Global Principal Typing in Partially Commutative Asynchronous Sessions

ESOP '09

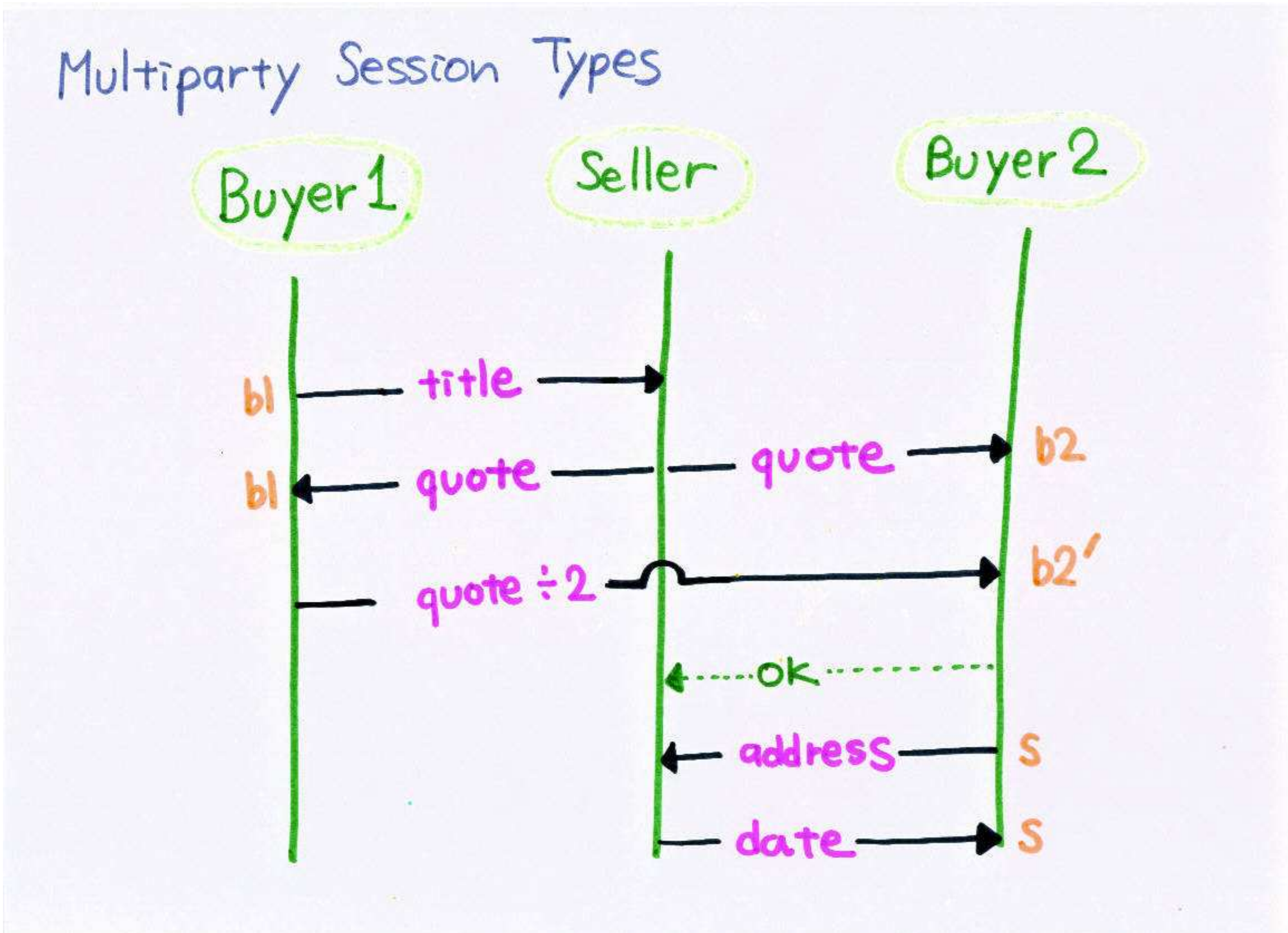
Dimitris Mostrous[1] Nobuko Yoshida[1] Kohei Honda[2]

(1) Imperial College London
(2) Queen Mary, University of London

Friday 27 March 2009

Overview

- asynchronous communication subtyping for structured multi-party interactions
  - flexibility / (type-safe) optimisation
- language is buffered π-calculus with typed m-party sessions
- main points:
  - session typing guarantees *conformance* to specification (e.g. type safety, session fidelity);
  - *top-down* refinement of specification, preserving conformance;
  - *bottom-up* synthesis with inference of global scenario;
  - sound & complete subtyping algorithm (terminating)
  - principal type inference for the synthesis of bottom-up specifications


Multiparty Session Types
Buyer 1
Seller
Buyer 2
b1
title
b1
quote
quote
b2
quote ÷ 2
b2′
ok
address
S
date
S

Two Buyers - Seller Example

Global Type

$B1 \to S : s\langle \mathrm{String} \rangle.$

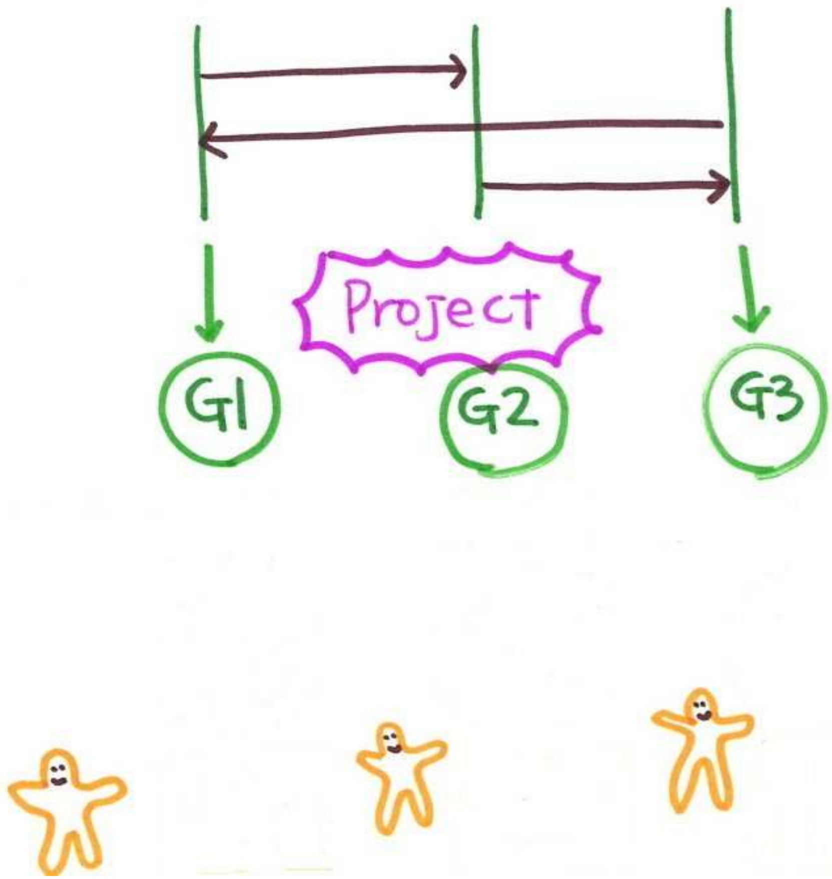
$S \to B1 : b1\langle \mathrm{Int} \rangle.$

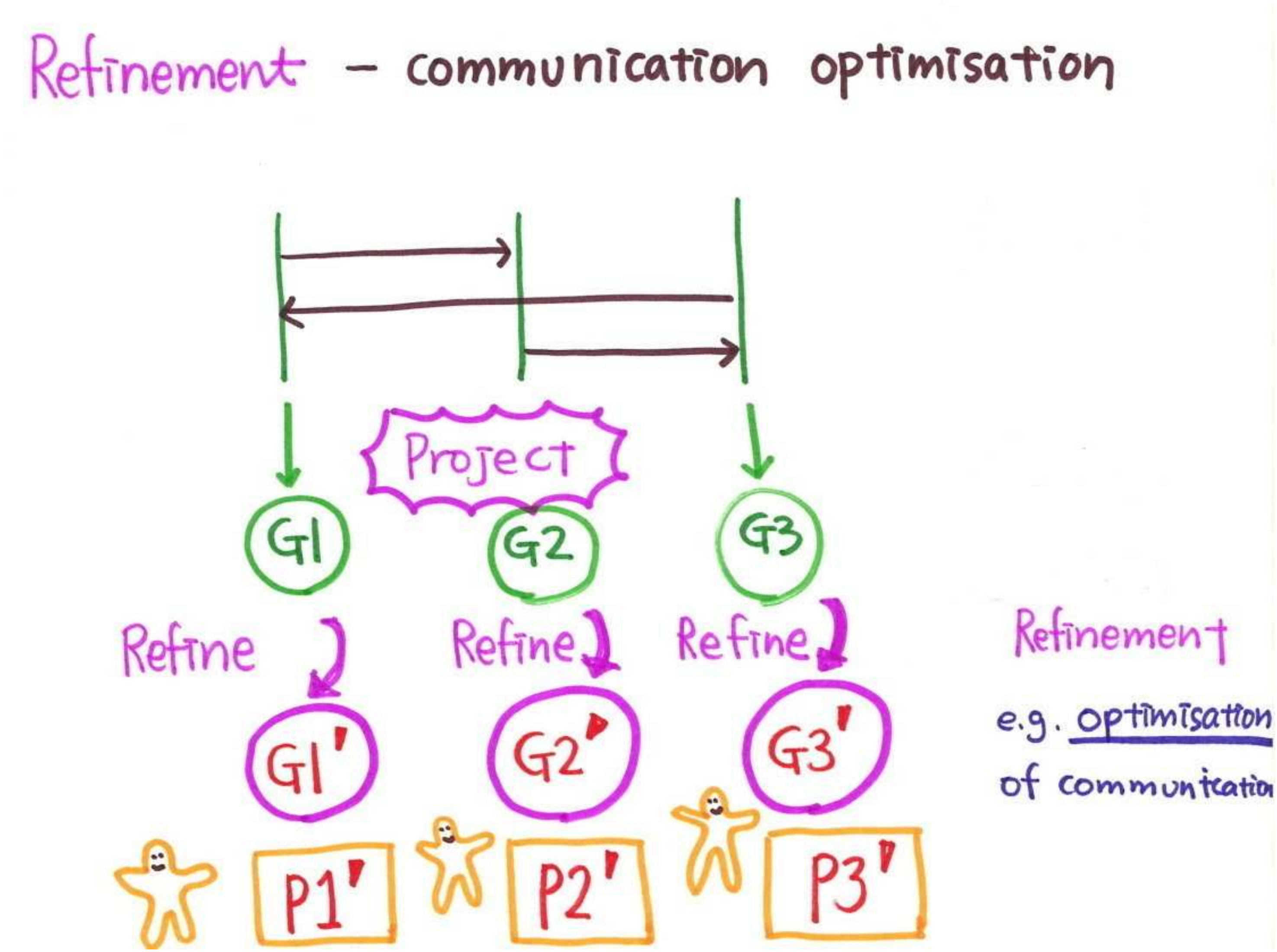
$S \to B2 : b2\langle \mathrm{Int} \rangle.$

$B1 \to B2 : b2'\langle \mathrm{Int} \rangle.$

$B2 \to S : s\{\mathrm{ok} : B2 \to S : s\langle \mathrm{String} \rangle.$
$\qquad S \to B2 : b2\langle \mathrm{Date} \rangle.\ \mathrm{end},$
$\quad \mathrm{quit} : \mathrm{end}\}$

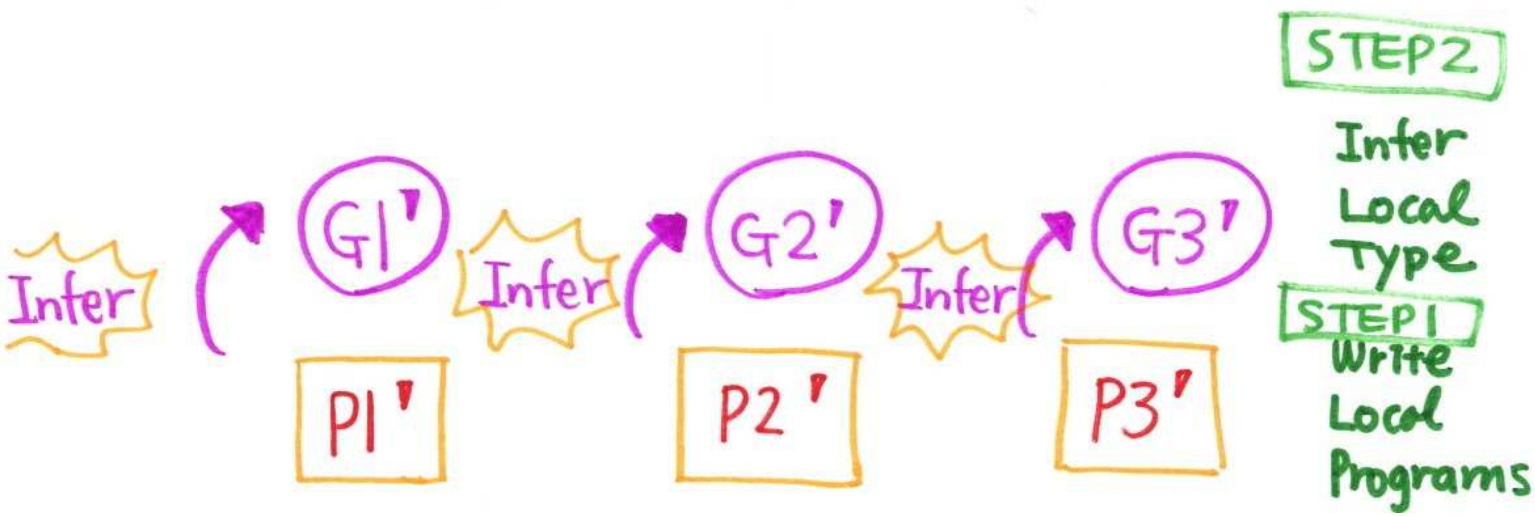
Refinement – communication optimisation



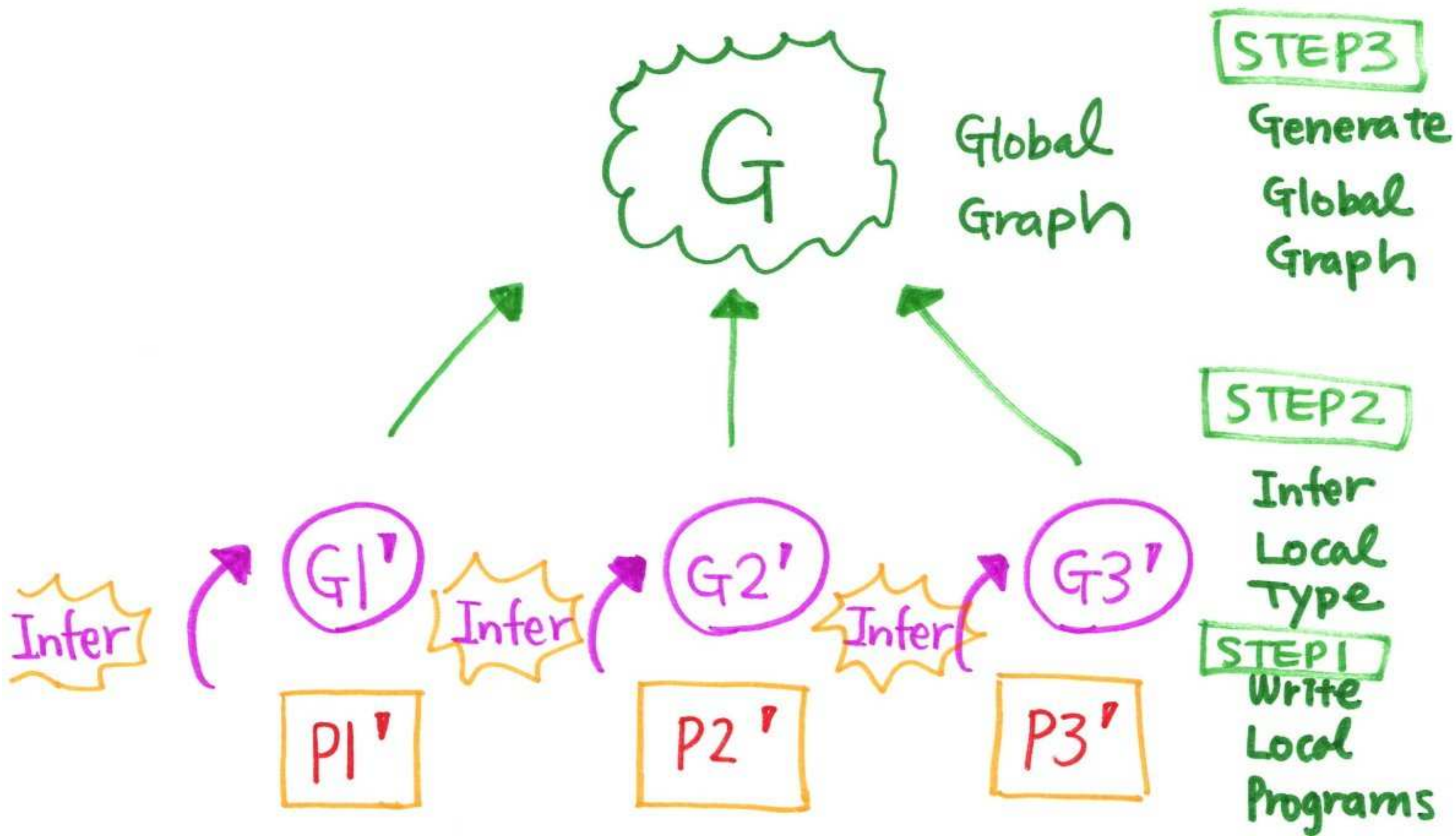

Refinement - communication optimisation
Project
G1
G2
G3
Refine
Refine
Refine
G1'
G2'
G3'
P1'
P2'
P3'
Refinement
e.g. optimisation
of communication

Global Inference – Bottom Up Inference

Global Inference – Bottom Up Inference



Global Inference – Bottom Up Inference



Types

Global	G	$::=$	$\mathtt{p} \to \mathtt{p}' : k\,\langle U \rangle ; G'$	values	$\mid$	$\mu\mathbf{t}.G$
		$\mid$	$\mathtt{p} \to \mathtt{p}' : k\,\{l_j : G_j\}_{j\in J}$	branching	$\mid$	$\mathbf{t}$
		$\mid$	G, G'	parallel	$\mid$	end
Value	U	$::=$	bool $\mid$ nat $\mid$ $\cdots$ $\mid$ G			

Local types for each participant from Global type using $G \upharpoonright \mathtt{p}$

Local

T	$::=$	$k!\,\langle U \rangle ; T$	send	$\mid$	$k\&\{l_i : T_i\}_{i\in I}$	branching
	$\mid$	$k?\,\langle U \rangle ; T$	receive	$\mid$	$\mu\mathbf{t}.T \mid \mathbf{t}$	recursion
	$\mid$	$k \oplus \{l_i : T_i\}_{i\in I}$	selection	$\mid$	end	end

(no delegation of local type T)

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example:

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1' = k!\langle U\rangle; k'?\langle U'\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example:

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1' = k!\langle U\rangle; k'?\langle U'\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example:

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1' = k!\langle U\rangle; k'?\langle U'\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example: $T_1' \ll T_1$

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1' = k!\langle U\rangle; k'?\langle U'\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example: $T_1' \ll T_1$

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1' = k!\langle U\rangle; k'?\langle U'\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

(OI) $\qquad k!\langle U\rangle; k'?\langle U'\rangle; T \quad \ll \quad k'?\langle U'\rangle; k!\langle U\rangle; T$

Partial Permutations

- top-level actions can be permuted using rules of $\ll$
- for example: $T_1' \ll T_1$ $\quad T_2' \not\ll T_2$

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2 = k'!\langle U'\rangle; k?\langle U\rangle; T_1''$$

$$T_1 = k'?\langle U'\rangle; k!\langle U\rangle; T_1' \qquad T_2' = k?\langle U\rangle; k'!\langle U'\rangle; T_1''$$

No progress

n-times nested unfolding

- action required for $\ll$ may be inside recursion
- type can be unrolled internally until it appears in the top-level
- for example, twice-unfold of guarded type:

$$\mathsf{unfold}^2(k?\langle U\rangle;\mu\mathbf{t}.k'!\langle U'\rangle;\mathbf{t}) = k?\langle U\rangle;k'!\langle U'\rangle;k'!\langle U'\rangle;\mu\mathbf{t}.k'!\langle U'\rangle$$

$$\mathsf{unfold}^0(T) = T \text{ for all } T \qquad \mathsf{unfold}^{1+n}(T) = \mathsf{unfold}^1(\mathsf{unfold}^n(T))$$

$$\mathsf{unfold}^1(k!\langle U\rangle;T) = k!\langle U\rangle;\mathsf{unfold}^1(T) \qquad \mathsf{unfold}^1(k\oplus\{l_i : T_i\}_{i\in I}) = k\oplus\{l_i : \mathsf{unfold}^1(T_i)\}_{i\in I}$$

$$\mathsf{unfold}^1(k?\langle U\rangle;T) = k?\langle U\rangle;\mathsf{unfold}^1(T) \qquad \mathsf{unfold}^1(k\&\{l_i : T_i\}_{i\in I}) = k\&\{l_i : \mathsf{unfold}^1(T_i)\}_{i\in I}$$

$$\mathsf{unfold}^1(\mu\mathbf{t}.T) = T[\mu\mathbf{t}.T/\mathbf{t}] \qquad \mathsf{unfold}^1(\mathbf{t}) = \mathbf{t} \qquad \mathsf{unfold}^1(\mathsf{end}) = \mathsf{end}$$

Coinductive Subtyping

- Simulation-based method: we say $T_1 \leqslant_c T_2$ if there exists simulation relation $\Re$ with $(T_1, T_2) \in \Re$.
- for example, if $(T_1, T_2) \in \Re$ we require:
  - If $T_1 = \mathsf{end}$, then $\mathsf{unfold}^n(T_2) = \mathsf{end}$.
  - If $T_1 = k!\langle U_1 \rangle; T_1'$, then $\mathsf{unfold}^n(T_2) \gg k!\langle U_2 \rangle; T_2'$, $(T_1', T_2') \in \Re$ and $(U_1, U_2) \in \Re$.
  - If $T_1 = k\&\{l_i : T_{1i}\}_{i \in I}$, then $\mathsf{unfold}^n(T_2) \gg k\&\{l_j : T_{2j}\}_{j \in J}$ and $J \subseteq I$ and $\forall j \in J.(T_{1j}, T_{2j}) \in \Re$.
- Example $T_1 = k'!; \mu\mathbf{t}.k'!; k?; \mathbf{t}$, $T_2 = \mu\mathbf{t}.k?; k'!; \mathbf{t}$.
  - T_1 represents more optimal communications than T_2 since it can output messages at k' without waiting.
  - We can prove $T_1 \leqslant_c T_2$.

Properties of $\leqslant_c$

- **Theorem** $\leqslant_c$ is a preorder.
  - If $T_1 \mathfrak{R}_1 T_2$ and $T_2 \mathfrak{R}_2 T_3$ for type simulations $\mathfrak{R}_1$ and $\mathfrak{R}_2$ then there exists a type simulation $\mathfrak{R}_3$ such that if $\text{unfold}^n(T_2) \gg T_2'$, then $T_2' \mathfrak{R}_3 T_3$.
  - We write $\mathbf{trc}(T_1 \mathfrak{R}_1 T_2 \mathfrak{R}_2 T_3)$ for $\mathfrak{R}_3$.
  - $\mathbf{trc}(\mathfrak{R}_1, \mathfrak{R}_2)$ is the smallest relation such that if $(T_1, T_2) \in \mathfrak{R}_1$ and $(T_2, T_3) \in \mathfrak{R}_2$, then $\mathbf{trc}(T_1 \mathfrak{R}_1 T_2 \mathfrak{R}_2 T_3) \subseteq \mathbf{trc}(\mathfrak{R}_1, \mathfrak{R}_2)$

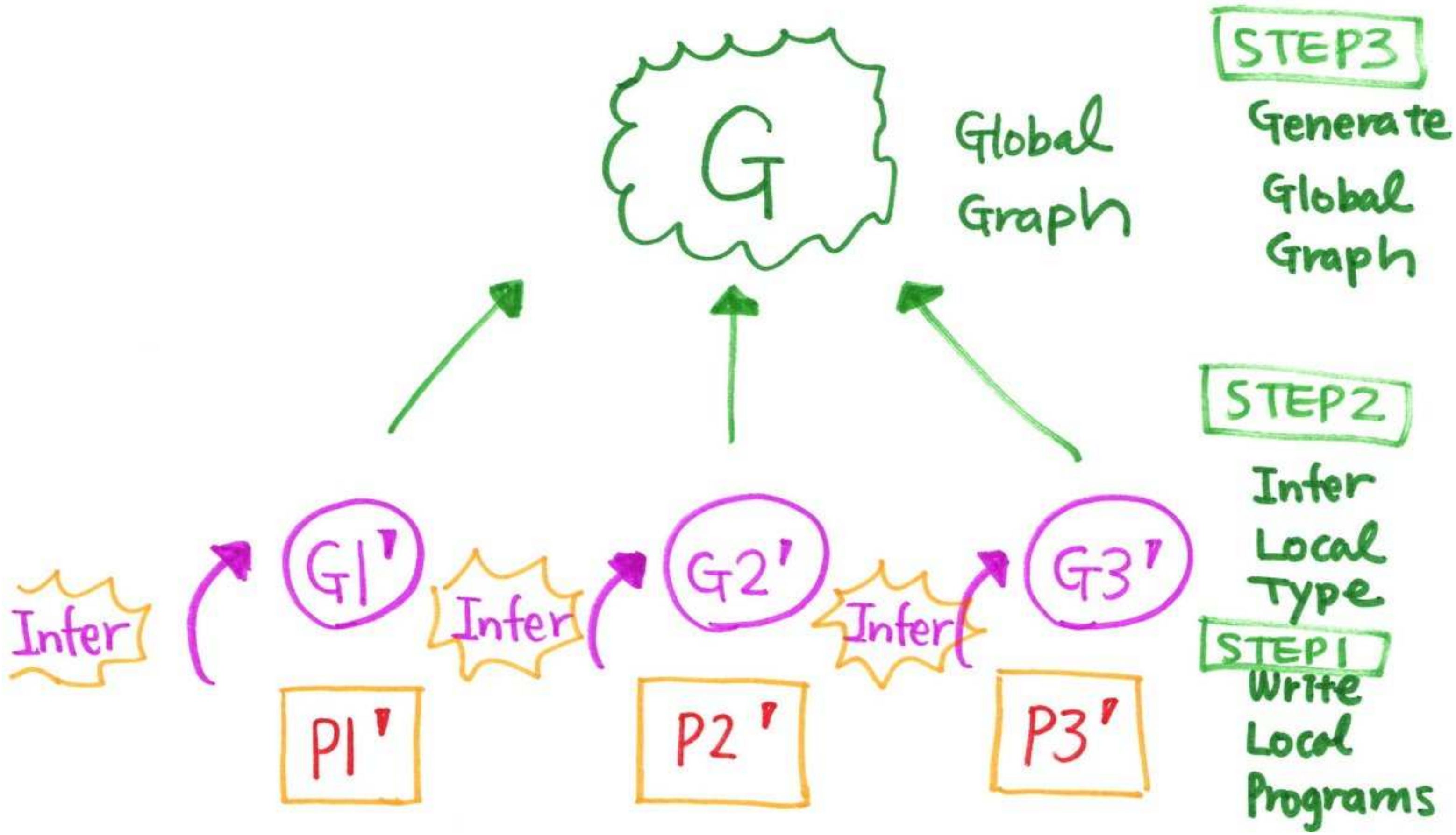
Algorithmic Subtyping

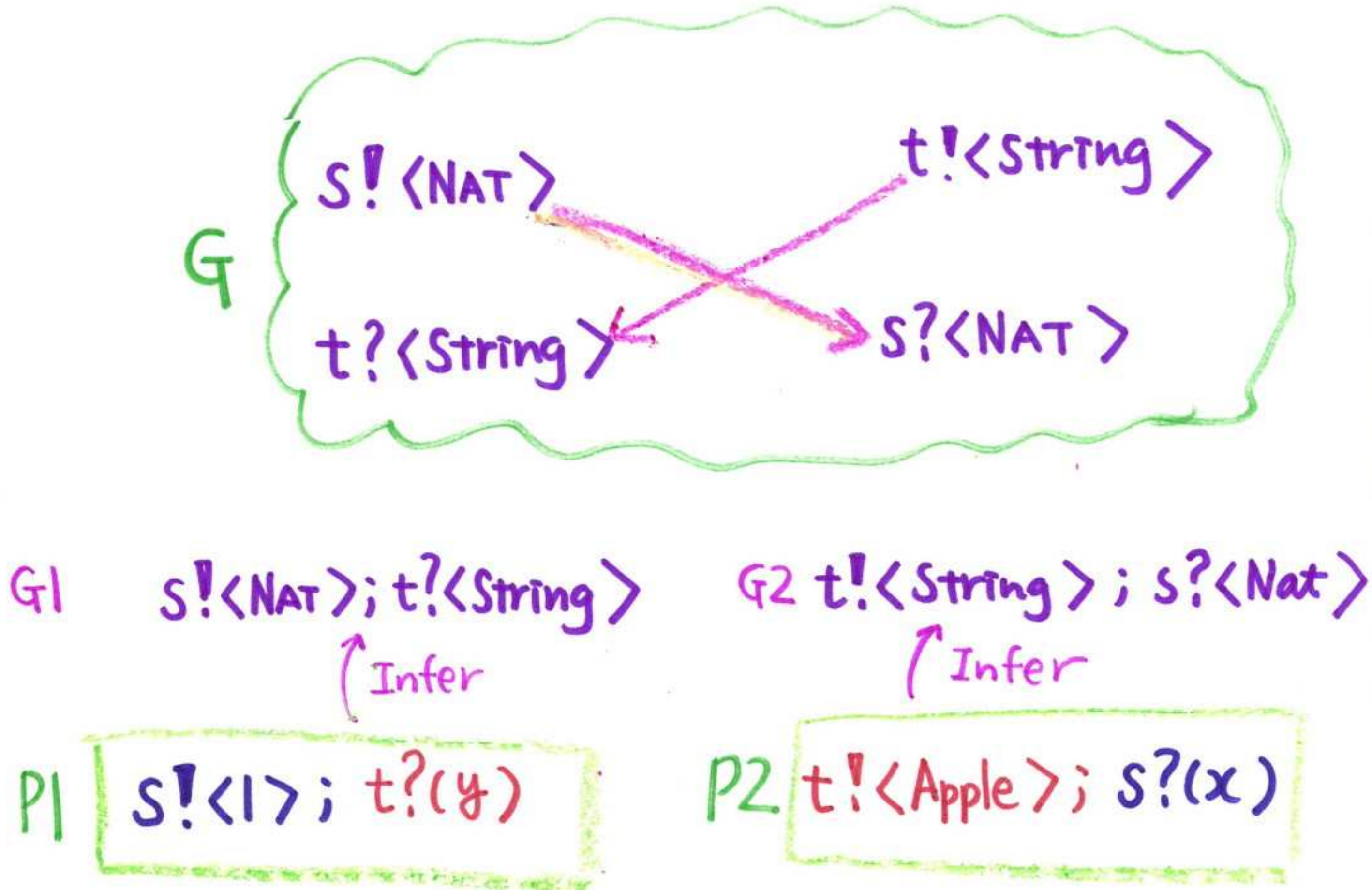
$$[\textsc{Out}]\quad \frac{\Sigma \vdash U_1 \leqslant U_2 \qquad \Sigma \vdash T_1 \leqslant \mathcal{T}\,[T'_{2h}]^{h\in H} \qquad \mathcal{T}\,[k!\,\langle U_2\rangle; T_{2h}]^{h\in H} \overset{k}{\longmapsto} k!\,\langle U_2\rangle; \mathcal{T}\,[T'_{2h}]^{h\in H}}{\Sigma \vdash k!\,\langle U_1\rangle; T_1 \leqslant \mathcal{T}\,[k!\,\langle U_2\rangle; T_{2h}]^{h\in H}}$$

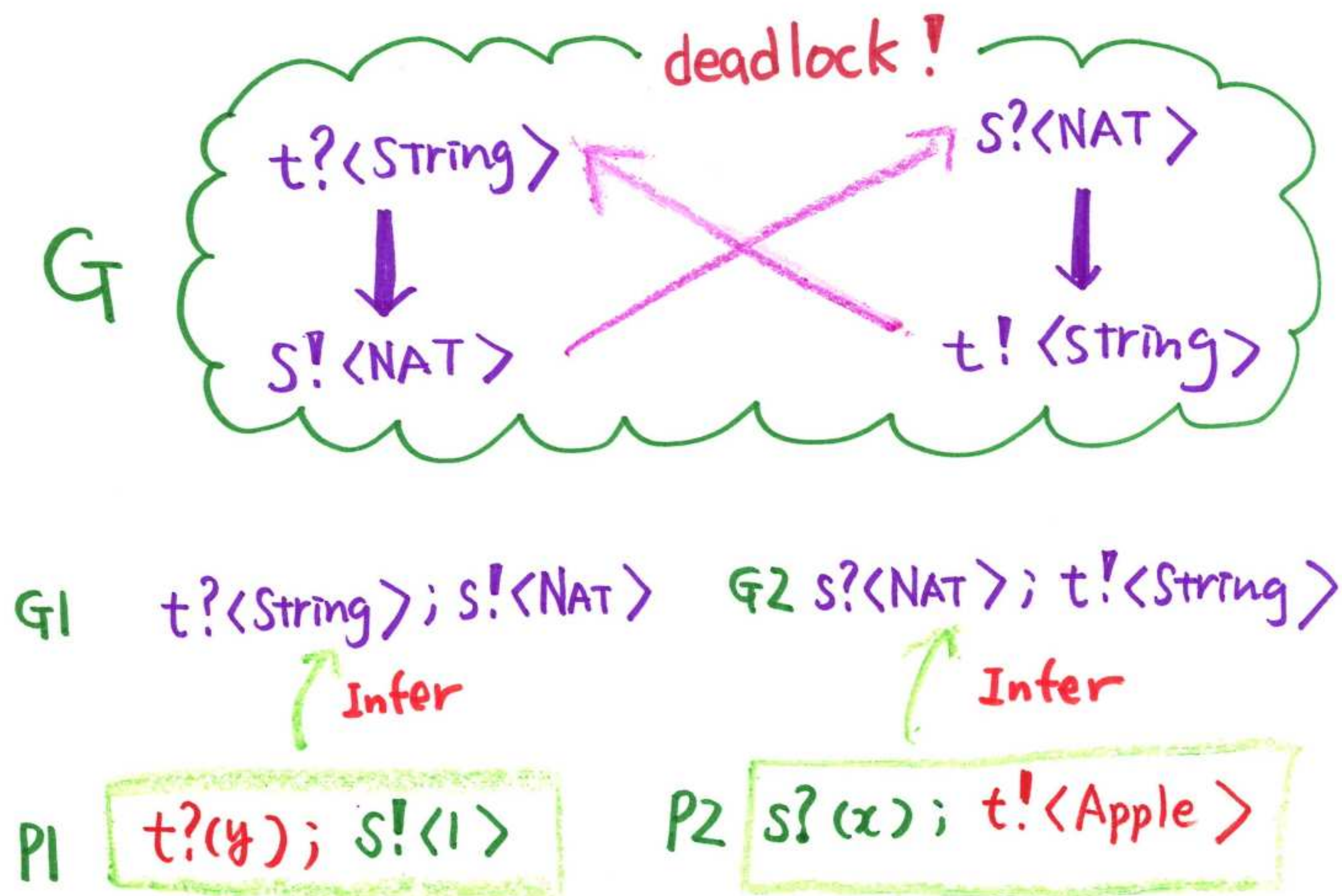
- **Lemma**
  - The subtyping algorithm always terminates.
  - If $T \leqslant_c T'$ then the algorithm does not return false when applied to $\Sigma \vdash T \leqslant T'$.

- **Theorem** For all closed types T and T', $T \leqslant_c T'$ if and only if $T \leqslant T'$

Global Inference – Bottom Up Inference




G
s!⟨NAT⟩
t!⟨String⟩
t?⟨String⟩
s?⟨NAT⟩
G1 s!⟨NAT⟩; t?⟨String⟩
G2 t!⟨String⟩; s?⟨Nat⟩
Infer
Infer
P1 s!⟨1⟩; t?(y)
P2 t!⟨Apple⟩; s?(x)

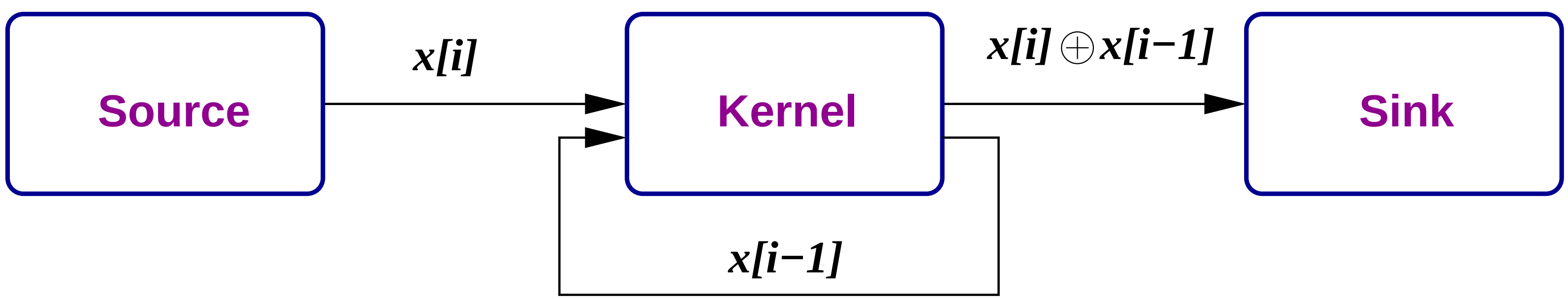

deadlock!
G
t?⟨String⟩
S!⟨NAT⟩
S?⟨NAT⟩
t!⟨String⟩
G1 t?⟨String⟩; S!⟨NAT⟩
G2 S?⟨NAT⟩; t!⟨String⟩
Infer
Infer
P1 t?(y); S!⟨1⟩
P2 S?(x); t!⟨Apple⟩

Global Principal Typing

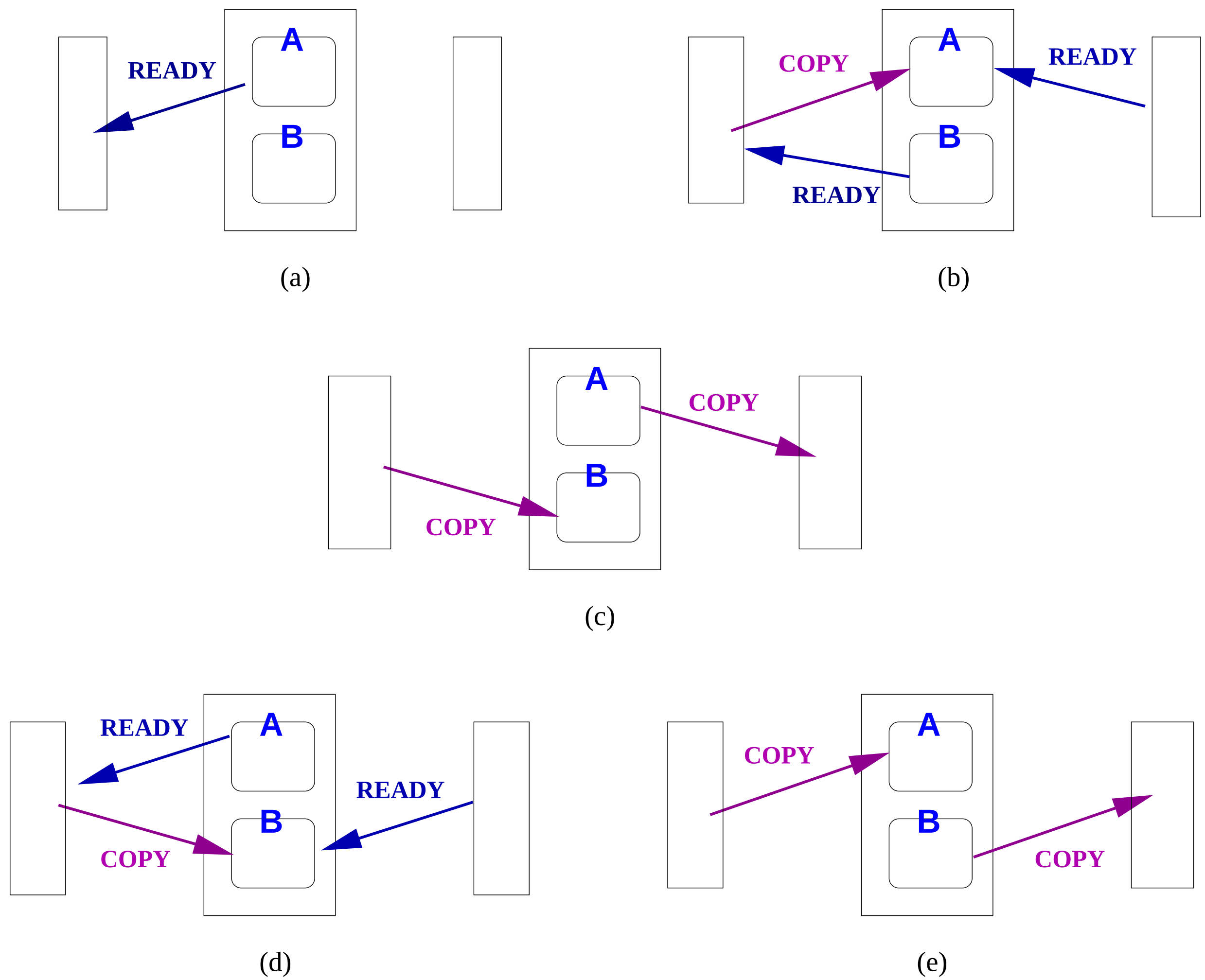
- **Theorem [principal global typing]**
  - The typability of P with respect to $\vdash_{\mathrm{g}}$ is decidable.
  - If P is typable then P has a *principal global typing* Γ_0 in the sense that
    - $\Gamma_0 \vdash_{\mathrm{g}} P \triangleright \emptyset$ holds and;
    - $\Gamma \vdash_{\mathrm{g}} P \triangleright \emptyset$ implies $\Gamma_0 \leqslant \Gamma$.

Example: Double Buffering Algorithm

- Optimisation by overlapping computation and communication
- Source — Kernel — Sink
  - Source sends data to Kernel
  - Kernel computes on data
  - Kernel sends to Sink
  - Use of 2 buffers at Kernel allows Sink to write in one while Sink reads from the other.



Example: Double Buffering Algorithm



Types for Double Buffering

- **Original Local Types**

$$T_{\mathsf{source}} = \mu \mathbf{t}.r_1?\langle\rangle;s_1!\langle U\rangle;r_2?\langle\rangle;s_2!\langle U\rangle;\mathbf{t}$$

$$T_{\mathsf{kernel}} = \mu \mathbf{t}.r_1!\langle\rangle;s_1?\langle U\rangle;t_1?\langle\rangle;u_1!\langle U\rangle; r_2!\langle\rangle;s_2?\langle U\rangle;t_2?\langle\rangle;u_2!\langle U\rangle;\mathbf{t}$$

$$T_{\mathsf{sink}} = \mu \mathbf{t}.t_1!\langle\rangle;u_1?\langle U\rangle;t_2!\langle\rangle;u_2?\langle U\rangle;\mathbf{t}$$

- **Optimized Kernel Type**

$$T_{\mathsf{opt}} = r_1!\langle\rangle;r_2!\langle\rangle;\mu \mathbf{t}.s_1?\langle U\rangle;t_1?\langle\rangle;u_1!\langle U\rangle;r_1!\langle\rangle; s_2?\langle U\rangle;t_2?\langle\rangle;u_2!\langle U\rangle;r_2!\langle\rangle;\mathbf{t}$$

- **Theorem** $T_{\mathsf{opt}} \leqslant_c T_{\mathsf{kernel}}$

References

- Asynchronous Multiparty Session Types (Marco Carbone, Kohei Honda and Nobuko Yoshida) [POPL'08]
- Global Progress in Dynamically Interleaved Multiparty Sessions (Lorenzo Bettini, Mario Coppo, Loris D'Antoni, Marco De Luca, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida) [CONCUR'08]
- *Synchronous* Multiparty Session Types (Andi Bejleri, Nobuko Yoshida) [PLACES'08]
- Global Principal Typing in Partially Commutative Asynchronous Sessions [ESOP'09]
- Session-Based Communication Optimisation for Higher-Order Mobile Processes (Dimitris Mostrous, Nobuko Yoshida) [TLCA'09] www.doc.ic.ac.uk/ ∼mostrous