Type Systems for a Network Specification Language With Multiple-Choice Let *

Yarom Gabay, Assaf J. Kfoury, Likai Liu, Azer Bestavros, Adam D. Bradley, and Ibrahim Matta
{yarom, kfoury, liulk, best, artdodge, matta}@cs.bu.edu
Department of Computer Science, Boston University

Technical Report: BUCS-TR-2005-034

Abstract

When analysing the behavior of complex networked systems, it is often the case that some components within that network are only known to the extent that they belong to one of a set of possible "implementations" – e.g., versions of a specific protocol, class of schedulers, etc. In this report we augment the specification language considered in BUCS-TR-2004-021, BUCS-TR-2005-014, BUCS-TR-2005-015, and BUCS-TR-2005-033, to include a non-deterministic multiple-choice let-binding, which allows us to consider compositions of networking subsystems that allow for looser component specifications.

1 Introduction

In this report we augment the specification language considered in [1], [2] and [3], to include a non-deterministic multiple-choice let-binding. Briefly, with such a let-binding, we can write a specification – call it $\mathcal{A}$ – of the form:

let $x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ **in** $\mathcal{B}$

where $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are flow specifications and $\mathcal{B}$ is a flow specification that may mention x as a free flow variable. Intuitively, the specification $\mathcal{A}$ says that any of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ can be selected non-deterministically and substituted for every occurrence of x in $\mathcal{B}$. We say that $\mathcal{A}$ is safe iff, no matter which of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is selected, the resulting specification after the substitution is safe. These ideas are made precise in due course.

We present three systems for the analysis of flow specifications written in a language augmented with non-deterministic multiple-choice let-binding: System A, System B, and the system **Exact**. Following the approach proposed in [1], [2] and [3], an analysis of a specification corresponds to inferring a typing of the specification in an appropriately defined type system. We show that System A is sound but not complete with respect to System B, so that all flow specifications that can be type-checked in System A can also be typed-checked in System B but not vice-versa. We also show that System B is sound but not complete with respect to **Exact**, thus establishing an ordering on the expressiveness of these 3 systems. As will become clear from our examination, this increase of expressiveness from System A to System B, and from System B to **Exact**, comes with a significant increase in the cost.

We envision a strategy whereby the user will ascertain the safety of a flow specification by first trying System A – technically, the specification has a typing in System A. If the attempt using System A fails, the user will next try System B, but with an added cost. And if the attempt using System B fails, the user will last try **Exact**, provided the prohibitive cost is also tolerable.

*This work is partially supported by NSF award CCR-0205294.

2 Semantic Understanding of Multiple Choices

2.1 Syntax of Flow Specifications

Recall from [1], the language of global-flow is defined using the following BNF:

$$
\begin{array}{rcll}
x, y, z & \in & \mathsf{FlowVar} & \text{flow variable} \\
A, B, C & \in & \mathsf{LocalFlow} & \text{local flow} \\
\mathcal{A}, \mathcal{B}, \mathcal{C} & \in & \mathsf{GlobalFlow} ::= A \mid x & \\
 & & \mid \mathcal{A}; \mathcal{B} & \text{sequential flow} \\
 & & \mid \mathcal{A} \| \mathcal{B} & \text{parallel flow} \\
 & & \mid \textbf{let } x = \mathcal{A} \textbf{ in } \mathcal{B} & \text{let-binding}
\end{array}
$$

For the purpose of identification, we refer to this language as *single-choice let*, or simply Single. We now define the language of *multiple-choice let* as follows:

$$
\begin{array}{rcll}
\mathcal{A}, \mathcal{B}, \mathcal{C} & \in & \mathsf{GlobalFlow} ::= A \mid x & \\
 & & \mid \mathcal{A}; \mathcal{B} & \text{sequential flow} \\
 & & \mid \mathcal{A} \| \mathcal{B} & \text{parallel flow} \\
 & & \mid \textbf{let } x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \textbf{ in } \mathcal{B} & \text{let-binding}
\end{array}
$$

and we call this language Multi. Somewhat ambiguously, we use Single and Multi to denote the above BNF grammars as well as the sets of expressions generated by these grammars.

2.2 Safety of Multiple-Choice Let

Let $\{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$ and $\{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$ be finite sets of flow specifications, all written in the specification language with *single-choice let*. We write:

$$
\begin{array}{lll}
\{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \,\|\, \{\mathcal{B}_1, \ldots, \mathcal{B}_n\} & \text{to denote} & \{\mathcal{A}_i \| \mathcal{B}_j \mid 1 \le i \le m, 1 \le j \le n\} \\
\{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \,;\, \{\mathcal{B}_1, \ldots, \mathcal{B}_n\} & \text{to denote} & \{\mathcal{A}_i ; \mathcal{B}_j \mid 1 \le i \le m, 1 \le j \le n\} \\
\textbf{let}^* \; x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \;\textbf{ in }\; \{\mathcal{B}_1, \ldots, \mathcal{B}_n\} & \text{to denote} & \{\textbf{let } x = \mathcal{A}_i \textbf{ in } \mathcal{B}_j \mid 1 \le i \le m, 1 \le j \le n\}
\end{array}
$$

Define a transformation $[\![\,]\!] : \mathsf{Multi} \to \mathcal{P}(\mathsf{Single})$, where Multi is the flow specification language with *multiple-choice* let, Single is the flow specification language with *single-choice* let. Note that the transformation returns a finite subset of expressions in Single

$$
\begin{array}{rcl}
[\![x]\!] & \triangleq & \{x\} \\
[\![A]\!] & \triangleq & \{A\} \\
[\![\mathcal{A}_1; \mathcal{A}_2]\!] & \triangleq & [\![\mathcal{A}_1]\!] \,;\, [\![\mathcal{A}_2]\!] \\
[\![\mathcal{A}_1 \| \mathcal{A}_2]\!] & \triangleq & [\![\mathcal{A}_1]\!] \,\|\, [\![\mathcal{A}_2]\!] \\
[\![\textbf{let } x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_m\} \textbf{ in } \mathcal{B}]\!] & \triangleq & \textbf{let}^* \; x \in [\![\mathcal{A}_1]\!] \cup \cdots \cup [\![\mathcal{A}_m]\!] \;\textbf{ in }\; [\![\mathcal{B}]\!]
\end{array}
$$

The transformation $[\![\,]\!]$ defines the semantic of multiple-choice let in terms of single-choice let. This allows us to express the desired safety property of multiple-choice let in terms of the safety of single-choice let established in [2].

Definition 2.1 (Safety of Multiple-Choice Let)**.** *Let $\mathcal{A} \in$* Multi*. Then $\mathcal{A}$ is* safe *iff every $\mathcal{B} \in [\![\mathcal{A}]\!]$ is* safe *(safety of* Single *is discussed in [1]).*

Assessing the safety of non-deterministic multiple-choice flows requires the type system to explore the choice-space, which explodes combinatorially. Hence, performing an exact type-checking is time consuming. An approximation type-checking algorithm would reduce the exploration of search-space while rejecting some safe flows as a trade-off. There are safe specifications in Multi that will not be derived a type in an approximation type-system. On the other hand, in 4. We show that **Exact** derives a type for a specification if and only if the specification is safe.

3 The Approximation Type-Systems

In this section, we introduce System A and System B that provide a coarse approximation of safety property. Approximation is done in the form of taking the *least common supertype* of all choices. System A and System B differ only in the place for computing a least common supertype when analyzing a multiple-choice let-binding. System A computes the least common supertype of all the choices before analyzing the let-body, in order to avoid exploring the choice-space in the body. System B applies all the choices to the body and takes the least common supertype of all outcomes. The difference of the two approaches reflect in the fact that System A is sound but not complete with respect to System B. On the other hand, System B is more costly to compute.

3.1 Least Common Supertype

For the purpose of typing *let-choice* we define the *least common supertype*.

Definition 3.1 (Least Common Supertype)**.** *Let τ, τ_1, τ_2, …, τ_n be types. The least common supertype of τ_1, τ_2, …, τ_n denoted LCSup$\{\tau_1, \tau_2, \ldots, \tau_n\}$, is a type τ such that:*

(i) for every $i \in \{1, \ldots, n\}$ $\tau_i <: \tau$, and

(ii) for every τ', if for every $i \in \{1, \ldots, n\}$ it holds that $\tau_i <: \tau'$ then $\tau <: \tau'$.

Similarly, we define least common supertype for flow types.

Definition 3.2 (Greatest Common Subtype)**.** *Let τ, τ_1, τ_2, …, τ_n be types. The greatest common subtype of τ_1, τ_2, …, τ_n denoted GCSub$\{\tau_1, \tau_2, \ldots, \tau_n\}$, is a type τ such that:*

(i) for every $i \in \{1, \ldots, n\}$ $\tau <: \tau_i$, and

(ii) for every τ', if for every $i \in \{1, \ldots, n\}$ it holds that $\tau' <: \tau_i$ then $\tau' <: \tau$.

Similarly, we define greatest common subtype for flow types.

Lemma 3.3 (LCSup Calculation)**.** *Let T_1 and T_2 be the flow types $\begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}$ and $\begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}$ respectively. The following holds:*

(i) $LCSup\{T_1, T_2\}$ is defined iff $GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\}$, $LCSup\{\sigma_1, \sigma_3\}$ and $GCSub\{\sigma_2, \sigma_4\}$ are defined.

(ii) if $LCSup\{T_1, T_2\}$ is defined then

$$LCSup\left\{\begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}, \begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}\right\} = \begin{bmatrix}GCSub\{\rho_1, \rho_3\} & LCSup\{\rho_2, \rho_4\}\\ LCSup\{\sigma_1, \sigma_3\} & GCSub\{\sigma_2, \sigma_4\}\end{bmatrix}$$

3.2 Typing Rules

The typing rules in [1] are:

$$\frac{\Gamma(x) = T}{\Gamma, \Delta \vdash x : T}\ (\mathsf{var}) \quad \frac{type(A) = T}{\Gamma, \Delta \vdash A : T}\ (\mathsf{local}) \quad \frac{\Gamma, \Delta \vdash \mathcal{A}_1 : T_1 \quad \Gamma, \Delta \vdash \mathcal{A}_2 : T_2}{\Gamma, \Delta \vdash \mathcal{A}_1 \| \mathcal{A}_2 : T_1 \bullet T_2}\ (\mathsf{par})$$

$$\frac{\Gamma, \Delta \vdash \mathcal{A}_1 : \begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix} \quad \Gamma, \Delta \vdash \mathcal{A}_2 : \begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix} \quad \Delta \vdash \rho_2 <: \rho_3 \quad \Delta \vdash \sigma_3 <: \sigma_2}{\Gamma, \Delta \vdash \mathcal{A}_1; \mathcal{A}_2 : \begin{bmatrix}\rho_1 & \rho_4\\ \sigma_1 & \sigma_4\end{bmatrix}}\ (\mathsf{seq})$$

$$\frac{\Gamma, \Delta \vdash \mathcal{A} : T \quad \Gamma \cup \{x : T'\}, \Delta \vdash \mathcal{B} : T'' \quad \Delta \vdash T <: T'}{\Gamma, \Delta \vdash let\ x = \mathcal{A}\ in\ B : T''}\ (\mathsf{let})$$

We consider two alternatives for typing let-choice.

1. $$\frac{\Gamma, \Delta \vdash \mathcal{A}_i : T_i \text{ for every } i \in \{1, \ldots, n\} \quad T = LCSup\{T_1, \ldots, T_n\} \quad \Gamma \cup \{x : T\}, \Delta \vdash \mathcal{B} : T'}{\Gamma, \Delta \vdash let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B} : T'}\ (\mathsf{let\text{-}choice\text{-}A})$$

2. $$\frac{\Gamma, \Delta \vdash \mathcal{A}_i : T_i \quad \Gamma \cup \{x : T_i\}, \Delta \vdash \mathcal{B} : T_i' \text{ for every } i \in \{1, \ldots, n\} \quad T' = LCSup\{T_1', \ldots, T_n'\}}{\Gamma, \Delta \vdash let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B} : T'}\ (\mathsf{let\text{-}choice\text{-}B})$$

While (let-choice-A) first calculates the LCSup of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and then types $\mathcal{B}$ with x bound to the calculated LCSup, (let-choice-B) first types $\mathcal{B}$ with every one of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and then calculates the LCSup of all the resulting types.

Definition 3.4 (System A)**.** *Let System A be the type system with the set of rules* (var), (local), (par), (seq), *and* (let-choice-A).

Definition 3.5 (System B)**.** *Let System B be the type system with the set of rules* (var), (local), (par), (seq), *and* (let-choice-B).

Whenever we need to distinguish a derivation in System A from a derivation in System B we use the symbols $\vdash_A$ and $\vdash_B$ respectively.

We no longer need the rule (let) in System A and System B since (let) can be expressed by the new (let-choice-A) and (let-choice-B) rules respectively. The equivalence of System A and System B is a required property since System B more accurately expresses the semantics of let-choice where System A has a more algorithmic approach. If the two systems were equivalent then we could say System A achieves the same results as System B with much less computation since in the typing of let-choice it includes one LCSup calculation and one type calculation for $\mathcal{B}$ while System B includes n type calculations for $\mathcal{B}$ and one LCSup calculation. This is obviously not a proof for the efficiency of System A relative to System B but merely an intuition.

Although the equivalence of the two systems is a required property they are really different. Section 3 presents a proof for the soundness of System A relative to System B, meaning that every global-flow specification that is typable in System A is typable in System B. Section 4 shows that completeness of System A does not hold by showing a counter example where a simple global-flow specification is typable in System B but not typable in System A. The global-flow specification in the example is a valid one that should not be filtered out by the type system and so it shows System A might be too strict.

3.3 Soundness of System A to System B

Lemma 3.6. *Let Γ be a context, Δ a set of subtyping assumptions, $\mathcal{A}_1, \ldots, \mathcal{A}_n$ global-flow specifications, $\mathcal{B}$ a global-flow specification, T, T' flow types, and $T_{A_1}, \ldots, T_{A_n}$ flow types.*
Hypothesis:

(a1) $\Gamma, \Delta \vdash_A \mathcal{A}_i : T_{A_i}$ *for every* $i \in \{1, \ldots, n\}$

(a2) $T = LCSup\{T_{A_1}, \ldots, T_{A_n}\}$

(a3) $\Gamma \cup \{x : T\}, \Delta \vdash_A \mathcal{B} : T'$

Conclusion:

(c1) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B} : T_i'$ *for every* $i \in \{1, \ldots, n\}$, *for some* $T_1', \ldots, T_n'$

(c2) $T' = LCSup\{T_1', \ldots, T_n'\}$

The proof is presented in the appendix.

Theorem 3.7 (Soundness of System A Relative to System B)**.** *Let Γ be a context, Δ a subtyping assumption set, $\mathcal{A}$ a global-flow specification, and T a flow type. If $\Gamma, \Delta \vdash_A \mathcal{A} : T'$ then $\Gamma, \Delta \vdash_B \mathcal{A} : T'$.*

Proof. By induction on type derivation. Consider the last derivation rule,

- Case (var), (local), (seq) or (par). The result here is immediate since we have the subderivation by IH and the very same last derivation can be made in System B.
- Case (let-choice-A). By the assumption we have

  (s1) $\mathcal{A} = let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B}$
  (s2) $\Gamma, \Delta \vdash_A \mathcal{A}_i : T_i$ for every $i \in \{1, \ldots, n\}$
  (s3) $T = LCSup\{T_1, \ldots, T_n\}$
  (s4) $\Gamma \cup \{x : T\}, \Delta \vdash_A \mathcal{B} : T'$

  By (s2),(s3),(s4) and Lemma 3.6 we infer

  (s5) $\Gamma \cup \{x : T_i\}, \Delta \vdash_B \mathcal{B} : T'_i$ for every $i \in \{1, \ldots, n\}$, for some $T'_1, \ldots, T'_n$
  (s6) $T' = LCSup\{T'_1, \ldots, T'_n\}$

  By applying IH on (s2) we infer

  (s7) $\Gamma, \Delta \vdash_B \mathcal{A}_i : T_i$ for every $i \in \{1, \ldots, n\}$

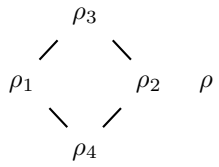
  By (let-choice-B) and (s7),(s5),(s6) we conclude
  $\Gamma, \Delta \vdash_B let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B} : T'$

□

3.4 Incompleteness of System A to System B

In this section we show the incompleteness of System A relative to System B by a counter example. In the example we present a global-flow specification that is derivable in System B but not derivable in System A. The global-flow specification presented in the example is a valid one that should not be filtered out by the type system and so it implies that System A is too strict.

Counter Example Showing System A is Incomplete Let $\rho, \rho_1, \rho_2, \rho_3, \rho_4$ be forward types; let σ be a backward type. The subtyping relations between the different forward types are depicted by the following figure (the forward type ρ has no subtyping relations with any of the other types),



Note 3.8 (Subtyping Relation for Counter-Example)**.** *According to these subtyping relations, we formally write Δ (not including reflexive closures) as*

$$\Delta = \{\rho_1 <: \rho_3, \rho_2 <: \rho_3, \rho_4 <: \rho_1, \rho_4 <: \rho_2, \rho_4 <: \rho_3\}.$$

Let C_1, C_2, C_3, C_4 be local flows with the flow types T_1, T_2, T_3, T_4 respectively, where

$$T_1 = \begin{bmatrix} \rho_1 & \rho_1 \\ \sigma & \sigma \end{bmatrix}, \quad T_2 = \begin{bmatrix} \rho_2 & \rho_2 \\ \sigma & \sigma \end{bmatrix}, \quad T_3 = \begin{bmatrix} \rho & \rho_4 \\ \sigma & \sigma \end{bmatrix}, \quad T_4 = \begin{bmatrix} \rho_3 & \rho \\ \sigma & \sigma \end{bmatrix}.$$

Let T be $LCSup\{T_1, T_2\}$.

So we have

$$\begin{aligned} T &= \begin{bmatrix} GCSub\{\rho_1, \rho_2\} & LCSup\{\rho_1, \rho_2\} \\ LCSup\{\sigma, \sigma\} & GCSub\{\sigma, \sigma\} \end{bmatrix} && \text{(according to Lemma 3.3)} \\ &= \begin{bmatrix} \rho_4 & \rho_3 \\ \sigma & \sigma \end{bmatrix} && \text{(according to the subtyping relations presented above)} \end{aligned}$$

We examine the following global-flow specification $\mathcal{A}$

$$\mathcal{A} \triangleq \ let\ x \in \{C_1, C_2\}\ in\ (C_3; x); (x; C_4)$$

We try to type $\mathcal{A}$ in both systems, System A and System B.

Type Derivation in System A

We have $T = LCSup\{T_1, T_2\}$ and we need to find
$\{x : T\}, \Delta \vdash (C_3; x); (x; C_4) : ?$

(1)

$$\frac{\dfrac{type(C_3) = T_3}{\{x:T\}, \Delta \vdash C_3 : T_3} \quad \dfrac{\Gamma(x) = T}{\{x:T\}, \Delta \vdash x : T} \quad \Delta \vdash \rho_4 <: \rho_4 \quad \Delta \vdash \sigma <: \sigma}{\{x:T\}, \Delta \vdash C_3; x : \begin{bmatrix} \rho & \rho_3 \\ \sigma & \sigma \end{bmatrix}}$$

(2)

$$\frac{\dfrac{\Gamma(x) = T}{\{x:T\}, \Delta \vdash x : T} \quad \dfrac{type(C_4) = T_4}{\{x:T\}, \Delta \vdash C_4 : T_4} \quad \Delta \vdash \rho_3 <: \rho_3 \quad \Delta \vdash \sigma <: \sigma}{\{x:T\}, \Delta \vdash x; C_4 : \begin{bmatrix} \rho_4 & \rho \\ \sigma & \sigma \end{bmatrix}}$$

(3) In order to complete the type derivation, the following judgment must be made.

$$\frac{(1) \quad (2) \quad \Delta \vdash \rho_3 <: \rho_4 \quad \Delta \vdash \sigma <: \sigma}{\{x:T\}, \Delta \vdash (C_3; x); (x; C_4) : \begin{bmatrix} \rho & \rho \\ \sigma & \sigma \end{bmatrix}}$$

But this derivation cannot be carried out since ρ_3 is not a subtype of ρ_4 and so a type derivation for $\mathcal{A}$ in System A does not exist.

Type Derivation in System B

Here, we need to find T_1' such that $\{x : T_1\}, \Delta \vdash (C_3; x); (x; C_4) : T_1'$, then find T_2' such that $\{x : T_2\}, \Delta \vdash (C_3; x); (x; C_4) : T_2'$, and then find T' such that $T' = LCSup\{T_1', T_2'\}$

(1)

$$\frac{\dfrac{type(C_3) = T_3}{\{x : T_1\}, \Delta \vdash C_3 : T_3} \quad \dfrac{\Gamma(x) = T_1}{\{x : T_1\}, \Delta \vdash x : T_1} \quad \Delta \vdash \rho_4 <: \rho_1 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_1\}, \Delta \vdash C_3; x : \begin{bmatrix} \rho & \rho_1 \\ \sigma & \sigma \end{bmatrix}}$$

(2)

$$\frac{\dfrac{\Gamma(x) = T_1}{\{x : T_1\}, \Delta \vdash x : T_1} \quad \dfrac{type(C_4) = T_4}{\{x : T_1\}, \Delta \vdash C_4 : T_4} \quad \Delta \vdash \rho_1 <: \rho_3 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_1\}, \Delta \vdash x; C_4 : \begin{bmatrix} \rho_1 & \rho \\ \sigma & \sigma \end{bmatrix}}$$

(3)

$$\frac{(1) \quad (2) \quad \Delta \vdash \rho_1 <: \rho_1 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_1\}, \Delta \vdash (C_3; x); (x; C_4) : \begin{bmatrix} \rho & \rho \\ \sigma & \sigma \end{bmatrix}}$$

(4)

$$\frac{\dfrac{type(C_3) = T_3}{\{x : T_2\}, \Delta \vdash C_3 : T_3} \quad \dfrac{\Gamma(x) = T_2}{\{x : T_2\}, \Delta \vdash x : T_2} \quad \Delta \vdash \rho_4 <: \rho_2 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_2\}, \Delta \vdash C_3; x : \begin{bmatrix} \rho & \rho_2 \\ \sigma & \sigma \end{bmatrix}}$$

(5)

$$\frac{\dfrac{\Gamma(x) = T_2}{\{x : T_2\}, \Delta \vdash x : T_2} \quad \dfrac{type(C_4) = T_4}{\{x : T_2\}, \Delta \vdash C_4 : T_4} \quad \Delta \vdash \rho_2 <: \rho_3 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_2\}, \Delta \vdash x; C_4 : \begin{bmatrix} \rho_2 & \rho \\ \sigma & \sigma \end{bmatrix}}$$

(6)

$$\frac{(4) \quad (5) \quad \Delta \vdash \rho_2 <: \rho_2 \quad \Delta \vdash \sigma <: \sigma}{\{x : T_2\}, \Delta \vdash (C_3; x); (x; C_4) : \begin{bmatrix} \rho & \rho \\ \sigma & \sigma \end{bmatrix}}$$

From (3) and (6) we get T_1' and T_2' respectively. Since $T_1' = T_2'$ it is trivial that $T' = LCSup\{T_1', T_2'\} = \begin{bmatrix} \rho & \rho \\ \sigma & \sigma \end{bmatrix}$. We conclude,

$$\varnothing, \Delta \vdash let\ x \in \{C_1, C_2\}\ in\ (C_3; x); (x; C_4) : T'$$

Theorem 3.9 (Incompleteness of System A Relative to System B)**.** *There is a global-flow specification $\mathcal{A}$ such that $\mathcal{A}$ is typable in System B and $\mathcal{A}$ is not typable in System A.*

Proof. Shown by counter example in section 5.1 □

4 Type System "Exact"

4.1 Syntax of Types

$$\begin{array}{rcl} \mathscr{T} & ::= & T \\ & | & \mathscr{T}, T \end{array}$$

Type Selectors

We define the following type selectors:

1. Given a flow type $T = \begin{bmatrix} \rho_1 & \rho_2 \\ \sigma_1 & \sigma_2 \end{bmatrix}$, the following operations are defined: $\mathsf{f\text{-}in}(T) = \rho_1$, $\mathsf{f\text{-}out}(T) = \rho_2$, $\mathsf{b\text{-}out}(T) = \sigma_1$ and $\mathsf{b\text{-}in}(T) = \sigma_2$.
2. Given a type $\mathscr{T} = T_1, \ldots, T_n$, the following operation is defined: $\mathsf{s}_i(\mathscr{T}) = T_i$ for $1 \leq i \leq n$. $\mathsf{s}_i(\mathscr{T})$ is undefined for improper index.
3. Given a type $\mathscr{T} = T_1, \ldots, T_n$, the following operation is defined: $|\mathscr{T}| = n$.
4. The type $\mathscr{T} = T_1, \ldots, T_n$ is denoted by $(T_i)^{1 \leq i \leq n}$. We also use this notation in the extended version as follows: For every $1 \leq j \leq m$, let $\mathscr{T}_j = T_{j,1}, \ldots, T_{j,n_j}$. The type $\mathscr{T}_1, \ldots, \mathscr{T}_m$ is denoted by $(\mathscr{T}_j)^{1 \leq j \leq m}$.

4.2 Typing Rules

$$(\mathsf{var}) \quad \frac{\Gamma(x) = T}{\Gamma, \Delta \vdash x : T}$$

$$(\mathsf{local}) \quad \frac{type(A) = T}{\Gamma, \Delta \vdash A : T}$$

$$(\mathsf{par}) \quad \frac{\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T} \quad \Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}'}{\Gamma, \Delta \vdash \mathcal{A}_1 \| \mathcal{A}_2 : (\mathsf{s}_i(\mathscr{T})\mathsf{s}_j(\mathscr{T}'))^{1 \leq i \leq |\mathscr{T}|,\, 1 \leq j \leq |\mathscr{T}'|}}$$

$$(\mathsf{seq}) \quad \frac{\begin{array}{l} \Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T} \quad \Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_i(\mathscr{T})) <: \mathsf{f\text{-}in}(\mathsf{s}_j(\mathscr{T}')) \\ \Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}' \quad \Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_j(\mathscr{T}')) <: \mathsf{b\text{-}in}(\mathsf{s}_i(\mathscr{T})) \quad \text{for every } 1 \leq i \leq |\mathscr{T}|,\, 1 \leq j \leq |\mathscr{T}'| \end{array}}{\Gamma, \Delta \vdash \mathcal{A}_1 ; \mathcal{A}_2 : \left(\begin{bmatrix} \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T})) & \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}')) \\ \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T})) & \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}')) \end{bmatrix} \right)^{1 \leq i \leq |\mathscr{T}|,\, 1 \leq j \leq |\mathscr{T}'|}}$$

$$(\mathsf{let\text{-}choice}) \quad \frac{\Gamma, \Delta \vdash \mathcal{A}_i : \mathscr{T}_i \quad \Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash \mathcal{B} : \mathscr{T}'_{i,j} \quad \text{for every } 1 \leq i \leq n,\, 1 \leq j \leq |\mathscr{T}_i|}{\Gamma, \Delta \vdash let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B} : (\mathscr{T}'_{i,j})^{1 \leq i \leq n,\, 1 \leq j \leq |\mathscr{T}_i|}}$$

4.3 "Exact" Characterizes Safety

In order to justify the name of **Exact**, we need to show that **Exact** has the property of typing a specification if and only if the specification is safe (according to definition 2.1). This implies that the set of typable specifications in **Exact** is the same as the set of safe specifications. Hence, the name Exact. For this purpose we extend the notation of a judgment as follows

Definition 4.1.

$$\Gamma, \Delta \vdash\!\!\rightarrow \{A_1, \ldots, A_n\} : \mathscr{T} \triangleq \textit{ for every } 1 \leq i \leq n \quad \Gamma, \Delta \vdash A_i : \mathsf{s}_i(\mathscr{T})$$

Note:

- The expression "$\Gamma, \Delta \vdash\!\!\rightarrow \{A_1, \ldots, A_n\} : \mathscr{T}$" is not a formal judgment, which can be defined using inference rules, but rather an abbreviation for $n \geq 1$ distinct judgments.
- Implicitly we impose $|\mathscr{T}| = n$.

Lemma 4.2. *Let $\mathcal{A} \in \mathsf{Multi}$, $\mathscr{T}$ be a type in* **Exact**, *Γ be a context, and Δ be a set of subtyping assumptions. If the judgment $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ can be derived in* **Exact** *then $\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}]\!] : \mathscr{T}$ holds.*

Lemma 4.3. *Let $\mathcal{A} \in \mathsf{Multi}$, $\mathscr{T}$ be a type in* **Exact**, Γ *be a context, and Δ be a set of subtyping assumptions. If $\Gamma, \Delta \vdash\!\!\rightsquigarrow [\![\mathcal{A}]\!] : \mathscr{T}$ holds then the judgment $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ can be derived in* **Exact**.

Theorem 4.4 ("Exact" Characterizes Safety)**.** *Let $\mathcal{A} \in \mathsf{Multi}$, $\mathscr{T}$ be a type in* **Exact**, Γ *be a context, and Δ be a set of subtyping assumptions. The judgment $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ can be derived iff $\Gamma, \Delta \vdash\!\!\rightsquigarrow [\![\mathcal{A}]\!] : \mathscr{T}$ holds.*

Proof. Implied by Lemma 4.2 and Lemma 4.3 □

The accuracy of **Exact** is implied by Theorem 4.4.

4.4 Soundness of System B relative to "Exact"

Lemma 4.5 (Parallel Flows Preserve Suptyping)**.** *Let T_1, T_2, T_3 and T_4 be flow types and Δ a set of subtyping assumptions. If $\Delta \vdash T_1 <: T_3$ and $\Delta \vdash T_2 <: T_4$ then $\Delta \vdash T_1 \bullet T_2 <: T_3 \bullet T_4$.*

Theorem 4.6 (Soundness of System B relative to "**Exact**")**.** *Let Γ be a context, Δ a set of subtyping assumptions, $\mathcal{A}$ a global-flow specification, T a flow type in System B and $\mathscr{T}$ a flow type in* **Exact**. *$\Gamma, \Delta \vdash_B \mathcal{A} : T$ implies $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$, where $\Delta \vdash \mathsf{s}_i(\mathscr{T}) <: T$ for every $1 \leq i \leq |\mathscr{T}|$.*

Proof. By induction on the structure of $\mathcal{A}$:

1. $\mathcal{A} = A$

   The same derivation can be made in **Exact**.

2. $\mathcal{A} = x$

   The same derivation can be made in **Exact**.

3. $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$

   By the assumptions we have

   $\Gamma, \Delta \vdash_B \mathcal{A}_1 : T_1$ for some type flow T_1 and (1)

   $\Gamma, \Delta \vdash_B \mathcal{A}_2 : T_2$ for some type flow T_2 where (2)

   $T = T_1 \bullet T_2$ (3)

   By the IH and (1) we make the following judgment

   $\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1$ where $\Delta \vdash \mathsf{s}_i(\mathscr{T}_1) <: T_1$ for every $1 \leq i \leq |\mathscr{T}_1|$ (4)

   By the IH and (2) we make the following judgment

   $\Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}_2$ where $\Delta \vdash \mathsf{s}_j(\mathscr{T}_2) <: T_2$ for every $1 \leq j \leq |\mathscr{T}_2|$ (5)

   Using (4),(5) and (par) we make the following judgment

   $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$, where $\mathscr{T} = (\mathsf{s}_i(\mathscr{T}_1) \bullet \mathsf{s}_j(\mathscr{T}_2))^{1 \leq i \leq |\mathscr{T}_1|,\, 1 \leq j \leq |\mathscr{T}_2|}$ (6)

   By (4),(5) and Lemma 4.5 we make the following judgments

   for every $1 \leq i \leq |\mathscr{T}_1|$, for every $1 \leq j \leq |\mathscr{T}_2|$ $\quad \Delta \vdash \mathsf{s}_i(\mathscr{T}_1) \bullet \mathsf{s}_j(\mathscr{T}_2) <: T_1 \bullet T_2$ (7)

4. $\mathcal{A} = \mathcal{A}_1 ; \mathcal{A}_2$

   By the assumptions we have

   $\Gamma, \Delta \vdash_B \mathcal{A}_1 : T_1$ for some type flow T_1 and (1)

   $\Gamma, \Delta \vdash_B \mathcal{A}_2 : T_2$ for some type flow T_2 with (2)

   $\Delta \vdash \mathsf{f\text{-}out}(T_1) <: \mathsf{f\text{-}in}(T_2)$ and (3)

   $\Delta \vdash \mathsf{b\text{-}out}(T_2) <: \mathsf{b\text{-}in}(T_1)$ and (4)

   $$T = \begin{bmatrix} \mathsf{f\text{-}in}(T_1) & \mathsf{f\text{-}out}(T_2) \\ \mathsf{b\text{-}out}(T_1) & \mathsf{b\text{-}in}(T_2) \end{bmatrix} \quad (5)$$

   By the IH and (1) we make the following judgment

$\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1$ where $\Delta \vdash \mathsf{s}_i(\mathscr{T}_1) <: T_1$ for every $1 \leq i \leq |\mathscr{T}_1|$ (6)

By the IH and (2) we make the following judgment

$\Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}_2$ where $\Delta \vdash \mathsf{s}_j(\mathscr{T}_2) <: T_2$ for every $1 \leq j \leq |\mathscr{T}_2|$ (7)

From (6) and (ftype-lift) we have the following for every $1 \leq i \leq |\mathscr{T}_1|$

$\Delta \vdash \mathsf{f\text{-}in}(T_1) <: \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T}_1))$ and (8)

$\Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) <: \mathsf{f\text{-}out}(T_1)$ and

$\Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) <: \mathsf{b\text{-}out}(T_1)$ and

$\Delta \vdash \mathsf{b\text{-}in}(T_1) <: \mathsf{b\text{-}in}(\mathsf{s}_i(\mathscr{T}_1))$

From (7) and (ftype-lift) we have the following for every for every $1 \leq j \leq |\mathscr{T}_2|$

$\Delta \vdash \mathsf{f\text{-}in}(T_2) <: \mathsf{f\text{-}in}(\mathsf{s}_j(\mathscr{T}_2))$ and (9)

$\Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) <: \mathsf{f\text{-}out}(T_2)$ and

$\Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) <: \mathsf{b\text{-}out}(T_2)$ and

$\Delta \vdash \mathsf{b\text{-}in}(T_2) <: \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}_2))$

The following subtyping relations hold for every $1 \leq i \leq |\mathscr{T}_1|$, for every $1 \leq j \leq |\mathscr{T}_2|$

(10)

$$\begin{array}{lcll} \Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) & <: & \mathsf{f\text{-}out}(T_1) & \text{(by (8))} \\ & <: & \mathsf{f\text{-}in}(T_2) & \text{(by (3))} \\ & <: & \mathsf{f\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) & \text{(by (9))} \end{array}$$

In a similar way for every $1 \leq i \leq |\mathscr{T}_1|$, for every $1 \leq j \leq |\mathscr{T}_2|$

$\Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) <: \mathsf{b\text{-}in}(\mathsf{s}_i(\mathscr{T}_1))$ (11)

By (6), (7), (10), (11) and (seq) we make the following judgment

$$\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T} \text{ where } \mathscr{T} = \left(\begin{bmatrix} \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) \\ \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) \end{bmatrix} \right)^{1 \leq i \leq |\mathscr{T}_1|,\, 1 \leq j \leq |\mathscr{T}_2|} \quad (12)$$

By (5), (8), (9) and (ftype-lift) we have the following

for every $1 \leq i \leq |\mathscr{T}_1||\mathscr{T}_2|$ $\Delta \vdash \mathsf{s}_i(\mathscr{T}) <: T$ (13)

5. $\mathcal{A}$ = let $x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ in $\mathcal{B}$.

By the assumption we have

$\Gamma, \Delta \vdash_B \mathcal{A}_i : T_i$ for every $1 \leq i \leq n$ and (1)

$\Gamma \cup \{x : T_i\}, \Delta \vdash_B \mathcal{B} : T'_i$ for every $1 \leq i \leq n$ and (2)

$T = LCSup\{T'_1, \ldots, T'_n\}$ (3)

By the IH and (1) we have

$\Gamma, \Delta \vdash \mathcal{A}_i : \mathscr{T}_i$ where $\Delta \vdash \mathsf{s}_j(\mathscr{T}_i) <: T_i$ for every $1 \leq i \leq n$, for every $1 \leq j \leq |\mathscr{T}_i|$ (4)

By the IH and (2) we have

$\Gamma \cup \{x : T_i\}, \Delta \vdash \mathcal{B} : \mathscr{T}'_i$ where $\Delta \vdash \mathsf{s}_k(\mathscr{T}'_i) <: T'_i$ for every $1 \leq i \leq n$, for every $1 \leq k \leq |\mathscr{T}'_i|$ (5)

And furthermore using (3) we also have

$\Delta \vdash \mathsf{s}_k(\mathscr{T}'_i) <: T$ for every $1 \leq i \leq n$, for every $1 \leq k \leq |\mathscr{T}'_i|$ (6)

By (4),(5) and the subsumption rule we make the following judgments

$\Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash \mathcal{B} : \mathscr{T}'_i$ for every $1 \leq i \leq n$, for every $1 \leq j \leq |\mathscr{T}_i|$ (7)

Let $\mathscr{T} = (\mathscr{T}'_i)^{1 \leq i \leq n}$. According to (let-choice) and (6) and (7) we make the judgment

$\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ and by (6) we have $\Delta \vdash \mathsf{s}_\ell(\mathscr{T}) <: T$ for every $1 \leq \ell \leq |\mathscr{T}|$

□

4.5 Incompleteness of System B relative to "Exact"

Using the same setup of Δ in Note 3.8, we examine a different flow example that exploits the weakness of System B. Let C_1, C_2 be local flows with the following types T_1, T_2 respectively, where

$$T_1 = \begin{bmatrix} \rho_1 & \rho_1 \\ \sigma & \sigma \end{bmatrix}, \quad T_2 = \begin{bmatrix} \rho_2 & \rho_2 \\ \sigma & \sigma \end{bmatrix}.$$

Let T be $LCSup\{T_1, T_2\}$.

So we have

$$\begin{aligned} T &= \begin{bmatrix} GCSub\{\rho_1, \rho_2\} & LCSup\{\rho_1, \rho_2\} \\ LCSup\{\sigma, \sigma\} & GCSub\{\sigma, \sigma\} \end{bmatrix} && \text{(according to Lemma 3.3)} \\ &= \begin{bmatrix} \rho_4 & \rho_3 \\ \sigma & \sigma \end{bmatrix} && \text{(according to the subtyping relations presented above)} \end{aligned}$$

We examine the following global-flow specification $\mathcal{A}$

$$\begin{aligned} \mathcal{A} \triangleq\ & let\ y = \\ & \quad let\ x \in \{C_1, C_2\} in\ x;x \\ & in\ y;y \end{aligned}$$

We try to type $\mathcal{A}$ in both systems, System B and **Exact**.

Type Derivation in System B

We use the following sub-derivations

$$\frac{type(C_1) = T_1}{\varnothing, \Delta \vdash C_1 : T_1} \tag{1}$$

$$\frac{type(C_2) = T_2}{\varnothing, \Delta \vdash C_2 : T_2} \tag{2}$$

$$\frac{\dfrac{\Gamma(x) = T_1}{\{x : T_1\}, \Delta \vdash x : T_1} \quad \dfrac{\Gamma(x) = T_1}{\{x : T_1\}, \Delta \vdash x : T_1} \quad \Delta \vdash \mathsf{f\text{-}out}(T_1) <: \mathsf{f\text{-}in}(T_1) \quad \Delta \vdash \mathsf{b\text{-}out}(T_1) <: \mathsf{b\text{-}in}(T_1)}{\{x : T_1\}, \Delta \vdash x;x : T_1} \tag{3}$$

$$\frac{\dfrac{\Gamma(x) = T_2}{\{x : T_2\}, \Delta \vdash x : T_2} \quad \dfrac{\Gamma(x) = T_2}{\{x : T_2\}, \Delta \vdash x : T_2} \quad \Delta \vdash \mathsf{f\text{-}out}(T_2) <: \mathsf{f\text{-}in}(T_2) \quad \Delta \vdash \mathsf{b\text{-}out}(T_2) <: \mathsf{b\text{-}in}(T_2)}{\{x : T_2\}, \Delta \vdash x;x : T_2} \tag{4}$$

We attempt to use the above sub-derivations in the following derivation

$$\dfrac{\dfrac{(1)\quad(2)\quad(3)\quad(4)}{\varnothing, \Delta \vdash let\ x \in \{C_1, C_2\}\ in\ x; x : T} \quad \dfrac{\dfrac{\Gamma(y) = T}{\{y : T\}, \Delta \vdash y : T} \quad \dfrac{\Gamma(y) = T}{\{y : T\}, \Delta \vdash y : T} \quad \begin{array}{l}\Delta \vdash \mathsf{f\text{-}out}(T) <: \mathsf{f\text{-}in}(T) \\ \Delta \vdash \mathsf{b\text{-}out}(T) <: \mathsf{b\text{-}in}(T)\end{array}}{\{y : T\}, \Delta \vdash y; y : T}}{\varnothing, \Delta \vdash let\ y = (let\ x \in \{C_1, C_2\}\ in\ x; x)\ in\ y; y : T}$$

However, this derivation cannot be carried out. We don't have the assumptions $\Delta \vdash \mathsf{f\text{-}out}(T) <: \mathsf{f\text{-}in}(T)$, since $\Delta \vdash \rho_3 \not<: \rho_4$.

Type Derivation in "Exact"

By using the above sub-derivations, which also hold in **Exact**, and the following sub-derivations

(5)

$$\dfrac{\dfrac{\Gamma(y) = T_1}{\{y : T_1\}, \Delta \vdash y : T_1} \quad \dfrac{\Gamma(y) = T_1}{\{y : T_1\}, \Delta \vdash y : T_1} \quad \Delta \vdash \mathsf{f\text{-}out}(T_1) <: \mathsf{f\text{-}in}(T_1) \quad \Delta \vdash \mathsf{b\text{-}out}(T_1) <: \mathsf{b\text{-}in}(T_1)}{\{y : T_1\}, \Delta \vdash y; y : T_1}$$

(6)

$$\dfrac{\dfrac{\Gamma(y) = T_2}{\{y : T_2\}, \Delta \vdash y : T_2} \quad \dfrac{\Gamma(y) = T_2}{\{y : T_2\}, \Delta \vdash y : T_2} \quad \Delta \vdash \mathsf{f\text{-}out}(T_2) <: \mathsf{f\text{-}in}(T_2) \quad \Delta \vdash \mathsf{b\text{-}out}(T_2) <: \mathsf{b\text{-}in}(T_2)}{\{y : T_2\}, \Delta \vdash y; y : T_2}$$

We derive the following

$$\dfrac{\dfrac{(1)\quad(2)\quad(3)\quad(4)}{\varnothing, \Delta \vdash let\ x \in \{C_1, C_2\}\ in\ x; x : T_1, T_2} \quad (5) \quad (6)}{\varnothing, \Delta \vdash let\ y = (let\ x \in \{C_1, C_2\}\ in\ x; x)\ in\ y; y : T_1, T_2}$$

Theorem 4.7 (Incompleteness of System B Relative to "**Exact**")**.** *There is a global-flow specification $\mathcal{A}$ such that $\mathcal{A}$ is typable in* **Exact** *and $\mathcal{A}$ is not typable in System B.*

Proof. As shown by the counter example above. □

A Proofs

A.1 Lemma 3.3

For the purpose of proving this lemma we use (ftype-lift), defined in [1], which expresses a subtyping relation between two flow types.

$$\frac{\Delta \vdash \Delta <: \rho_3\rho_1 \quad \Delta \vdash \Delta <: \rho_2\rho_4 \quad \Delta \vdash \Delta <: \sigma_1\sigma_3 \quad \Delta \vdash \Delta <: \sigma_4\sigma_2}{\Delta \vdash \Delta <: \begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}\begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}} \text{ (ftype-lift)}$$

Lemma. *Let T_1 and T_2 be the flow types $\begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}$ and $\begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}$ respectively. The following holds:*

(i) *$LCSup\{T_1, T_2\}$ is defined iff $GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\}$, $LCSup\{\sigma_1, \sigma_3\}$ and $GCSub\{\sigma_2, \sigma_4\}$ are defined.*

(ii) *if $LCSup\{T_1, T_2\}$ is defined then*

$$LCSup\left\{\begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}, \begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}\right\} = \begin{bmatrix}GCSub\{\rho_1, \rho_3\} & LCSup\{\rho_2, \rho_4\}\\ LCSup\{\sigma_1, \sigma_3\} & GCSub\{\sigma_2, \sigma_4\}\end{bmatrix}$$

Proof.

(i) (1) *First we assume $LCSup\{T_1, T_2\}$ is defined and prove $GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\}$, $LCSup\{\sigma_1, \sigma_3\}$ and $GCSub\{\sigma_2, \sigma_4\}$ are defined.*

Let T be $LCSup\{T_1, T_2\}$ with the following form $\begin{bmatrix}\rho_5 & \rho_6\\ \sigma_5 & \sigma_6\end{bmatrix}$. We prove that $GCSub\{\rho_1, \rho_3\}$ exists and is in fact ρ_5.

$T = LCSup\{T_1, T_2\}$ and so by (ftype-lift)we have

(s1) $\rho_5 <: \rho_1$, $\rho_2 <: \rho_6$, $\sigma_1 <: \sigma_5$ and $\sigma_6 <: \sigma_2$

(s2) $\rho_5 <: \rho_3$, $\rho_4 <: \rho_6$, $\sigma_3 <: \sigma_5$ and $\sigma_6 <: \sigma_4$

So ρ_5 holds the first part of GCSub definition, $\rho_5 <: \rho_1$ and $\rho_5 <: \rho_3$. In order to prove ρ_5 is $GCSub\{\rho_1, \rho_3\}$ we still need to show that for every type ρ'_5 that satisfies (s3), it holds that $\rho'_5 <: \rho_5$.

(s3) $\rho'_5 <: \rho_1$ and $\rho'_5 <: \rho_3$

Let T' be $\begin{bmatrix}\rho'_5 & \rho_6\\ \sigma_5 & \sigma_6\end{bmatrix}$.

From (s1), (s3) and (ftype-lift)we derive

(s4)
$$\frac{\rho'_5 <: \rho_1 \quad \rho_2 <: \rho_6 \quad \sigma_1 <: \sigma_5 \quad \sigma_6 <: \sigma_2}{\Delta \vdash \Delta <: T_1T'}$$

From (s2), (s3) and (ftype-lift)we derive

(s5)
$$\frac{\rho'_5 <: \rho_3 \quad \rho_4 <: \rho_6 \quad \sigma_3 <: \sigma_5 \quad \sigma_6 <: \sigma_4}{\Delta \vdash \Delta <: T_2T'}$$

(s4), (s5) and the fact that $T = LCSup\{T_1, T_2\}$ imply that

(s6) $T <: T'$

From (s6) and (ftype-lift)we have, $\rho'_5 <: \rho_5$ and we conclude ρ_5 is $GCSub\{\rho_1, \rho_3\}$.
Similarly we can prove that, ρ_6 is $LCSup\{\rho_2, \rho_4\}$, σ_5 is $LCSup\{\sigma_1, \sigma_3\}$ and σ_6 is $GCSub\{\sigma_2, \sigma_4\}$ hence the conclusion of (1).

(2) *Next we assume* $GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\}$, $LCSup\{\sigma_1, \sigma_3\}$ *and* $GCSub\{\sigma_2, \sigma_4\}$ *are defined and prove* $LCSup\{T_1, T_2\}$ *is defined*

Let T be $\begin{bmatrix}\rho_5 & \rho_6\\ \sigma_5 & \sigma_6\end{bmatrix}$ where $\rho_5 = GCSub\{\rho_1, \rho_3\}$, $\rho_6 = LCSup\{\rho_2, \rho_4\}$, $\sigma_5 = LCSup\{\sigma_1, \sigma_3\}$ and $\sigma_6 = GCSub\{\sigma_2, \sigma_4\}$.

We show that $LCSup\{T_1, T_2\}$ exists and is in fact T. By the definition of LCSup and GCSub and by using (ftype-lift)we derive

(s1)

$$\frac{GCSub\{\rho_1, \rho_3\} <: \rho_1 \quad \rho_2 <: LCSup\{\rho_2, \rho_4\} \quad \sigma_1 <: LCSup\{\sigma_1, \sigma_3\} \quad GCSub\{\sigma_1, \sigma_3\} <: \sigma_2}{\Delta \vdash \Delta <: T_1 T}$$

Similarly we derive

(s2)

$$\frac{GCSub\{\rho_1, \rho_3\} <: \rho_3 \quad \rho_4 <: LCSup\{\rho_2, \rho_4\} \quad \sigma_3 <: LCSup\{\sigma_1, \sigma_3\} \quad GCSub\{\sigma_1, \sigma_3\} <: \sigma_4}{\Delta \vdash \Delta <: T_2 T}$$

So T holds the first part of LCSup definition, $T_1 <: T$ and $T_2 <: T$. In order to prove T is $LCSup\{T_1, T_2\}$ we still need to show that for some flow-type T' such that $T_1 <: T'$ and $T_2 <: T'$ it holds that $T <: T'$.

Assume T' has the following form $\begin{bmatrix}\rho_{5'} & \rho_{6'}\\ \sigma_{5'} & \sigma_{6'}\end{bmatrix}$.

Since $T_1 <: T'$, by (ftype-lift)we have

(s3) $\rho'_5 <: \rho_1$, $\rho_2 <: \rho'_6$, $\sigma_1 <: \sigma'_5$ and $\sigma'_6 <: \sigma_2$

Similarly, since $T_2 <: T'$ we have

(s4) $\rho'_5 <: \rho_3$, $\rho_4 <: \rho'_6$, $\sigma_3 <: \sigma'_5$ and $\sigma'_6 <: \sigma_4$

From (s3), (s4) and the definition of LCSup and BCSub, we have

(s5) $\rho'_5 <: GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\} <: \rho'_6$, $LCSup\{\sigma_1, \sigma_3\} <: \sigma'_5$ and $\sigma'_6 <: GCSub\{\sigma_2, \sigma_4\}$

From (s5) and (ftype-lift)we derive

(s6) $\Delta \vdash \Delta <: TT'$

We conclude that $T = LCSup\{T_1, T_2\}$.

(ii) As we saw in part *(i)* of the lemma, since $T = LCSup\{T_1, T_2\}$ we have the following results:

1. $GCSub\{\rho_1, \rho_3\}$, $LCSup\{\rho_2, \rho_4\}$, $LCSup\{\sigma_1, \sigma_3\}$ and $GCSub\{\sigma_2, \sigma_4\}$ are defined.
2. $LCSup\left\{\begin{bmatrix}\rho_1 & \rho_2\\ \sigma_1 & \sigma_2\end{bmatrix}, \begin{bmatrix}\rho_3 & \rho_4\\ \sigma_3 & \sigma_4\end{bmatrix}\right\} = \begin{bmatrix}GCSub\{\rho_1, \rho_3\} & LCSup\{\rho_2, \rho_4\}\\ LCSup\{\sigma_1, \sigma_3\} & GCSub\{\sigma_2, \sigma_4\}\end{bmatrix}$

□

A.2 Lemma 3.6

Lemma. *Let* Γ *be a context,* Δ *a set of subtyping assumptions,* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *global-flow specifications,* $\mathcal{B}$ *a global-flow specification,* T, T' *flow types, and* $T_{A_1}, \ldots, T_{A_n}$ *flow types.*
Hypothesis:

(a1) $\Gamma, \Delta \vdash_A \mathcal{A}_i : T_{A_i}$ *for every* $i \in \{1, \ldots, n\}$

(a2) $T = LCSup\{T_{A_1}, \ldots, T_{A_n}\}$

(a3) $\Gamma \cup \{x : T\}, \Delta \vdash_A \mathcal{B} : T'$

Conclusion:

(c1) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B} : T'_i$ *for every* $i \in \{1, \ldots, n\}$, *for some* $T'_1, \ldots, T'_n$

(c2) $T' = LCSup\{T'_1, \ldots, T'_n\}$

Note: if x does not occur free in $\mathcal{B}$ the result is immediate.

Proof. by induction on the structure of $\mathcal{B}$.

- **case** $\mathcal{B} = B$
  Immediate.
- **case** $\mathcal{B} = y$
  for some variable y. If $y \neq x$ then the result is immediate. If $y = x$, then by (a3) and by (var) rule we get $T' = T$. We get (c1) by (var) rule and by assigning $T'_i = T_{A_i}$ for every $i \in \{1, \ldots, n\}$. We have (c2) by, $T' = T = LCSup\{T'_{A_1}, \ldots, T'_{A_n}\} = LCSup\{T'_1, \ldots, T'_n\}$
- **case** $\mathcal{B} = \mathcal{B}_1; \mathcal{B}_2$.

  Assume $T' = \begin{bmatrix} \rho_1 & \rho'_2 \\ \sigma_1 & \sigma'_2 \end{bmatrix}$.

  According to the assumptions (a1), (a2), (a3), and (seq) rule, we must have

  (s1) $\Gamma \cup \{x : T\}, \Delta \vdash_A \mathcal{B}_1 : T_1$ with $T_1 = \begin{bmatrix} \rho_1 & \rho'_1 \\ \sigma_1 & \sigma'_1 \end{bmatrix}$

  (s2) $\Gamma \cup \{x : T\}, \Delta \vdash_A \mathcal{B}_2 : T_2$ with $T_2 = \begin{bmatrix} \rho_2 & \rho'_2 \\ \sigma_2 & \sigma'_2 \end{bmatrix}$

  With

  (s3) $\Delta \vdash_B \rho'_1 <: \rho_2$

  (s4) $\Delta \vdash_B \sigma_2 <: \sigma'_1$

  The IH and (a1),(a2),(s1) imply

  (s5) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T'_{1,i}$ for every $i \in \{1, \ldots, n\}$

  (s6) $T_1 = LCSup\{T'_{1,1}, \ldots, T'_{1,n}\}$

  The IH and (a1),(a2),(s2) imply

  (s7) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T'_{2,i}$ for every $i \in \{1, \ldots, n\}$

  (s8) $T_2 = LCSup\{T'_{2,1}, \ldots, T'_{2,n}\}$

  From (s5) and (s6) we derive

  (s9) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T_1$ for every $i \in \{1, \ldots, n\}$

  From (s7) and (s8) we derive

  (s10) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T_2$ for every $i \in \{1, \ldots, n\}$

  We can now make the following derivations, according to (s9),(s10) and (seq) rule,
  for every $i \in \{1, \ldots, n\}$

$$\frac{\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T_1 \quad \Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T_2 \quad (s3) \quad (s4)}{\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B} : T'}$$

  So we have (c1). Since $T' = LCSup\{T', \ldots, T'\}$ is trivial, we have (c2) as well.

- **case** $\mathcal{B} = \mathcal{B}_1 \parallel \mathcal{B}_2$
  According to the assumptions (a1), (a2), (a3), and (par) rule, we must have,

  (s1) $\Gamma \cup \{x : T'\}, \Delta \vdash_A \mathcal{B}_1 : T_1$
  (s2) $\Gamma \cup \{x : T'\}, \Delta \vdash_A \mathcal{B}_2 : T_2$

  With $T' = T_1 \bullet T_2$.

  The IH and (a1),(a2),(s1) imply

  (s3) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T'_{1,i}$ for every $i \in \{1, \ldots, n\}$
  (s4) $T_1 = LCSup\{T'_{1,1}, \ldots, T'_{1,n}\}$

  The IH and (a1),(a2),(s2) imply

  (s5) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T'_{2,i}$ for every $i \in \{1, \ldots, n\}$
  (s6) $T_2 = LCSup\{T'_{2,1}, \ldots, T'_{2,n}\}$

  From (s3) and (s4) we derive

  (s7) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T_1$ for every $i \in \{1, \ldots, n\}$

  From (s5) and (s6) we derive

  (s8) $\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T_2$ for every $i \in \{1, \ldots, n\}$

  We can now make the following derivations, according to (s7),(s8) and (par) rule,
  for every $i \in \{1, \ldots, n\}$

$$\frac{\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_1 : T_1 \quad \Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B}_2 : T_2}{\Gamma \cup \{x : T_{A_i}\}, \Delta \vdash_B \mathcal{B} : T'}$$

  So we have (c1). Since $T' = LCSup\{T', \ldots, T'\}$ is trivial, we have (c2) as well.

- **case** $\mathcal{B} = \ let\ y \in \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}\ in\ \mathcal{B}_0$.
  Assume $y \neq x$. this is easy to do since if $y = x$ then we could just rename every free occurrence of x in $\mathcal{B}_0$ to a different variable. Moreover, assume that y does not occur free in $\mathcal{B}_1, \ldots, \mathcal{B}_m$, and in $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Again, this is easy to achieve with renaming.

  According to the assumptions and $(\mathsf{let\text{-}choice\text{-}A})$ rule, we must have

  (s1) $\Gamma \cup \{x : T\}, \Delta, \Delta \vdash_A \mathcal{B}_j : T''_j$ for every $j \in \{1, \ldots, m\}$
  (s2) $T'' = LCSup\{T''_1, \ldots, T''_m\}$
  (s3) $\Gamma \cup \{x : T, y : T''\}, \Delta, \Delta \vdash_A \mathcal{B}_0 : T'$

  For every $j \in \{1, \ldots, m\}$, by IH and (a1),(a2),(s1), we derive (s4.j),(s5.j) as followed:

  (s4.j) $\Gamma \cup \{x : T_{A_i}\}, \Delta, \Delta \vdash_B \mathcal{B}_j : T_{j,i}$ for every $i \in \{1, \ldots, n\}$
  (s5.j) $T''_j = LCSup\{T_{j,1}, \ldots, T_{j,n}\}$

  By IH and (s1),(s2),(s3) we derive

  (s6) $\Gamma \cup \{x : T, y : T''_j\}, \Delta, \Delta \vdash_B \mathcal{B}_0 : T'''_j$ for every $j \in \{1, \ldots, m\}$
  (s7) $T' = LCSup\{T'''_1, \ldots, T'''_m\}$

  For every $j \in \{1, \ldots, m\}$, from (a1) and the weakening lemma we derive (s8.j) as followed:

(s8.j) $\Gamma \cup \{y : T''_j\}, \Delta, \Delta \vdash_A \mathcal{A}_i : T_{A_i}$ for every $i \in \{1, \ldots, n\}$

For every $j \in \{1, \ldots, m\}$, by IH and (s8.j),(a2),(s6), we derive (s9.j),(s10.j) as followed:

(s9.j) $\Gamma \cup \{x : T_{A_i}, y : T''_j\}, \Delta, \Delta \vdash_B \mathcal{B}_0 : T'_{j,i}$ for every $i \in \{1, \ldots, n\}$

(s10.j) $T'''_j = LCSup\{T'_{j,1}, \ldots, T'_{j,n}\}$

For every $j \in \{1, \ldots, m\}$, by (s9.j),(s10.j) we derive

(s11.j) $\Gamma \cup \{x : T_{A_i}, y : T''_j\}, \Delta, \Delta \vdash_B \mathcal{B}_0 : T'''_j$ for every $i \in \{1, \ldots, n\}$

For every $j \in \{1, \ldots, m\}$, by (s4.j),(s5.j) we derive

(s12.j) $\Gamma \cup \{x : T_{A_i}\}, \Delta, \Delta \vdash_B \mathcal{B}_j : T''_j$ for every $i \in \{1, \ldots, n\}$

By (let-choice-B) we can make the following derivation

For every $i \in \{1, \ldots, n\}$

$$\frac{(s12.1), \ldots, (s12.m) \quad (s11.j), \ldots, (s11.m) \quad (s7)}{\Gamma \cup \{x : T_{A_i}\}, \Delta, \Delta \vdash_B let\ y \in \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}\ in\ \mathcal{B}_0 : T'}$$

We have (c1) by assigning $T'_i = T'$ for every $i \in \{1, \ldots, n\}$. We have (c2) by the trivial fact $T' = LCSup\{T', \ldots, T'\}$

□

A.3 Lemma 4.2

Lemma. *Let $\mathcal{A} \in$ Multi, $\mathscr{T}$ be a type in* **Exact**, Γ *be a context, and Δ be a set of subtyping assumptions. If the judgment $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ can be derived in* **Exact** *then $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}]\!] : \mathscr{T}$ holds.*

Proof. By induction on the structure of $\mathcal{A}$.

1. $\mathcal{A} = x$.
   In this case we know $\mathscr{T} = \Gamma(x)$. Since $[\![x]\!] = \{x\}$ it means $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![x]\!] : \Gamma(x)$, which is exactly the same as $\mathscr{T}$.

2. $\mathcal{A} = A$.
   Very similar to case 1.

3. $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$.
   By applying IH we get
   $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}_1]\!] : \mathscr{T}_1$, where $\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1$
   We do the same with $\mathcal{A}_2$ to get
   $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}_2]\!] : \mathscr{T}_2$
   That means that
   $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}_1]\!] \,\|\, [\![\mathcal{A}_2]\!] : (\mathsf{S}_i(\mathscr{T}_1)\mathsf{S}_j(\mathscr{T}_2))^{1 \le i \le |\mathscr{T}_1|,\, 1 \le j \le |\mathscr{T}_2|}$
   Which is the same type as $\mathscr{T}$.

4. $\mathcal{A} = \mathcal{A}_1; \mathcal{A}_2$.
   By applying IH we get
   $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}_1]\!] : \mathscr{T}_1$, where $\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1$ (1)
   We do the same with $\mathcal{A}_2$ to get
   $\Gamma, \Delta \vdash\!\!\rightarrowtail [\![\mathcal{A}_2]\!] : \mathscr{T}_2$ (2)

By the assumption we have that the following constraints hold

$$\text{for every } 1 \le i \le |\mathscr{T}_1|,\ j \le 1 \le |\mathscr{T}_2|, \tag{3}$$

$$\Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) <: \mathsf{f\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) \text{ and } \Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) <: \mathsf{b\text{-}in}(\mathsf{s}_i(\mathscr{T}_1))$$

By (1), (2) and (3) we have

$$\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}_1]\!] ; [\![\mathcal{A}_2]\!] : \left(\begin{bmatrix} \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) \\ \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) \end{bmatrix} \right)^{1 \le i \le |\mathscr{T}_1|,\, 1 \le j \le |\mathscr{T}_2|}$$

Which is the same type as $\mathscr{T}$.

5. $\mathcal{A} = let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B}$.

   By assumption we have

   $$\Gamma, \Delta \vdash \mathcal{A}_i : \mathscr{T}_i \text{ for every } 1 \le i \le n \text{ and} \tag{1}$$

   $$\Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash \mathcal{B} : \mathscr{T}'_{i,j} \text{ for every } 1 \le i \le n,\ j \le 1 \le |\mathscr{T}_i| \tag{2}$$

   By IH and (1) we have

   $\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}_i]\!] : \mathscr{T}_i$ for every $1 \le i \le n$

   By IH and (2) we have

   $\Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash\!\!\rightarrow [\![\mathcal{B}]\!] : \mathscr{T}'_{i,j}$ for every $1 \le i \le n,\ j \le 1 \le |\mathscr{T}_i|$

   That means we have

   $\Gamma, \Delta \vdash\!\!\rightarrow \mathbf{let}^*\ \ x \in [\![\mathcal{A}_1]\!] \cup \cdots \cup [\![\mathcal{A}_n]\!]\ \ \mathbf{in}\ \ [\![\mathcal{B}]\!] : (\mathscr{T}'_{i,j})^{1 \le i \le n,\, 1 \le j \le |\mathscr{T}_i|}$

   By that type is exactly the same as $\mathscr{T}$.

□

A.4 Lemma 4.3

Lemma. *Let $\mathcal{A} \in$ Multi, $\mathscr{T}$ be a type in* **Exact**, *Γ be a context, and Δ be a set of subtyping assumptions. If $\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}]\!] : \mathscr{T}$ holds then the judgment $\Gamma, \Delta \vdash \mathcal{A} : \mathscr{T}$ can be derived in* **Exact**.

Proof.

1. $\mathcal{A} = x$.
   In this case we know $\mathscr{T} = \Gamma(x)$ Since $[\![x]\!] = \{x\}$. It means $\Gamma, \Delta \vdash x : \mathscr{T}$ in **Exact** as well.

2. $\mathcal{A} = A$.
   Very similar to case 1.

3. $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$.
   By the definition of $[\![\ ]\!]$ we have $\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}_1]\!] \,\|\, [\![\mathcal{A}_2]\!] : \mathscr{T}$

   Assume

   $$\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}_1]\!] : \mathscr{T}_1 \text{ and } \Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}_2]\!] : \mathscr{T}_2 \tag{1}$$

   By IH and (1) we have

   $$\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1 \text{ and } \Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}_2 \tag{2}$$

   By the definition of $[\![\ ]\!]$ and (1) we have $\Gamma, \Delta \vdash\!\!\rightarrow [\![\mathcal{A}]\!] : (\mathsf{s}_i(\mathscr{T}_1)\mathsf{s}_j(\mathscr{T}_2))^{1 \le i \le |\mathscr{T}_1|, j \le 1 \le |\mathscr{T}_2|}$

   and by (2) we have $\Gamma, \Delta \vdash \mathcal{A} : (\mathsf{s}_i(\mathscr{T}_1)\mathsf{s}_j(\mathscr{T}_2))^{1 \le i \le |\mathscr{T}_1|, j \le 1 \le |\mathscr{T}_2|}$

4. $\mathcal{A} = \mathcal{A}_1;\mathcal{A}_2$.
We have $\Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}_1;\mathcal{A}_2]\!] : \mathscr{T}$ or by definition $\Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}_1]\!];[\![\mathcal{A}_2]\!] : \mathscr{T}$
Assume
$\Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}_1]\!] : \mathscr{T}_1$ and $\Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}_2]\!] : \mathscr{T}_2$ (1)
By IH and (1) we have
$\Gamma, \Delta \vdash \mathcal{A}_1 : \mathscr{T}_1$ and $\Gamma, \Delta \vdash \mathcal{A}_2 : \mathscr{T}_2$ (2)
By (1) and the definition of $[\![\ ]\!]$ we have
$$\Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}]\!] : \left(\begin{bmatrix} \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) \\ \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) \end{bmatrix} \right)^{1 \le i \le |\mathscr{T}_1|,\, 1 \le j \le |\mathscr{T}_2|}$$
Further more, it means the following constraints hold
for every $1 \le i \le |\mathscr{T}_1|,\ j \le 1 \le |\mathscr{T}_2|$,
$\Delta \vdash \mathsf{f\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) <: \mathsf{f\text{-}in}(\mathsf{s}_j(\mathscr{T}_2))$ and $\Delta \vdash \mathsf{b\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) <: \mathsf{b\text{-}in}(\mathsf{s}_i(\mathscr{T}_1))$
Using that and (2) we can say
$$\Gamma, \Delta \vdash \mathcal{A} : \left(\begin{bmatrix} \mathsf{f\text{-}in}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{f\text{-}out}(\mathsf{s}_j(\mathscr{T}_2)) \\ \mathsf{b\text{-}out}(\mathsf{s}_i(\mathscr{T}_1)) & \mathsf{b\text{-}in}(\mathsf{s}_j(\mathscr{T}_2)) \end{bmatrix} \right)^{1 \le i \le |\mathscr{T}_1|,\, 1 \le j \le |\mathscr{T}_2|}$$
5. $\mathcal{A} = let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B}$.

By the definition of $[\![\ ]\!]$ we have $\Gamma, \Delta \vdash\!\rightarrowtail \mathbf{let}^*\ x \in [\![\mathcal{A}_1]\!] \cup \cdots \cup [\![\mathcal{A}_n]\!]\ \ \mathbf{in}\ \ [\![\mathcal{B}]\!] : \mathscr{T}$.
Assume
for every $1 \le i \le n\ \ \Gamma, \Delta \vdash\!\rightarrowtail [\![\mathcal{A}_i]\!] : \mathscr{T}_i$. (1)
Further assume
for every $1 \le i \le n,\ j \le 1 \le |\mathscr{T}_i| \qquad \Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash\!\rightarrowtail [\![\mathcal{B}]\!] : \mathscr{T}'_{i,j}$. (2)
By the definition of $[\![\ ]\!]$ we have $\mathscr{T} = (\mathscr{T}'_{i,j})^{1 \le i \le n,\, 1 \le j \le |\mathscr{T}_i|}$.
By applying IH on (1) and then on (2) we have
for every $1 \le i \le n\ \ \Gamma, \Delta \vdash \mathcal{A}_i : \mathscr{T}_i$ and
for every $1 \le i \le n,\ j \le 1 \le |\mathscr{T}_i| \qquad \Gamma \cup \{x : \mathsf{s}_j(\mathscr{T}_i)\}, \Delta \vdash \mathcal{B} : \mathscr{T}'_{i,j}$.
That means
$\Gamma, \Delta \vdash \mathcal{A} = let\ x \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}\ in\ \mathcal{B} : \mathscr{T}$.

□

A.5 Lemma 4.5

Lemma. *Let T_1, T_2, T_3 and T_4 be flow types and Δ a set of subtyping assumptions. If $\Delta \vdash T_1 <: T_3$ and $\Delta \vdash T_2 <: T_4$ then $\Delta \vdash T_1 \bullet T_2 <: T_3 \bullet T_4$.*

Proof. Let $T_1 = \begin{bmatrix} \rho_1 & \rho'_1 \\ \sigma_1 & \sigma'_1 \end{bmatrix}$, $T_2 = \begin{bmatrix} \rho_2 & \rho'_2 \\ \sigma_2 & \sigma'_2 \end{bmatrix}$, $T_3 = \begin{bmatrix} \rho_3 & \rho'_3 \\ \sigma_3 & \sigma'_3 \end{bmatrix}$ and $T_4 = \begin{bmatrix} \rho_4 & \rho'_4 \\ \sigma_4 & \sigma'_4 \end{bmatrix}$. Since $\Delta \vdash T_1 <: T_3$ and $\Delta \vdash T_2 <: T_4$, using (ftype-lift) we have

$\Delta \vdash \rho_3 <: \rho_1, \Delta \vdash \rho'_1 <: \rho'_3, \Delta \vdash \sigma_1 <: \sigma_3$ and $\Delta \vdash \sigma'_3 <: \sigma'_1$ and (1)

$\Delta \vdash \rho_4 <: \rho_2, \Delta \vdash \rho'_2 <: \rho'_4, \Delta \vdash \sigma_2 <: \sigma_4$ and $\Delta \vdash \sigma'_4 <: \sigma'_2$ (2)

Using (fwtype-lift), (bwtype-lift) and (1) and (2) we get

$$\Delta \vdash \begin{bmatrix} \rho_1 \cdot \rho_2 & \rho'_1 \cdot \rho'_2 \\ \sigma_1 \cdot \sigma_2 & \sigma'_1 \cdot \sigma'_2 \end{bmatrix} <: \begin{bmatrix} \rho_3 \cdot \rho_4 & \rho'_3 \cdot \rho'_4 \\ \sigma_3 \cdot \sigma_4 & \sigma'_3 \cdot \sigma'_4 \end{bmatrix}$$

which is the same as $\Delta \vdash T_1 \bullet T_2 <: T_3 \bullet T_4$.

□

References

[1] Azer Bestavros, Adam Bradley, Assaf Kfoury, and Ibrahim Matta. Safe Compositional Specification of Networking Systems. *ACM SIGCOMM Computer Communication Review (CCR)*, 34(3), July 2004.

[2] Likai Liu, Assaf Kfoury, Azer Bestavros, Adam Bradley, Yarom Gabay, and Ibrahim Matta. Safe Compositional Specification of Networking Systems: TRAFFIC The Language and Its Type Checking. Technical Report BUCS-TR-2005-015, CS Department, Boston University, May 12 2005.

[3] Likai Liu, Assaf Kfoury, Azer Bestavros, Yarom Gabay, Adam Bradley, and Ibrahim Matta. Safe Compositional Specification of Networking Systems: A Compositional Analysis Approach. Technical Report BUCS-TR-2005-033, CS Department, Boston University, Dec 28 2005.