Quantitative Types

QCOMICAL School 2025

November, 2025

Pablo Barenbaum

Instituto de Ciencias de la Computación
FCEyN, UBA, Argentina

Universidad Nacional de Quilmes / CONICET
Argentina

Quantitative Types

Topic of this course

non-idempotent intersection types

a.k.a. **quantitative types**

a.k.a. **multi-types**

a.k.a. **tensor types**

Comparison (in one slide)

"Typical" type systems

- guarantee properties of programs (typable $\implies$ has property P)
- capture qualitative properties of programs
  (termination, productivity, deadlock-freeness, ...)
- each fragment of a program is typed exactly once
- type inference is decidable (useful for static analysis)

Quantitative type systems

- characterise properties of programs (typable $\iff$ has property P)
- capture quantitative properties of programs
  (reduction length, size of the normal form, # memory accesses, ...)
- each fragment of a program is typed zero, one, or more times
  (as many times as it is used in runtime)
- type inference is undecidable (but they are useful as models)

Failure of subject expansion

Consider the following interpretation in the **simply typed** λ-calculus:

$$[\![t]\!] = \{A \mid \ \vdash t : A\}$$

Does $t =_\beta s$ imply $[\![t]\!] = [\![s]\!]$? We require:

- Subject reduction: $t \to_\beta s$ and $\vdash t : A$ implies $\vdash s : A$.
- Subject expansion: $t \to_\beta s$ and $\vdash s : A$ implies $\vdash t : A$.

Failure of subject expansion

Does $\vdash p\{x := q\} : A$ imply $\vdash (\lambda x.\, p)\, q : A$?

Problem: $p\{x := q\}$ may produce zero, one, or more copies of q.

$$(\lambda x.\, \mathtt{id})\ \underset{???}{\Omega}\ \to_\beta \mathtt{id}$$

$$(\lambda x.\, x\, x)\ \underset{???}{\mathtt{id}}\ \to_\beta\ \underset{A \to A}{\mathtt{id}}\ \ \underset{A}{\mathtt{id}}$$

Idea: the identity on the left could be typed with $(A \to A) \cap A$.

Intersection type systems

Coppo & Dezani-Ciancaglini (1980)

Syntax

$$\begin{array}{ll} \textsc{Terms} & t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \\ \textsc{Types} & A, B, \ldots ::= \alpha \mid \{A_1, \ldots, A_n\} \to B \qquad (n \geq 1) \end{array}$$

- $\{A_1, \ldots, A_n\}$ is a non-empty **set** of types.
- Intuitively, it represents a finite *intersection* $A_1 \cap \ldots \cap A_n$.

Typing rules of $\lambda_\cap^{\mathrm{CD}}$

$$\frac{}{\Gamma, x : \{A_1, \ldots, A_i, \ldots, A_n\} \vdash x : A_i}$$

$$\frac{\Gamma, x : \{A_1, \ldots, A_n\} \vdash t : B}{\Gamma \vdash \lambda x.\, t : \{A_1, \ldots, A_n\} \to B}$$

$$\frac{\Gamma \vdash t : \{A_1, A_2, \ldots, A_n\} \to B \quad \Gamma \vdash s : A_1 \quad \Gamma \vdash s : A_2 \quad \ldots \quad \Gamma \vdash s : A_n}{\Gamma \vdash t\, s : B}$$

Coppo & Dezani-Ciancaglini (1980)

Example

Let:

- $\mathtt{id} = \lambda x.\, x$
- $A = \{\alpha\} \to \alpha$
- $B = \{A\} \to A = \{\{\alpha\} \to \alpha\} \to \{\alpha\} \to \alpha$

Then:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \{A, B\} \vdash x : \underbrace{\{A\} \to A}_{B}} \qquad \dfrac{}{x : \{A, B\} \vdash x : A}}{x : \{A, B\} \vdash x\, x : A}}{\vdash \lambda x.\, x\, x : \{A,\ B\} \to A} \qquad \dfrac{\dfrac{}{x : \{\alpha\} \vdash x : \alpha}}{\vdash \mathtt{id} : A} \qquad \dfrac{\dfrac{}{x : \{A\} \vdash x : A}}{\vdash \mathtt{id} : B}}{\vdash (\lambda x.\, x\, x)\, \mathtt{id} : A}$$

"Finitistic" polymorphism.

Note: $\lambda x.\, x\, x$ is SN but not typable using simple types.

Intersection type systems

Coppo & Dezani-Ciancaglini (1980)

Theorem (Characterisation of Strong Normalisation)

The following are equivalent:

1. **Typability.**
   There exist Γ, A such that $\Gamma \vdash t : A$ holds in $\lambda_{\cap}^{\mathrm{CD}}$.
2. **Strong $\rightarrow_\beta$-normalisation.**
   There are no infinite reduction sequences $t \rightarrow_\beta t_1 \rightarrow_\beta t_2 \ldots$

Note: connection with denotational semantics

- Any Scott $\mathcal{D}_\infty$ model can be described as a filter model $\mathcal{F}^{\mathrm{TT}}$ for some intersection type theory TT.
- In a filter model, $[\![t]\!] = \{A \mid \vdash_{\mathrm{TT}} t : A\}$ holds for closed t.

For a survey, see Barendregt *et al.* 's *Lambda Calculus with Types* (2010)

Linear Logic

Girard (1987)

Sequent calculi usually include **structural rules**:

Weakening	Contraction	Exchange
$\dfrac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathtt{LW}$	$\dfrac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathtt{LC}$	$\dfrac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta}\,\mathtt{LX}$
$\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\,\mathtt{RW}$	$\dfrac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\,\mathtt{RC}$	$\dfrac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2}\,\mathtt{RX}$

Linear Logic

- **Resource-aware** logic.
- No weakening: $(A \otimes B) \multimap A$ is not a theorem.
- No contraction: $A \multimap (A \otimes A)$ is not a theorem.
- Exchange: contexts can be understood as **multisets** of formulae. (Not completely equivalent).
- Intuitively, each hypothesis must be used exactly once.

Linear Logic

Girard (1987)

MLL (Multiplicative fragment)

$$\textsc{Formulae} \quad A, B, \ldots \quad ::= \alpha \mid \overline{\alpha} \mid A \otimes B \mid A \parr B$$

$$\begin{array}{rclrcl} \alpha^\perp & := & \overline{\alpha} & (A \otimes B)^\perp & := & A^\perp \parr B^\perp \\ \overline{\alpha}^\perp & := & \alpha & (A \parr B)^\perp & := & A^\perp \otimes B^\perp \end{array}$$

$A \multimap B$ abbreviates $A^\perp \parr B$.

Contexts $(\Gamma, \Delta, \ldots)$ are **multisets** of formulae (implicit exchange).

Inference rules

$$\frac{}{\vdash A, A^\perp}\,\mathtt{ax} \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\,\otimes \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,\parr$$

- No implicit weakening in the rule $\mathtt{ax}$.
- No implicit contraction in the rule $\otimes$.
- The rule $\otimes$ requires to choose how to split the context.
  For example: $\vdash A^\perp, B^\perp, C^\perp, B \otimes (C \otimes A)$.

Linear Logic

Girard (1987)

Definition (Approximation)

A formula in MLL approximates an intuitionistic formula according to the inductive definition[1]:

$$\frac{}{\alpha \sqsubset \alpha} \qquad \frac{A_1 \sqsubset X \quad \dots \quad A_n \sqsubset X \quad B \sqsubset Y}{(A_1 \otimes \dots \otimes A_n) \multimap B \sqsubset X \to Y}$$

[1] More precisely: MLL with units and minimal logic.

Theorem (Girard's translation + approximation theorem)

If X is a valid intuitionistic formula, there is a valid MLL formula $A \sqsubset X$.

$$\alpha \multimap \mathbf{1} \multimap \alpha \qquad \sqsubset \quad \alpha \to \beta \to \alpha$$

$$(\alpha \multimap \alpha \multimap \beta) \multimap (\alpha \otimes \alpha) \multimap \beta \qquad \sqsubset \quad (\alpha \to \alpha \to \beta) \to \alpha \to \beta$$

Quantitative type systems embody approximation theorems.

Head reduction

Remark. Every λ-term is of exactly one of the two following forms:

1. $\lambda x_1 \dots x_n .\, y\, t_1 \dots t_m$
2. $\lambda x_1 \dots x_n .\, (\lambda y.\, p)\, q\, t_1 \dots t_m$

Nomenclature

$\underbrace{\lambda x_1 \dots x_n .\, y\, t_1 \dots t_m}_{\text{head normal form}}$ HNF is the set of head normal forms

$$\lambda x_1 \dots x_n .\, \underbrace{s}_{\text{subterm in head position}}\, t_1 \dots t_m$$

$$\lambda x_1 \dots x_n .\, \underbrace{y}_{\text{head variable}}\, t_1 \dots t_m \qquad \lambda x_1 \dots x_n .\, \underbrace{(\lambda y.\, p)\, q}_{\text{head redex}}\, t_1 \dots t_m$$

$$\lambda x_1 \dots x_n .\, (\lambda y.\, p)\, q\, t_1 \dots t_m \ \underbrace{\rightarrow_{\mathtt{h}}}_{\text{head reduction}}\ \lambda x_1 \dots x_n .\, p\{y := q\}\, t_1 \dots t_m$$

t is head-normalising $\overset{\text{def}}{\iff} \exists s.\, t \rightarrow_{\mathtt{h}}^{*} s \in \text{HNF}$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

TERMS	$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s$
TYPES	$A, B, \ldots ::= \alpha \mid \mathcal{M} \to A$
MULTI-TYPES	$\mathcal{M}, \mathcal{N}, \ldots ::= [A_i]_{i \in I}$

- A multi-type is a (possibly empty) **finite multiset** of types.
- $\mathcal{M} + \mathcal{N}$ is the union of multi-types.
- A context $(\Gamma, \Delta, \ldots)$ is a function mapping variables to multi-types.
- We use sequential notation to write contexts. For instance:

$$\Gamma \quad = \quad (\, x : [[\alpha] \to \beta, \alpha], y : [\beta, \beta, \gamma]\,)$$

  is the context that maps:

$$x \mapsto [[\alpha] \to \beta, \alpha] \qquad y \mapsto [\beta, \beta, \gamma] \qquad z \mapsto [\,] \qquad \ldots$$

- We assume that contexts are of **finite support**.
- $\Gamma + \Delta$ is the context defined by $(\Gamma + \Delta)(x) = \Gamma(x) + \Delta(x)$.

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

We have two forms of judgment:

$$\Gamma \vdash t : A \qquad \Gamma \Vdash t : \mathcal{M}$$

Typing rules of System $\mathcal{H}$

$$\frac{}{x : [A] \vdash x : A}\,\mathtt{var} \qquad \frac{\Gamma, x : \mathcal{M} \vdash t : A}{\Gamma \vdash \lambda x.\, t : \mathcal{M} \to A}\,\mathtt{lam}$$

$$\frac{\Gamma \vdash t : \mathcal{M} \to A \qquad \Delta \Vdash s : \mathcal{M}}{\Gamma + \Delta \vdash t\, s : A}\,\mathtt{app} \qquad \frac{\Gamma_1 \vdash t : A_1 \qquad \ldots \qquad \Gamma_n \vdash t : A_n}{\Gamma_1 + \ldots + \Gamma_n \Vdash t : [A_1, \ldots, A_n]}\,\mathtt{many}$$

- "Linear logic in disguise".
- Rules are multiplicative: no implicit weakening nor contraction.
- Rules are logically sound w.r.t. the translation to MLL (with units):

$$\underline{\mathcal{M} \to A} \quad = \quad \underline{\mathcal{M}} \multimap \underline{A}$$

$$\underline{[A_1, \ldots, A_n]} \quad = \quad \underline{A_1} \otimes \ldots \otimes \underline{A_n}$$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Sometimes instead of:

$$\dfrac{\Gamma \vdash t : [A_1, \ldots, A_n] \to B \qquad \dfrac{\Delta_1 \vdash s : A_1 \qquad \ldots \qquad \Delta_n \vdash s : A_n}{\Delta_1 + \ldots + \Delta_n \Vdash s : [A_1, \ldots, A_n]}\texttt{many}}{\Gamma + \Delta_1 + \ldots + \Delta_n \vdash t\, s : B}\texttt{app}$$

we write:

$$\dfrac{\Gamma \vdash t : [A_1, \ldots, A_n] \to B \qquad \Delta_1 \vdash s : A_1 \qquad \ldots \qquad \Delta_n \vdash s : A_n}{\Gamma + \Delta_1 + \ldots + \Delta_n \vdash t\, s : B}\texttt{app}$$

This is just a minor abuse of notation.

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Example (1)

$$\dfrac{\dfrac{\overline{x : [[A] \to A] \vdash x : [A] \to A}}{\vdash \mathtt{id} : [[A] \to A] \to [A] \to A} \qquad \dfrac{\overline{x : [A] \vdash x : A}}{\vdash \mathtt{id} : [A] \to A}}{\vdash \mathtt{id}\,\mathtt{id} : [A] \to A}$$

$$\rightsquigarrow$$

$$\dfrac{\overline{x : [A] \vdash x : A}}{\vdash \mathtt{id} : [A] \to A}$$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Example (2)

Let:

- $A = [\alpha] \to \alpha$
- $B = [A] \to A = [[\alpha] \to \alpha] \to [\alpha] \to \alpha$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : [B] \vdash x : B} \quad \dfrac{\dfrac{}{x : [B] \vdash x : B} \quad \dfrac{}{x : [A] \vdash x : A}}{x : [A, B] \vdash x\,x : A}}{x : [A, B, B] \vdash x\,(x\,x) : A}}{\vdash \lambda x.\, x\,(x\,x) : [A, B, B] \to A} \quad \vdash \mathtt{id}\,\mathtt{id} : A \quad \vdash \mathtt{id}\,\mathtt{id} : B \quad \vdash \mathtt{id}\,\mathtt{id} : B}{\vdash (\lambda x.\, x\,(x\,x))(\mathtt{id}\,\mathtt{id}) : A}$$

$$\rightsquigarrow$$

$$\dfrac{\vdash \mathtt{id}\,\mathtt{id} : B \qquad \dfrac{\vdash \mathtt{id}\,\mathtt{id} : B \qquad \vdash \mathtt{id}\,\mathtt{id} : A}{\vdash \mathtt{id}\,\mathtt{id}\,(\mathtt{id}\,\mathtt{id}) : A}}{\vdash \mathtt{id}\,\mathtt{id}\,(\mathtt{id}\,\mathtt{id}\,(\mathtt{id}\,\mathtt{id})) : A}$$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Example (3)

$$\dfrac{\overline{x : [[\,] \to A] \vdash x : [\,] \to A}}{x : [[\,] \to A] \vdash x\,x : A}$$

$$\dfrac{\overline{x : [[B] \to A] \vdash x : [B] \to A} \qquad \overline{x : [B] \vdash x : B}}{x : [[B] \to A, B] \vdash x\,x : A}$$

$$\dfrac{\overline{x : [[B, C] \to A] \vdash x : [B, C] \to A} \qquad \overline{x : [B] \vdash x : B} \qquad \overline{x : [C] \vdash x : C}}{x : [[B, C] \to A, B, C] \vdash x\,x : A}$$

More in general:

$$\vdash \lambda x.\, x\,x : [[B_1, \ldots, B_n] \to A, B_1, \ldots, B_n] \to A$$

However, $\Omega = (\lambda x.\, x\,x)\,\lambda x.\, x\,x$ is **not** typable.
Intuitively, the argument should be typed an infinite number of times.

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Example (4)

$$\cfrac{\cfrac{\cfrac{\cfrac{\overline{x : [[\,] \to A] \vdash x : [\,] \to A}}{x : [[\,] \to A] \vdash x\,\Omega : A}}{\vdash \lambda x.\, x\,\Omega : [[\,] \to A] \to A}}{\vdash \lambda y.\, \lambda x.\, x\, y : [\,] \to [[\,] \to A] \to A}}{\vdash (\lambda y.\, \lambda x.\, x\, y)\,\Omega : [[\,] \to A] \to A}$$

$$\rightsquigarrow$$

$$\cfrac{\cfrac{\overline{x : [[\,] \to A] \vdash x : [\,] \to A}}{x : [[\,] \to A] \vdash x\,\Omega : A}}{\vdash \lambda x.\, x\,\Omega : [[\,] \to A] \to A}$$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

We shall show that System $\mathcal{H}$ characterises **head normalising** terms.
Three key lemmas:

Lemma 1 (Weighted Subject Reduction)

If $t \rightarrow_{\mathtt{h}} s$ is a head step and $\Gamma \vdash t : A$ then $\Gamma \vdash s : A$.
Moreover, the **size** of the typing derivation decreases.
(The size is the number of inference rules, not counting the `many` rule).

Lemma 2 (Subject Expansion for head steps)

If $t \rightarrow_{\mathtt{h}} s$ is a head step and $\Gamma \vdash s : A$ then $\Gamma \vdash t : A$.

Lemma 3 (Typability of head normal forms)

If t is a head normal form, then t is typable.

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Assuming the lemmas on the previous slide, we have:

Theorem (System $\mathcal{H}$ characterises head normalisation)

The following are equivalent:

1. t is typable in System $\mathcal{H}$.
2. t is head normalising.

Proof of Soundness ($1 \implies 2$).

- Let $D \triangleright \Gamma \vdash t : A$ for some Γ, A.
- Proceed by induction on the size of D.
- If t is a head normal form, we are done.
- Otherwise, consider the head step $t \to_{\mathtt{h}} s$.
- By Weighted Subject Reduction, there is a typing derivation D' that concludes $\Gamma \vdash s : A$ and such that $\mathtt{sz}(D) > \mathtt{sz}(D')$.
- By IH, s is head normalising.
- Hence t is also head normalising.

Theorem (System $\mathcal{H}$ characterises head normalisation)

The following are equivalent:

1. t is typable in System $\mathcal{H}$.
2. t is head normalising.

Proof of Completeness $(2 \implies 1)$.

- Let $t \to_{\mathtt{h}} t_1 \to_{\mathtt{h}} t_2 \ldots \to_{\mathtt{h}} t_n$ with t_n a head normal form.
- Proceed by induction on n.
- If $n = 0$, t is a head normal form, so it is typable (by Lemma 3).
- If $n > 0$, by IH there exist Γ, A such that $\Gamma \vdash t_1 : A$.
- But $t \to_{\mathtt{h}} t_1$, so by Subject Expansion $\Gamma \vdash t : A$.

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

Lemma 1 (Weighted Subject Reduction)

If $t \to_{\mathtt{h}} s$ is a head step and $\Gamma \vdash t : A$ then $\Gamma \vdash s : A$.
Moreover, the **size** of the typing derivation decreases.

Proof.

- The head step is of the form:

$$t = (\lambda x_1 \dots x_n.\, (\lambda y.\, p)\, q\, t_1 \dots t_m) \to_{\mathtt{h}} (\lambda x_1 \dots x_n.\, p\{x := q\}\, t_1 \dots t_m) = s$$

- It is easy to reduce the general case to the root case ($n = m = 0$):

$$t = (\lambda y.\, p)\, q \to_{\mathtt{h}} p\{x := q\} = s$$

$$\dfrac{\dfrac{\dfrac{D_1}{\Gamma, x : \mathcal{M} \vdash p : A}}{\Gamma \vdash \lambda y.\, p : \mathcal{M} \to A}\,\mathtt{lam} \qquad \dfrac{D_2}{\Delta \Vdash q : \mathcal{M}}}{\Gamma + \Delta \vdash (\lambda y.\, p)\, q : A}\,\mathtt{app} \rightsquigarrow \dfrac{D'}{\Gamma + \Delta \vdash p\{x := q\} : A}$$

- The property is reduced to a Substitution Lemma.
- The rules on the left (lam, app) are erased — the size decreases.

Lemma 1' (Substitution Lemma)

Let $D_1 \triangleright \Gamma, x : \mathcal{M} \vdash t : A$ and $D_2 \triangleright \Delta \Vdash s : \mathcal{M}$.
Then there exists a derivation D' such that $D' \triangleright \Gamma + \Delta \vdash t\{x := s\} : A$ and $\mathtt{sz}(D') = \mathtt{sz}(D_1) - |\mathcal{M}| + \mathtt{sz}(D_2)$.

Proof.

- Proceed by induction on D_1.
- We only show some interesting cases:

D_1	D_2		D'
$\dfrac{}{x : [A] \vdash x : A}\,\mathtt{var}$	$\dfrac{\dfrac{\vdots}{\Delta \vdash s : A}}{\Delta \Vdash s : [A]}\,\mathtt{many}$	$\leadsto$	$\dfrac{\vdots}{\Delta \vdash s : A}$
$\dfrac{}{y : [A] \vdash y : A}\,\mathtt{var}$	$\dfrac{\text{(no premises)}}{\Vdash s : []}\,\mathtt{many}$	$\leadsto$	$\dfrac{}{y : [A] \vdash y : A}\,\mathtt{var}$

System $\mathcal{H}$

Gardner (1994), de Carvalho (2007)

The most interesting part is the substitution lemma on the many rule:

$$D_1 \quad = \quad \dfrac{\dfrac{D_{1,1}}{\Gamma_1, x : \mathcal{M}_1 \vdash t : A_1} \quad \ldots \quad \dfrac{D_{1,n}}{(\Gamma_n, x : \mathcal{M}_n \vdash t : A_n)}}{(\Gamma_1 + \ldots + \Gamma_n), x : (\mathcal{M}_1 + \ldots + \mathcal{M}_n) \vdash t : [A_1, \ldots, A_n]}\,\texttt{many}$$

$$D_2 \quad = \quad \dfrac{\vdots}{\Delta \Vdash s : (+_{i \in I} \mathcal{M}_i)}\,\texttt{many}$$

Then there exist contexts $\Delta_1, \ldots, \Delta_n$ and derivations $D_{2,1}, \ldots, D_{2,n}$ s.t.:

$$\dfrac{D_{2,1}}{\Delta_1 \Vdash s : \mathcal{M}_1}\,\texttt{many} \quad \ldots \quad \dfrac{D_{2,n}}{\Delta_n \Vdash s : \mathcal{M}_n}\,\texttt{many}$$

where $\Delta = +_{i=1}^{n} \Delta_i$ and $\mathtt{sz}(D_2) = +_{i=1}^{n} \mathtt{sz}(D_{2,i})$.

Applying the IH on each pair $D_{1,i}$ / $D_{2,i}$ to obtain $D_{3,i}$, we conclude:

$$D_3 = \dfrac{\dfrac{D_{3,1}}{\Gamma_1 + \Delta_1 \vdash t\{x := s\} : A_1} \quad \ldots \quad \dfrac{D_{3,n}}{\Gamma_n + \Delta_n \vdash t\{x := s\} : A_n}}{(\Gamma_1 + \ldots + \Gamma_n) + (\Delta_1 + \ldots + \Delta_n) \vdash t\{x := s\} : [A_1, \ldots, A_n]}\,\texttt{many}$$

Lemma 2 (Subject Expansion)

If $t \to_{\mathtt{h}} s$ is a head step and $\Gamma \vdash s : A$ then $\Gamma \vdash t : A$.

Proof. Similar to the proof of Subject Reduction.
Relies on an Anti-Substitution Lemma.

Lemma 2' (Anti-Substitution Lemma)

If $\Gamma \vdash t\{x := s\} : A$, there exist $\Gamma_1, \Gamma_2, \mathcal{M}$ such that:

- $\Gamma_1, x : \mathcal{M} \vdash t : A$
- $\Gamma_2 \Vdash s : \mathcal{M}$
- $\Gamma = \Gamma_1 + \Gamma_2$

 Gardner (1994), de Carvalho (2007)

Lemma 3 (Typability of head normal forms)

If t is a head normal form, then t is typable.

Proof. Since t is a head normal form, it is of the form:

$$t = \lambda x_1 \ldots x_n . \, y \, t_1 \ldots t_m$$

Let $A = \underbrace{[\,] \to \ldots \to [\,]}_{m \text{ times}} \to \alpha$.

Regardless of the shapes of $t_1, \ldots, t_m$:

$$y : [A] \vdash y \, t_1 \ldots t_m : \alpha$$

We consider two cases:

1. If $y \notin \{x_1, \ldots, x_n\}$, then:

$$y : [A] \vdash \lambda x_1 \ldots x_n . \, y \, t_1 \ldots t_m : \underbrace{[\,] \to \ldots \to [\,]}_{n \text{ times}} \to \alpha$$

2. If $y = x_i$ for some $1 \leq i \leq n$, then:

$$\vdash \lambda x_1 \ldots x_n . \, x_i \, t_1 \ldots t_m : \underbrace{[\,] \to \ldots \to [\,]}_{i-1 \text{ times}} \to [A] \to \underbrace{[\,] \to \ldots \to [\,]}_{n-i \text{ times}} \to \alpha$$

Summary of proof technique

Substitution Lemma
t, s typable $\Longrightarrow$ $t\{x := s\}$ typable
size is preserved

Anti-Substitution Lemma
$t\{x := s\}$ typable $\Longrightarrow$ t, s typable

Weighted Subject Reduction
$t \to_{\mathtt{h}} s$, t typable $\Longrightarrow$ s typable
size decreases

Subject Expansion
$t \to_{\mathtt{h}} s$, s typable $\Longrightarrow$ t typable

Typability of HNFs
t HNF $\Longrightarrow$ t typable

Soundness
t typable $\Longrightarrow$ t head normalising

Completeness
t head normalising $\Longrightarrow$ t typable

The same techniques are extended to other systems:

`Cf. Mazza, Pellissier & Vial (POPL'18)`

typable in System $\mathcal{H}$	$\rightsquigarrow$	typable in System X
head normalising	$\rightsquigarrow$	$\to_X$ normalising
head normal form	$\rightsquigarrow$	$\to_X$ normal form

$\vdots$

Head normalisation

Remark

Subject reduction and expansion hold for **arbitrary** reduction steps.

$$\text{Let } t \to_\beta s. \text{ Then } \Gamma \vdash t : A \text{ if and only if } \Gamma \vdash s : A.$$

(Only slightly revising the proofs).

Corollary (Head normalisation)

If $t \to_\beta^* s \in \mathrm{HNF}$ then there exists $s' \in \mathrm{HNF}$ such that $t \to_{\mathtt{h}}^* s'$.

Remark

Weighted subject reduction does not hold for arbitrary reduction steps.

Subject reduction may yield a derivation of the same size when the reduction occurs in an **untyped** subterm:

$$\dfrac{\dfrac{}{x : [\,] \to A \vdash x : [\,] \to A}\,\mathtt{var}}{x : [\,] \to A \vdash x\, t : A}\,\mathtt{app} \rightsquigarrow \dfrac{\dfrac{}{x : [\,] \to A \vdash x : [\,] \to A}\,\mathtt{var}}{x : [\,] \to A \vdash x\, s : A}\,\mathtt{app}$$

Quantitative upper bounds

The Weighted Subject Reduction lemma ensures that the size of the typing derivation decreases after each head reduction step.

Theorem (Upper bounds for reduction lengths)

Let $D \triangleright \Gamma \vdash t : A$ in System $\mathcal{H}$ and let $t \rightarrow_{\mathtt{h}}^{*} s \in \mathsf{HNF}$.
Then the number of steps in the reduction is at most $\mathtt{sz}(D)$.

Weakly and strongly normalising terms

We consider now full β-reduction, closed by arbitrary contexts:

$$(\lambda x.\, t)\, s \to_\beta t\{x := s\}$$

Definition

t is **weakly normalising** (WN) $\stackrel{\text{def}}{\iff}$ $\exists s.\ t \to_\beta^* s \in \mathsf{NF}$.

Definition

t is **strongly normalising** (SN) if there is no infinite sequence

$$t \to_\beta t_1 \to_\beta t_2 \ldots$$

Remark

- $t \in \mathsf{SN} \implies t \in \mathsf{WN}$ (nonconstructively)
- The converse does not hold; *e.g.* $(\lambda x.\, y)\, \Omega \in \mathsf{WN} \setminus \mathsf{SN}$.

We shall characterise WN and SN using quantitative type systems.

For a survey, see Bucciarelli, Kesner & Ventura (2017).

Normal forms

Theorem (Characterisation of normal forms)

A term is a $\to_\beta$-normal form if and only if $t \in \mathsf{NF}$ can be derived using the following inductive rule:

$$\frac{t_1 \in \mathsf{NF} \qquad \dots \qquad t_m \in \mathsf{NF} \qquad (n, m \geq 0)}{\lambda x_1 \dots x_n.\, y\; t_1\; \dots\; t_m \in \mathsf{NF}}$$

Characterising weak normalisation

Goal

Characterise weak normalisation (t typable in $\mathcal{W}$ iff $t \in \mathsf{WN}$).

System $\mathcal{W}$ has the **same** grammar of types and rules as System $\mathcal{H}$:

$$\begin{array}{ll} \textsc{Types} & A, B, \ldots ::= \alpha \mid \mathcal{M} \to A \\ \textsc{Multi-types} & \mathcal{M}, \mathcal{N}, \ldots ::= [A_i]_{i \in I} \end{array}$$

$$\frac{}{x : [A] \vdash x : A}\,\texttt{var} \qquad \frac{\Gamma, x : \mathcal{M} \vdash t : A}{\Gamma \vdash \lambda x.\, t : \mathcal{M} \to A}\,\texttt{lam}$$

$$\frac{\Gamma \vdash t : \mathcal{M} \to A \qquad \Delta \Vdash s : \mathcal{M}}{\Gamma + \Delta \vdash t\, s : A}\,\texttt{app} \qquad \frac{\Gamma_1 \vdash t : A_1 \qquad \ldots \qquad \Gamma_n \vdash t : A_n}{\Gamma_1 + \ldots + \Gamma_n \Vdash t : [A_1, \ldots, A_n]}\,\texttt{many}$$

Problem: $x\,\Omega$ is typable but not WN.

We need to impose further conditions on the derivation.

Characterising weak normalisation

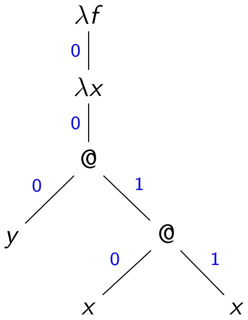
- We know:
  Any **head** reduction step decreases the size of the typing derivation.
  We have used this to show that typable terms are head normalising.

- But **arbitrary** reduction steps do not decrease the size.
  - $x\,\Omega$ is typable under the context $x : [\,] \to \alpha$.
  - The typing derivation does not give any type to Ω.

- To show that typable terms are weakly normalising, we generalize Weighted Subject Reduction for **typed** reduction steps.

Characterising weak normalisation

Positions inside a λ-term can be identified using strings $p \in \{0, 1\}^*$:



Characterising weak normalisation

Definition (Typed positions)

Let $D \triangleright \Gamma \vdash t : A$.

The set of **typed positions** of D is a subset of the positions of t.

$$\mathtt{TP}\left(\frac{}{\_ \vdash x : \_}\right) \quad := \quad \{\epsilon\}$$

$$\mathtt{TP}\left(\frac{D' \triangleright \_ \vdash t : \_}{\_ \vdash \lambda x. t : \_}\right) \quad := \quad \{\epsilon\} \cup \{0 \cdot p \mid p \in \mathtt{TP}(D' \triangleright \_ \vdash t : \_)\}$$

$$\mathtt{TP}\left(\frac{D_1 \triangleright \_ \vdash t : \_ \qquad D_2 \triangleright \_ \Vdash s : \_}{\_ \vdash t\, s : \_}\right) \quad := \quad \{\epsilon\} \cup \{0 \cdot p \mid p \in \mathtt{TP}(D_1 \triangleright \_ \vdash t : \_)\} \cup \{1 \cdot p \mid p \in \mathtt{TP}(D_2 \triangleright \_ \Vdash s : \_)\}$$

$$\mathtt{TP}\left(\frac{(D_i \triangleright \_ \vdash t : \_)_{i \in \{1,\dots,n\}}}{\Vdash t : }\mathtt{many}\right) \quad := \quad \bigcup_{i=1}^{n} \mathtt{TP}(D_i \triangleright \_ \vdash t : \_)$$

Characterising weak normalisation

Lemma (Weighted Subject Reduction in System $\mathcal{W}$)

Let $t \rightarrow_\beta s$ be a step contracting a redex at position p.

If $D \triangleright \Gamma \vdash t : A$, then there exists $D' \triangleright \Gamma \vdash s : A$.

Moreover:

- If $p \in \mathtt{TP}(D)$, then $\mathtt{sz}(D) > \mathtt{sz}(D')$.
- If $p \notin \mathtt{TP}(D)$, then $\mathtt{sz}(D) = \mathtt{sz}(D')$.

A redex is **typed** w.r.t. D if it occurs at a position $p \in \mathtt{TP}(D)$.

Corollary

Reduction of typed redexes terminates in a term without typed redexes.

But are terms without typed redexes in normal form?

Characterising weak normalisation

Problem

Terms without typed redexes are not always in normal form.

Again, this is because there are derivations like:

$$\dfrac{\dfrac{}{x : [\,] \to \alpha \vdash x : [\,] \to \alpha}\,\texttt{var}}{x : [\,] \to \alpha \vdash x\,\Omega : \alpha}\,\texttt{app}$$

$x\,\Omega$ has no typed redexes w.r.t. D, but is not a normal form.

First (failed) attempt to address the problem

Forbid the empty multiset altogether.

Characterising weak normalisation

Problem

If we forbid the empty multiset... some WN terms became untypable.

For example, to type $(\lambda x.\, y)\, \Omega$ we are forced to use $[\,]$:

$$\dfrac{\dfrac{\dfrac{}{y : [\alpha] \vdash y : \alpha}\,\texttt{var}}{y : [\alpha] \vdash \lambda x.\, y : [\,] \to \alpha}\,\texttt{lam}}{y : [\alpha] \vdash (\lambda x.\, y)\, \Omega : \alpha}\,\texttt{app}$$

All WN terms that are not SN become untypable.

In general, terms containing erasing subterms become untypable.

(In fact, this restriction allows typing *all and only* terminating λI terms).

Characterising weak normalisation

Second (successful) attempt to address the problem

Some occurrences of $[\,]$ are good, some are evil:

$$\dfrac{\dfrac{\overline{y : [\alpha] \vdash y : \alpha}\ \texttt{var}}{y : [\alpha] \vdash \lambda x.\, y : [\,] \to \alpha}\ \texttt{lam}}{y : [\alpha] \vdash (\lambda x.\, y)\,\Omega : \alpha}\ \texttt{app}
\qquad
\dfrac{\overline{x : [\,] \to \alpha \vdash x : [\,] \to \alpha}\ \texttt{var}}{x : [\,] \to \alpha \vdash x\,\Omega : \alpha}\ \texttt{app}$$

Negative occurrences are good. Positive occurrences are evil.

An occurrence of $[\,]$ is:

negative	if it is an	odd	number of times to the left of $\to / \vdash$
positive	if it is an	even	number of times to the left of $\to / \vdash$

- $[\,] \to \alpha$ should be allowed:
  $\mathcal{A}$ can choose to erase a non-terminating argument provided by $\mathcal{B}$.
- $[[\,] \to \alpha] \to \beta$ should be forbidden:
  $\mathcal{A}$ cannot assume that $\mathcal{B}$ will erase a non-terminating argument.

Characterising weak normalisation

Definition (Good typing derivations)

A typing derivation $D \triangleright \Gamma \vdash t : A$ is good if there are **no** positive occurrences of $[\,]$ in $(\Gamma \vdash A)$.

Theorem (System $\mathcal{W}$ characterises weak normalisation)

The following are equivalent:

1. t is typable in System $\mathcal{W}$ with a good derivation.
2. t is weakly normalising.

Theorem (Upper bounds for l.o. reduction lengths)

Let $D \triangleright \Gamma \vdash t : A$ be good. Then the number of steps in the *leftmost-outermost* (l.o.) reduction $t \to_\beta{}^* s \in \mathrm{NF}$ is at most $\mathtt{sz}(D)$.

Characterising strong normalisation

Goal

Characterise strong normalisation (t typable in System $\mathcal{S}$ iff $t \in \mathsf{SN}$).

Problem

- Functions that erase their arguments (such as $\lambda x.\, y$) have types with the empty multiset as the domain (such as $[\,] \to \alpha$).
- System $\mathcal{W}$ does not type the argument t in an application $(\lambda x.\, y)\, t$.
- So t can be any term.
  For example, $(\lambda x.\, y)\, \Omega$ is typable in $\mathcal{W}$ but not SN.

Characterising strong normalisation

System $\mathcal{S}$ has the **same** grammar of types Systems $\mathcal{H}$ and $\mathcal{W}$.
The variable, abstraction and multi-typing ("`many`") rules are as before.

Application is split into two rules, for erasing and non-erasing functions:

$$\dfrac{\Gamma \vdash t : [\,] \to A \qquad \Delta \vdash s : B}{\Gamma + \Delta \vdash t\,s : A}\,\texttt{app0}$$

$$\dfrac{\Gamma \vdash t : \mathcal{M} \to A \qquad \Delta \Vdash s : \mathcal{M}}{\Gamma + \Delta \vdash t\,s : A}\,\texttt{app}$$

$$\dfrac{\Gamma \vdash t : \mathcal{M} \to A \qquad \Delta \Vdash s : \mathcal{M} \qquad |\mathcal{M}| > 0}{\Gamma + \Delta \vdash t\,s : A}\,\texttt{app+}$$

- The erasing rule types the argument even though it is not used.
- The non-erasing rule is as before, but requires that the function has non-empty domain, *i.e.* that the argument is used.

Characterising strong normalisation

System $\mathcal{S}$ is designed in such a way that **all** redexes are in typed positions.

Lemma (Weighted Subject Reduction for System $\mathcal{S}$)

Let $t \to_\beta s$ be an **arbitrary** step.
If $D \rhd \Gamma \vdash t : A$ then there exists $D' \rhd \Gamma \vdash s : A$.
Moreover, $\mathtt{sz}(D) > \mathtt{sz}(D')$.

This entails soundness (typable terms are SN).

Characterising strong normalisation

For completeness (SN terms are typable), the situation is subtler:

Problem

Subject Expansion does not hold for erasing steps.

- Consider an erasing step $(\lambda x.\, t)\, s \to t\{x := s\}$.
- Suppose that $t\{x := s\}$ is typable.
- If the step is erasing (*i.e.* $x \notin \mathsf{fv}(t)$) then $t\{x := s\} = t$ is typable. But we know nothing about s.
- To type $(\lambda x.\, t)\, s$ in System $\mathcal{S}$ we require s to be typable.

An example is:

$$\underbrace{(\lambda x.\, y)\, \Omega}_{\text{untypable}} \to \underbrace{y}_{\text{typable}}$$

The following weaker variant of the lemma holds:

Lemma (Subject Expansion for for System $\mathcal{S}$)

If $t \to_\beta s$ is non-erasing and $\Gamma \vdash s : A$ then $\Gamma \vdash t : A$.

Characterising strong normalisation

However, an SN term such as $(\lambda x.\, y)\, z$ may contain erasing redexes.

To show that all SN terms are typable,
proceed by induction on the inductive characterisation of SN terms:

$$\frac{t_1 \in \mathsf{SN} \qquad \ldots \qquad t_n \in \mathsf{SN}}{x\, t_1 \ldots t_n \in \mathsf{SN}}$$

$$\frac{t \in \mathsf{SN}}{\lambda x.\, t \in \mathsf{SN}}$$

$$\frac{t \in \mathsf{SN} \qquad s \in \mathsf{SN} \qquad u_1 \in \mathsf{SN} \qquad \ldots \qquad u_n \in \mathsf{SN} \qquad t\{x := s\}\, u_1 \ldots u_n \in \mathsf{SN}}{(\lambda x.\, t)\, s\, u_1 \ldots u_n}$$

The key rule is the last one: when the step is erasing, by IH we know that the erased argument is typable.

Characterising strong normalisation

Theorem (System $\mathcal{S}$ characterises strong normalisation)

The following are equivalent:

1. t is typable in System $\mathcal{S}$.
2. t is strongly normalising.

Theorem (Upper bounds for arbitrary reduction lengths)

Let $D \triangleright \Gamma \vdash t : A$ in System $\mathcal{S}$. Then the number of steps in **any** reduction $t \to_{\beta}{}^{*} s \in \mathsf{NF}$ is at most $\mathtt{sz}(D)$.

Exact bounds

Accattoli, Graham-Lengrand, Kesner (2018)

The systems seen so far provide **upper bounds** for reduction lengths.

Can they be adapted to obtain the **exact** length of reductions?

For simplicity, here we discuss head reduction (System $\mathcal{H}$) only.

Erasure, persistence and consumption

In a reduction of a term t to head normal form:

- Some fragments of t are erased (or **potentially** erased):

$$(\lambda x.\, y)\, \Omega \to y \qquad (\lambda y.\, y\, \Omega)\, x \to x\, \Omega$$

- Some fragments of t are persistent, becoming the head of the HNF:

$$(\lambda w.\, (\lambda x.\, (\lambda y.\, y)\, w))\, z \to \lambda x.\, (\lambda y.\, y)\, z \to \lambda x.\, z$$

- Some fragments of t are consumed, forming contracted β-redexes:

$$\lambda x.\, ((\lambda w.\, w) \mathbin{@} (\lambda y.\, y)) \mathbin{@} z \to \lambda x.\, (\lambda y.\, y) \mathbin{@} z \to \lambda x.\, z$$

Exact bounds

Accattoli, Graham-Lengrand, Kesner (2018)

Each typing rule corresponds to an erased, a persistent or a consumed fragment.

$$(\lambda x.\, x \mathbin{@} y) \mathbin{@} (\lambda z.\, \lambda w.\, z \mathbin{@} w) \to (\lambda z.\, \lambda w.\, z \mathbin{@} w) \mathbin{@} y \to \lambda w.\, y \mathbin{@} w$$

- This example is easy because there is no duplication.
- In general, a subterm may have many descendants along a reduction (some erased, some persistent, some consumed).
- Typing derivations record this by giving many types to each term. (Intuitively: one per each descendant).

The idea of a **tight** quantitative type system is to:

- Enforce the condition that erased terms are not typed.
- Distinguish between persistent and consuming typing rules.
- Each persistent typing rule accounts for the size of the normal form.
- Each consuming typing rule accounts for a reduction step.

Exact bounds

Accattoli, Graham-Lengrand, Kesner (2018)

System $\mathcal{H}_{\texttt{tight}}$ extends types with persistent constants:

$$
\begin{array}{lrcl}
\textsc{Persistent Types} & \mathtt{p}, \mathtt{q}, \ldots & ::= & \mathtt{neutral} \mid \mathtt{lambda} \\
\textsc{Types} & A, B, \ldots & ::= & \alpha \mid \mathtt{p} \mid \mathcal{M} \rightarrow A \\
\textsc{Multi-types} & \mathcal{M}, \mathcal{N}, \ldots & ::= & [A_i]_{i \in I} \\
\textsc{Persistent Multi-types} & \mathcal{P} & ::= & [\mathtt{p}_i]_{i \in I}
\end{array}
$$

Abstraction and application have persistent and consuming forms:

$$
\frac{}{x : [A] \vdash x : A}\,\texttt{var}
$$

$$
\frac{\Gamma, x : \mathcal{P} \vdash t : \mathtt{q}}{\Gamma \vdash \lambda x.\, t : \mathtt{lambda}}\,\texttt{lamP}
\qquad
\frac{\Gamma, x : \mathcal{M} \vdash t : A}{\Gamma \vdash \lambda x.\, t : \mathcal{M} \rightarrow A}\,\texttt{lamC}
$$

$$
\frac{\Gamma \vdash t : \mathtt{neutral}}{\Gamma \vdash t\, s : \mathtt{neutral}}\,\texttt{appP}
\qquad
\frac{\Gamma \vdash t : \mathcal{M} \rightarrow A \qquad \Gamma \Vdash s : \mathcal{M}}{\Gamma \vdash t\, s : A}\,\texttt{appC}
$$

$$
\frac{\Gamma_1 \vdash t : A_1 \qquad \ldots \qquad \Gamma_n \vdash t : A_n}{\Gamma_1 + \ldots + \Gamma_n \Vdash t : [A_1, \ldots, A_n]}\,\texttt{many}
$$

Exact bounds

Accattoli, Graham-Lengrand, Kesner (2018)

Definition (Tightness)

A derivation $D \triangleright \Gamma \vdash t : A$ is **tight** if A is a persistent type, and Γ maps all variables to persistent multi-types.

Theorem (System $\mathcal{H}_{\texttt{tight}}$ characterises head normalisation)

The following are equivalent:

1. t is typable in System $\mathcal{H}_{\texttt{tight}}$ with a **tight** derivation.
2. t is head normalising.

Theorem (Exact bounds)

Let $D \triangleright \Gamma \vdash t : A$ be a **tight** derivation in System $\mathcal{H}_{\texttt{tight}}$.
Let $t \rightarrow_{\texttt{h}}^{*} s \in \text{HNF}$. Then:

- The length of the reduction is **exactly** $\frac{1}{2} \cdot \#_{\texttt{consuming}}(D)$.
- The size of the HNF of t is **exactly** $\#_{\texttt{persistent}}(D)$.

Call-by-value reduction

Call-by-name

Reduction rule	$(\lambda x.\, t)\, s \to_{\mathtt{cbn}} t\{x := s\}$
Evaluation contexts	$\mathtt{E}_{\mathtt{cbn}} ::= \Box \mid \mathtt{E}_{\mathtt{cbn}}\, t$

Call-by-value Plotkin (1975)

Values	$v ::= \lambda x.\, t$
Reduction rule	$(\lambda x.\, t)\, v \to_{\mathtt{cbv}} t\{x := v\}$
Evaluation contexts	$\mathtt{E}_{\mathtt{cbv}} ::= \Box \mid \mathtt{E}_{\mathtt{cbv}}\, t \mid v\, \mathtt{E}_{\mathtt{cbv}}$

CBN is also called weak head reduction.
Termination in CBN can be characterised with a variant of System $\mathcal{H}$.

Characterising call-by-value termination

CBN and CBV correspond to translations of intuitionistic logic into LL:

Girard (1987), Maraist *et al.* (1999)

$$(A \to B)^{\mathtt{CBN}} := !A^{\mathtt{CBN}} \multimap B^{\mathtt{CBN}} \qquad (A \to B)^{\mathtt{CBV}} := !A^{\mathtt{CBV}} \multimap !B^{\mathtt{CBV}}$$

Quantitative type systems are connected with these translations.

Specifically:

- Arrow types in CBN systems are of the form $\mathcal{M} \to A$.
- Perhaps...

  arrow types in CBV systems should be of the form $\mathcal{M} \to \mathcal{N}$.

Characterising call-by-value termination

System $\mathcal{V}$

Buciarelli *et al.* (2020), Accattoli *et al.* (2023)

Inspired by Ehrhard's relational model (2012).

$$\begin{array}{ll} \textsc{Types} & A, B, \ldots ::= \alpha \mid \mathcal{M} \to \mathcal{N} \\ \textsc{Multi-types} & \mathcal{M}, \mathcal{N}, \ldots ::= [A_i]_{i \in I} \end{array}$$

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\,\mathtt{var} \qquad \frac{(\Gamma_i, x : \mathcal{M}_i \vdash t : \mathcal{N}_i)_{i=1}^{n}}{\Gamma_1 + \ldots + \Gamma_n \vdash \lambda x.\, t : [\mathcal{M}_i \to \mathcal{N}_i]_{i=1}^{n}}\,\mathtt{lam}$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \to \mathcal{N}] \qquad \Delta \vdash s : \mathcal{M}}{\Gamma + \Delta \vdash t\, s : \mathcal{N}}\,\mathtt{app}$$

- Multitypes have a different semantics than in CBN systems.
- Multitypes are the **types of values**.
- An abstraction of type $[A_1, \ldots, A_n]$ will have n descendants that take part as the function in a contracted redex.
- Note that $\vdash x : []$ holds.

Characterising call-by-value termination

Example

Let us write $\mathbf{0} := [\,]$ and $\mathcal{M} = [\mathbf{0} \to \mathbf{0}]$.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{x : \mathcal{M} \vdash x : \mathcal{M}}
      \qquad
      \dfrac{}{\vdash x : \mathbf{0}}
    }{x : \mathcal{M} \vdash x\,x : \mathbf{0}}
  }{\vdash \lambda x.\, x\,x : [\mathcal{M} \to \mathbf{0}]}
  \qquad
  \dfrac{
    \dfrac{\dfrac{}{x : \mathcal{M} \vdash x : \mathcal{M}}}{\vdash \mathtt{id} : [\mathcal{M} \to \mathcal{M}]}
    \qquad
    \dfrac{\dfrac{}{\vdash x : \mathbf{0}}}{\vdash \mathtt{id} : \mathcal{M}}
  }{\vdash \mathtt{id}\,\mathtt{id} : \mathcal{M}}
}{\vdash (\lambda x.\, x\,x)\,(\mathtt{id}\,\mathtt{id}) : \mathbf{0}}
$$

$$\rightsquigarrow$$

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{x : \mathcal{M} \vdash x : \mathcal{M}}
      \qquad
      \dfrac{}{\vdash x : \mathbf{0}}
    }{x : \mathcal{M} \vdash x\,x : \mathbf{0}}
  }{\vdash \lambda x.\, x\,x : [\mathcal{M} \to \mathbf{0}]}
  \qquad
  \dfrac{\dfrac{}{\vdash x : \mathbf{0}}}{\vdash \mathtt{id} : \mathcal{M}}
}{\vdash (\lambda x.\, x\,x)\,\mathtt{id} : \mathbf{0}}
$$

Characterising call-by-value termination

Theorem (System $\mathcal{V}$ characterises CBV termination)

Let t be a closed term. The following are equivalent:

1. t is typable in System $\mathcal{V}$.
2. t reduces to an abstraction in CBV.

Theorem (Upper bounds for CBV reduction)

Let t be a closed term and let $D \rhd \Gamma \vdash t : A$ in System $\mathcal{V}$. Then the number of steps in a reduction $t \rightarrow^*_{\mathtt{cbv}} \lambda x.\, s$ is at most $\mathtt{sz}(D)$.

- Preservation of strong normalisation in ES calculi.
- Completeness of evaluation strategies.
- Genericity and observational equivalence.

Explicit substitutions and PSN

The β-reduction rule is very high level:

$$(\lambda x.\, t)\, s \to_\beta t\{x := s\}$$

Typically, implementations of functional programming languages:

- Are not based on meta-level substitution $t\{x := s\}$.
- They are based on **environments** that bind variables to expressions.

Some calculi incorporate **explicit substitutions** (ESs):

Abadi, Cardelli, Curien, Lévy (1996)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid t[x := s]$$

- ESs implement fine-grained substitution.
- They can be seen as environments in **abstract machines**.
- Many calculi with ESs appeared in the 1990s and 2000s.

(Lescanne *et al.* , Rose, Ríos *et al.* , Kesner, ...)

Explicit substitutions and PSN

In a calculus $\mathcal{L}$ with ESs, typically $\to_{\mathcal{L}} = \to_{\mathtt{beta}} \cup \to_{\mathtt{subst}}$, where:

$$\begin{array}{lll} (\lambda x.\, t)\, s & \to_{\mathtt{beta}} & t[x := s] \\ t[x := s] & \to^{*}_{\mathtt{subst}} & t\{x := s\} \end{array}$$

Preservation of Strong Normalization (PSN)

A calculus $\mathcal{L}$ with ESs enjoys PSN if and only if:

for every pure term t (without ESs):

if $t \in \mathsf{SN}(\to_{\beta})$

then $t \in \mathsf{SN}(\to_{\mathcal{L}})$.

Trickier than it may seem

For example, Rose's $\lambda_{\mathtt{x}}$ requires the following rule to be confluent:

$$t[x := s][y := u] \;\to_{\mathtt{x}}\; t[y := u][x := s[y := u]]$$

But the RHS is again an instance of the LHS, breaking SN.

Explicit substitutions and PSN

Showing that a calculus $\mathcal{L}$ with ESs enjoys PSN is difficult.
In the 1990s and 2000s, this was done using heavy rewriting techniques.

Proof strategy (PSN via intersection types)

Design a type system $\mathcal{T}$ such that:

1. Weighted subject reduction holds for arbitrary $\rightarrow_{\mathcal{L}}$-steps.
2. Subject expansion holds for arbitrary $\rightarrow_{\beta}$-steps.
3. $\rightarrow_{\beta}$-normal forms are typable.

Then:

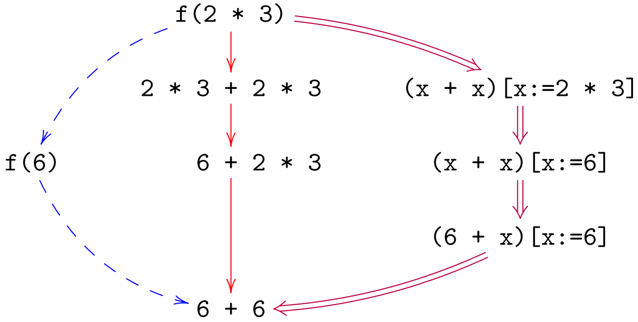
- Suppose that $t \in \mathsf{SN}(\rightarrow_{\beta})$.
- By 2. and 3., t is typable in System $\mathcal{T}$.
- By 1., $t \in \mathsf{SN}(\rightarrow_{\mathcal{L}})$.

This can be used to show that various ES calculi enjoy PSN.

`Cf. Kesner & Conchúir (2023)`

Call-by-need and completeness of strategies

Let `f(x) = x + x`. Then:



$\dashrightarrow$ = **call-by-value** $\longrightarrow$ = **call-by-name** $\Longrightarrow$ = **call-by-need**

Wadsworth (1971)

Call-by-need and completeness of strategies

Call-by-need and completeness of strategies

Using explicit substitutions, a CBNeed strategy may be defined.

Ariola *et al.* (1995), Accattoli *et al.* (2014)

$$
\begin{array}{lrcl}
\textsc{Terms} & t, s, \ldots & ::= & x \mid \lambda x.\, t \mid t\, s \mid t[x := s] \\
\textsc{Values} & v & ::= & \lambda x.\, t \\
\text{ES } \textsc{lists} & \mathtt{L} & ::= & \Box \mid \mathtt{L}[x := t] \\
\textsc{Ev. ctxs.} & \mathtt{E}_{\mathtt{cbnd}} & ::= & \Box \mid \mathtt{E}_{\mathtt{cbnd}}\, t \mid \mathtt{E}_{\mathtt{cbnd}}[x := t] \mid \mathtt{E}_{\mathtt{cbnd}}\langle x \rangle[x := \mathtt{E}_{\mathtt{cbnd}}]
\end{array}
$$

Reduction rules

$$(\lambda x.\, t)\mathtt{L}\, s \quad \to_{\mathtt{db}} \quad t[x := s]\mathtt{L}$$

$$\mathtt{E}_{\mathtt{cbnd}}\langle x \rangle[x := v\mathtt{L}] \quad \to_{\mathtt{lsv}} \quad \mathtt{E}_{\mathtt{cbnd}}\langle v \rangle[x := v]\mathtt{L}$$

Call-by-need and completeness of strategies

Example (Call-by-need reduction)

Let $\texttt{id} = \lambda x.\ x$. Then:

$$\begin{array}{rcl}
(\lambda x.\ x\,x)(\texttt{id}\,\texttt{id}) & \to & (x\,x)[x := \texttt{id}\,\texttt{id}] \\
& \to & (x\,x)[x := y[y := \texttt{id}]] \\
& \to & (x\,x)[x := \texttt{id}[y := \texttt{id}]] \\
& \to & (\texttt{id}\,x)[x := \texttt{id}][y := \texttt{id}] \\
& \to & z[z := x][x := \texttt{id}][y := \texttt{id}] \\
& \to & z[z := \texttt{id}][x := \texttt{id}][y := \texttt{id}] \\
& \to & \texttt{id}[z := \texttt{id}][x := \texttt{id}][y := \texttt{id}]
\end{array}$$

Call-by-need and completeness of strategies

CBV is not complete with respect to CBN:

- $(\lambda x.\, y)\,\Omega$ terminates in CBN but not in CBV.

Theorem (Completeness of CBNd) Ariola *et al.* (1995)

If t terminates in CBN then it terminates in CBNd.

Proof. Very hard, using rewriting techniques.

Much simpler proof, via intersection types Kesner (2014)

If t terminates in CBN then it terminates in CBNd.

Proof.

- Let t be CBN-terminating.
- Then t is typable in a variant of System $\mathcal{H}$.
- Then t is CBNd-terminating.

These results have been extended to **strong** CBNd.

With Balabonski *et al.* (2017), Bonelli *et al.* (2018)

Genericity and observational equivalence

- Let $\mathcal{E}$ be an equational theory over terms.
  (*e.g.* λ-terms with β-convertibility).
- Let $\mathcal{T}$ be a set of **testing contexts**.
  (*e.g.* head contexts $\lambda x_1 \dots x_n. \Box\, t_1 \dots t_n$).
- A term t is **meaningful** iff $\forall s. \exists \mathtt{C} \in \mathcal{T}.\ \mathtt{C}\langle t\rangle \equiv_{\mathcal{E}} s$.
  (*e.g.* solvable terms).

  A term is **meaningless** iff it is not meaningful.
- $\mathcal{E}$ enjoys **genericity** w.r.t. $\mathcal{T}$ iff, for an arbitrary context $\mathtt{C}$:

  if $\mathtt{C}\langle t\rangle$ is meaningful for some meaningless t
  then $\mathtt{C}\langle t'\rangle$ is meaningful for every t'.

Genericity and observational equivalence

Proof strategy (Genericity via intersection types)

Design a type system $\mathcal{T}$ such that:

- A term is typable if and only if it is meaningful.

  Intuitively, meaningful terms are head normalising terms.

Then:

- Suppose that t is meaningless, hence untypable.
- Suppose, moreover, that $\mathtt{C}\langle t\rangle$ is meaningful, hence typable.
- The subterm t in the typing derivation for $\mathtt{C}\langle t\rangle$ must necessarily occur in an untyped position.
- Then the same typing derivation also types $\mathtt{C}\langle s\rangle$.
- Hence $\mathtt{C}\langle s\rangle$ is meaningful.

Recently used to study genericity in CBN and CBV.
It has been refined to a **quantitative** notion of genericity.

Accattoli & Guerrieri (2022)

Arrial, Guerrieri & Kesner (2024)

Genericity and observational equivalence

- Let $\mathcal{E}$ be an equational theory between terms.
- Define observational equivalence $t \approx_{\mathcal{E}} s$ as follows:

$$\forall \mathtt{C}. \quad (\mathtt{C}\langle t \rangle \text{ is meaningful} \quad \text{if and only if} \quad \mathtt{C}\langle s \rangle \text{ is meaningful})$$

- Two theories $\mathcal{E}_1, \mathcal{E}_2$ are observational equivalent iff $\approx_{\mathcal{E}_1} = \approx_{\mathcal{E}_2}$.

Proof strategy (Observational equiv. via intersection types)

Design a type system $\mathcal{T}$ such that:

- A term is typable if and only if it is $\mathcal{E}_1$-meaningful.
- A term is typable if and only if it is $\mathcal{E}_2$-meaningful.

Then it is immediate that $\approx_{\mathcal{E}_1} = \approx_{\mathcal{E}_2}$.

Other applications and extensions (recent)

- Extension to classical systems ($\lambda\mu$).

  Kesner & Vial

- Extension to open and strong strategies.

  Accattoli, Guerrieri & Leberle; B., Kesner & Milicich

- Extension to call-by-push-value.

  Bucciarelli, Kesner, Ríos, Viso

- Intersection types as a "big-step semantics".

  Bernadet & Graham-Lengrand; Bonelli, B. & Milicich

- Factorisation of reduction graphs.

  B. & Ciruelos

- Relationship with simple types.

  Pautasso & Ronchi Della Rocca

- Inhabitation problem.

  Bucciarelli, Kesner, Ronchi Della Rocca

- Interaction equivalence.

  Accattoli, Lancelot, Manzonetto, Vanoni

- ...