \parallel p.l \rangle$$

$$\rightarrow \langle p \mapsto [\ell = \zeta(x)x.l \Leftarrow \zeta(y)x] \parallel p.l \Leftarrow \zeta(y)p \rangle$$

$$\rightarrow \langle p \mapsto [\ell = \zeta(y)p] \parallel p \rangle$$

where  $p$  is any fresh name.

## Our Hidden Agenda

We seek a concurrent extension of the **imp** $\zeta$ -calculus whose semantics has the simplicity and elegance of the Berry/Boudol/Milner chemical semantics of  $\pi$ .

Moreover, we seek to avoid:

- Auxiliary notions of stores, threads, configurations, evaluation contexts, labelled transitions
- Continuation-passing encodings of expressions
- Encodings of objects (All the known typed encodings of the  $\zeta$ -calculus into the  $\lambda$ -calculus or the  $\pi$ -calculus are fairly complex.)

## Our Solution

$$\mathbf{conc}\zeta \triangleq \mathbf{imp}\zeta + \overbrace{\{a \vec{\Gamma} b, (\nu p)a\}}^{\text{from } \pi} + \overbrace{\{p \mapsto [\ell_i = \zeta(x_i)b_i]\}}^{\text{from } \mathbf{imp}\zeta \text{ semantics}}$$

- Syntax is a superset of the  $\mathbf{imp}\zeta$ -calculus
- Channels, and hence the  $\pi$ -calculus, may be encoded
- Like in CML semantics,  $a \vec{\Gamma} b$  is asymmetric
- Well-formedness conditions enforced by a type system
- A small-step substitution-based semantics:  $a \equiv b, a \rightarrow b$

## Our $\text{conc}\zeta$ -calculus

### Syntax of the $\text{conc}\zeta$ -calculus

| $a, b, c ::=$                                      | terms                                  |
|--|--|
| $u$  | result (a variable $x$ or a name $p$ ) |
| $p \mapsto [\ell_i = \zeta(x_i) b_i]^{i \in 1..n}$ | object denomination                    |
| $u.l$  | select method                          |
| $u.l \Leftarrow \zeta(x) b$                        | update method                          |
| $\text{clone}(u)$                                  | clone object                           |
| $\text{let } x = a \text{ in } b$                  | let                                    |
| $a \uparrow b$                                     | composition                            |
| $(\nu p) a$  | restriction                            |

## Chemical Reductions of the Earlier Example

The following illustrates some conventions, structural congruences, and reductions of **conc** $\zeta$ :

$$\begin{aligned}
 & [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x].\ell \\
 & = \text{let } z = [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x] \text{ in } z.\ell \\
 & = \text{let } z = (\nu p)(p \mapsto [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x] \vec{\Gamma} p) \text{ in } z.\ell \\
 & \equiv (\nu p)(p \mapsto [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x] \vec{\Gamma} \text{let } z = p \text{ in } z.\ell) \\
 & \rightarrow (\nu p)(p \mapsto [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x] \vec{\Gamma} p.\ell) \\
 & \rightarrow (\nu p)(p \mapsto [\ell = \zeta(x)x.\ell \Leftarrow \zeta(y)x] \vec{\Gamma} p.\ell \Leftarrow \zeta(y)p) \\
 & \rightarrow (\nu p)(p \mapsto [\ell = \zeta(y)p] \vec{\Gamma} p)
 \end{aligned}$$

## Example of Interdependent Denominations

We may generate a cyclic dependency between denominations:

$$\begin{array}{l}
 \text{let } x_1 = [\ell = \zeta(y_1)y_1] \text{ in} \quad \rightarrow^* \quad (\nu p_1)(\nu p_2)( \\
 \text{let } x_2 = [\ell = \zeta(y_2)x_1] \text{ in} \quad \quad \quad p_1 \mapsto [\ell = \zeta(y_1)p_2] \dot{\mapsto} \\
 x_1.\ell \leftarrow \zeta(y_1)x_2 \quad \quad \quad p_2 \mapsto [\ell = \zeta(y_2)p_1] \dot{\mapsto} p_1)
 \end{array}$$

This illustrates the need in our calculus for name scoping (e.g.,  $(\nu p_1)$ ) to be syntactically separate from denomination (e.g.,  $p_1 \mapsto [\ell = \zeta(y_1)p_2]$ ).

## Semantics Rules of $\text{conc}\zeta$

### Structural congruence $a \equiv b$

$$(a \dot{\rightarrow} b) \dot{\rightarrow} c \equiv a \dot{\rightarrow} (b \dot{\rightarrow} c)$$

$$(a \dot{\rightarrow} b) \dot{\rightarrow} c \equiv (b \dot{\rightarrow} a) \dot{\rightarrow} c$$

$$(\nu p)(\nu q)a \equiv (\nu q)(\nu p)a$$

$$(\nu p)(a \dot{\rightarrow} b) \equiv a \dot{\rightarrow} (\nu p)b \text{ if } p \notin \text{fn}(a)$$

$$(\nu p)(a \dot{\rightarrow} b) \equiv ((\nu p)a) \dot{\rightarrow} b \text{ if } p \notin \text{fn}(b)$$

$$\text{let } x = (\text{let } y = a \text{ in } b) \text{ in } c \equiv \text{let } y = a \text{ in } (\text{let } x = b \text{ in } c) \text{ if } y \notin \text{fv}(c)$$

$$(\nu p)\text{let } x = a \text{ in } b \equiv \text{let } x = (\nu p)a \text{ in } b \text{ if } p \notin \text{fn}(b)$$

$$a \dot{\rightarrow} \text{let } x = b \text{ in } c \equiv \text{let } x = (a \dot{\rightarrow} b) \text{ in } c$$

**Reduction  $a \rightarrow b$ : basic reduction rules**

For the first three rules, let  $d = [\ell_i = \zeta(x_i)b_i \quad i \in 1..n]$ .

$(p \mapsto d) \dot{\mapsto} p.\ell_j \rightarrow (p \mapsto d) \dot{\mapsto} b_j\{x_j \leftarrow p\}$  if  $j \in 1..n$

$(p \mapsto d) \dot{\mapsto} (p.\ell_j \leftarrow \zeta(x)b) \rightarrow (p \mapsto d') \dot{\mapsto} p$

if  $j \in 1..n$ ,  $d' = [\ell_j = \zeta(x)b, \ell_i = \zeta(x_i)b_i \quad i \in (1..n) - \{j\}]$

$(p \mapsto d) \dot{\mapsto} \text{clone}(p) \rightarrow (p \mapsto d) \dot{\mapsto} (\forall q)(q \mapsto d \dot{\mapsto} q)$  if  $q \notin \text{fn}(d)$

$\text{let } x=p \text{ in } b \rightarrow b\{x \leftarrow p\}$

**Reduction  $a \rightarrow b$ : congruence and structural rules**

$(\forall p)a \rightarrow (\forall p)a'$  if  $a \rightarrow a'$

$a \dot{\mapsto} b \rightarrow a' \dot{\mapsto} b$  if  $a \rightarrow a'$

$b \dot{\mapsto} a \rightarrow b \dot{\mapsto} a'$  if  $a \rightarrow a'$

$\text{let } x=a \text{ in } b \rightarrow \text{let } x=a' \text{ in } b$  if  $a \rightarrow a'$

$a \rightarrow b$  if  $a \equiv a'$ ,  $a' \rightarrow b'$ ,  $b' \equiv b$

## Synchronisation primitives

$$\mathbf{conc}\zeta_m \stackrel{\Delta}{=} \mathbf{conc}\zeta + \overbrace{\{ \mathit{acquire}(u), \mathit{release}(u) \} + \{ p \mapsto \mathit{locked}, p \mapsto \mathit{unlocked} \}}^{\text{mutex denominations}}$$

**Reduction  $a \rightarrow b$ : additional rules for mutexes**

$$\begin{array}{l} (p \mapsto \mathit{unlocked}) \dot{\vdash} \mathit{acquire}(p) \rightarrow (p \mapsto \mathit{locked}) \dot{\vdash} p \\ (p \mapsto d) \dot{\vdash} \mathit{release}(p) \rightarrow (p \mapsto \mathit{unlocked}) \dot{\vdash} p \quad \text{if } d \in \{\mathit{locked}, \mathit{unlocked}\} \end{array}$$

- Mutexes may be implemented using a single queue.
- Mutexes may be encoded within  $\mathbf{conc}\zeta$  using a shared memory synchronisation algorithm (Dijkstra 1965).

## Example: Critical regions

Many programming languages offer support for critical regions:

$$\mathit{lock} \ u \ \mathit{in} \ a \ \triangleq \ \mathit{acquire}(u); \ \mathit{let} \ y=a \ \mathit{in} \ (\mathit{release}(u); y)$$

(where  $a; b \triangleq \mathit{let} \ x=a \ \mathit{in} \ b$  for  $x \notin \mathit{fv}(b)$ )

For example, we synchronise access to a shared resource:

$$\mathit{let} \ x=\mathit{unlocked} \ \mathit{in} \ \mathit{let} \ f=\mathit{fact} \ \mathit{in}$$

$$(\mathit{lock} \ x \ \mathit{in} \ f(10)) \ \vec{\Gamma} \ (\mathit{lock} \ x \ \mathit{in} \ f(20))$$

## Example: Asynchronous channels

$newChan \triangleq$

$(\nu rd)(\nu wr)(rd \mapsto locked \vec{\Gamma} \quad wr \mapsto unlocked \vec{\Gamma})$

$[val = \zeta(s)s.val,$

$read = \zeta(s)acquire(rd); let \chi = s.val in (release(wr); \chi),$

$write = \zeta(s)\lambda(\chi)acquire(wr); (s.val \leftarrow \zeta(s)\chi); release(rd); \chi]$

- Invariant: at any time at most one of  $rd$  and  $wr$  is unlocked.
- If  $rd$  is unlocked, the result in  $val$  is the contents of the channel.  
If  $wr$  is unlocked, the channel is empty.

## Encoding the asynchronous $\pi$ -calculus

$$\llbracket \bar{x}y \rrbracket = x.\mathit{write}(y)$$

$$\llbracket x(y).P \rrbracket = \mathit{let } y=x.\mathit{read} \mathit{ in } \llbracket P \rrbracket$$

$$\llbracket P \mid Q \rrbracket = \llbracket P \rrbracket \dot{\vdash} \llbracket Q \rrbracket$$

$$\llbracket (\mathit{new } x)P \rrbracket = \mathit{let } x=\mathit{newChan} \mathit{ in } \llbracket P \rrbracket$$

$$\llbracket !x(y).P \rrbracket = [\mathit{rep} = \zeta(s)\mathit{let } y=x.\mathit{read} \mathit{ in } (\llbracket P \rrbracket \dot{\vdash} s.\mathit{rep})].\mathit{rep}$$

- We conjecture that this translation is sound with respect to a suitable notion of contextual equivalence.
- This particular translation is not fully abstract.

## Well-formed Terms

A simple type system for well-formed terms makes these guarantees:

- (1) The top-level denominations in a term represent a partial function, whose domain is preserved by reduction.

Neither  $p \mapsto d_1 \vec{r} p \mapsto d_2$  nor  $p \mapsto [\ell = \zeta(x)q \mapsto d]$  is well-formed.

- (2) A denomination does not occur where a result is expected.

The term *let*  $x=p \mapsto d$  *in*  $b$  is not well-formed.

- (3) Each restriction of  $p$  includes a denomination of  $p$  in its scope.

The term  $(\nu p)p.\ell$  is not well-formed.

**Well-formed terms:**  $a : T$  where  $T ::= \text{Exp} \mid \text{Proc}$

(Well Result) (Well Object)

$$\frac{}{u : \text{Exp}} \quad \frac{b_i : \text{Exp} \quad \text{dom}(b_i) = \emptyset \quad \forall i \in 1..n}{p \mapsto [\ell_i = \zeta(x_i)b_i]^{i \in 1..n} : \text{Proc}}$$

(Well Select)

(Well Update)

(Well Clone)

(Well Res)

$$\frac{}{u.\ell : \text{Exp}} \quad \frac{b : \text{Exp} \quad \text{dom}(b) = \emptyset}{u.\ell \Leftarrow \zeta(x)b : \text{Exp}} \quad \frac{}{\text{clone}(u) : \text{Exp}} \quad \frac{a : T \quad p \in \text{dom}(a)}{(\forall p)a : T}$$

(Well Let) (for  $\text{dom}(b) = \emptyset$ ) (Well Par)

(Well Concur)

$$\frac{a : \text{Exp} \quad b : \text{Exp}}{\text{let } x=a \text{ in } b : \text{Exp}} \quad \frac{a : \text{Proc} \quad b : T \quad \text{dom}(a) \cap \text{dom}(b) = \emptyset}{a \dot{\mapsto} b : T} \quad \frac{a : \text{Exp}}{a : \text{Proc}}$$

**Lemma** Suppose  $a : T$ . If  $a \equiv b$  or  $a \rightarrow b$  then  $b : T$  and  $\text{dom}(a) = \text{dom}(b)$ .

## A Single-Threaded Fragment

We define  $a :^1 T$  by the same rules but omitting (Well Concur).

**Lemma** Suppose  $a :^1 T$ . If  $a \equiv b$  or  $a \rightarrow b$  then  $b :^1 T$  and  $\text{dom}(a) = \text{dom}(b)$ .

If  $a$  is a term of the  $\text{imp}\zeta$ -calculus, then  $a :^1 \text{Exp}$ .

**Theorem** Suppose  $a :^1 \text{Exp}$ . If  $a \rightarrow a'$  and  $a \rightarrow a''$  then  $a' \equiv a''$ .

If  $a :^1 \text{Exp}$  then  $a \equiv (\nu \vec{p})(q_1 \mapsto d_1 \vec{\Gamma} \cdots \vec{\Gamma} q_n \mapsto d_n \vec{\Gamma} t)$ , where the thread  $t$  is not a denomination.

## A First-Order Type System

- Types  $A ::= Proc \mid Exp \mid [l_i:A_i^{i \in 1..n}]$ , where the  $l_i$  are distinct, and no  $A_i = Proc$ .
- Subtyping:  
 $[l:A, l_i:A_i^{i \in 1..n}] <: [l_i:A_i^{i \in 1..n}] <: Exp <: Proc$ .
- Environments  $E ::= x_1 : A_1, \dots, x_n : A_n$ .
- Typing judgment  $E \vdash a : A$ .
- If  $E \vdash a : A$ ,  $A <: T$ , and  $T \in \{Proc, Exp\}$  then  $a : T$ .
- Suppose  $E \vdash a : A$ . If  $a \equiv b$  or  $a \rightarrow b$  then  $E \vdash b : A$ .

$$\begin{array}{c}
\text{(Val Subsumption)} \\
\frac{E \vdash a : A \quad A <: B}{E \vdash a : B}
\end{array}
\quad
\begin{array}{c}
\text{(Val } u) \\
\frac{E, u : A, E' \vdash \diamond}{E, u : A, E' \vdash u : A}
\end{array}
\quad
\begin{array}{c}
\text{(Val Select) (where } j \in 1..n) \\
\frac{E \vdash u : [\ell_i : B_i \text{ }^{i \in 1..n}]}{E \vdash u.\ell_j : B_j}
\end{array}$$

(Val Object) (where  $A = [\ell_i : B_i \text{ }^{i \in 1..n}]$  and  $E = E', p : A, E''$ )

$$\frac{E, x_i : A \vdash b_i : B_i \quad \text{dom}(b_i) = \emptyset \quad \forall i \in 1..n}{E \vdash p \mapsto [\ell_i = \zeta(x_i)b_i \text{ }^{i \in 1..n}] : \text{Proc}}$$

(Val Let)

$$\frac{E \vdash a : A \quad E, x : A \vdash b : B \quad \text{dom}(b) = \emptyset \quad A <: \text{Exp} \quad B <: \text{Exp}}{E \vdash \text{let } x=a \text{ in } b : B}$$

(Val Par) (where  $\text{dom}(a) \cap \text{dom}(b) = \emptyset$ ) (Val Res)

$$\frac{E \vdash a : \text{Proc} \quad E \vdash b : B}{E \vdash a \dot{\vdash} b : B}
\quad
\frac{E, p : A \vdash a : B \quad p \in \text{dom}(a)}{E \vdash (\nu p)a : B}$$

## Examples of Typing and Subtyping

$$A \rightarrow B \triangleq [\mathit{arg}:A, \mathit{val}:B]$$

$$\updownarrow A \triangleq [\mathit{read}:A, \mathit{write}:A \rightarrow A]$$

$$\uparrow A \triangleq [\mathit{write}:A \rightarrow A]$$

$$\downarrow A \triangleq [\mathit{read}:A]$$

- $\emptyset \vdash \mathit{newChan} : \updownarrow A$
- $\emptyset \vdash \mathit{newChan} : \uparrow A$  from  $\updownarrow A <: \uparrow A$
- $\emptyset \vdash \mathit{newChan} : \downarrow A$  from  $\updownarrow A <: \downarrow A$

## Alternative Styles of Semantics

Our chemical semantics based on  $\alpha \mapsto b$  is an alternative to:

- Classical SOS semantics based on configurations: Plotkin (1981), Di Blasio and Fisher (1996)
- Other chemical semantics: Amadio, Leth, and Thomsen (1994), Vasconcelos (1994), Peyton Jones, Gordon, and Finne (1996), Fournet and Gonthier (1996), Boudol (1997)
- Labelled transitions: Ferreira, Hennessy, and Jeffrey (1995)
- Encoding in an existing calculus: Jones (1993), Walker (1995), Sangiorgi and Kleist (1997), Dal-Zilio (1998)

## Conclusion

We solved  $\mathbf{conc}\zeta = \mathbf{imp}\zeta + \pi$  by setting

$$\mathbf{conc}\zeta \triangleq \mathbf{imp}\zeta + \{a \dot{\rightarrow} b, (\nu p)a\} + \{p \mapsto [l_i = \zeta(x_i)b_i]\}$$

This appears to be the first detailed study of a chemical semantics for an asymmetric  $a \dot{\rightarrow} b$ , and the first typed concurrent extension of  $\mathbf{imp}\zeta$ .

As a sanity check, we proved the equivalence of our chemical semantics and a classical SOS semantics.

Future work: equivalence, enriched type systems, mobility.