Mathematical and Computational Linguistics for Proofs

Structural Rules and Algebraic Properties of Intersection Types

Sandra Alves
University of Porto
(joint work with Mário Florido)

September 16, 2025

A long time ago...

Non-idempotent Intersections and Linear Logic Seminal work by Kfoury (2001), which was latter highlighted by de Carvalho (2007).

Intersection types and Simple types Bucciarelli, Piperno and Salvo (1999): Translation of intersection typing derivations into Curry typeable terms, preserving β-reduction.

Intersection types and Linear terms Damas and Florido (2004): Expansion relation between terms typable with intersection types and linear terms.

This started my long lasting interest in resource aware type systems...

A long time ago...

Non-idempotent Intersections and Linear Logic Seminal work by Kfoury (2001), which was latter highlighted by de Carvalho (2007).

Intersection types and Simple types Bucciarelli, Piperno and Salvo (1999): Translation of intersection typing derivations into Curry typeable terms, preserving β-reduction.

Intersection types and Linear terms Damas and Florido (2004): Expansion relation between terms typable with intersection types and linear terms.

This started my long lasting interest in resource aware type systems...

Structural Rules and Algebraic Properties (of Intersection Types)

Substructural Rules: in type/logic systems, these correspond to weakening (**W**), exchange (**E**), and contraction (**C**) rules:

	W	**E**	**C**	**Use**
Normal	✓	✓	✓	unrestricted
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				exactly once in order

Structural Rules and Algebraic Properties (of Intersection Types)

Substructural Rules: in type/logic systems, these correspond to weakening (**W**), exchange (**E**), and contraction (**C**) rules:

	W	**E**	**C**	**Use**
Normal	✓	✓	✓	unrestricted
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				exactly once in order

Structural Rules and Algebraic Properties (of Intersection Types)

Algebraic Properties: in intersection type systems the intersection operator $\cap$ can be:

- associative (**A**)
- commutative (**C**)
- and idempotent (**I**)

Our language

The untyped λ-calculus:

$$\begin{array}{rcll} x \in \mathcal{V} & \Rightarrow & x \in \Lambda & \\ M, N \in \Lambda & \Rightarrow & (MN) \in \Lambda & \text{(Application)} \\ M \in \Lambda,\ x \in \mathcal{V} & \Rightarrow & (\lambda x M) \in \Lambda & \text{(Abstraction)} \end{array}$$

The usual notion of β-reduction:

$$\beta : (\lambda x.M)N \to M[N/x]$$

Our language

The untyped λ-calculus:

$$\begin{array}{rcll} x \in \mathcal{V} & \Rightarrow & x \in \Lambda & \\ M, N \in \Lambda & \Rightarrow & (MN) \in \Lambda & \text{(Application)} \\ M \in \Lambda,\, x \in \mathcal{V} & \Rightarrow & (\lambda x M) \in \Lambda & \text{(Abstraction)} \end{array}$$

The usual notion of β-reduction:

$$\beta : (\lambda x.M)N \to M[N/x]$$

Simple types:

$$
\begin{array}{rcl}
\alpha, \beta \in \mathbb{V} & \Rightarrow & \alpha, \beta \in \mathbb{T}_C \\
\sigma, \tau \in \mathbb{T}_C & \Rightarrow & (\tau \to \sigma) \in \mathbb{T}_C
\end{array}
$$

A typing environment Γ is a finite list of pairs $x : \tau$ where **all** term variables x are distinct.

A typing:

$$\Gamma \vdash M : \sigma$$

means that M has type σ assuming the type declarations in Γ.

Simple types:

$$\begin{array}{rcl} \alpha, \beta \in \mathbb{V} & \Rightarrow & \alpha, \beta \in \mathbb{T}_C \\ \sigma, \tau \in \mathbb{T}_C & \Rightarrow & (\tau \to \sigma) \in \mathbb{T}_C \end{array}$$

A typing environment Γ is a finite list of pairs $x : \tau$ where **all** term variables x are distinct.

A typing:

$$\Gamma \vdash M : \sigma$$

means that M has type σ assuming the type declarations in Γ.

Simple types:

$$\begin{array}{rcl} \alpha, \beta \in \mathbb{V} & \Rightarrow & \alpha, \beta \in \mathbb{T}_C \\ \sigma, \tau \in \mathbb{T}_C & \Rightarrow & (\tau \to \sigma) \in \mathbb{T}_C \end{array}$$

A typing environment Γ is a finite list of pairs $x : \tau$ where **all** term variables x are distinct.

A typing:

$$\Gamma \vdash M : \sigma$$

means that M has type σ assuming the type declarations in Γ.

The Simple Type System (Logical Rules)

$$\frac{}{x : \tau \vdash_S x : \tau}\,(\text{Axiom})$$

$$\frac{\Gamma, x : \tau \vdash_S M : \sigma}{\Gamma \vdash_S \lambda x.M : \tau \to \sigma}\,(\to \text{Intro})$$

$$\frac{\Gamma_1 \vdash_S M : \tau \to \sigma \qquad \Gamma_2 \vdash_S N : \tau}{\Gamma_1, \Gamma_2 \vdash_S MN : \sigma}\,(\to \text{Elim})$$

The Simple Type System (Logical Rules)

$$\frac{}{\underbrace{x : \tau}_{\text{a single assumption}} \quad \vdash_S x : \tau}\ (\text{Axiom})$$

$$\frac{\overbrace{\Gamma, x : \tau}^{\text{there is an assumption}} \quad \vdash_S M : \sigma}{\Gamma \vdash_S \lambda x.M : \tau \to \sigma}\ (\to \text{Intro})$$

$$\frac{\Gamma_1 \vdash_S M : \tau \to \sigma \qquad \Gamma_2 \vdash_S N : \tau}{\underbrace{\Gamma_1, \Gamma_2}_{\text{list concatenation}} \quad \vdash_S MN : \sigma}\ (\to \text{Elim})$$

The Simple Type System (Logical Rules)

$$\frac{}{\underbrace{x : \tau}_{\text{a single assumption}} \vdash_S x : \tau}\ (\text{Axiom})$$

$$\frac{\overbrace{\Gamma, x : \tau}^{\text{there is an assumption}} \vdash_S M : \sigma}{\Gamma \vdash_S \lambda x.M : \tau \to \sigma}\ (\to \text{Intro})$$

$$\frac{\Gamma_1 \vdash_S M : \tau \to \sigma \qquad \Gamma_2 \vdash_S N : \tau}{\underbrace{\Gamma_1, \Gamma_2}_{\text{list concatenation}} \vdash_S MN : \sigma}\ (\to \text{Elim})$$

The Simple Type System (Logical Rules)

$$\frac{}{\underbrace{x : \tau}_{\text{a single assumption}} \quad \vdash_S x : \tau}\ (\text{Axiom})$$

$$\frac{\overbrace{\Gamma, x : \tau}^{\text{there is an assumption}} \quad \vdash_S M : \sigma}{\Gamma \vdash_S \lambda x.M : \tau \to \sigma}\ (\to \text{Intro})$$

$$\frac{\Gamma_1 \vdash_S M : \tau \to \sigma \qquad \Gamma_2 \vdash_S N : \tau}{\underbrace{\Gamma_1, \Gamma_2}_{\text{list concatenation}} \quad \vdash_S MN : \sigma}\ (\to \text{Elim})$$

The Simple Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M : \sigma} \text{(Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_S M : \sigma} \text{(Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M[x/x_1, x/x_2] : \sigma} \text{(Contraction)}$$

The Simple Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M : \sigma} \text{ (Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_S M : \sigma} \text{ (Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M[x/x_1, x/x_2] : \sigma} \text{ (Contraction)}$$

The Simple Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M : \sigma} \text{(Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_S M : \sigma} \text{(Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_S M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_S M[x/x_1, x/x_2] : \sigma} \text{(Contraction)}$$

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha} \qquad \dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}$$

The Simple Type System - Weakening

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{x:\alpha\to\alpha \vdash_S x:\alpha\to\alpha}{x:\alpha\to\alpha, \mathbf{y}:\beta \vdash_S x:\alpha\to\alpha}}{x:\alpha\to\alpha \vdash_S \lambda y.x:\beta\to\alpha\to\alpha}}{\vdash_S \lambda xy.x:(\alpha\to\alpha)\to\beta\to\alpha\to\alpha} \qquad \dfrac{x:\alpha\vdash_S x:\alpha}{\vdash_S \lambda x.x:\alpha\to\alpha}}{\vdash_S (\lambda xy.x)(\lambda x.x):\beta\to\alpha\to\alpha}$$

The Simple Type System - Weakening

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha} \qquad \dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}$$

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}
}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}
}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha}
\qquad
\dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}
}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}
$$

The Simple Type System - Weakening

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}
}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}
}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha}
\qquad
\dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}
}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}
$$

The Simple Type System - Weakening

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$
\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha} \qquad \dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}
$$

The Simple Type System - Weakening

For the λ-term $(\lambda xy.x)(\lambda x.x)$ the following derivation is obtained:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{x : \alpha \to \alpha \vdash_S x : \alpha \to \alpha}{x : \alpha \to \alpha, \mathbf{y} : \beta \vdash_S x : \alpha \to \alpha}
}{x : \alpha \to \alpha \vdash_S \lambda y.x : \beta \to \alpha \to \alpha}
}{\vdash_S \lambda xy.x : (\alpha \to \alpha) \to \beta \to \alpha \to \alpha}
\qquad
\dfrac{x : \alpha \vdash_S x : \alpha}{\vdash_S \lambda x.x : \alpha \to \alpha}
}{\vdash_S (\lambda xy.x)(\lambda x.x) : \beta \to \alpha \to \alpha}
$$

The Simple Type System - Exchange

For the λ-term $\lambda xy.yx$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{y:\alpha\to\beta \vdash_S y:\alpha\to\beta \qquad x:\alpha \vdash_S x:\alpha}{y:\alpha\to\beta, x:\alpha \vdash_S yx:\beta}}{x:\alpha, y:\alpha\to\beta \vdash_S yx:\beta}}{x:\alpha \vdash_S \lambda y.yx:(\alpha\to\beta)\to\beta}}{\vdash_S (\lambda xy.yx):\alpha\to(\alpha\to\beta)\to\beta}$$

The Simple Type System - Exchange

For the λ-term $\lambda xy.yx$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{y : \alpha \to \beta \vdash_S y : \alpha \to \beta \qquad x : \alpha \vdash_S x : \alpha}{y : \alpha \to \beta, x : \alpha \vdash_S yx : \beta}}{x : \alpha, y : \alpha \to \beta \vdash_S yx : \beta}}{x : \alpha \vdash_S \lambda y.yx : (\alpha \to \beta) \to \beta}}{\vdash_S (\lambda xy.yx) : \alpha \to (\alpha \to \beta) \to \beta}$$

For the λ-term $\lambda xy.yx$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{y : \alpha \to \beta \vdash_S y : \alpha \to \beta \qquad x : \alpha \vdash_S x : \alpha}{y : \alpha \to \beta, x : \alpha \vdash_S yx : \beta}}{x : \alpha, y : \alpha \to \beta \vdash_S yx : \beta}}{x : \alpha \vdash_S \lambda y.yx : (\alpha \to \beta) \to \beta}}{\vdash_S (\lambda xy.yx) : \alpha \to (\alpha \to \beta) \to \beta}$$

The Simple Type System - Exchange

For the λ-term $\lambda xy.yx$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{y : \alpha \to \beta \vdash_S y : \alpha \to \beta \qquad x : \alpha \vdash_S x : \alpha}{y : \alpha \to \beta, x : \alpha \vdash_S yx : \beta}}{x : \alpha, y : \alpha \to \beta \vdash_S yx : \beta}}{x : \alpha \vdash_S \lambda y.yx : (\alpha \to \beta) \to \beta}}{\vdash_S (\lambda xy.yx) : \alpha \to (\alpha \to \beta) \to \beta}$$

The Simple Type System - Exchange

For the λ-term $\lambda xy.yx$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{\dfrac{y:\alpha\to\beta\vdash_S y:\alpha\to\beta \qquad x:\alpha\vdash_S x:\alpha}{y:\alpha\to\beta, x:\alpha\vdash_S yx:\beta}}{x:\alpha, y:\alpha\to\beta\vdash_S yx:\beta}}{x:\alpha\vdash_S \lambda y.yx:(\alpha\to\beta)\to\beta}}{\vdash_S (\lambda xy.yx):\alpha\to(\alpha\to\beta)\to\beta}$$

For the λ-term $\lambda fx.f(fx)$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{f_1 : \alpha \to \alpha \vdash_S f_1 : \alpha \to \alpha \qquad \dfrac{f_2 : \alpha \to \alpha \vdash_S f_2 : \alpha \to \alpha \qquad x : \alpha \vdash_S x : \alpha}{f_2 : \alpha \to \alpha, x : \alpha \vdash_S (f_2 x) : \alpha}}{\mathbf{f_1} : \alpha \to \alpha, \mathbf{f_2} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f_1}(\mathbf{f_2} x) : \alpha}}{\dfrac{\mathbf{f} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f}(\mathbf{f} x) : \alpha}{f : \alpha \to \alpha \vdash_S \lambda x.f(fx) : \alpha \to \alpha}}}{\vdash_S \lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha}$$

The Simple Type System - Contraction

For the λ-term $\lambda fx.f(fx)$ the following derivation is obtained:

$$\dfrac{f_1 : \alpha \to \alpha \vdash_S f_1 : \alpha \to \alpha \qquad \dfrac{f_2 : \alpha \to \alpha \vdash_S f_2 : \alpha \to \alpha \qquad x : \alpha \vdash_S x : \alpha}{f_2 : \alpha \to \alpha, x : \alpha \vdash_S (f_2 x) : \alpha}}{\dfrac{\mathbf{f_1} : \alpha \to \alpha, \mathbf{f_2} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f_1}(\mathbf{f_2}x) : \alpha}{\dfrac{\mathbf{f} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f}(\mathbf{f}x) : \alpha}{\dfrac{f : \alpha \to \alpha \vdash_S \lambda x.f(fx) : \alpha \to \alpha}{\vdash_S \lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha}}}}$$

For the λ-term $\lambda fx.f(fx)$ the following derivation is obtained:

$$\dfrac{\dfrac{\dfrac{f_1 : \alpha \to \alpha \vdash_S f_1 : \alpha \to \alpha \qquad \dfrac{f_2 : \alpha \to \alpha \vdash_S f_2 : \alpha \to \alpha \qquad x : \alpha \vdash_S x : \alpha}{f_2 : \alpha \to \alpha, x : \alpha \vdash_S (f_2 x) : \alpha}}{\color{blue}{\mathbf{f_1} : \alpha \to \alpha, \mathbf{f_2} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f_1}(\mathbf{f_2} x) : \alpha}}}{\color{blue}{\mathbf{f} : \alpha \to \alpha, x : \alpha \vdash_S \mathbf{f}(\mathbf{f} x) : \alpha}}}{\dfrac{f : \alpha \to \alpha \vdash_S \lambda x.f(fx) : \alpha \to \alpha}{\vdash_S \lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha}}$$

From Simple Types to Substructural Types

Simple Types are not expressive enough to reason about restricted use of computational resources.

What happens when we remove one (or more) structural rule(s)?

Substructural Type Systems are related to **Substructural Logics**

- Linear logic: the basis of resource aware formalisms.
- Lambek ordered logic: applications to natural language processing.
- Relevant logic.

Simple Types are not expressive enough to reason about restricted use of computational resources.

What happens when we remove one (or more) structural rule(s)?

Substructural Type Systems are related to **Substructural Logics**

- Linear logic: the basis of resource aware formalisms.
- Lambek ordered logic: applications to natural language processing.
- Relevant logic.

Simple Types are not expressive enough to reason about restricted use of computational resources.

What happens when we remove one (or more) structural rule(s)?

Substructural Type Systems are related to **Substructural Logics**

- Linear logic: the basis of resource aware formalisms.
- Lambek ordered logic: applications to natural language processing.
- Relevant logic.

Type System	**W**	**E**	**C**	**Use of assumptions**
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				in order

The Relevant Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_R M : \sigma} \text{ (Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_R M[x/x_1, x/x_2] : \sigma} \text{ (Contraction)}$$

No weakening implies that any typed term is a λI-term (in every $\lambda x.N$ in M, x occurs free in N at least once).

For example, $\lambda y.x$ is not typable in the *Relevant Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are typable.

The Relevant Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_R M : \sigma} \text{(Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_R M[x/x_1, x/x_2] : \sigma} \text{(Contraction)}$$

No weakening implies that any typed term is a λI-term (in every $\lambda x.N$ in M, x occurs free in N at least once).

For example, $\lambda y.x$ is not typable in the *Relevant Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are typable.

The Relevant Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_R M : \sigma} \text{(Exchange)}$$

$$\frac{\Gamma_1, x_1 : \tau, x_2 : \tau, \Gamma_2 \vdash_R M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_R M[x/x_1, x/x_2] : \sigma} \text{(Contraction)}$$

No weakening implies that any typed term is a λI-term (in every $\lambda x.N$ in M, x occurs free in N at least once).

For example, $\lambda y.x$ is not typable in the *Relevant Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are typable.

Substructural Type Systems

Type System	W	E	C	Use of assumptions
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				in order

The Affine Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_A M : \sigma} \text{(Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_A M : \sigma} \text{(Exchange)}$$

No contraction, means that each variable cannot occur more than once.

For example, $\lambda x.x$ and $\lambda x.y$ are typable in the *Affine Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are not typable.

The Affine Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_A M : \sigma} \text{(Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_A M : \sigma} \text{(Exchange)}$$

No contraction, means that each variable cannot occur more than once.

For example, $\lambda x.x$ and $\lambda x.y$ are typable in the *Affine Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are not typable.

The Affine Type System (Structural Rules)

$$\frac{\Gamma_1, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, x : \tau, \Gamma_2 \vdash_A M : \sigma} \text{(Weakening)}$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_A M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_A M : \sigma} \text{(Exchange)}$$

No contraction, means that each variable cannot occur more than once.

For example, $\lambda x.x$ and $\lambda x.y$ are typable in the *Affine Type System*, whereas $\lambda xyz.xz(yz)$ and $\lambda fx.f(fx)$ are not typable.

Substructural Type Systems

Type System	W	E	C	Use of assumptions
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				in order

The Linear Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_L M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_L M : \sigma} \text{ (Exchange)}$$

No weakening and no contraction means that:

- for each subterm $\lambda x.N$ of M, x occurs free in N exactly once;
- each free variable of M has just one occurrence free in M.

For example $\lambda x.x$ and $\lambda xy.xy$ are typable in the *Linear Type System*, whereas $\lambda x.y$ and $\lambda fx.f(fx)$ are not.

The *Linear Type System* enjoys both **Subject Reduction** and **Subject Expansion**.

The Linear Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_L M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_L M : \sigma} \text{(Exchange)}$$

No weakening and no contraction means that:

- for each subterm $\lambda x.N$ of M, x occurs free in N exactly once;
- each free variable of M has just one occurrence free in M.

For example $\lambda x.x$ and $\lambda xy.xy$ are typable in the *Linear Type System*, whereas $\lambda x.y$ and $\lambda fx.f(fx)$ are not.

The *Linear Type System* enjoys both **Subject Reduction** and **Subject Expansion**.

The Linear Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_L M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_L M : \sigma} \text{(Exchange)}$$

No weakening and no contraction means that:

- for each subterm $\lambda x.N$ of M, x occurs free in N exactly once;
- each free variable of M has just one occurrence free in M.

For example $\lambda x.x$ and $\lambda xy.xy$ are typable in the *Linear Type System*, whereas $\lambda x.y$ and $\lambda fx.f(fx)$ are not.

The *Linear Type System* enjoys both **Subject Reduction** and **Subject Expansion**.

The Linear Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_L M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_L M : \sigma} \text{(Exchange)}$$

No weakening and no contraction means that:

- for each subterm $\lambda x.N$ of M, x occurs free in N exactly once;
- each free variable of M has just one occurrence free in M.

For example $\lambda x.x$ and $\lambda xy.xy$ are typable in the *Linear Type System*, whereas $\lambda x.y$ and $\lambda fx.f(fx)$ are not.

The *Linear Type System* enjoys both **Subject Reduction** and **Subject Expansion**.

The Linear Type System (Structural Rules)

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \vdash_L M : \sigma}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \vdash_L M : \sigma} \text{ (Exchange)}$$

No weakening and no contraction means that:

- for each subterm $\lambda x.N$ of M, x occurs free in N exactly once;
- each free variable of M has just one occurrence free in M.

For example $\lambda x.x$ and $\lambda xy.xy$ are typable in the *Linear Type System*, whereas $\lambda x.y$ and $\lambda fx.f(fx)$ are not.

The *Linear Type System* enjoys both **Subject Reduction** and **Subject Expansion**.

Type System	**W**	**E**	**C**	**Use of assumptions**
Relevant		✓	✓	at least once
Affine	✓	✓		at most once
Linear		✓		exactly once
Ordered				exactly once in order

The Ordered Type System (Logical Rules)

$$\frac{}{x : \tau \vdash x : \tau}\ (\text{Axiom})$$

$$\frac{x : \tau_1, \Gamma \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_l \tau_2}\ (\rightarrow_{\mathrm{l}}\ \text{Intro}) \qquad \frac{\Gamma, x : \tau_1 \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_r \tau_2}\ (\rightarrow_{\mathrm{r}}\ \text{Intro})$$

$$\frac{\Gamma_2 \vdash_O N : \tau \qquad \Gamma_1 \vdash_O M : \tau \rightarrow_l \sigma}{\Gamma_2, \Gamma_1 \vdash_O MN : \sigma}\ (\rightarrow_{\mathrm{l}}\ \text{Elim})$$

$$\frac{\Gamma_1 \vdash_O M : \tau \rightarrow_r \sigma \qquad \Gamma_2 \vdash_O N : \tau}{\Gamma_1, \Gamma_2 \vdash_O MN : \sigma}\ (\rightarrow_{\mathrm{r}}\ \text{Elim})$$

The Ordered Type System (Logical Rules)

$$\frac{}{x : \tau \vdash x : \tau}\ (\text{Axiom})$$

$$\frac{x : \tau_1, \Gamma \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_l \tau_2}\ (\rightarrow_{\mathrm{l}}\ \text{Intro}) \qquad \frac{\Gamma, x : \tau_1 \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_r \tau_2}\ (\rightarrow_{\mathrm{r}}\ \text{Intro})$$

$$\frac{\Gamma_2 \vdash_O N : \tau \qquad \Gamma_1 \vdash_O M : \tau \rightarrow_l \sigma}{\Gamma_2, \Gamma_1 \vdash_O MN : \sigma}\ (\rightarrow_{\mathrm{l}}\ \text{Elim})$$

$$\frac{\Gamma_1 \vdash_O M : \tau \rightarrow_r \sigma \qquad \Gamma_2 \vdash_O N : \tau}{\Gamma_1, \Gamma_2 \vdash_O MN : \sigma}\ (\rightarrow_{\mathrm{r}}\ \text{Elim})$$

The Ordered Type System (Logical Rules)

$$\frac{}{x : \tau \vdash x : \tau}\ (\text{Axiom})$$

$$\frac{x : \tau_1, \Gamma \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_l \tau_2}\ (\rightarrow_\mathrm{l}\ \text{Intro}) \qquad \frac{\Gamma, x : \tau_1 \vdash_O M : \tau_2}{\Gamma \vdash_O \lambda x.M : \tau_1 \rightarrow_r \tau_2}\ (\rightarrow_\mathrm{r}\ \text{Intro})$$

$$\frac{\Gamma_2 \vdash_O N : \tau \qquad \Gamma_1 \vdash_O M : \tau \rightarrow_l \sigma}{\Gamma_2, \Gamma_1 \vdash_O MN : \sigma}\ (\rightarrow_\mathrm{l}\ \text{Elim})$$

$$\frac{\Gamma_1 \vdash_O M : \tau \rightarrow_r \sigma \qquad \Gamma_2 \vdash_O N : \tau}{\Gamma_1, \Gamma_2 \vdash_O MN : \sigma}\ (\rightarrow_\mathrm{r}\ \text{Elim})$$

The Ordered Type System- Properties

No contraction (it is a linear system) and no weakening (it is a relevant system)

Plus, no exchange: the order of the assumptions matter!

Is $(\lambda x.xz_2)z_1$ typable in the *Ordered Type System*? Yes!

In fact, we have two (valid) typings:

$$z_1 : \alpha \rightarrow_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

But the following typings are not valid:

$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_r \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_1 : \alpha \rightarrow_l \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$

The Ordered Type System- Properties

No contraction (it is a linear system) and no weakening (it is a relevant system)

Plus, no exchange: the order of the assumptions matter!

Is $(\lambda x.xz_2)z_1$ typable in the *Ordered Type System*? Yes!

In fact, we have two (valid) typings:

$z_1 : \alpha \to_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$

$z_2 : \alpha,\ z_1 : \alpha \to_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$

But the following typings are not valid:

$z_2 : \alpha,\ z_1 : \alpha \to_r \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$

$z_1 : \alpha \to_l \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$

The Ordered Type System- Properties

No contraction (it is a linear system) and no weakening (it is a relevant system)

Plus, no exchange: the order of the assumptions matter!

Is $(\lambda x.xz_2)z_1$ typable in the *Ordered Type System*? Yes!

In fact, we have two (valid) typings:

$$z_1 : \alpha \rightarrow_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$

$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

But the following typings are not valid:

$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_r \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

$$z_1 : \alpha \rightarrow_l \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$

The Ordered Type System- Properties

No contraction (it is a linear system) and no weakening (it is a relevant system)

Plus, no exchange: the order of the assumptions matter!

Is $(\lambda x.xz_2)z_1$ typable in the *Ordered Type System*? Yes!

In fact, we have two (valid) typings:

$$z_1 : \alpha \rightarrow_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

But the following typings are not valid:

$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_r \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_1 : \alpha \rightarrow_l \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$

Now let's slightly detour and talk about Intersection Types

Intersection Types System (ITS)

Intersection types [Barendregt, Coppo and Dezani, 1983] give us a characterization of the strongly normalizable λ-terms:

$$\Gamma \vdash_{\cap} M : \sigma \iff M \text{ is strongly normalizable}$$

A term is strongly normalizing if every reduction sequence ends with an irreducible term (a normal form).

Note that, in the Simple Type System:

$$\Gamma \vdash M : \sigma \Rightarrow M \text{ is strongly normalizing}$$

... but the opposite does not hold: the strongly normalizable term $\lambda x.xx$ is not simply typable.

Intersection Types System (ITS)

Intersection types [Barendregt, Coppo and Dezani, 1983] give us a characterization of the strongly normalizable λ-terms:

$$\Gamma \vdash_{\cap} M : \sigma \iff M \text{ is strongly normalizable}$$

A term is strongly normalizing if every reduction sequence ends with an irreducible term (a normal form).

Note that, in the Simple Type System:

$$\Gamma \vdash M : \sigma \Rightarrow M \text{ is strongly normalizing}$$

... but the opposite does not hold: the strongly normalizable term $\lambda x.xx$ is not simply typable.

Intersection Types System (ITS)

Intersection types [Barendregt, Coppo and Dezani, 1983] give us a characterization of the strongly normalizable λ-terms:

$$\Gamma \vdash_{\cap} M : \sigma \iff M \text{ is strongly normalizable}$$

A term is strongly normalizing if every reduction sequence ends with an irreducible term (a normal form).

Note that, in the Simple Type System:

$$\Gamma \vdash M : \sigma \Rightarrow M \text{ is strongly normalizing}$$

... but the opposite does not hold: the strongly normalizable term $\lambda x.xx$ is not simply typable.

Intersection Types

$$\frac{}{x : \tau \vdash x : \tau} \text{(Axiom)}$$

$$\frac{\Gamma \cup \{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma} (\to \text{Intro}_\text{I})$$

$$\frac{\Gamma \vdash M : \sigma \qquad x \text{ does not occur in } \Gamma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} (\to \text{Intro}_\text{K})$$

$$\frac{\Gamma_0 \vdash M : \tau_1 \cap \cdots \cap \tau_n \to \sigma \qquad \Gamma_1 \vdash N : \tau_1 \ \cdots \ \Gamma_n \vdash N : \tau_n}{\Gamma_0 \wedge \Gamma_1 \wedge \cdots \wedge \Gamma_n \vdash MN : \sigma} (\to \text{Elim})$$

Intersection Types

$$\frac{}{x : \tau \vdash x : \tau} \text{(Axiom)}$$

$$\frac{\Gamma \cup \{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma} (\to \mathrm{Intro_I})$$

$$\frac{\Gamma \vdash M : \sigma \qquad x \text{ does not occur in } \Gamma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} (\to \mathrm{Intro_K})$$

$$\frac{\Gamma_0 \vdash M : \tau_1 \cap \cdots \cap \tau_n \to \sigma \qquad \Gamma_1 \vdash N : \tau_1 \ \cdots \ \Gamma_n \vdash N : \tau_n}{\Gamma_0 \wedge \Gamma_1 \wedge \cdots \wedge \Gamma_n \vdash MN : \sigma} (\to \mathrm{Elim})$$

Intersection Types

$$\frac{}{x : \tau \vdash x : \tau} \text{(Axiom)}$$

$$\frac{\Gamma \cup \{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma} (\to \mathsf{Intro_I})$$

$$\frac{\Gamma \vdash M : \sigma \qquad x \text{ does not occur in } \Gamma}{\Gamma \vdash \lambda x.M : \tau \to \sigma} (\to \mathsf{Intro_K})$$

$$\frac{\Gamma_0 \vdash M : \tau_1 \cap \cdots \cap \tau_n \to \sigma \qquad \Gamma_1 \vdash N : \tau_1 \ \cdots \ \Gamma_n \vdash N : \tau_n}{\Gamma_0 \wedge \Gamma_1 \wedge \cdots \wedge \Gamma_n \vdash MN : \sigma} (\to \text{Elim})$$

Intersection Types

$$\frac{}{x : \tau \vdash x : \tau}\,(\text{Axiom})$$

$$\frac{\Gamma \cup \{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma}\,(\to \text{Intro}_{\mathsf{I}})$$

$$\frac{\Gamma \vdash M : \sigma \qquad x \text{ does not occur in } \Gamma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}\,(\to \text{Intro}_{\mathsf{K}})$$

$$\frac{\Gamma_0 \vdash M : \tau_1 \cap \cdots \cap \tau_n \to \sigma \qquad \Gamma_1 \vdash N : \tau_1 \;\; \cdots \;\; \Gamma_n \vdash N : \tau_n}{\Gamma_0 \wedge \Gamma_1 \wedge \cdots \wedge \Gamma_n \vdash MN : \sigma}\,(\to \text{Elim})$$

Intersection Types-Example

The λ-term $(\lambda x.xx)$ is typable in the Intersection Type System:

$$\dfrac{\dfrac{x : \alpha \to \beta \vdash x : \alpha \to \beta \qquad x : \alpha \vdash x : \alpha}{x : (\alpha \to \beta) \cap \alpha \vdash xx : \beta}}{\vdash (\lambda x.xx) : ((\alpha \to \beta) \cap \alpha) \to \beta}$$

Intersection Types-Example

The λ-term $(\lambda x.xx)$ is typable in the Intersection Type System:

$$\dfrac{\dfrac{x : \alpha \to \beta \vdash x : \alpha \to \beta \qquad x : \alpha \vdash x : \alpha}{x : (\alpha \to \beta) \cap \alpha \vdash xx : \beta}}{\vdash (\lambda x.xx) : ((\alpha \to \beta) \cap \alpha) \to \beta}$$

Intersection Types-Example

The λ-term $(\lambda x.xx)$ is typable in the Intersection Type System:

$$\dfrac{\dfrac{x : \alpha \to \beta \vdash x : \alpha \to \beta \qquad x : \alpha \vdash x : \alpha}{x : (\alpha \to \beta) \cap \alpha \vdash xx : \beta}}{\vdash (\lambda x.xx) : ((\alpha \to \beta) \cap \alpha) \to \beta}$$

Intersection Types-Example

The λ-term $(\lambda x.xx)$ is typable in the Intersection Type System:

$$\dfrac{\dfrac{x : \alpha \to \beta \vdash x : \alpha \to \beta \qquad x : \alpha \vdash x : \alpha}{x : (\alpha \to \beta) \cap \alpha \vdash xx : \beta}}{\vdash (\lambda x.xx) : ((\alpha \to \beta) \cap \alpha) \to \beta}$$

$$
\begin{tikzcd}
 & \text{Intersection} \arrow[d, dash] & \\
 & \text{Simple} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Affine} \arrow[dr, dash] & & \text{Relevant} \arrow[dl, dash] \\
 & \text{Linear} \arrow[d, dash] & \\
 & \text{Ordered} &
\end{tikzcd}
$$

Algebraic properties of Intersection and Substructural Systems

Expansion based on Intersection types

Given the ITS typing:

$$\vdash_{\cap} (\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$$

Consider the non-linear term:

$$\vdash_{\cap} \lambda x.xx : (\underbrace{(\alpha \to \alpha) \to (\alpha \to \alpha)}_{\text{1st occ. of } x} \cap \underbrace{(\alpha \to \alpha)}_{\text{2nd occ. of } x}) \to \alpha \to \alpha$$

We expand this into:

$$\vdash_{L} \lambda x_1 x_2 . x_1 x_2 : \underbrace{((\alpha \to \alpha) \to (\alpha \to \alpha))}_{x_1} \to \underbrace{(\alpha \to \alpha)}_{x_2} \to \alpha \to \alpha$$

Obtaining the following typing in the *Linear System*:

$$\vdash_{L} (\lambda x_1 x_2 . x_1 x_2)(\lambda y.y)(\lambda y.y) : \alpha \to \alpha$$

Expansion based on Intersection types

Given the ITS typing:

$$\vdash_{\cap} (\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$$

Consider the non-linear term:

$$\vdash_{\cap} \lambda x.xx : \underbrace{(\alpha \to \alpha) \to (\alpha \to \alpha)}_{\textit{1st occ. of } x} \cap \underbrace{(\alpha \to \alpha)}_{\textit{2nd occ. of } x}\ ) \to \alpha \to \alpha$$

We expand this into:

$$\vdash_{L} \lambda x_1 x_2 . x_1 x_2 : \underbrace{((\alpha \to \alpha) \to (\alpha \to \alpha))}_{x_1} \to \underbrace{(\alpha \to \alpha)}_{x_2} \to \alpha \to \alpha$$

Obtaining the following typing in the *Linear System*:

$$\vdash_{L} (\lambda x_1 x_2 . x_1 x_2)(\lambda y.y)(\lambda y.y) : \alpha \to \alpha$$

Expansion based on Intersection types

Given the ITS typing:

$$\vdash_\cap (\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$$

Consider the non-linear term:

$$\vdash_\cap \lambda x.xx : (\underbrace{(\alpha \to \alpha) \to (\alpha \to \alpha)}_{\text{1st occ. of } x} \cap \underbrace{(\alpha \to \alpha)}_{\text{2nd occ. of } x}) \to \alpha \to \alpha$$

We expand this into:

$$\vdash_L \lambda x_1 x_2 . x_1 x_2 : \underbrace{((\alpha \to \alpha) \to (\alpha \to \alpha))}_{x_1} \to \underbrace{(\alpha \to \alpha)}_{x_2} \to \alpha \to \alpha$$

Obtaining the following typing in the *Linear System*:

$$\vdash_L (\lambda x_1 x_2 . x_1 x_2)(\lambda y.y)(\lambda y.y) : \alpha \to \alpha$$

Expansion based on Intersection types

Given the ITS typing:

$$\vdash_\cap (\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$$

Consider the non-linear term:

$$\vdash_\cap \lambda x.xx : \underbrace{(\alpha \to \alpha) \to (\alpha \to \alpha)}_{\textit{1st occ. of } x} \cap \underbrace{(\alpha \to \alpha)}_{\textit{2nd occ. of } x}\ ) \to \alpha \to \alpha$$

We expand this into:

$$\vdash_L \lambda x_1 x_2 . x_1 x_2 : \underbrace{((\alpha \to \alpha) \to (\alpha \to \alpha))}_{x_1} \to \underbrace{(\alpha \to \alpha)}_{x_2} \to \alpha \to \alpha$$

Obtaining the following typing in the *Linear System*:

$$\vdash_L (\lambda x_1 x_2 . x_1 x_2)(\lambda y.y)(\lambda y.y) : \alpha \to \alpha$$

ACI-Expansion

ACI - Associative, Commutative and Idempotent $(\tau \cap \tau = \tau)$

$$\begin{array}{rcl}
\mathcal{E}_I(x:\tau) & \lhd & (y, \{x : \{y : \tau\}\}) \\
& & \quad \text{if } x \neq y \\[1em]
\mathcal{E}_I(\lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma) & \lhd & (\lambda x_1 \ldots x_n.M^*, A) \\
& & \quad \text{if } x \text{ occurs in } M \text{ and} \\
& & \quad \mathcal{E}_I(M:\sigma) \lhd (M^*, A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\}) \\[1em]
\mathcal{E}_I(\lambda x.M : \tau \to \sigma) & \lhd & (\lambda y.M^*, A) \\
& & \quad \text{if } x \text{ does not occur in } M, \\
& & \quad y \text{ is a fresh variable and} \\
& & \quad \mathcal{E}_I(M:\sigma) \lhd (M^*, A) \\[1em]
\mathcal{E}_I(MN:\sigma) & \lhd & (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \cdots \uplus A_n) \\
& & \quad \text{if for some } k > 0 \text{ and } \tau_1, \ldots \tau_k, \\
& & \quad \mathcal{E}_I(M : \tau_1 \cap \cdots \cap \tau_k \to \sigma) \lhd (M_0, A_0) \text{ and} \\
& & \quad \mathcal{E}_I(N:\tau_i) \lhd (N_i, A_i), (1 \leq i \leq k)
\end{array}$$

ACI-Expansion

ACI - Associative, Commutative and Idempotent $(\tau \cap \tau = \tau)$

$$
\begin{array}{rcl}
\mathcal{E}_I(x : \tau) & \lhd & (y, \{x : \{y : \tau\}\}) \\
 & & \quad \text{if } x \neq y \\
 & & \\
\mathcal{E}_I(\lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma) & \lhd & (\lambda x_1 \ldots x_n.M^*, A) \\
 & & \quad \text{if } x \text{ occurs in } M \text{ and} \\
 & & \quad \mathcal{E}_I(M : \sigma) \lhd (M^*, A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\}) \\
 & & \\
\mathcal{E}_I(\lambda x.M : \tau \to \sigma) & \lhd & (\lambda y.M^*, A) \\
 & & \quad \text{if } x \text{ does not occur in } M, \\
 & & \quad y \text{ is a fresh variable and} \\
 & & \quad \mathcal{E}_I(M : \sigma) \lhd (M^*, A) \\
 & & \\
\mathcal{E}_I(MN : \sigma) & \lhd & (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \cdots \uplus A_n) \\
 & & \quad \text{if for some } k > 0 \text{ and } \tau_1, \ldots \tau_k, \\
 & & \quad \mathcal{E}_I(M : \tau_1 \cap \cdots \cap \tau_k \to \sigma) \lhd (M_0, A_0) \text{ and} \\
 & & \quad \mathcal{E}_I(N : \tau_i) \lhd (N_i, A_i), (1 \leq i \leq k)
\end{array}
$$

ACI-Expansion

ACI - Associative, Commutative and Idempotent $(\tau \cap \tau = \tau)$

$$\mathcal{E}_I(x : \tau) \quad \lhd \quad (y, \{x : \{y : \tau\}\}) \quad \text{if } x \neq y$$

$$\mathcal{E}_I(\lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma) \quad \lhd \quad (\lambda x_1 \ldots x_n.M^*, A)$$
if x occurs in M and
$\mathcal{E}_I(M : \sigma) \lhd (M^*, A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\})$

$$\mathcal{E}_I(\lambda x.M : \tau \to \sigma) \quad \lhd \quad (\lambda y.M^*, A)$$
if x does not occur in M,
y is a fresh variable and
$\mathcal{E}_I(M : \sigma) \lhd (M^*, A)$

$$\mathcal{E}_I(MN : \sigma) \quad \lhd \quad (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \cdots \uplus A_n)$$
if for some $k > 0$ and $\tau_1, \ldots \tau_k$,
$\mathcal{E}_I(M : \tau_1 \cap \cdots \cap \tau_k \to \sigma) \lhd (M_0, A_0)$ and
$\mathcal{E}_I(N : \tau_i) \lhd (N_i, A_i), (1 \leq i \leq k)$

ACI-Expansion

ACI - Associative, Commutative and Idempotent $(\tau \cap \tau = \tau)$

$$
\begin{array}{rcl}
\mathcal{E}_I(x : \tau) & \lhd & (y, \{x : \{y : \tau\}\}) \\
 & & \quad \text{if } x \neq y \\[1em]
\mathcal{E}_I(\lambda x.M : \tau_1 \cap \cdots \cap \tau_n \to \sigma) & \lhd & (\lambda x_1 \ldots x_n.M^*, A) \\
 & & \quad \text{if } x \text{ occurs in } M \text{ and} \\
 & & \quad \mathcal{E}_I(M : \sigma) \lhd (M^*, A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\}) \\[1em]
\mathcal{E}_I(\lambda x.M : \tau \to \sigma) & \lhd & (\lambda y.M^*, A) \\
 & & \quad \text{if } x \text{ does not occur in } M, \\
 & & \quad y \text{ is a fresh variable and} \\
 & & \quad \mathcal{E}_I(M : \sigma) \lhd (M^*, A) \\[1em]
\mathcal{E}_I(MN : \sigma) & \lhd & (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \cdots \uplus A_n) \\
 & & \quad \text{if for some } k > 0 \text{ and } \tau_1, \ldots \tau_k, \\
 & & \quad \mathcal{E}_I(M : \tau_1 \cap \cdots \cap \tau_k \to \sigma) \lhd (M_0, A_0) \text{ and} \\
 & & \quad \mathcal{E}_I(N : \tau_i) \lhd (N_i, A_i), (1 \leq i \leq k)
\end{array}
$$

ACI-Expansion - Example

Let us show step by step how to calculate an expansion of $(\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$

$$\mathcal{E}_I(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathcal{E}_I(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

thus

$$\mathcal{E}_I(xx : \alpha \to \alpha) \lhd (x_1x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd (\lambda x_1x_2.x_1x_2, \varnothing)$$

It easy to show that

$$\mathcal{E}_I(\lambda y.y : \alpha \to \alpha) \lhd (\lambda z.z, \varnothing)$$

and

$$\mathcal{E}_I(\lambda y.y : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (\lambda z.z, \varnothing)$$

thus

$$\mathcal{E}_I((\lambda x.xx)(\lambda y.y) : \alpha \to \alpha) \lhd ((\lambda x_1x_2.x_1x_2)(\lambda z.z)(\lambda z.z), \varnothing)$$

ACI-Expansion - Example

Let us show step by step how to calculate an expansion of $(\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$

$$\mathcal{E}_I(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathcal{E}_I(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

thus

$$\mathcal{E}_I(xx : \alpha \to \alpha) \lhd (x_1x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd (\lambda x_1x_2.x_1x_2, \varnothing)$$

It easy to show that

$$\mathcal{E}_I(\lambda y.y : \alpha \to \alpha) \lhd (\lambda z.z, \varnothing)$$

and

$$\mathcal{E}_I(\lambda y.y : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (\lambda z.z, \varnothing)$$

thus

$$\mathcal{E}_I((\lambda x.xx)(\lambda y.y) : \alpha \to \alpha) \lhd ((\lambda x_1x_2.x_1x_2)(\lambda z.z)(\lambda z.z), \varnothing)$$

ACI-Expansion - Example

Let us show step by step how to calculate an expansion of $(\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$

$$\mathcal{E}_I(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathcal{E}_I(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

thus

$$\mathcal{E}_I(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd (\lambda x_1 x_2.x_1 x_2, \varnothing)$$

It easy to show that

$$\mathcal{E}_I(\lambda y.y : \alpha \to \alpha) \lhd (\lambda z.z, \varnothing)$$

and

$$\mathcal{E}_I(\lambda y.y : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (\lambda z.z, \varnothing)$$

thus

$$\mathcal{E}_I((\lambda x.xx)(\lambda y.y) : \alpha \to \alpha) \lhd ((\lambda x_1 x_2.x_1 x_2)(\lambda z.z)(\lambda z.z), \varnothing)$$

ACI-Expansion - Example

Let us show step by step how to calculate an expansion of $(\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$

$$\mathcal{E}_I(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathcal{E}_I(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

thus

$$\mathcal{E}_I(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd (\lambda x_1 x_2.x_1 x_2, \varnothing)$$

It easy to show that

$$\mathcal{E}_I(\lambda y.y : \alpha \to \alpha) \lhd (\lambda z.z, \varnothing)$$

and

$$\mathcal{E}_I(\lambda y.y : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (\lambda z.z, \varnothing)$$

thus

$$\mathcal{E}_I((\lambda x.xx)(\lambda y.y) : \alpha \to \alpha) \lhd ((\lambda x_1 x_2.x_1 x_2)(\lambda z.z)(\lambda z.z), \varnothing)$$

ACI-Expansion - Example

Let us show step by step how to calculate an expansion of $(\lambda x.xx)(\lambda y.y) : \alpha \to \alpha$

$$\mathcal{E}_I(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathcal{E}_I(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

thus

$$\mathcal{E}_I(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd (\lambda x_1 x_2.x_1 x_2, \varnothing)$$

It easy to show that

$$\mathcal{E}_I(\lambda y.y : \alpha \to \alpha) \lhd (\lambda z.z, \varnothing)$$

and

$$\mathcal{E}_I(\lambda y.y : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (\lambda z.z, \varnothing)$$

thus

$$\mathcal{E}_I((\lambda x.xx)(\lambda y.y) : \alpha \to \alpha) \lhd ((\lambda x_1 x_2.x_1 x_2)(\lambda z.z)(\lambda z.z), \varnothing)$$

ACI-Expansion - Example

Let us now look at one expansion of $\lambda fx.f(fx)$: ´

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

and,

$$\mathcal{E}_I(x : \alpha) \lhd (x_1, \{x : \{x_1 : \alpha\}\})$$

thus,

$$\mathcal{E}_I((fx) : \alpha) \lhd (f_1 x_1, \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

we also have,

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

therefore,

$$\mathcal{E}_I(f(fx) : \alpha) \lhd (f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.f(fx) : \alpha \to \alpha) \lhd (\lambda x_1.f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}\})$$

finally we have,

$$\mathcal{E}_I(\lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha) \lhd \lambda f_1 x_1.f_1(f_1 x_1)$$

ACI-Expansion - Example

Let us now look at one expansion of $\lambda fx.f(fx)$: ´

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

and,

$$\mathcal{E}_I(x : \alpha) \lhd (x_1, \{x : \{x_1 : \alpha\}\})$$

thus,

$$\mathcal{E}_I((fx) : \alpha) \lhd (f_1 x_1, \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

we also have,

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

therefore,

$$\mathcal{E}_I(f(fx) : \alpha) \lhd (f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.f(fx) : \alpha \to \alpha) \lhd (\lambda x_1.f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}\})$$

finally we have,

$$\mathcal{E}_I(\lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha) \lhd \lambda f_1 x_1.f_1(f_1 x_1)$$

ACI-Expansion - Example

Let us now look at one expansion of $\lambda fx.f(fx)$: ´

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

and,

$$\mathcal{E}_I(x : \alpha) \lhd (x_1, \{x : \{x_1 : \alpha\}\})$$

thus,

$$\mathcal{E}_I((fx) : \alpha) \lhd (f_1 x_1, \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

we also have,

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

therefore,

$$\mathcal{E}_I(f(fx) : \alpha) \lhd (f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.f(fx) : \alpha \to \alpha) \lhd (\lambda x_1.f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}\})$$

finally we have,

$$\mathcal{E}_I(\lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha) \lhd \lambda f_1 x_1.f_1(f_1 x_1)$$

ACI-Expansion - Example

Let us now look at one expansion of $\lambda fx.f(fx)$: ´

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

and,

$$\mathcal{E}_I(x : \alpha) \lhd (x_1, \{x : \{x_1 : \alpha\}\})$$

thus,

$$\mathcal{E}_I((fx) : \alpha) \lhd (f_1 x_1, \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

we also have,

$$\mathcal{E}_I(f : \alpha \to \alpha) \lhd (f_1, \{f : \{f_1 : \alpha \to \alpha\}\})$$

therefore,

$$\mathcal{E}_I(f(fx) : \alpha) \lhd (f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}, x : \{x_1 : \alpha\}\})$$

and

$$\mathcal{E}_I(\lambda x.f(fx) : \alpha \to \alpha) \lhd (\lambda x_1.f_1(f_1 x_1), \{f : \{f_1 : \alpha \to \alpha\}\})$$

finally we have,

$$\mathcal{E}_I(\lambda fx.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha) \lhd \lambda f_1 x_1.f_1(f_1 x_1)$$

ACI-Expansion - Properties

We consider the following translation $\mathcal{T}$ from intersection types to simple types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to \cdots \to \mathcal{T}(\tau_n) \to \mathcal{T}(\sigma)$.

We have the following properties regarding ACI expansion:

$$\mathcal{E}_I(M : \sigma) \lhd (N, A)$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_S N : \mathcal{T}(\sigma)$.
- If M is a λI-term , then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_R N : \mathcal{T}(\sigma)$.

We consider the following translation $\mathcal{T}$ from intersection types to simple types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to \cdots \to \mathcal{T}(\tau_n) \to \mathcal{T}(\sigma)$.

We have the following properties regarding ACI expansion:

$$\mathcal{E}_I(M : \sigma) \lhd (N, A)$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_S N : \mathcal{T}(\sigma)$.
- If M is a λI-term , then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_R N : \mathcal{T}(\sigma)$.

ACI-Expansion - Properties

We consider the following translation $\mathcal{T}$ from intersection types to simple types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to \cdots \to \mathcal{T}(\tau_n) \to \mathcal{T}(\sigma)$.

We have the following properties regarding ACI expansion:

$$\mathcal{E}_I(M : \sigma) \lhd (N, A)$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_S N : \mathcal{T}(\sigma)$.
- If M is a λI-term , then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_R N : \mathcal{T}(\sigma)$.

AC-Expansion

AC - Associative, Commutative but not Idempotent ($\tau \cap \tau \neq \tau$)

$$\underbrace{\mathcal{E}_I(x : \tau)}_{\text{ACI}} \quad \lhd \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } x \neq y$$

$$\underbrace{\mathcal{E}_C(x : \tau)}_{\text{AC}} \quad \lhd \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } y \text{ is a fresh variable}$$

For example:

$$\mathcal{E}_C(\lambda x.x(xx) : ((\alpha \to \alpha) \cap (\alpha \to \alpha) \cap \alpha) \to \alpha)$$
$$\lhd(\lambda x_1 x_2 x_3.x_1(x_2 x_3), \{x : \{x_1 : \alpha \to \alpha, x_2 : \alpha \to \alpha, x_3 : \alpha\}\})$$

AC-Expansion

AC - Associative, Commutative but not Idempotent ($\tau \cap \tau \neq \tau$)

$$\underbrace{\mathcal{E}_I(x : \tau)}_{\text{ACI}} \quad \lhd \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } x \neq y$$

$$\underbrace{\mathcal{E}_C(x : \tau)}_{\text{AC}} \quad \lhd \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } y \text{ is a fresh variable}$$

For example:

$$\mathcal{E}_C(\lambda x.x(xx) : ((\alpha \to \alpha) \cap (\alpha \to \alpha) \cap \alpha) \to \alpha)$$
$$\lhd(\lambda x_1 x_2 x_3.x_1(x_2 x_3), \{x : \{x_1 : \alpha \to \alpha, x_2 : \alpha \to \alpha, x_3 : \alpha\}\})$$

AC-Expansion

AC - Associative, Commutative but not Idempotent ($\tau \cap \tau \neq \tau$)

$$\underbrace{\mathcal{E}_I(x:\tau)}_{\text{ACI}} \quad \triangleleft \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } x \neq y$$

$$\underbrace{\mathcal{E}_C(x:\tau)}_{\text{AC}} \quad \triangleleft \quad (y, \{x : \{y : \tau\}\}), \quad \text{if } y \text{ is a fresh variable}$$

For example:

$$\begin{aligned}&\mathcal{E}_C(\lambda x.x(xx) : ((\alpha \to \alpha) \cap (\alpha \to \alpha) \cap \alpha) \to \alpha)\\ &\quad \triangleleft(\lambda x_1 x_2 x_3.x_1(x_2 x_3), \{x : \{x_1 : \alpha \to \alpha, x_2 : \alpha \to \alpha, x_3 : \alpha\}\})\end{aligned}$$

AC-Expansion - Properties

In AC expansion the number of types in the intersection is the same as the free occurrences of the parameter in the function body.

We have the following properties regarding AC expansion:

$$\mathcal{E}_C(M : \sigma) \lhd (N, \mathcal{C})$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_A N : \mathcal{T}(\sigma)$.
- If M is a λI-term, then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_L N : \mathcal{T}(\sigma)$.

AC-Expansion - Properties

In AC expansion the number of types in the intersection is the same as the free occurrences of the parameter in the function body.

We have the following properties regarding AC expansion:

$$\mathcal{E}_C(M : \sigma) \lhd (N, \mathcal{C})$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_A N : \mathcal{T}(\sigma)$.
- If M is a λI-term, then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_L N : \mathcal{T}(\sigma)$.

AC-Expansion - Properties

In AC expansion the number of types in the intersection is the same as the free occurrences of the parameter in the function body.

We have the following properties regarding AC expansion:

$$\mathcal{E}_C(M : \sigma) \lhd (N, \mathcal{C})$$

- $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_A N : \mathcal{T}(\sigma)$.
- If M is a λI-term, then $\Gamma \vdash_\cap M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_L N : \mathcal{T}(\sigma)$.

Ordered A-Expansion

A - Associative, but not Commutative ($\tau \cap \sigma \neq \sigma \cap \tau$) nor Idempotent ($\tau \cap \tau \neq \tau$)

$$\begin{array}{lll}
\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) & \lhd & (\lambda y_1 \dots y_n.M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_n) \to_r \mathcal{T}(\sigma)}, A), \\
& & \text{if } x \in \mathtt{fv}(M) \text{ and} \\
& & \mathcal{E}_O(M : \sigma) \lhd (M_0^{\mathcal{T}(\sigma)}, A + [x : [x_1 : \mathcal{T}(\sigma_1), \dots, x_n : \mathcal{T}(\sigma_n)]])
\end{array}$$

$$\begin{array}{lll}
\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) & \lhd & (\lambda x_1 \dots x_n.M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_n) \to_l \mathcal{T}(\sigma)}, A), \\
& & \text{if } x \in \mathtt{fv}(M) \text{ and} \\
& & \mathcal{E}_O(M : \sigma) \lhd (M_0^{\mathcal{T}(\sigma)}, [x : [x_n : \mathcal{T}(\sigma_n), \dots, x_1 : \mathcal{T}(\sigma_1)]] + A)
\end{array}$$

$$\begin{array}{lll}
\mathcal{E}_O(MN : \sigma) & \lhd & ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_0 + A_1 + \dots + A_m) \\
& & \text{if for some } m > 0 \text{ and } \sigma_1, \dots, \sigma_m \\
& & \mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \lhd (M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_m) \to_r \mathcal{T}(\sigma)}, A_0) \\
& & \text{and } \left(\mathcal{E}_O(N : \sigma_i) \lhd (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}
\end{array}$$

$$\begin{array}{lll}
\mathcal{E}_O(MN : \sigma) & \lhd & ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_m + \dots + A_1 + A_0), \\
& & \text{if for some } m > 0 \text{ and } \sigma_1, \dots, \sigma_m \\
& & \mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \lhd (M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_m) \to_l \mathcal{T}(\sigma)}, A_0) \\
& & \text{and } \left(\mathcal{E}_O(N : \sigma_i) \lhd (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}
\end{array}$$

Ordered A-Expansion

A - Associative, but not Commutative ($\tau \cap \sigma \neq \sigma \cap \tau$) nor Idempotent ($\tau \cap \tau \neq \tau$)

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \cdots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda y_1 \ldots y_n.M_0^{\mathcal{T}(\sigma_1)\to_r \cdots \to_r \mathcal{T}(\sigma_n) \to_r \mathcal{T}(\sigma)}, A),$$
if $x \in \mathtt{fv}(M)$ and
$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, A + [x : [x_1 : \mathcal{T}(\sigma_1), \ldots, x_n : \mathcal{T}(\sigma_n)]])$

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \cdots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda x_1 \ldots x_n.M_0^{\mathcal{T}(\sigma_1)\to_l \cdots \to_l \mathcal{T}(\sigma_n) \to_l \mathcal{T}(\sigma)}, A),$$
if $x \in \mathtt{fv}(M)$ and
$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, [x : [x_n : \mathcal{T}(\sigma_n), \ldots, x_1 : \mathcal{T}(\sigma_1)]] + A)$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \ldots N_m)^{\mathcal{T}(\sigma)}, A_0 + A_1 + \cdots + A_m)$$
if for some $m > 0$ and $\sigma_1, \ldots, \sigma_m$
$\mathcal{E}_O(M : \sigma_1 \cap \cdots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1)\to_r \cdots \to_r \mathcal{T}(\sigma_m) \to_r \mathcal{T}(\sigma)}, A_0)$
and $\left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\ldots m}$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \ldots N_m)^{\mathcal{T}(\sigma)}, A_m + \cdots + A_1 + A_0),$$
if for some $m > 0$ and $\sigma_1, \ldots, \sigma_m$
$\mathcal{E}_O(M : \sigma_1 \cap \cdots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1)\to_l \cdots \to_l \mathcal{T}(\sigma_m) \to_l \mathcal{T}(\sigma)}, A_0)$
and $\left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\ldots m}$

Ordered A-Expansion

A - Associative, but not Commutative ($\tau \cap \sigma \neq \sigma \cap \tau$) nor Idempotent ($\tau \cap \tau \neq \tau$)

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda y_1 \dots y_n.M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_n) \to_r \mathcal{T}(\sigma)}, A),$$
$$\text{if } x \in \mathtt{fv}(M) \text{ and}$$
$$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, A + [x : [x_1 : \mathcal{T}(\sigma_1), \dots, x_n : \mathcal{T}(\sigma_n)]])$$

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda x_1 \dots x_n.M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_n) \to_l \mathcal{T}(\sigma)}, A),$$
$$\text{if } x \in \mathtt{fv}(M) \text{ and}$$
$$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, [x : [x_n : \mathcal{T}(\sigma_n), \dots, x_1 : \mathcal{T}(\sigma_1)]] + A)$$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_0 + A_1 + \dots + A_m)$$
$$\text{if for some } m > 0 \text{ and } \sigma_1, \dots, \sigma_m$$
$$\mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_m) \to_r \mathcal{T}(\sigma)}, A_0)$$
$$\text{and } \left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}$$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_m + \dots + A_1 + A_0),$$
$$\text{if for some } m > 0 \text{ and } \sigma_1, \dots, \sigma_m$$
$$\mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_m) \to_l \mathcal{T}(\sigma)}, A_0)$$
$$\text{and } \left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}$$

Ordered A-Expansion

A - Associative, but not Commutative ($\tau \cap \sigma \neq \sigma \cap \tau$) nor Idempotent ($\tau \cap \tau \neq \tau$)

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda y_1 \dots y_n.M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_n) \to_r \mathcal{T}(\sigma)}, A),$$
if $x \in \mathtt{fv}(M)$ and
$$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, A + [x : [x_1 : \mathcal{T}(\sigma_1), \dots, x_n : \mathcal{T}(\sigma_n)]])$$

$$\mathcal{E}_O(\lambda x.M : \sigma_1 \cap \dots \cap \sigma_n \to \sigma) \quad \triangleleft \quad (\lambda x_1 \dots x_n.M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_n) \to_l \mathcal{T}(\sigma)}, A),$$
if $x \in \mathtt{fv}(M)$ and
$$\mathcal{E}_O(M : \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma)}, [x : [x_n : \mathcal{T}(\sigma_n), \dots, x_1 : \mathcal{T}(\sigma_1)]] + A)$$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_0 + A_1 + \dots + A_m)$$
if for some $m > 0$ and $\sigma_1, \dots, \sigma_m$
$$\mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1) \to_r \dots \to_r \mathcal{T}(\sigma_m) \to_r \mathcal{T}(\sigma)}, A_0)$$
and $\left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}$

$$\mathcal{E}_O(MN : \sigma) \quad \triangleleft \quad ((M_0 N_1 \dots N_m)^{\mathcal{T}(\sigma)}, A_m + \dots + A_1 + A_0),$$
if for some $m > 0$ and $\sigma_1, \dots, \sigma_m$
$$\mathcal{E}_O(M : \sigma_1 \cap \dots \cap \sigma_m \to \sigma) \triangleleft (M_0^{\mathcal{T}(\sigma_1) \to_l \dots \to_l \mathcal{T}(\sigma_m) \to_l \mathcal{T}(\sigma)}, A_0)$$
and $\left(\mathcal{E}_O(N : \sigma_i) \triangleleft (N_i^{\mathcal{T}(\sigma_i)}, A_i)\right)_{i=1\dots m}$

Ordered Expansion - Example

Let $M \equiv (\lambda x.xz)z$. The ordered expansion of M is calculated step by step as:

$$\mathcal{E}_O(x : \alpha \to \beta) = ({x_1}^{\alpha \to_r \beta}, [x : [x_1 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O(z : \beta) = ({z_1}^{\beta}, [z : [z_1 : \beta]])$$

$$\mathcal{E}_O(xz : \beta) = ((x_1 z_1)^{\beta}, [x : [x_1 : \alpha \to_r \beta], z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(\lambda x.xz : (\alpha \to \beta) \to \beta) = ((\lambda x_1.x_1 z_1)^{(\alpha \to_r \beta) \to_l \beta}, [z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(z : \alpha \to \beta) = ({z_2}^{\alpha \to_r \beta}, [z : [z_2 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O((\lambda x.xz)z : \beta) = (((\lambda x_1.x_1 z_1)z_2)^{\beta}, [z : [z_2 : \alpha \to_r \beta, z_1 : \alpha]])$$

Ordered Expansion - Example

Let $M \equiv (\lambda x.xz)z$. The ordered expansion of M is calculated step by step as:

$$\mathcal{E}_O(x : \alpha \to \beta) = ({x_1}^{\alpha \to_r \beta}, [x : [x_1 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O(z : \beta) = ({z_1}^{\beta}, [z : [z_1 : \beta]])$$

$$\mathcal{E}_O(xz : \beta) = ((x_1 z_1)^{\beta}, [x : [x_1 : \alpha \to_r \beta], z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(\lambda x.xz : (\alpha \to \beta) \to \beta) = ((\lambda x_1.x_1 z_1)^{(\alpha \to_r \beta) \to_l \beta}, [z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(z : \alpha \to \beta) = ({z_2}^{\alpha \to_r \beta}, [z : [z_2 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O((\lambda x.xz)z : \beta) = (((\lambda x_1.x_1 z_1)z_2)^{\beta}, [z : [z_2 : \alpha \to_r \beta, z_1 : \alpha]])$$

Ordered Expansion - Example

Let $M \equiv (\lambda x.xz)z$. The ordered expansion of M is calculated step by step as:

$$\mathcal{E}_O(x : \alpha \to \beta) = ({x_1}^{\alpha \to_r \beta}, [x : [x_1 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O(z : \beta) = ({z_1}^{\beta}, [z : [z_1 : \beta]])$$

$$\mathcal{E}_O(xz : \beta) = ((x_1 z_1)^{\beta}, [x : [x_1 : \alpha \to_r \beta], z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(\lambda x.xz : (\alpha \to \beta) \to \beta) = ((\lambda x_1.x_1 z_1)^{(\alpha \to_r \beta) \to_l \beta}, [z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(z : \alpha \to \beta) = ({z_2}^{\alpha \to_r \beta}, [z : [z_2 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O((\lambda x.xz)z : \beta) = (((\lambda x_1.x_1 z_1)z_2)^{\beta}, [z : [z_2 : \alpha \to_r \beta, z_1 : \alpha]])$$

Ordered Expansion - Example

Let $M \equiv (\lambda x.xz)z$. The ordered expansion of M is calculated step by step as:

$$\mathcal{E}_O(x : \alpha \to \beta) = ({x_1}^{\alpha \to_r \beta}, [x : [x_1 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O(z : \beta) = ({z_1}^{\beta}, [z : [z_1 : \beta]])$$

$$\mathcal{E}_O(xz : \beta) = ((x_1 z_1)^{\beta}, [x : [x_1 : \alpha \to_r \beta], z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(\lambda x.xz : (\alpha \to \beta) \to \beta) = ((\lambda x_1.x_1 z_1)^{(\alpha \to_r \beta) \to_l \beta}, [z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(z : \alpha \to \beta) = ({z_2}^{\alpha \to_r \beta}, [z : [z_2 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O((\lambda x.xz)z : \beta) = (((\lambda x_1.x_1 z_1)z_2)^{\beta}, [z : [z_2 : \alpha \to_r \beta, z_1 : \alpha]])$$

Ordered Expansion - Example

Let $M \equiv (\lambda x.xz)z$. The ordered expansion of M is calculated step by step as:

$$\mathcal{E}_O(x : \alpha \to \beta) = ({x_1}^{\alpha \to_r \beta}, [x : [x_1 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O(z : \beta) = ({z_1}^{\beta}, [z : [z_1 : \beta]])$$

$$\mathcal{E}_O(xz : \beta) = ((x_1 z_1)^{\beta}, [x : [x_1 : \alpha \to_r \beta], z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(\lambda x.xz : (\alpha \to \beta) \to \beta) = ((\lambda x_1.x_1 z_1)^{(\alpha \to_r \beta) \to_l \beta}, [z : [z_1 : \alpha]])$$

$$\mathcal{E}_O(z : \alpha \to \beta) = ({z_2}^{\alpha \to_r \beta}, [z : [z_2 : \alpha \to_r \beta]])$$

$$\mathcal{E}_O((\lambda x.xz)z : \beta) = (((\lambda x_1.x_1 z_1)z_2)^{\beta}, [z : [z_2 : \alpha \to_r \beta, z_1 : \alpha]])$$

Ordered Expansion - Properties

We have the following property regarding A expansion:

$$\mathcal{E}_{\mathcal{O}}(M : \sigma) \lhd (N^{\mathcal{T}(\sigma)}, A)$$

If M is a λI-term, then $\Gamma \vdash_{\cap} M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_{O} N : \mathcal{T}(\sigma)$.

But now $\mathcal{T}$ goes from intersection types to ordered types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to_r \cdots \to_r \mathcal{T}(\tau_n) \to_r \mathcal{T}(\sigma)$.

We have the following property regarding A expansion:

$$\mathcal{E}_{\mathcal{O}}(M : \sigma) \lhd (N^{\mathcal{T}(\sigma)}, A)$$

If M is a λI-term, then $\Gamma \vdash_{\cap} M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_{O} N : \mathcal{T}(\sigma)$.

But now $\mathcal{T}$ goes from intersection types to ordered types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to_r \cdots \to_r \mathcal{T}(\tau_n) \to_r \mathcal{T}(\sigma)$.

Ordered Expansion - Properties

We have the following property regarding A expansion:

$$\mathcal{E}_{\mathcal{O}}(M : \sigma) \lhd (N^{\mathcal{T}(\sigma)}, A)$$

If M is a λI-term, then $\Gamma \vdash_{\cap} M : \sigma \Rightarrow \mathcal{T}(\Gamma) \vdash_{O} N : \mathcal{T}(\sigma)$.

But now $\mathcal{T}$ goes from intersection types to ordered types:

- $\mathcal{T}(\alpha) = \alpha$, if α is a type variable;
- $\mathcal{T}((\tau_1 \cap \cdots \cap \tau_n) \to \sigma) = \mathcal{T}(\tau_1) \to_r \cdots \to_r \mathcal{T}(\tau_n) \to_r \mathcal{T}(\sigma)$.

What about reduction?

Consider *weak head reduction* $\underset{w}{\rightarrow}$ is defined by:

$$(\lambda x.M)N \underset{w}{\rightarrow} M[N/x]$$

and

$$\frac{M \underset{w}{\rightarrow} M'}{MN \underset{w}{\rightarrow} M'N}$$

In functional programming languages, reduction is weak.

Expansion (ACI, AC and A) preserves weak head reduction, thus the following diagram commutes:

$$\begin{tikzcd}
M_1 \arrow[r, "w"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\
N_1 \arrow[r, two heads, "w"'] & N_2
\end{tikzcd}$$

What about reduction?

Consider *weak head reduction* $\underset{w}{\rightarrow}$ is defined by:

$$(\lambda x.M)N \underset{w}{\rightarrow} M[N/x]$$

and

$$\frac{M \underset{w}{\rightarrow} M'}{MN \underset{w}{\rightarrow} M'N}$$

In functional programming languages, reduction is weak.

Expansion (ACI, AC and A) preserves weak head reduction, thus the following diagram commutes:

$$\begin{tikzcd} M_1 \arrow[r, "w"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\ N_1 \arrow[r, twoheadrightarrow, "w"'] & N_2 \end{tikzcd}$$

What about reduction?

Consider *weak head reduction* $\underset{w}{\rightarrow}$ is defined by:

$$(\lambda x.M)N \underset{w}{\rightarrow} M[N/x]$$

and

$$\frac{M \underset{w}{\rightarrow} M'}{MN \underset{w}{\rightarrow} M'N}$$

In functional programming languages, reduction is weak.

Expansion (ACI, AC and A) preserves weak head reduction, thus the following diagram commutes:

$$\begin{tikzcd} M_1 \arrow[r, "w"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\ N_1 \arrow[r, two heads, "w"'] & N_2 \end{tikzcd}$$

What about reduction?

Consider *weak head reduction* $\underset{w}{\rightarrow}$ is defined by:

$$(\lambda x.M)N \underset{w}{\rightarrow} M[N/x]$$

and

$$\frac{M \underset{w}{\rightarrow} M'}{MN \underset{w}{\rightarrow} M'N}$$

In functional programming languages, reduction is weak.

Expansion (ACI, AC and A) preserves weak head reduction, thus the following diagram commutes:

$$\begin{tikzcd} M_1 \arrow[r, "w"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\ N_1 \arrow[r, two heads, "w"'] & N_2 \end{tikzcd}$$

Correctness

Expansion commutes with β-reduction in the λI-calculus,

$$\begin{tikzcd} M_1 \arrow[r, "\beta"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\ N_1 \arrow[r, two heads, "\beta"'] & N_2 \end{tikzcd}$$

but not in the λ-calculus:

$$\begin{tikzcd} \lambda x.(\lambda y.z)xx \arrow[r, "\beta"'] \arrow[d, "\mathcal{E}"'] & \lambda x.zx \\ \lambda x_1 x_2.(\lambda y.z)x_1 x_2 \arrow[r, "\beta"'] & \lambda x_1 x_2.z x_2 \end{tikzcd}$$

Correctness

Expansion commutes with β-reduction in the λI-calculus,

$$\begin{tikzcd} M_1 \arrow[r, "\beta"'] \arrow[d, "\mathcal{E}"'] & M_2 \arrow[d, "\mathcal{E}"] \\ N_1 \arrow[r, twoheadrightarrow, "\beta"'] & N_2 \end{tikzcd}$$

but not in the λ-calculus:

$$\begin{tikzcd} \lambda x.(\lambda y.z)xx \arrow[r, "\beta"'] \arrow[d, "\mathcal{E}"'] & \lambda x.zx \\ \lambda x_1 x_2.(\lambda y.z)x_1 x_2 \arrow[r, "\beta"'] & \lambda x_1 x_2.zx_2 \end{tikzcd}$$

To summarize...

How does reduction relates to the different notions of expansion:

$\cap$	**Source**	**Target**	**Preserves reductions**
ACI	λ	Simple Types	Weak Head Reduction
ACI	λI	Relevant Types	β-reduction
AC	λ	Affine Types	Weak Head Reduction
AC	λI	Linear Types	β-reduction
A	λI	Ordered Types	β-reduction

To summarize...

How does reduction relates to the different notions of expansion:

$\cap$	**Source**	**Target**	**Preserves reductions**
ACI	λ	Simple Types	Weak Head Reduction
ACI	λI	Relevant Types	β-reduction
AC	λ	Affine Types	Weak Head Reduction
AC	λI	Linear Types	β-reduction
A	λI	Ordered Types	β-reduction

To summarize...

How does reduction relates to the different notions of expansion:

$\cap$	**Source**	**Target**	**Preserves reductions**
ACI	λ	Simple Types	Weak Head Reduction
ACI	λI	Relevant Types	β-reduction
AC	λ	Affine Types	Weak Head Reduction
AC	λI	Linear Types	β-reduction
A	λI	Ordered Types	β-reduction

To summarize...

How does reduction relates to the different notions of expansion:

$\cap$	**Source**	**Target**	**Preserves reductions**
ACI	λ	Simple Types	Weak Head Reduction
ACI	λI	Relevant Types	β-reduction
AC	λ	Affine Types	Weak Head Reduction
AC	λI	Linear Types	β-reduction
A	λI	Ordered Types	β-reduction

To summarize...

How does reduction relates to the different notions of expansion:

$\cap$	**Source**	**Target**	**Preserves reductions**
ACI	λ	Simple Types	Weak Head Reduction
ACI	λI	Relevant Types	β-reduction
AC	λ	Affine Types	Weak Head Reduction
AC	λI	Linear Types	β-reduction
A	λI	Ordered Types	β-reduction

To summarize...

How the different structural rules relate to the different expansion relations:

Type System	**W**	**E**	**C**	**Assumptions**	**Intersection**
Relevant		✓	✓	at least once	ACI
Affine	✓	✓		at most once	AC
Linear		✓		exactly once	AC
Ordered				in order	A

Current and Future Work

What are we currently looking at...

Remember the two (valid) typings:

$$z_1 : \alpha \rightarrow_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

We would like to be able to have a notion of principal-pair for the ordered type system and a type-inference algorithm.

What else we would like to look at...

- Study in more detail the relation between the linear and the ordered system;
- In theory there are more possible substructural systems... is any of the remaining ones interesting?

What are we currently looking at...

Remember the two (valid) typings:

$$z_1 : \alpha \rightarrow_r \beta,\ z_2 : \alpha \vdash_O (\lambda x.xz_2)z_1 : \beta$$
$$z_2 : \alpha,\ z_1 : \alpha \rightarrow_l \beta \vdash_O (\lambda x.xz_2)z_1 : \beta$$

We would like to be able to have a notion of principal-pair for the ordered type system and a type-inference algorithm.

What else we would like to look at...

- Study in more detail the relation between the linear and the ordered system;
- In theory there are more possible substructural systems... is any of the remaining ones interesting?

Thank you!