

On affine usages in signal-based communication

Roberto AMADIO

Mehdi DOGGUY

Université Paris Diderot - CNRS
Laboratoire Preuves, Programmes et Systèmes



February 2009

CHOCO

Plan

Introduction

Usages

Typing system

Typed examples

Linear case

Conclusion

1 Introduction

Goal

Concurrency model

Examples

Non-deterministic cases

Goal

- ▶ Design a type system that guarantees **determinacy**
- ▶ **Folklore** (see, e.g., Esterel and SL)
- ▶ **Challenge** Have a type theory that works when signals are taken as *first-class* values.
- ▶ **Ref** We build on work of Girard 1987, Wadler 1993, Kobayashi et al. 1999, ...

Why ?

- ▶ Deterministic programs are easier to debug and verify
- ▶ Determinacy in concurrent programming is like side-effect-freedom in functional programming

Concurrency model

▶ Synchronous/Timed model

- ▶ Notion of instant, phase, round, ...
- ▶ Time passes (from an instant to the next) exactly when there is nothing to do.

▶ Signal-based interaction

- ▶ **Emission** a value emitted on a signal persists during the instant and disappears at the end of it.
- ▶ **Reception** Two possibilities:
 - ▶ During the instant, select non-deterministically a value emitted on the signal.
 - ▶ At the end of the instant, collect non-deterministically all the values emitted on the signal.

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

$\Downarrow$

$$P\{x/v\}$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

$\Downarrow$

$$\bar{s}\langle v \rangle \mid P\{x/v\}$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

$$\Downarrow N$$

$$K$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

Concurrency model

- ▶ Outputs are persistent

$$\bar{s}\langle v \rangle \mid s(x).(P, K)$$

- ▶ Inputs depend on the instant

$$s(x).(P, K)$$

- ▶ Input at the end of the instant

$$\bar{s}\langle v_1 \rangle \mid \bar{s}\langle v_2 \rangle \mid \bar{s}\langle v_3 \rangle \mid \text{pause}.A(!s)$$

$$\Downarrow N$$

$$A(\{v_1, v_2, v_3\})$$

Concurrency model: $S\pi$ -calculus

Synchronous π -calculus

<i>processes</i>	$P ::=$	0	
		$\bar{s}\langle v \rangle$	Output
		$s(x).P, K$	Input
		$P_1 \mid P_2$	Parallel
		$\nu s P$	Restriction
		$[u \trianglerighteq p]P_1, P_2$	Pattern matching
		$[s_1 = s_2]P_1, P_2$	Signal matching
		$A(e)$	Thread call

continuation $K ::= A(r)$

*expression at end
of instant* $e ::=$

	Var
	$Constr(e_1, \dots, e_n)$
	$Fun(e_1, \dots, e_n)$
	$!s$

Dereferenciation

Example #1

$$\nu_{s_1, s_2} (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid \\ s_1(x). (s_1(y). (s_2(z). (A(x, y), B(!s_1)) \\ , Nil_2())) \\ , Nil_1()) \\)$$

Example #1

$$\nu_{s_1, s_2} (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid \\ s_1(x). (s_1(y). (s_2(z). (A(x, y), B(!s_1)) \\ , Nil_2())) \\ , Nil_1()) \\)$$

$\Downarrow$

$$\nu_{s_1, s_2} (s_2(z). A(\sigma(x), \sigma(y)))$$

Where $\sigma(x), \sigma(y) \in \{v_1, v_2\}$ and $\sigma(x) \neq \sigma(y)$

Example #1

$$\nu s_1, s_2 (\quad \overline{s_1} < v_1 > \mid \overline{s_1} < v_2 > \mid \\ s_1(x). (s_1(y). (s_2(z). (A(x, y), B(!s_1)) \\ \quad , Nil_2())) \\ \quad , Nil_1()) \\)$$

$\Downarrow$

$$\nu s_1, s_2 (\overline{s_1} < v_1 > \mid \overline{s_1} < v_2 > \mid s_2(z). (A(\sigma(x), \sigma(y)), B(!s_1)))$$

Example #1

$$\nu s_1, s_2 (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid \\ s_1(x). (s_1(y). (s_2(z). (A(x, y), B(!s_1)) \\ , Nil_2())) \\ , Nil_1()) \\)$$

$\Downarrow$

$$\nu s_1, s_2 (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid s_2(z). (A(\sigma(x), \sigma(y)), B(!s_1)))$$

$\Downarrow N$

Example #1

$$\nu s_1, s_2 (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid \\ s_1(x). (s_1(y). (s_2(z). (A(x, y), B(!s_1)) \\ , Nil_2())) \\ , Nil_1()) \\)$$

$\Downarrow$

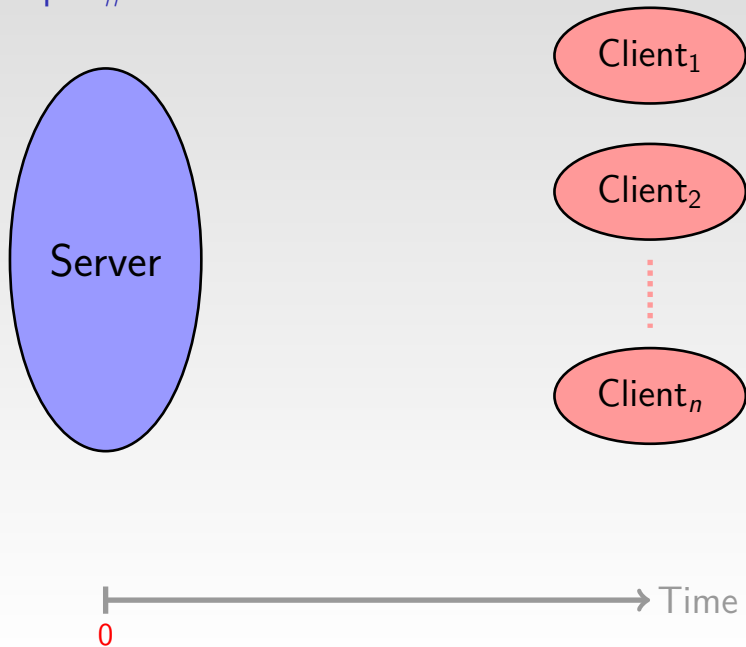
$$\nu s_1, s_2 (\overline{s_1} \langle v_1 \rangle \mid \overline{s_1} \langle v_2 \rangle \mid s_2(z). (A(\sigma(x), \sigma(y)), B(!s_1)))$$

$\Downarrow N$

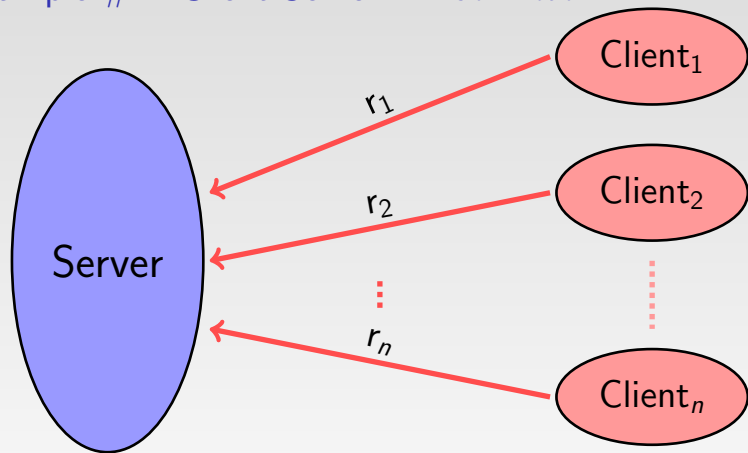
$$\nu s_1, s_2. B(I)$$

Where $I \in \{[v_1; v_2], [v_2; v_1]\}$

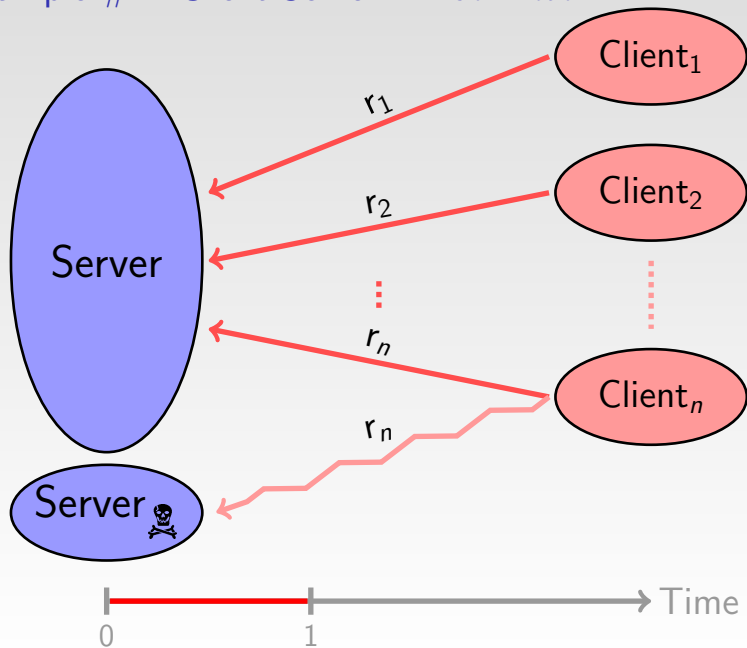
Example #2: Client-Server ...in slow motion



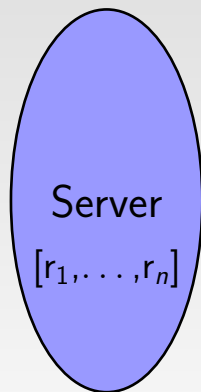
Example #2: Client-Server ...in slow motion



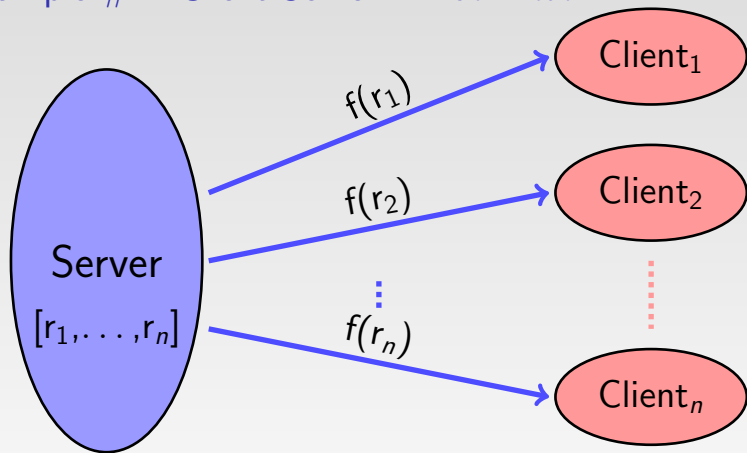
Example #2: Client-Server ...in slow motion



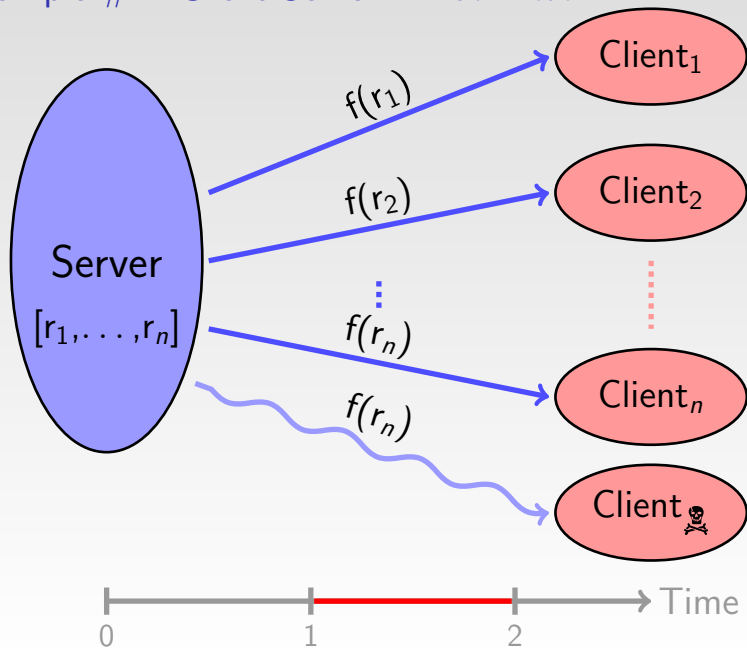
Example #2: Client-Server ...in slow motion



Example #2: Client-Server ...in slow motion



Example #2: Client-Server ...in slow motion



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid pause.s'(x). \bar{t} \langle x \rangle, 0)$$

$$\begin{aligned} Server(s) &= pause.Handle(s, !s) \\ Handle(s, \ell) &= [\ell \sqsupseteq cons(req(s', x), \ell')] \\ &\quad (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ &\quad , Server(s) \end{aligned}$$



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid pause.s'(x). \bar{t} \langle x \rangle, 0)$$

$$\begin{aligned} Server(s) &= pause.Handle(s, !s) \\ Handle(s, \ell) &= [\ell \sqsupseteq cons(req(s', x), \ell')] \\ &\quad (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ &\quad , Server(s) \end{aligned}$$



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid pause.s'(x). \bar{t} \langle x \rangle, 0)$$

$$\begin{aligned} Server(s) &= pause.Handle(s, !s) \\ Handle(s, \ell) &= [\ell \sqsupseteq cons(req(s', x), \ell')] \\ &\quad (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ &\quad , Server(s) \end{aligned}$$



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid pause.s'(x). \bar{t} \langle x \rangle, 0)$$

$$\begin{aligned} Server(s) &= pause.Handle(s, !s) \\ Handle(s, \{r_1, \dots, r_n\}) &= [\ell \geq \text{cons}(req(s', x), \ell')] \\ &\quad (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ &\quad , Server(s) \end{aligned}$$



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid pause.s'(x). \bar{t} \langle x \rangle, 0)$$

$$Server(s) = pause.Handle(s, !s)$$

$$Handle(s, \{r_1, \dots, r_n\}) = [\ell \sqsupseteq cons(req(s', x), \ell')] \\ (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ , Server(s)$$



Example #2: Client-Server

$$\nu s \text{ (} Server(s) \mid Client(x, s, t) \mid \dots \mid Client(x, s, t) \text{)}$$

$$Client(x, s, t) = \nu s' (\bar{s} \langle req(s', x) \rangle \mid \text{pause}.s'(x).\bar{t} \langle x \rangle, 0)$$

$$Server(s) = \text{pause}.Handle(s, !s)$$

$$Handle(s, \{r_1, \dots, r_n\}) = [\ell \sqsupseteq \text{cons}(req(s', x), \ell')] \\ (\bar{s'} \langle f(x) \rangle \mid Handle(s, \ell')) \\ , Server(s)$$



Non-deterministic cases

- ▶ At least two different values are emitted on the same signal, at the same instant:

$$\bar{s} \langle v \rangle \mid \bar{s} \langle v' \rangle \mid s(x).P, K$$

- ▶ At the end of the instant, at least two different values are available:

$$\bar{s} \langle v \rangle \mid \bar{s} \langle v' \rangle \mid \text{pause}.A(!s)$$

2 Usages

Signal usage

Main usages

How to combine them

1. Signal usage

$$s : \text{Sig} \left(\begin{smallmatrix} w \\ r \\ e \end{smallmatrix} \right)^\omega (\sigma)$$

$$r, w, e \in L = \left\{ \begin{array}{ll} 0 & \text{Cannot be used} \\ 1 & \text{Used at most once} \\ \infty & \text{Used an arbitrary number of times} \end{array} \right\}$$

Usages: All of them?

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ \infty \end{pmatrix}$$

Usages: All of them?

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ \infty \end{pmatrix}$$

Usages: All of them?

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ 1 \\ \infty \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 0 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ 1 \end{pmatrix} \quad \begin{pmatrix} \infty \\ \infty \\ \infty \end{pmatrix}$$

Usages: All of them?

$$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 1 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ \infty \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ \infty \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ \infty \\ \infty \end{pmatrix}$$

Usages: All of them?

$$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 0 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix}$$

Usages: All of them?

$$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

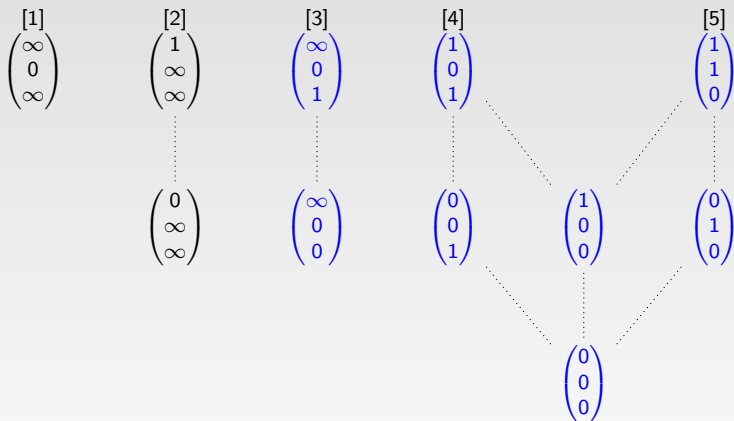
$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \infty \\ 0 \\ \infty \end{pmatrix}$$

Main usages



- ▶ Leafs are called **neutral usages** (without 1)
- ▶ Usages containing **1** are called affine
- ▶ **Blue** ones are affine preserving

How to combine them

$$\begin{pmatrix} w \\ r \\ e \end{pmatrix} \oplus \begin{pmatrix} w' \\ r' \\ e' \end{pmatrix} = \begin{pmatrix} w \oplus w' \\ r \oplus r' \\ e \oplus e' \end{pmatrix}$$

$$u \oplus u' = \begin{cases} 0 \oplus x = x & x \in L \\ x \oplus 0 = x \\ 1 \oplus 1 = \uparrow \\ \infty \oplus 1 = \uparrow \\ \infty \oplus y = \infty \\ y \oplus \infty = \infty \end{cases} \quad y \in \{0, \infty\}$$

3 Typing system

Types

Type system with usages

Set-types and proof obligations

Properties

Types

Intuition: A constructor can be applied to an affine type only if it is affine preserving.

$$\kappa ::= 1 \mid \text{List}_{\infty}(\kappa) \mid \text{Set}_{\infty}(\kappa) \mid \text{Sig}_u(\kappa)$$

Classical uniform types
(*u is neutral*)

Types

Intuition: A constructor can be applied to an affine type only if it is affine preserving.

$$\kappa ::= 1 \mid List_{\infty}(\kappa) \mid Set_{\infty}(\kappa) \mid Sig_u(\kappa)$$

Classical uniform types

(u is neutral)

$$\lambda ::= List_1(\sigma) \mid Set_1(\sigma) \mid Sig_u(\kappa) \mid Sig_v(\lambda)$$

Affine uniform types

(u is uniform and affine, v is uniform and affine preserving)

Types

Intuition: A constructor can be applied to an affine type only if it is affine preserving.

$$\kappa ::= 1 \mid \text{List}_{\infty}(\kappa) \mid \text{Set}_{\infty}(\kappa) \mid \text{Sig}_u(\kappa)$$

Classical uniform types

(u is neutral)

$$\lambda ::= \text{List}_1(\sigma) \mid \text{Set}_1(\sigma) \mid \text{Sig}_u(\kappa) \mid \text{Sig}_v(\lambda)$$

Affine uniform types

(u is uniform and affine, v is uniform and affine preserving)

$$\sigma ::= \kappa \mid \lambda$$

Uniform types

$$\rho ::= \sigma \mid \text{Sig}_{u \cdot u' \omega}(\kappa) \mid \text{Sig}_{v \cdot v' \omega}(\lambda)$$

Non uniform types

(u and u' are neutral ; v and v' are affine)

Type system with usages

We must guarantee what we emit and we can rely on what we receive.

$$(out) \frac{\Gamma_1 \vdash s : Sig_u(\sigma) \quad u(0)_1 \neq 0 \quad \Gamma_2 \vdash e : \sigma}{\Gamma_1 \oplus \Gamma_2 \vdash \bar{s} \langle e \rangle}$$

$$(in) \frac{\Gamma_1 \vdash s : Sig_u(\sigma) \quad u(0)_2 \neq 0 \quad \Gamma_{2,x:\sigma} \vdash P \quad (\Gamma_1 \oplus \Gamma_2) \vdash [A(\mathbf{r})]}{(\Gamma_1 \oplus \Gamma_2) \vdash s(x).P, A(\mathbf{r})}$$

Set-types and proof obligations

If a thread identifier A waits for an argument of type $Set_x(D)$ (where $x \in \{1, \infty\}$) then we have a proof obligation:

$$\frac{\ell \text{ a permutation of } \ell'}{A(\ell) \approx A(\ell')}$$

In the client-server example it suffices to appeal to simple algebraic properties of bisimulation: associativity and commutativity of parallel composition and preservation under static contexts.

Subject reduction

The **subject reduction** lemma is stated as follows:

$$\frac{\Gamma \vdash P \quad P \xrightarrow{act} P' \quad (\Gamma, act) \downarrow}{\Gamma(act) \vdash P'}$$

- ▶ The **compatibility** of typing context Γ with an action act : $(\Gamma, act) \downarrow$. Unconstrained transitions may fail to preserve typing.
- ▶ The **residual context** $\Gamma(act)$ after an action act compatible with Γ .

Typed programs are *deterministic*

Theorem

Let P be a program and C a static context such that $C[P]$ is typable. If $C[P]$ reduces to Q_1 and Q_2 after n instants then Q_1 and Q_2 are typed bisimilar.

Proof steps

- ▶ Reductions within the instant commute strongly:

$$\frac{P \xrightarrow{\tau}_{\Gamma} P_i, \quad i = 1, 2}{\exists Q, P_i (\xrightarrow{\tau}_{\Gamma} \cup =) Q, \quad i = 1, 2}$$

- ▶ Reductions at the end of the instant preserve typed bisimulation:

$$\frac{P \xrightarrow{N}_{\Gamma} P_i, \quad i = 1, 2}{P_1 \approx_{\uparrow(\Gamma)} P_2}$$

4 Typed examples

Client-Server

Single-assignment references

Synchronous data flow

Client-Server

$$\begin{aligned}
 \text{Server}(s) &= \text{pause}.\text{Handle}(s, !s) \\
 \text{Handle}(s, \ell) &= [\ell \sqsupseteq \text{cons}(\text{req}(s', x), \ell')] \\
 &\quad (\overline{s'} < f(x) > \mid \text{Handle}(s, \ell')) \\
 &\quad , \text{Server}(s)
 \end{aligned}$$

$$\text{Client}(x, s, t) = \nu s' (\overline{s} < \text{req}(s', x) > \mid \text{pause}.s'(x).\overline{t} < x >, 0)$$

$$s : \text{Sig} \begin{pmatrix} \infty \\ 0 \\ 1 \end{pmatrix}^{\omega}(\text{Req}) \quad I : \text{Set}_1(\text{Req}) \quad s' : \text{Sig} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}^{\omega}(\text{Data})$$

$$\text{Req} = \text{req of } \text{Sig} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^{\omega}(\text{Data}) * \text{Data}$$

Single-assignment references

$$Ref(s, s', x) = \overline{s} \langle x \rangle \mid s'(y).pause.Ref(s, s', y), Ref(s, s', x)$$

$$\langle newref(s, e).P \rangle = \nu \overline{s'} (Ref(s, s', e) \mid \langle P \rangle)$$

$$\langle read(s, x).P \rangle = s(x). \langle P \rangle, 0$$

$$\langle write(s, e).P \rangle = \overline{s'} \langle e \rangle \mid \langle P \rangle$$

$$Ref : (Sig \begin{pmatrix} 1 \\ \infty \\ \infty \end{pmatrix}^{\omega(Data)}, \quad Sig \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^{\omega(Data)}, \quad Data)$$

$$read : (Sig \begin{pmatrix} 0 \\ \infty \\ \infty \end{pmatrix}^{\omega(Data)}, \quad Data) \quad write : (Sig \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^{\omega(Data)}, \quad Data)$$

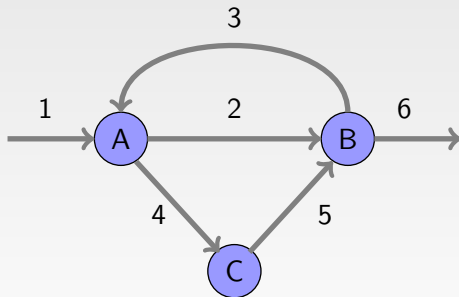
Synchronous data flow

$$\nu s_2, s_3, s_4, s_5 (A(s_1, s_2, s_3, s_4) \mid B(s_2, s_3, s_5, s_6) \mid C(s_4, s_5))$$

$$A(s_1, s_2, s_3, s_4) = s_1(x).(\overline{s_2} \langle f(x) \rangle \mid s_3(y).(\overline{s_4} \langle g(y) \rangle \mid \text{pause}.A(s_1, s_2, s_3, s_4)), 0), 0$$

$$B(s_2, s_3, s_5, s_6) = s_2(x).(\overline{s_3} \langle i(x) \rangle \mid s_5(y).(\overline{s_6} \langle l(y) \rangle \mid \text{pause}.B(s_2, s_3, s_5, s_6)), 0), 0$$

$$C(s_4, s_5) = s_4(x).(\overline{s_5} \langle h(x) \rangle \mid \text{pause}.C(s_4, s_5)), 0$$



Synchronous data flow

$$\nu \mathbf{s_2, s_3, s_4, s_5} \ (A(s_1, s_2, s_3, s_4) \mid B(s_2, s_3, s_5, s_6) \mid C(s_4, s_5))$$

$$\begin{aligned} A(s_1, s_2, s_3, s_4) &= s_1(x).(\overline{s_2} \langle f(x) \rangle \mid s_3(y).(\overline{s_4} \langle g(y) \rangle \mid \text{pause}.A(s_1, s_2, s_3, s_4)), 0), 0 \\ B(s_2, s_3, s_5, s_6) &= s_2(x).(\overline{s_3} \langle i(x) \rangle \mid s_5(y).(\overline{s_6} \langle l(y) \rangle \mid \text{pause}.B(s_2, s_3, s_5, s_6)), 0), 0 \\ C(s_4, s_5) &= s_4(x).(\overline{s_5} \langle h(x) \rangle \mid \text{pause}.C(s_4, s_5)), 0 \end{aligned}$$

$$A : (I, O, I, O) \quad B : (I, O, I, O) \quad C : (I, O)$$

$$I : \text{Sig}_{(0,1,0)}(\text{Data}) \quad O : \text{Sig}_{(1,0,0)}(\text{Data})$$

5 Linear case

Linear case

$$\nu s: \textit{Sig}(1, 1, 0) (\textit{Data}) (s(x). \bar{s} \langle v \rangle, 0)$$

Linear case

$\nu s: \text{Sig}_{(1,1,0)}(\text{Data}) (s(x).\bar{s}\langle v \rangle, 0)$

$$\frac{\Gamma \text{ neutral}}{\Gamma \vdash 0} + \text{Deadlock-freedom}$$

6 Conclusion

Conclusion

Few remarks

- ▶ We need to make a distinction between affine usages and affine preserving usages.
- ▶ Distinctive differences with respect to affine systems for, say, the π -calculus is that:
 - ▶ signals allow for (some form of) multi-point interaction while preserving determinacy.
 - ▶ signal usages depend on time.

Thank you!

Thank you!

Questions ?

$$\langle k \rangle \frac{\Gamma_i \vdash \langle r_i \rangle : \sigma_i \quad i = 1, \dots, n \quad k : (\sigma_1, \dots, \sigma_n) \rightarrow \sigma \quad k = f \text{ or } k = c}{\Gamma_0 \oplus \Gamma_1 \oplus \dots \oplus \Gamma_n \vdash \langle k(r_1, \dots, r_n) \rangle : \sigma} \quad (k) \frac{\Gamma_i \vdash e_i : \sigma_i \quad i = 1, \dots, n \quad k : (\sigma_1, \dots, \sigma_n) \rightarrow \sigma \quad k = f \text{ or } k = c}{\Gamma_0 \oplus \Gamma_1 \oplus \dots \oplus \Gamma_n \vdash k(e_1, \dots, e_n) : \sigma}$$

$$\langle var_C \rangle \frac{Op = C \quad Op = Set}{\Gamma, x : Op_u(\sigma) \vdash \langle x \rangle : Op_u(\sigma)} \quad \langle var_{sig} \rangle \frac{y^\omega \geq u}{\Gamma, s : Sig_{xy^\omega}(\sigma) \vdash \langle s \rangle : Sig_u(\sigma)}$$

$$(var) \frac{u \geq u' \quad Op \in \{Sig, Set, C\}}{\Gamma, x : Op_u(\sigma) \vdash x : Op_{u'}(\sigma)}$$

$$\langle !_{Set} \rangle \frac{(u(0) \geq (\infty, 0, \infty) \wedge x = \infty) \vee (u(0) \geq (\infty, 0, 1) \wedge x = 1)}{\Gamma, s : Sig_u(\sigma) \vdash \langle !s \rangle : Set_x(\sigma)} \quad \langle !_{List} \rangle \frac{(u(0) \geq (0, \infty, \infty) \wedge x = \infty) \vee (u(0) \geq (0, 0, 1) \wedge x = 1)}{\Gamma, s : Sig_u(\sigma) \vdash \langle !s \rangle : List_x(\sigma)}$$

$$(0) \frac{}{\Gamma \vdash 0} \quad (out) \frac{\Gamma_1 \vdash s : Sig_u(\sigma) \quad u(0)_1 \neq 0 \quad \Gamma_2 \vdash e : \sigma}{\Gamma_1 \oplus \Gamma_2 \vdash \bar{s} \langle e \rangle}$$

$$(\nu) \frac{\Gamma, s : Sig_u(\sigma) \vdash P}{\Gamma \vdash \nu s : Sig_u(\sigma) P} \quad (in) \frac{\Gamma_1 \vdash s : Sig_u(\sigma) \quad u(0)_2 \neq 0 \quad \Gamma_2, x : \sigma \vdash P \quad (\Gamma_1 \oplus \Gamma_2) \vdash [A(\mathbf{r})]}{(\Gamma_1 \oplus \Gamma_2) \vdash s(x).P, A(\mathbf{r})}$$

$$(m_s) \frac{s_1, s_2 \in dom(\Gamma) \quad \Gamma \vdash P_i \quad i = 1, 2}{\Gamma \vdash [s_1 = s_2]P_1, P_2} \quad (m_c) \frac{c : (\sigma_1, \dots, \sigma_n) \rightarrow \sigma \quad \Gamma_1 \vdash u : \sigma \quad \Gamma_2, x_1 : \sigma_1, \dots, x_n : \sigma_n \vdash P_1 \quad \Gamma_1 \oplus \Gamma_2 \vdash P_2}{\Gamma_1 \oplus \Gamma_2 \vdash [u \sqsupseteq c(x_1, \dots, x_n)]P_1, P_2}$$

$$(par) \frac{\Gamma_i \vdash P_i \quad i = 1, 2}{\Gamma_1 \oplus \Gamma_2 \vdash P_1 \mid P_2} \quad (rec) \frac{A : (\sigma_1, \dots, \sigma_n) \quad \Gamma_i \vdash e_i : \sigma_i \quad i = 1, \dots, n}{\Gamma_1 \oplus \dots \oplus \Gamma_n \vdash A(e_1, \dots, e_n)}$$

$$(out) \frac{\Gamma \vdash s : Sig_u(\sigma) \quad u(0) = (1, 1, 0)}{\Gamma \vdash \bar{s} \langle e \rangle} \quad \langle rec \rangle \frac{A : (\sigma_1, \dots, \sigma_n) \quad \Gamma_i \vdash \langle r_i \rangle : \sigma_i \quad i = 1, \dots, n}{\Gamma_1 \oplus \dots \oplus \Gamma_n \vdash \langle A(r_1, \dots, r_n) \rangle}$$