Typed Connector Families

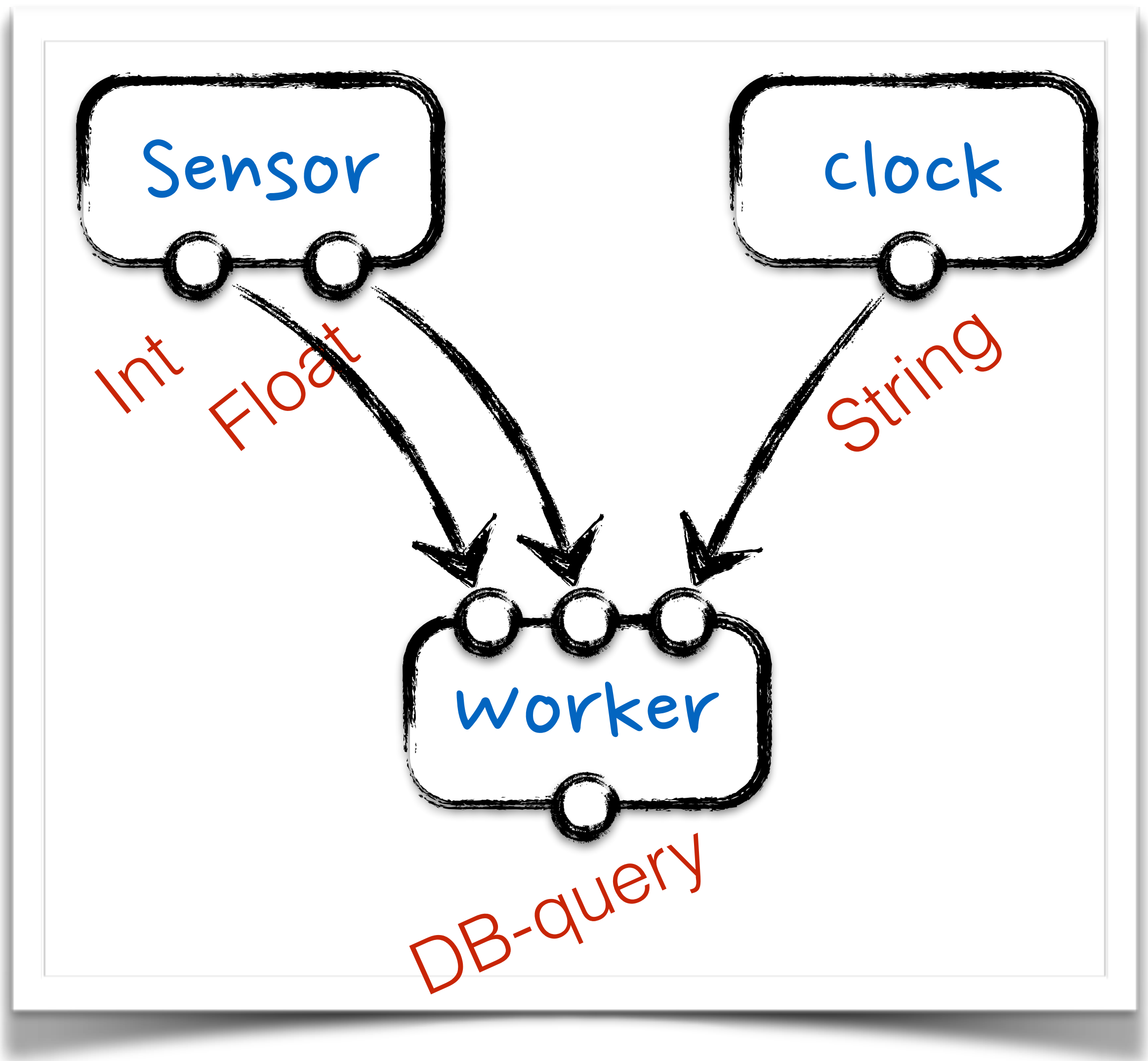
<u>José Proença</u> & Dave Clarke
(KU Leuven, Belgium) (UPPSALA University, Sweden)

DistriNet
iMinds KU LEUVEN

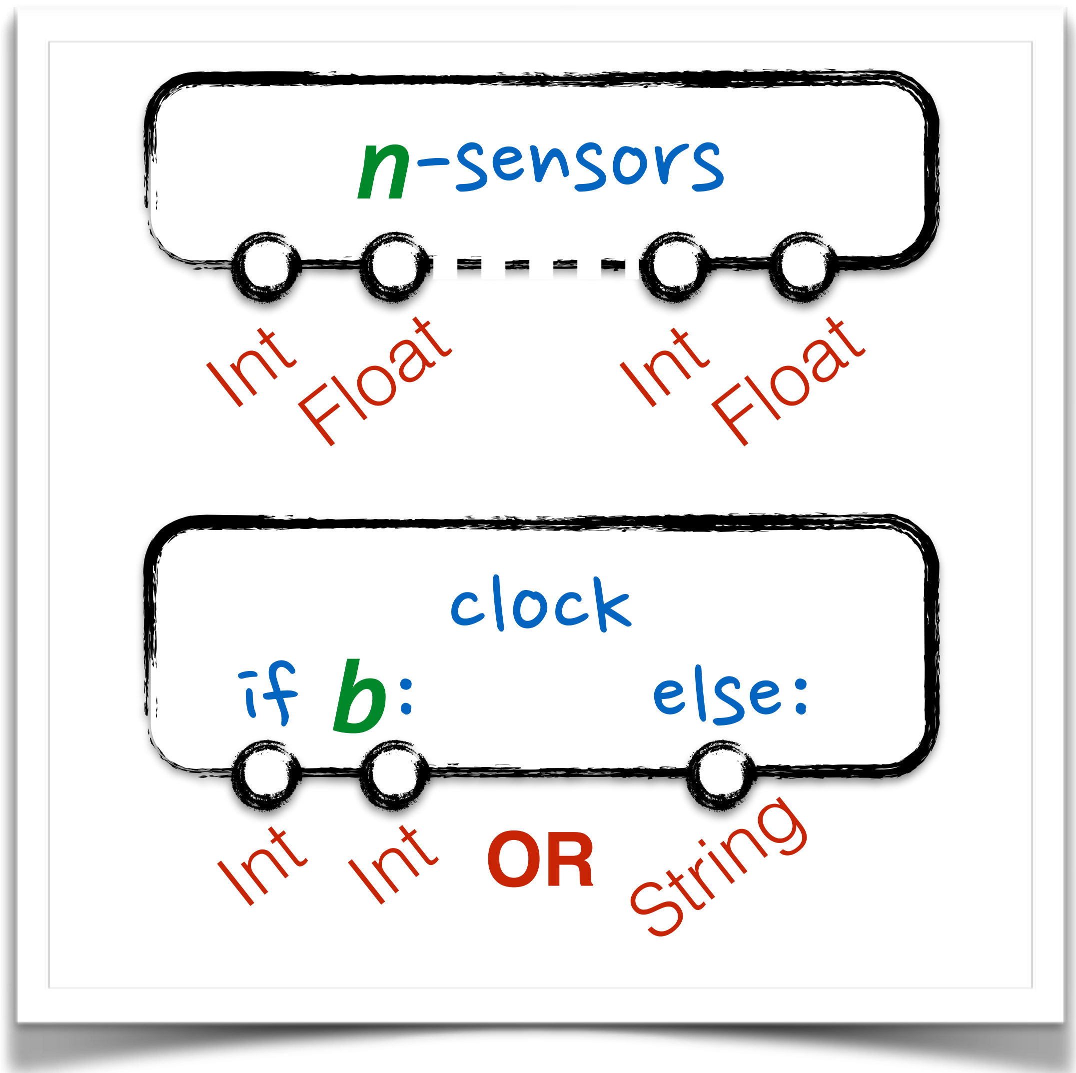
UPPSALA UNIVERSITET

Paper+slides: http://facs15.proenca.org

Motivation



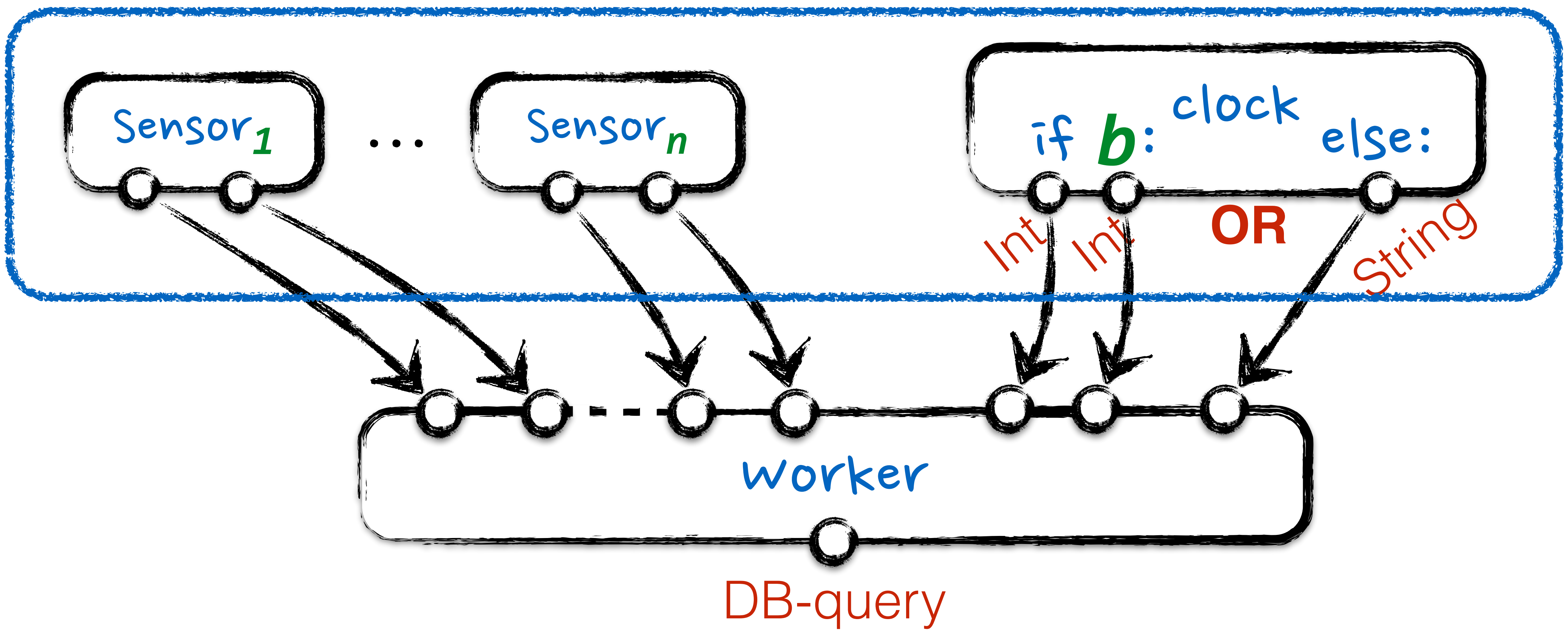
Static interfaces



Software product lines

Paper+slides: http://facs15.proenca.org

Motivation



$$\lambda n{:}\mathrm{Int}, b{:}\mathrm{Bool} \cdot \mathrm{Sensor}^n \otimes \mathrm{clock}(b)$$

$$: \forall\, n{:}\mathrm{Int}, b{:}\mathrm{bool} \cdot 0 \longrightarrow (\mathrm{Int} \otimes \mathrm{Float})^n \otimes (\mathrm{Int} \otimes \mathrm{Int} \oplus^b \mathrm{String})$$

Paper+slides: http://facs15.proenca.org

Outline

basic connector calculus

Sensor $\otimes$ clock ; worker

: $o \longrightarrow$ DB-query

parameterised connector calculus

$\lambda \boldsymbol{n}$:Int · Sensor^{n}

: $\forall \boldsymbol{n}$:Int · $o \longrightarrow \cdots$

connector families

composing

parameterised cc

Type-checking approach

for untyped ports

Paper+slides: http://facs15.proenca.org

Basic connector calculus

$c ::= c_1 \,;\, c_2$	sequential composition	category with
$\mid\; c_1 \otimes c_2$	parallel composition	a tensor (monoid)
$\mid\; \mathsf{id}_I$	identity connectors	
$\mid\; \gamma_{I,J}$	symmetries	symmetries
$\mid\; \mathsf{Tr}_I(c)$	traces	traces
$\mid\; p \in \mathcal{P}$	primitive connectors	

connectors: morphisms

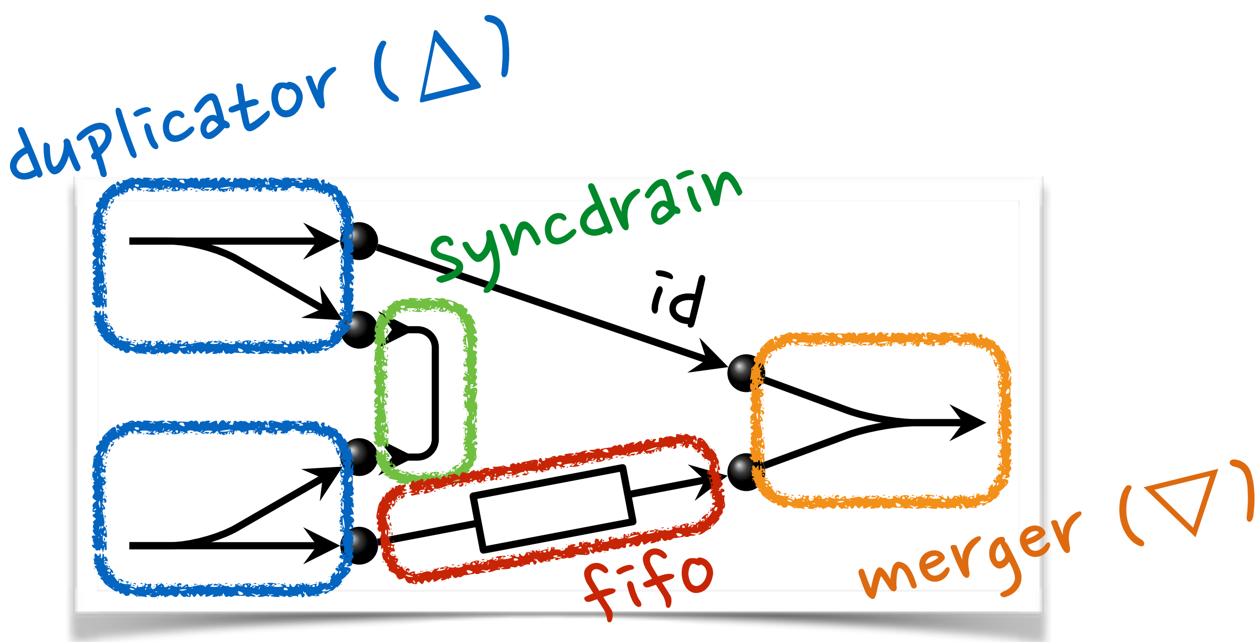
$I, J ::= I \otimes J$	tensor
$\mid\; \mathsf{0}$	empty interface
$\mid\; A$	port type

interfaces: objects

Based on connector algebra from Bruni et. al. (TCS'06)

Paper+slides: ***http://facs15.proenca.org***

Reo example



$\Delta(a, a_1, a_2)$
$\Delta(b, b_1, b_2)$
$\text{syncdrain}(a_2, b_1)$
$\text{id}(a_2, c_1)$
$\text{fifo}(b_2, c_2)$
$\nabla(c_1, c_2, c)$

traditional:
with port names

$\Delta \otimes \Delta\ ;$
$\text{id} \otimes \text{syncdrain} \otimes \text{fifo}\ ;$
∇

connector calculus

Visualisation of connectors

recall:

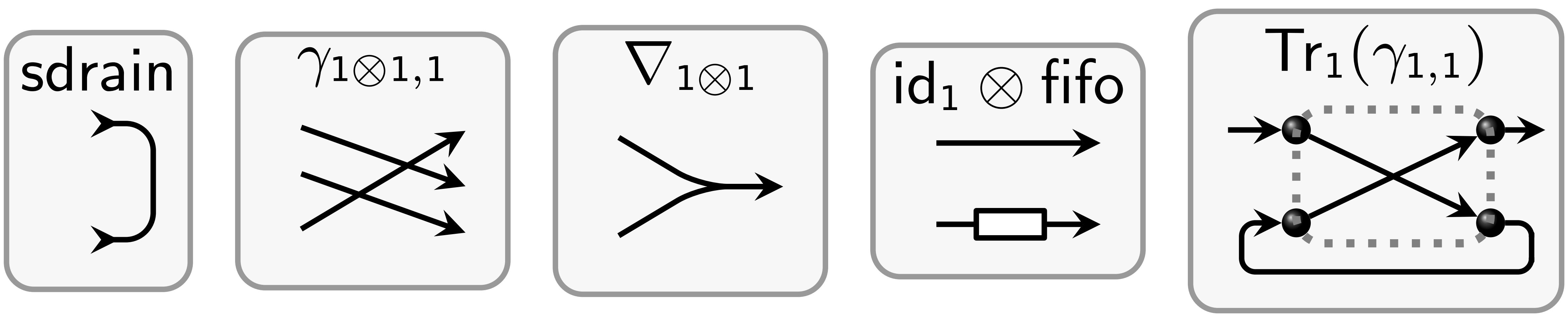
id_I

$\gamma_{I,J}$

$\mathsf{Tr}_I(c)$

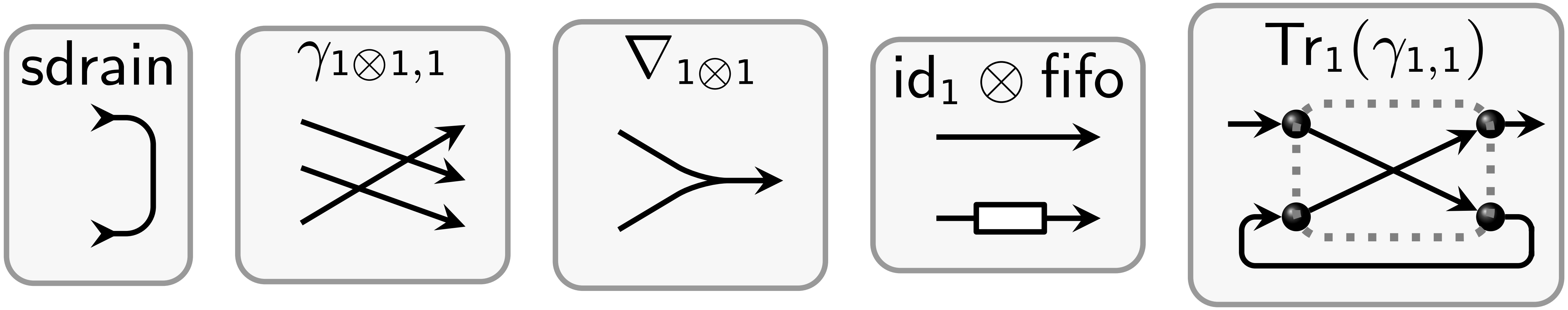
$$
\begin{array}{rlll}
I, J ::= & I \otimes J & & \text{tensor} \\
\mid & 0 & & \text{empty interface} \\
\mid & A & & \text{port type}
\end{array}
$$

1 is some port type



Data goes always from left to right

Typing connectors



$1\otimes 1 \to 0$ $\quad$ $1\otimes 1\otimes 1 \to 1\otimes 1\otimes 1$ $\quad$ $1\otimes 1 \to 1$ $\quad$ $1\otimes 1 \to 1\otimes 1$ $\quad$ $1 \to 1$

conn : $I \longrightarrow J$

IF $\quad$ c1 : $I_1 \longrightarrow J$ & c2 : $J \longrightarrow J_2$

THEN $\quad$ c1 ; c2 : $I_1 \longrightarrow J_2$

Constraint-based type rules

(sequence)

$$\frac{\Gamma \mid \phi \vdash c_1 : I_1 \to J_1 \quad \Gamma \mid \phi \vdash c_2 : I_2 \to J_2}{\Gamma \mid \phi, J_1 = I_2 \vdash c_1 \ ; \ c_2 : I_1 \to J_2}$$

(trace)

$$\frac{\Gamma \mid \phi \vdash c : J_1 \to J_2}{\Gamma \mid \phi, J_1{=}X{\otimes}I, J_2{=}Y{\otimes}I \vdash \mathsf{Tr}_I(c) : X \to Y}$$

Paper+slides: http://facs15.proenca.org

Parameterised connector calculus

$\lambda x : \mathrm{Int} \cdot c$

fifo^{exp} means: $\mathrm{fifo} \otimes \cdots \otimes \mathrm{fifo}$ ("exp" times)

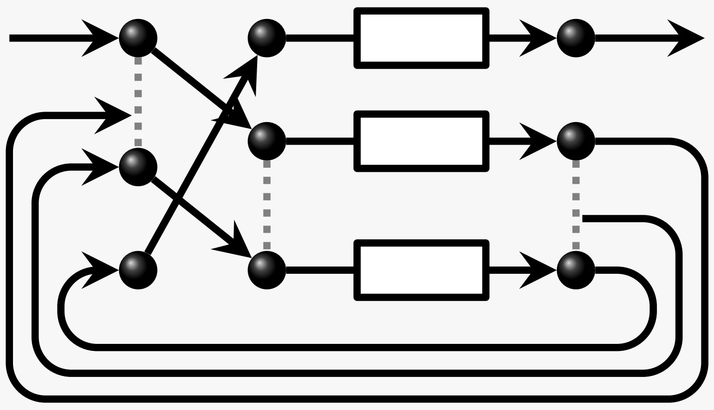
$(\Delta_{I^x})^{x \leftarrow exp}$ means: $\Delta_{I^0} \otimes \cdots \otimes \Delta_{I^{exp-1}}$

$\mathrm{fifo} \oplus^{exp} \mathrm{drain}$ means: if (exp) then (fifo) else (drain)

$c ::= \ldots$	connectors	$I ::= \ldots$	interfaces
$\mid\ c^{x \leftarrow \alpha}$	n-ary parallel replication	$\mid\ I^{\alpha}$	n-ary parallel replication
$\mid\ c_1 \oplus^{\phi} c_2$	conditional choice	$\mid\ I \oplus^{\phi} J$	conditional choice
$\mid\ \lambda x : P \cdot c$	parameterised connector		
$\mid\ c(\phi)$	bool-instantiation	α, β	integer expressions
$\mid\ c(\alpha)$	int-instantiation	ϕ, ψ	boolean expressions

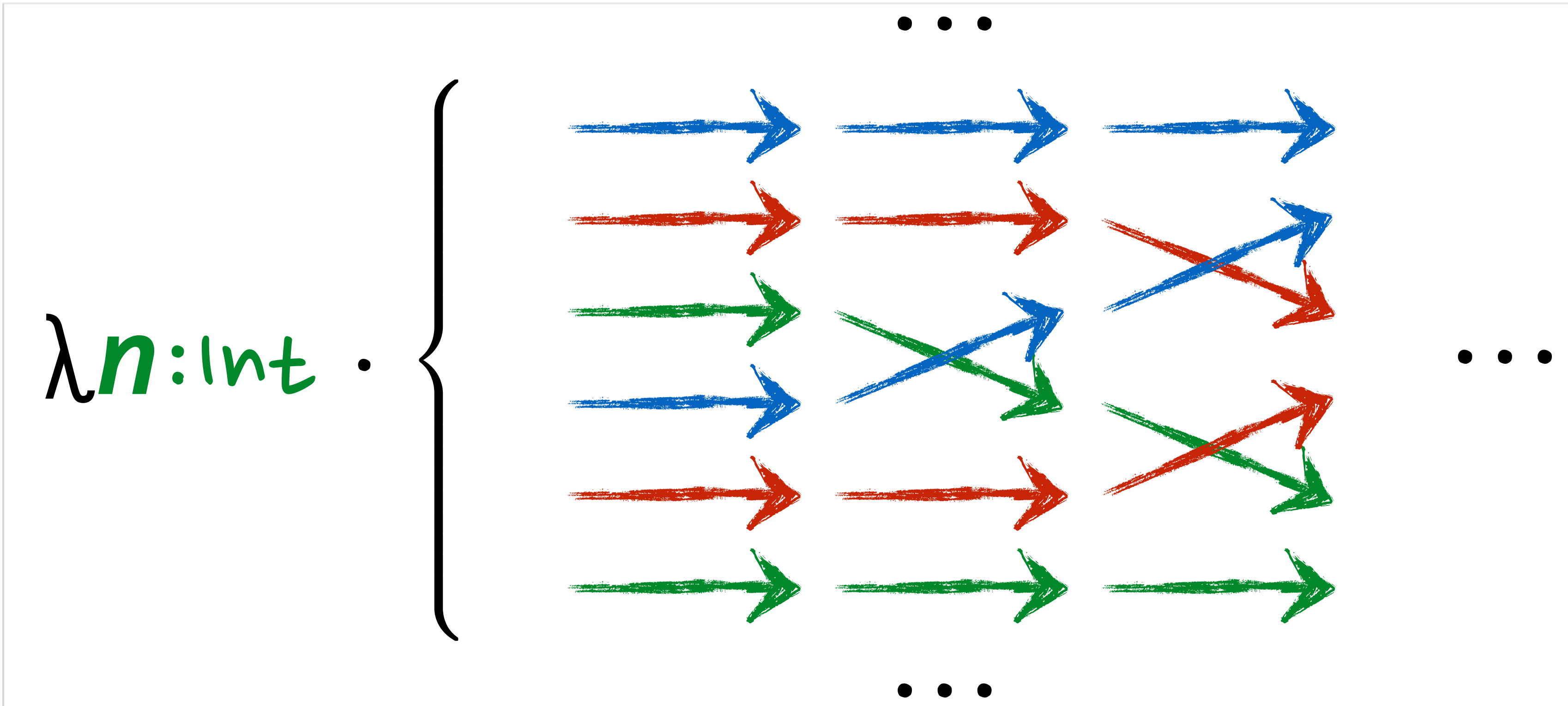
Example: *seq-fifo*

$$\textbf{\textit{seq-fifo}} = \lambda n : \mathbb{N} \cdot \mathsf{Tr}_{n-1}\,(\gamma_{n-1,1}\ ;\ \mathsf{fifo}^{n})$$



seq-fifo : $\forall$ n:Int $\cdot\ 1 \longrightarrow 1$

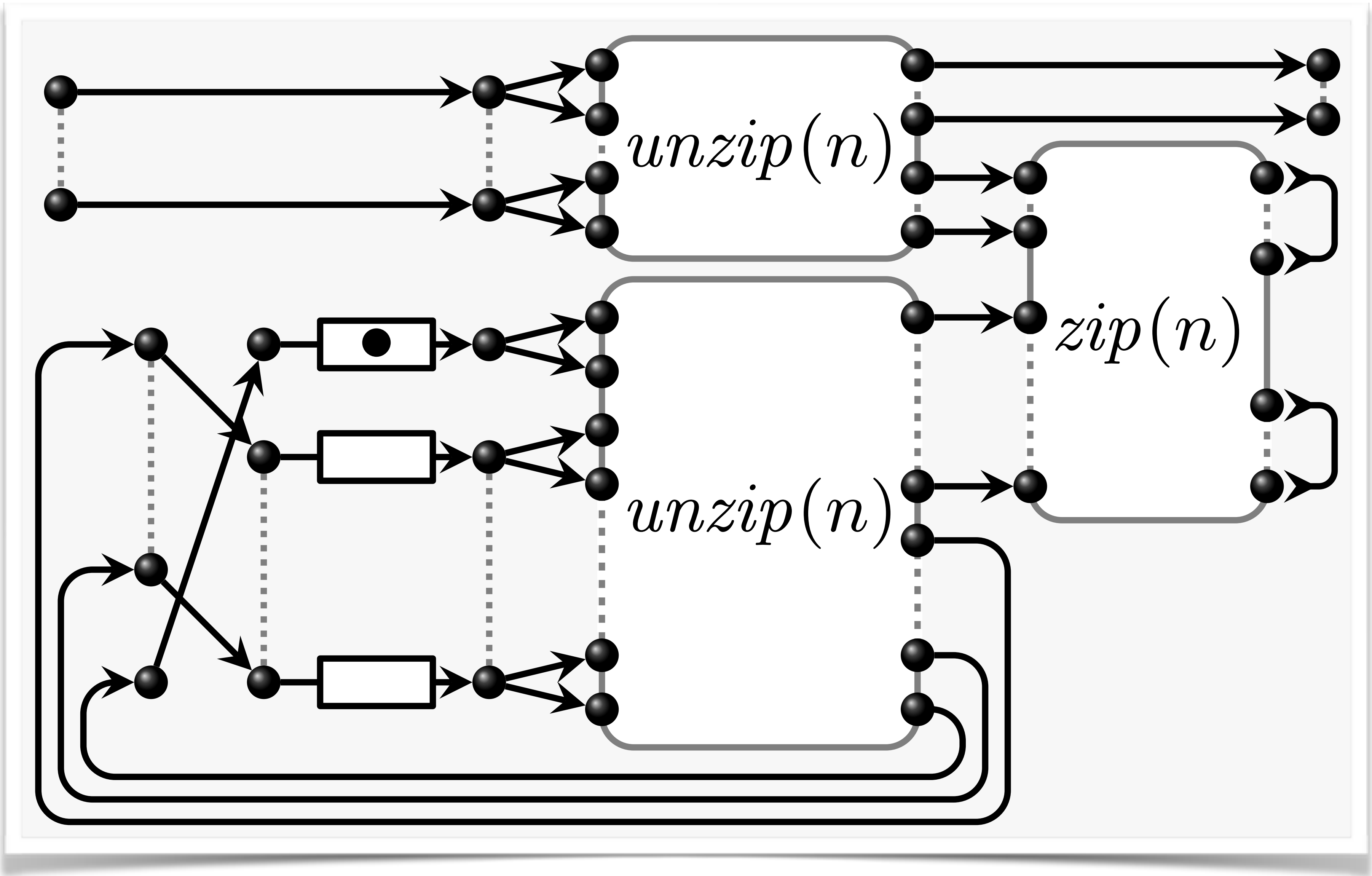
Example - *zip*



$$\boldsymbol{zip} = \lambda n : \mathbb{N} \cdot \mathsf{Tr}_{2n^2-2n}\big(\gamma_{2n^2-2n,2n} ; (\mathsf{id}_{n-x} \otimes \gamma^{x}_{1,1} \otimes \mathsf{id}_{n-x})^{x \leftarrow n} \big)$$

$zip \;:\; \forall\, n{:}Int \cdot (1^n)^2 \longrightarrow (1^2)^n$

Example - *sequencer*



sequencer : $\forall\, n{:}\mathrm{Int} \cdot 1^n \longrightarrow 1^n$

Connector Families

(restriction)

$$\frac{\Gamma \mid \phi \vdash \psi \qquad \Gamma \mid \phi, \psi \vdash c : T}{\Gamma \mid \phi \vdash c|_{\psi} : T|_{\psi}}$$

$\lambda n{:}\mathrm{Int} \cdot Tr_{n-1}(\gamma_{n-1,1} \ ; \ \mathit{fifo}^{n}) \ |_{n<5}$

(fam-sequence)

$$\frac{\Gamma \mid \phi \vdash c_1 : \forall \overline{x_1 : T_1} \cdot I_1 \to J_1 |_{\psi_1} \qquad \Gamma \mid \phi \vdash c_2 : \forall \overline{x_2 : T_2} \cdot I_2 \to J_2 |_{\psi_2} \qquad \overline{x_1} \cap \overline{x_2} = \varnothing}{\Gamma \mid \phi, J_1 = I_2 \vdash c_1 \ ; \ c_2 : \forall \overline{x_1 : T_1}, \overline{x_2 : T_2} \cdot I_1 \to J_2 |_{\psi_1, \psi_2}}$$

$(\lambda \underline{x{:}\mathrm{Int}} \cdot c1) \ ; \ (\lambda \underline{y{:}\mathrm{Int}} \cdot c2) \ : \ \forall \underline{x{:}\mathrm{Int}, y{:}\mathrm{Int}} \cdot I_1 \longrightarrow J_2$

Solving Type Constraints

$$\Gamma \mid \phi \vdash \mathbf{c} : T \mid_{\psi}$$

$\underline{\mathbf{c}}$ is well-typed if:

given an empty context $\underline{\Gamma}$

the type rules yield T, φ, ψ

such that $\boxed{\varphi \wedge \psi \text{ have some solution}}$

Solving Type Constraints

untyped ports:

interfaces as integers

$$(\![0]\!) = 0$$
$$(\![1]\!) = 1$$
$$(\![I \otimes J]\!) = (\![I]\!) + (\![J]\!)$$
$$(\![I^\alpha]\!) = (\![I]\!) * \alpha$$

$\underline{\mathbf{c}}$ is well-ty

given

the ty

such th

Solving Type Constraints

untyped ports:

interfaces as integers

$$(\!|0|\!) = 0$$
$$(\!|1|\!) = 1$$
$$(\!|I \otimes J|\!) = (\!|I|\!) + (\!|J|\!)$$
$$(\!|I^\alpha|\!) = (\!|I|\!) * \alpha$$

```
scala> import paramConnectors.DSL._
import paramConnectors.DSL._

scala> fifo
res1: paramConnectors.Prim =
fifo
   : 1 -> 1

scala> lam(n, fifo | n > 5)
res2: paramConnectors.IAbs =
\n.(fifo | (n > 5))
   : ∀n:I . 1 -> 1 | n > 5

scala> val sequencer = ...
sequencer: paramConnectors.IAbs =
\n.(...)
   : ∀n:I . n -> n

scala> lam(b, b? fifo + drain) &
       lam(c, c? fifo + id*fifo)
res3: paramConnectors.Seq = ...
   : ∀b:B,c:B . 1 -> 1 | c & b
```

Example

$$\text{seq-fifo} = \lambda n{:}\mathrm{Int} \cdot \mathrm{Tr}_{n-1}(\gamma_{n-1,1} \,;\, \mathrm{fifo}^n) \;|_{n<5}$$

$$\varnothing \mid \mathbf{1} \otimes (n-1) = \mathbf{1}^n \;\;,\;\; (n-1) \otimes \mathbf{1} = X \otimes (n-1) \;\;,\;\; \mathbf{1}^n = Y \otimes (n-1)$$
$$\vdash \text{seq-fifo} \;:\; \forall n : \mathbb{N} \cdot X \to Y \,|_{n<5}$$

Solution exists: well-typed.

Enough?

Example

$\text{seq-fifo} = \lambda n{:}\mathrm{Int} \cdot Tr_{n-1}(\gamma_{n-1,1} \,;\, \text{fifo}^n) \mid_{n<5}$

$\varnothing \mid \mathbf{1} \otimes (n-1) = \mathbf{1}^n \;,\; (n-1) \otimes \mathbf{1} = X \otimes (n-1) \;,\; \mathbf{1}^n = Y \otimes (n-1)$
$\vdash \text{seq-fifo} \;:\; \forall n : \mathbb{N} \cdot X \to Y \mid_{n<5}$

$\text{seq-fifo} \;:\; \forall n{:}\mathrm{Int} \cdot 1 \longrightarrow 1 \mid_{n<5}$

3-Phase Solver

1. Simplify	arithmetic rewrites
2. unify	most general unification (partial)
3. constraint solving	off-the-shelf constraint solver + check uniqueness

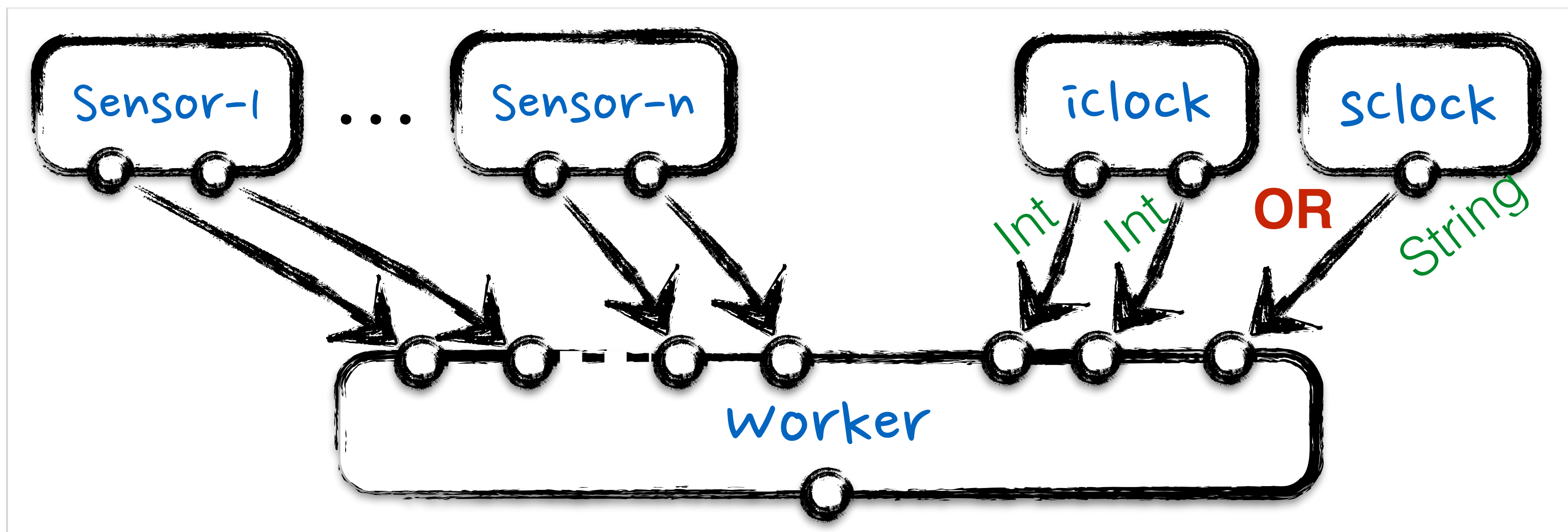
3-Phase Solver

```
scala> debug(seqfifo)
\n.Tr_(n - 1){sym(n - 1,1) ; (fifo^n)}
   : ∀n:I . 1 -> 1
 - type-rules:  ∀n:I . x1 -> x2 | ((x1 + (n - 1)) == ...
 - [ unification: [x1:I -> 1, x2:I -> 1] ]
 - [ missing:     true ]
 - substituted:   ∀n:I . 1 -> 1 | ((1 + (n - 1)) == ...
 - simplified:    ∀n:I . 1 -> 1
 - [ solution:    Some([]) ]
 - post-solver:   ∀n:I . 1 -> 1
 - instantiation: 1 -> 1

scala>
```

Paper+slides: http://facs15.proenca.org

Wrapping up



$(\lambda n{:}\mathrm{Int} \cdot \mathrm{Sensor}^n) \otimes (\lambda b{:}\mathrm{Bool} \cdot (\mathrm{iclock} \oplus^b \mathrm{sclock}))\ ;\ \mathrm{worker}$

parameterised calculus

restriction + composition

solver for type constraints

Paper+slides: http://facs15.proenca.org