Polymorphic sessions and sequential composition of types

Diogo Poças   Diana Costa   Andreia Mordido   Vasco Vasconcelos

University of Lisbon

PLACES'23

[PLACES '22] Higher-order Context-free Session types in System F

[PLACES '22] Higher-order Context-free Session types in System F

RECAP ON CFSTs

- CFSTs are **not** restricted to tail recursion

[PLACES '22] Higher-order Context-free Session types in System F

RECAP ON CFSTs

- CFSTs are not restricted to tail recursion
- ; sequential composition operator

[PLACES '22] Higher-order Context-free Session types in System F

RECAP ON CFSTs

- CFSTs are not restricted to tail recursion
- ; sequential composition operator
- skip corresponding neutral element

[PLACES '22] Higher-order Context-free Session types in System F

RECAP ON CFSTs

- CFSTs are not restricted to tail recursion
- ; sequential composition operator
- skip corresponding neutral element
- CFSTs are not characterized by regular languages

[PLACES '22] Higher-order Context-free Session types in System F

RECAP ON CFSTs

- CFSTs are <u>not</u> restricted to tail recursion
- ; sequential composition operator
- skip corresponding neutral element
- CFSTs are <u>not</u> characterized by regular languages
- Type equivalence : decidable (via translation to simple grammars)

[PLACES '22] Higher-order Context-free Session types in System F

[PLACES '22] Higher-order Context-free Session types in System F

- Promoted CFSTs to the higher-order setting

[PLACES '22] Higher-order Context-free Session types in System F

- Promoted CFSTs to the higher-order setting

  !(?int)

[PLACES '22] Higher-order Context-free Session types in System F

- Promoted CFSTs to the higher-order setting

$!(?\mathrm{int})$

$$\mathrm{InputTree} \doteq \oplus\{\mathrm{Node} : \mathrm{InputTree} \,;\, !(?\mathrm{int}) \,;\, \mathrm{InputTree},\ \mathrm{Leaf} : \mathrm{skip}\}$$

recursion incorporated using equations

[PLACES '22] Higher-order Context-free Session types in System F

- Added polymorphism, using De Bruijn indices to refer to polymorphic variables

[PLACES '22] Higher-order Context-free Session types in System F

- Added polymorphism, using De Bruijn indices to refer to polymorphic variables

$$\forall^{s}\, 0 ; !int$$

[PLACES '22] Higher-order Context-free Session types in System F

- Added polymorphism, using De Bruijn indices to refer to polymorphic variables

$$\forall^{s}\, 0 ;\, !\mathrm{int}$$

bound by the first enclosing $\forall$

[PLACES '22] Higher-order Context-free Session types in System F

- Added polymorphism, using De Bruijn indices to refer to polymorphic variables

$\forall^{S}\, 0 ; !\mathrm{int}$ $\qquad\qquad$ $\forall^{T}\, 0 \to !\mathrm{int}$

bound by the first enclosing $\forall$

[PLACES '22] Higher-order Context-free Session types in System F

- Added polymorphism, using De Bruijn indices to refer to polymorphic variables

$\forall^{S}\, 0 ;\, !\mathrm{int}$

bound by the first enclosing $\forall$

$\forall^{T}\, 0 \to\, !\mathrm{int}$

functional

$K ::= T \mid S$

Kinds

session

[PLACES '22] Higher-order Context-free Session types in System F

- But .. we only allowed functional polymorphism !

[PLACES '22] Higher-order Context-free Session types in System F

- We provided: syntactic (rule based) and semantic (bisimulation based) definitions of equivalence;

  + a type equivalence algorithm by reduction to the bisimilarity of simple grammars.

[PLACES'22] → [PLACES'23]

GOAL: Full polymorphism

$\forall \ldots\; T \longrightarrow$ of kind s

GOAL: Full polymorphism

$\forall \ldots \; T \;\longrightarrow$ of kind s

$\downarrow$

NEW DESIGN CHOICES

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$X = 0 \to \forall^{s} X$$

equation expansion

$$0 \to \forall^{s} 0 \to \forall^{s} X$$

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$T \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \rightarrow \forall \beta_2 . \alpha_1 \rightarrow \forall \beta_3 . \alpha_1 \rightarrow \cdots$$

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$T \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \beta_2 . \alpha_1 \to \forall \beta_3 . \alpha_1 \to \dots$$

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$T \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \beta_2 . \alpha_1 \to \forall \beta_3 . \alpha_1 \to \cdots$$

$$X \doteq \forall \alpha . Y$$

$$Y \doteq \forall \beta \; \alpha \to Y$$

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$T \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \beta_2 . \alpha_1 \to \forall \beta_3 . \alpha_1 \to \dots$$

representation using De Bruijn indices

$$\forall \; \forall \; 1 \to \forall \; 2 \to \forall \; 3 \to \dots$$

WHAT WE LEARNED

- Using De Bruijn indices restricts expressivity

$$T \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \beta_2 . \alpha_1 \to \forall \beta_3 . \alpha_1 \to \cdots$$

$$\forall \quad \forall \quad 1 \to \forall \quad 2 \to \forall \quad 3 \to \cdots$$

↳ NO FINITE REPRESENTATION

WHAT WE LEARNED

- Equations are more expressive than μ-types

$$U \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \alpha_2 . \beta_1 \to \forall \beta_2 . \alpha_2 \to \forall \alpha_3 . \beta_2 \to \cdots$$

WHAT WE LEARNED

- Equations are more expressive than μ-types

$$U \triangleq \forall \alpha_1 . \forall \beta_1 . \alpha_1 \to \forall \alpha_2 . \beta_1 \to \forall \beta_2 . \alpha_2 \to \forall \alpha_3 . \beta_2 \to \cdots$$

WHAT WE LEARNED

- Equations are more expressive than μ-types

$$U \triangleq \forall\alpha_1.\forall\beta_1.\alpha_1 \to \forall\alpha_2.\beta_1 \to \forall\beta_2.\alpha_2 \to \forall\alpha_3.\beta_2 \to \cdots$$

$\hookrightarrow$ $\forall\alpha.X$

$$X \doteq \forall\beta.\alpha \to \forall\alpha.\beta \to X \quad (*)$$

WHAT WE LEARNED

- Equations are more expressive than μ-types

$$U \triangleq \forall\alpha_1.\forall\beta_1.\alpha_1 \to \forall\alpha_2.\beta_1 \to \forall\beta_2.\alpha_2 \to \forall\alpha_3.\beta_2 \to \cdots$$

↳ $\forall\alpha.X$

$$X \doteq \forall\beta.\alpha \to \forall\alpha.\beta \to X \quad (*)$$

↳ CANNOT BE WRITTEN USING CONVENTIONAL μ-TYPES

WHAT WE DID WITH WHAT WE LEARNED

- We kept using equations

WHAT WE DID WITH WHAT WE LEARNED

- We kept using equations
- We dropped the use of De Bruijn indices and worked with variable names

WHAT WE DID WITH WHAT WE LEARNED

- We kept using equations
- we dropped the use of De Bruijn indices and worked with variable names
- we still do not rename variables when expanding equations — we do not adopt the variable convention

WHAT WE DID WITH WHAT WE LEARNED

- We kept using equations
- we dropped the use of De Bruijn indices and worked with variable names
- we still do not rename variables when expanding equations — we do not adopt the variable convention

$$U \triangleq \forall\alpha_1 . \forall\beta_1 . \alpha_1 \to \forall\alpha_2 . \beta_1 \to \forall\beta_2 . \alpha_2 \to \forall\alpha_3 . \beta_2 \to \cdots$$

$\hookrightarrow \forall\alpha . X$

$$X \doteq \forall\beta . \alpha \to \forall\alpha . \beta \to X \xrightarrow[\text{renaming}]{} X \doteq \forall\beta . \alpha \to \forall\alpha' . \beta \to X$$

≠ after expansion

BACK TO OUR GOAL

Full polymorphism

$$\text{HeteroTree} \doteq \&\{\ \text{Leaf} : \text{skip},\quad \text{Node} : \text{HeteroTree} ; \underbrace{(\forall \alpha : \mathsf{T}.\ \underbrace{?\alpha}_{\text{kind } s})}_{\text{kind } s} ; \text{HeteroTree}\ \}$$

TYPE EQUIVALENCE WITH DE BRUIJN INDICES

$$\frac{T \simeq U}{\forall^{K}.\,T \simeq \forall^{K}.\,U}$$

TYPE EQUIVALENCE WITH DE BRUIJN INDICES

$$\frac{T \simeq U}{\forall^{K}.T \simeq \forall^{K}.U}$$

TYPE EQUIVALENCE WITH VARIABLE NAMES

$$\frac{T \simeq U[\alpha/\beta]}{\forall \alpha : K.T \simeq \forall \beta : K.U}$$

TYPE EQUIVALENCE WITH DE BRUIJN INDICES

$$\frac{T \simeq U}{\forall^{K}.T \simeq \forall^{K}.U}$$

TYPE EQUIVALENCE WITH VARIABLE NAMES

$$\frac{T \simeq U[\alpha/\beta]}{\forall \alpha : K.T \simeq \forall \beta : K.U}$$

but
we do not
want to make
these substitutions
on the fly

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha])$$

fresh variable

α_i = first variable not free in $\forall \alpha : K . T$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : \kappa . T) = \forall \alpha_i : \kappa . N(T[\alpha_i / \alpha]),$$

$$\text{where } \alpha_i = \mathrm{first}(\forall \alpha : \kappa . T)$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha]) ,$$

$$\text{where } \alpha_i = \mathit{first}(\forall \alpha : K . T)$$

TYPE EQUIVALENCE

$$\frac{T \simeq U}{\forall \alpha : K . T \simeq \forall \alpha : K . U}$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha]),$$

where $\alpha_i = \text{first}(\forall \alpha : K . T)$

e.g.,

$$N(\forall \alpha : T . \forall \beta \cdot S . \beta)$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha]),$$

where $\alpha_i = \text{first}(\forall \alpha : K . T)$

e.g.,

$$N(\forall \alpha : T . \forall \beta : S . \beta) = \forall \gamma : T . N(\forall \beta : S . \beta [\gamma / \alpha])$$, where $\gamma < \cdots$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall\alpha : K . T) = \forall\alpha_i : K . N(T[\alpha_i/\alpha]),$$

where $\alpha_i = first(\forall\alpha : K . T)$

e.g.,

$$N(\forall\alpha : T . \forall\beta \cdot s . \beta) = \forall\gamma \cdot T . N(\forall\beta : s . \beta[\gamma/\alpha])$$

$$= \forall\gamma : T . N(\forall\beta : s . \beta)$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha]),$$

where $\alpha_i = \mathrm{first}(\forall \alpha : K . T)$

e.g.,

$$N(\forall \alpha : T . \forall \beta \cdot S . \beta) = \forall \gamma \cdot T . N(\forall \beta : S . \beta [\gamma / \alpha])$$

$$= \forall \gamma : T . N(\forall \beta : S . \beta)$$

$$= \forall \gamma \cdot T . \forall \gamma : S . \gamma$$

bound here

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall\alpha:K.T) = \forall\alpha_i:K.\,N(T[\alpha_i/\alpha]),$$

$$\text{where } \alpha_i = \mathrm{first}(\forall\alpha:K.T)$$

e.g.,

$$N(\forall\alpha:T.\ \forall\beta\cdot S.\beta) = \forall\gamma\cdot T.\ N(\forall\beta:S.\beta\,[\gamma/\alpha])$$

$$= \forall\gamma:T.\ N(\forall\beta:S.\beta)$$

$$= \forall\gamma\cdot T.\ \forall\gamma:S.\gamma$$

bound here

The canonical renaming uses the least amount of variable names possible.

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall\alpha : K . T) = \forall\alpha_i : K . N(T[\alpha_i/\alpha]) \,,$$

$$\text{where } \alpha_i = \text{first}(\forall\alpha : K . T)$$

$$N(\forall\alpha : K . T \,;\, U) = \forall\alpha_i : K . (N(T[\alpha_i/\alpha]) \,;\, N(U))$$

$$\text{where } \alpha_i = \text{first}(\forall\alpha : K . T \,;\, U)$$

CANONICAL RENAMING

$$N(\alpha) = \alpha$$

$$N(\forall \alpha : K . T) = \forall \alpha_i : K . N(T[\alpha_i / \alpha]) ,$$

where $\alpha_i = \mathrm{first}(\forall \alpha : K . T)$

$$N(\forall \alpha : K . T \,;\, U) = \forall \alpha_i : K . (N(T[\alpha_i / \alpha]) \,;\, N(U))$$

where $\alpha_i = \mathrm{first}(\forall \alpha : K . T \,;\, U)$

TYPE EQUIVALENCE

$$\frac{T \cong U}{\forall \alpha : K . T \cong \forall \alpha : K . U}$$

→ no need to introduce other rules

CANONICAL RENAMING

$$N(\forall\alpha : \kappa . T \; ; U) = \forall\alpha_i : \kappa . (N(T[\alpha_i/\alpha]) \; ; N(U))$$

$$\text{where } \alpha_i = \mathrm{first}(\forall\alpha : \kappa . T \; ; U)$$

e.g.,

$$N((\forall\alpha : s . \alpha) \; ; \alpha)$$

CANONICAL RENAMING

$$N(\forall\alpha : \kappa . T \,;\, U) = \forall\alpha_i : \kappa . (N(T[\alpha_i/\alpha]) \,;\, N(U))$$

where $\alpha_i = \mathrm{first}(\forall\alpha : \kappa . T \,;\, U)$

e.g.,

$$N((\forall\alpha : s . \alpha) \,;\, \alpha) = \forall\gamma : s . (N(\alpha[\gamma/\alpha]) \,;\, N(\alpha))$$

CANONICAL RENAMING

$$N(\forall\alpha : \kappa . T \; ; U) = \forall\alpha_i : \kappa . (N(T[\alpha_i/\alpha]) \; ; N(U))$$

where $\alpha_i = \mathrm{first}(\forall\alpha : \kappa . T \; ; U)$

e.g.,

$$N((\forall\alpha : s . \alpha) ; \alpha) = \forall\gamma : s . (N(\alpha[\gamma/\alpha]) \; ; N(\alpha))$$

$$= \forall\gamma . \; s . (N(\gamma) \; ; N(\alpha))$$

CANONICAL RENAMING

$$N(\forall\alpha : K . T \,;\, U) = \forall\alpha_i : K . (N(T[\alpha_i/\alpha]) \,;\, N(U))$$

where $\alpha_i = \text{first}(\forall\alpha : K . T \,;\, U)$

e.g.,

$$N((\forall\alpha : s . \alpha) ; \alpha) = \forall\gamma : s . (N(\alpha[\gamma/\alpha]) ; N(\alpha))$$

$$= \forall\gamma : s . (N(\gamma) ; N(\alpha))$$

$$= \forall\gamma : s . (\gamma ; \alpha)$$

CANONICAL RENAMING

$$N(\forall\alpha:\kappa.T\ ;U) = \forall\alpha_i:\kappa.(N(T[\alpha_i/\alpha])\ ;N(U))$$

where $\alpha_i = \mathrm{first}(\forall\alpha:\kappa.T\ ;U)$

e.g.,

$$N(\forall\alpha:s.\alpha\ ;\forall\beta:T\ \,'\beta)$$

CANONICAL RENAMING

$$N(\forall\alpha:K.T\,;U) = \forall\alpha_i:K.(N(T[\alpha_i/\alpha]);N(U))$$

where $\alpha_i = \text{first}(\forall\alpha:K.T\,;U)$

e.g.,

$$N(\forall\alpha:s.\alpha\,;\forall\beta:T\ !\beta) = \forall\gamma:s.\left(N(\alpha[\gamma/\alpha]);N(\forall\beta:T.!\beta)\right)$$

CANONICAL RENAMING

$$N(\forall\alpha:\kappa.T\ ;U) = \forall\alpha_i:\kappa.(N(T[\alpha_i/\alpha]);N(U))$$

where $\alpha_i = \text{first}(\forall\alpha:\kappa.T\ ;U)$

e.g.,

$$N(\forall\alpha:s.\alpha;\forall\beta:T\ !\beta) = \forall\gamma:s.(N(\alpha[\gamma/\alpha]);N(\forall\beta:T.!\beta))$$

$$= \forall\gamma.\ s.(\gamma;\forall\gamma:T.N(!\beta[\gamma/\beta]))$$

CANONICAL RENAMING

$$N(\forall\alpha : K . T \; ; U) = \forall\alpha_i : K . (N(T[\alpha_i/\alpha]) \; ; N(U))$$

where $\alpha_i = \mathit{first}(\forall\alpha : K . T \; ; U)$

e.g.,

$$N(\forall\alpha : S . \alpha \; ; \forall\beta : T \; !\beta) = \forall\gamma : S . (N(\alpha[\gamma/\alpha]) \; ; N(\forall\beta : T . !\beta))$$

$$= \forall\gamma . S . (\gamma \; ; \forall\gamma : T . N(!\beta[\gamma/\beta]))$$

$$= \forall\gamma : S . (\gamma \; ; \forall\gamma : T . N(!\gamma))$$

CANONICAL RENAMING

$$N(\forall\alpha:\kappa.T\ ;U) = \forall\alpha_i:\kappa.(N(T[\alpha_i/\alpha])\ ;N(U))$$

where $\alpha_i = \mathrm{first}(\forall\alpha:\kappa.T\ ;U)$

e.g.,

$$N(\forall\alpha:s.\alpha\ ;\forall\beta:T\ !\beta) = \forall\gamma:s.(N(\alpha[\gamma/\alpha])\ ;N(\forall\beta:T.!\beta))$$

$$= \forall\gamma:s.(\gamma\ ;\forall\gamma:T.N(!\beta[\gamma/\beta]))$$

$$= \forall\gamma:s.(\gamma\ ;\forall\gamma:T.N(!\gamma))$$

$$= \forall\gamma:s.(\gamma\ ;\forall\gamma:T.!\gamma)$$

MERCI !