Lambda calculus

Part II
Lambda Calculi with Types

Based on materials provided by H. Barendregt

Types

are certain objects, usually syntactic expressions (*e.g. boolean, integer, Char*), that may be assigned to terms denoting programs.

Types serve to classify the (objects denoted by the) terms.

Semantics.

Each type σ has as semantics a set D_σ of „objects of type σ". There are several systems of type assignment with different collections of types.

For more complicated type systems the semantics D_σ will in general be not a set, but an object in some category.

Type assignment is done for the following reasons.

Firstly, the type of a term F gives *partial specification* of what the function F is supposed to do. Usually specification of this type is given before the term as program is constructed.

Once this term has been constructed, the verification whether this term is indeed of the required type provides *partial correctness* proof for the program.

Secondly, types play a role in *efficiency*. If it is known that a subterm S of a program has a certain type, then S may be executed more efficiently by making use of the type information.

To explain the idea of type assignment, we present type systems of various strengths.

We start with the system $\lambda\!\!\rightarrow$ of simply typed lambda calculus. We shall distinguish between typing *a la Curry* and *a la Church* by introducing $\lambda\!\!\rightarrow$ in both ways.

Several other systems of typed lambda calculus exist in a Curry and a Church version. However it is not so for all systems.

For example, for the Curry system $\lambda\!\!\cap$ of intersection types it is not clear how to define its Church version and for the Church system λC (calculus of constructions) it is not clear how to define a Curry version.

For the systems that exist in both styles there is a clear relation.

The system $\lambda\to$ -Curry

is assigning elements of a given set **T** of types to type free lambda terms. For this reason the calculi *a la* Curry are sometimes called *systems of type assignement.*

The system $\lambda\to$ -Curry consists of

(i) the set of *types* of $\lambda\to$, notation Type($\lambda\to$). We write **T**= Type($\lambda\to$) for short.

(ii) the finite set of rules.

We shall start with a lot of definitions.

Definition.(The set of types of $\lambda\to$)

The set of types **T**=Type($\lambda\to$) is defined inductively,

$\alpha,\alpha',\alpha'',\ldots\in \mathbf{T}$ (type variables)

$\sigma,\tau\in \mathbf{T}\Rightarrow(\sigma\to\tau)\in \mathbf{T}$ (function space types)

or in abstract syntax

$$\mathbf{T}=\mathbf{V}\mid\mathbf{T}\to\mathbf{T}$$

where **V** is defined by

$\mathbf{V}=\alpha\mid\mathbf{V}'$ (type variables)

Notation.

(i) If $\sigma_1,\ldots,\sigma_n\in\mathbf{T}$ then

$$\sigma_1\to\sigma_2\to\ldots\to\sigma_n$$

stands for

$$(\sigma_1\to(\sigma_2\to\ldots\to(\sigma_{n-1}\to\sigma_n)\ldots)),$$

hence , we use association to the right.

(ii) $\alpha,\beta,\gamma,\ldots$ denote arbitrary type variables.

Definition ($\lambda\to$-Curry).

A statement $M{:}\sigma$ is *derivable from a basis* Γ, notation

$$\Gamma\mid-_{\lambda\to\text{-}Curry}M:\sigma$$

(or

$$\Gamma\mid-_{\lambda\to}M:\sigma$$

or

$$\Gamma\mid-M:\sigma$$

if there is no danger of confusion) if $\Gamma\mid-M:\sigma$ can be produced by the folowing rules

$\lambda\!\rightarrow$-Curry (version 0)

$$(x:\sigma)\in\Gamma \Rightarrow \Gamma \mid\!- x:\sigma$$

$$\Gamma \mid\!- M:(\sigma\rightarrow\tau), \Gamma \mid\!- N:\sigma \Rightarrow \Gamma \mid\!- (MN):\tau$$

$$\Gamma, x:\sigma \mid\!- M:\tau \Rightarrow \Gamma \mid\!- (\lambda x.M):(\sigma\rightarrow\tau)$$

Here $\Gamma, x:\sigma$ stands for $\Gamma\cup\{x:\sigma\}$ and $x\notin Dom(\Gamma)$ in order to $\Gamma\cup\{x:\sigma\}$ be a basis.

If $\Gamma=\{x_1:\sigma_1,\ldots,x_n:\sigma_n\}$ we can write instead of $\Gamma \mid\!- M:\sigma$ $x_1:\sigma_1,\ldots,x_n:\sigma_n \mid\!- M:\sigma$. If Γ is empty, we write $\mid\!- M:\sigma$.

We pronounce |- as "yields" or „is derivable".

The rules given in Version 0 are usually expressed as follows.

$\lambda\!\rightarrow$-Curry (Version 1)

(axiom) $\quad \Gamma \mid\!- (x:\sigma)$ if $(x:\sigma)\in\Gamma$

(→-elimination) $\quad \dfrac{\Gamma \mid\!- M:(\sigma\rightarrow\tau) \quad \Gamma \mid\!- N:\sigma}{\Gamma \mid\!- (MN):\tau}$

(→-introduction) $\quad \dfrac{\Gamma, x:\sigma \mid\!- M:\tau}{\Gamma \mid\!- (\lambda x.M):(\sigma\rightarrow\tau)}$

The following is the natural deduction formulation

$\lambda\!\rightarrow$-Curry (Version 2)

Elimination rule	Introduction rule
	$\begin{array}{c}[x:\sigma]\\ \vdots\\ M:\tau\end{array}$
$\dfrac{M:(\sigma\rightarrow\tau) \quad N:\sigma}{(MN):\tau}$	$\dfrac{M:\tau}{(\lambda x.M):(\sigma\rightarrow\tau)}$

The basic axiom of Versions 0 and 1 is considered here as implicit and is not mentioned. The notation

$$\begin{array}{c}x:\sigma\\ \vdots\\ M:\tau\end{array}$$

Means that from the assumption $x:\sigma$ together with a set Γ of other statements, one can derive $M:\tau$.

The basic axiom of Versions 0 and 1 is considered here as implicit and is not mentioned. The notation

$$x : \sigma$$
$$\vdots$$
$$M : \tau$$

Means that from the assumption $x : \sigma$ together with a set Γ of other statements, one can derive $M : \tau$.

The rule of $\rightarrow$-introduction in the table states that

$$(\lambda x.M) : (\sigma \rightarrow \tau)$$

is derivable even without the assumption $x : \sigma$ but still using Γ. This process is called *cancellation* of an assumption and is indicated by the striking through the statement $\cancel{x : \sigma}$.

Example.

(a) Using Version 1 of the system the *derivation*

$$\dfrac{\dfrac{x : \sigma, y : \tau \mid\!- x : \sigma}{x : \sigma \mid\!- (\lambda y.x) : (\tau \rightarrow \sigma)}}{\mid\!- (\lambda xy.x) : (\sigma \rightarrow \tau \rightarrow \sigma)}$$

Shows that $\mid\!- (\lambda xy.x) : (\sigma \rightarrow \tau \rightarrow \sigma)$ for all $\sigma, \tau \in \mathbf{T}$.

(b) Using Version 2, a *natural deduction derivation* of the same type assignment is

$$\dfrac{\dfrac{\dfrac{\cancel{x : \sigma}^{2} \qquad \cancel{y : \tau}^{1}}{x : \sigma}}{(\lambda y.x) : (\tau \rightarrow \sigma)^{1}}}{(\lambda xy.x) : (\sigma \rightarrow \tau \rightarrow \sigma)^{2}}$$

The indices 1 and 2 are bookkeeping devices that indicate at which application of a rule a particular assumption is being cancelled.

Example.

(a) Using Version 1 of the system the *derivation*

$$\dfrac{\dfrac{x : \sigma, y : \tau \mid\!- x : \sigma}{x : \sigma \mid\!- (\lambda y.x) : (\tau \rightarrow \sigma)}}{\mid\!- (\lambda xy.x) : (\sigma \rightarrow \tau \rightarrow \sigma)}$$

Shows that $\mid\!- (\lambda xy.x) : (\sigma \rightarrow \tau \rightarrow \sigma)$ for all $\sigma, \tau \in \mathbf{T}$.

(c) For all $\sigma \in \mathbf{T}$ we have

$$\mid\!- (\lambda x.x) : (\sigma \rightarrow \sigma)$$

(c) For all $\sigma \in \mathbf{T}$ we have

$$\vdash (\lambda x.x) : (\sigma \to \sigma)$$

Indeed

$$\frac{x : \sigma \vdash x : \sigma}{\vdash (\lambda x.x) : (\sigma \to \sigma)}$$

(c) For all $\sigma \in \mathbf{T}$ we have

$$\vdash (\lambda x.x) : (\sigma \to \sigma)$$

Indeed

$$\frac{x : \sigma \vdash x : \sigma}{\vdash (\lambda x.x) : (\sigma \to \sigma)}$$

(d) $y : \sigma \vdash (\lambda x.x) y : \sigma$

(c) For all $\sigma \in \mathbf{T}$ we have

$$\vdash (\lambda x.x) : (\sigma \to \sigma)$$

Indeed

$$\frac{x : \sigma \vdash x : \sigma}{\vdash (\lambda x.x) : (\sigma \to \sigma)}$$

(d) $y : \sigma \vdash (\lambda x.x) y : \sigma$

It follows from (c) and Elimination rule that

$$\frac{\vdash (\lambda x.x) : (\sigma \to \sigma) \qquad y : \sigma \vdash (\lambda x.x) y : \sigma}{}$$

Properties of $\lambda \to$ -Curry

Several properties of type assignment in $\lambda \to$ are valid. First, we analyse how much of a basis is necessary in order to derive a type assignment.

Definition.

(i) Let $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ be a basis. We consider Γ as a partial function from the set of term variables to the set of types.

(ii) Then $\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$ is the domain of Γ and we write $\Gamma(x_i) = \sigma_i$ for the value of Γ i.e. The type which is assigned to the variable x_i.

(iii) Let V' be a set of term variables, the restriction of Γ to V' is defined as follows $\Gamma \mid V' = \{x : \sigma \mid x \in V' \,\&\, \sigma = \Gamma(x)\}$.

(iv) For types $\sigma,\tau \in \mathbf{T}$, and a type variable α, the substitution of τ for α in σ is denoted by $\sigma[\alpha := \tau]$.

Basis lemma for $\lambda\!\rightarrow$-Curry

Let Γ be a basis.

(i) If $\Gamma', \Gamma \subseteq \Gamma'$ is another basis, then $\Gamma \vdash M : \sigma \Rightarrow \Gamma' \vdash M : \sigma$.

(ii) $\Gamma \vdash M : \sigma \Rightarrow FV(M) \subseteq dom(\Gamma)$.

(iii) $\Gamma \vdash M : \sigma \Rightarrow \Gamma \mid FV(M) \vdash M : \sigma$.

Proof.

Since such proofs will occur frequently, we produce it in full only for the first statement in order to be briefer later on.

(i) We proceed by induction on the derivation of $\Gamma \vdash M : \sigma$.

Case 1. $M : \sigma$ is $x : \sigma$ and this declaration is an element of Γ. Then also $x : \sigma \in \Gamma'$ and thus $\Gamma' \vdash M : \sigma$.

Case 2. $M : \sigma$ is $(M_1 M_2) : \sigma$ and it follows directly from two assignments $M_1 : (\tau \rightarrow \sigma)$ and $M_2 : \tau$ for some τ. By the Induction Hypothesis one has $\Gamma' \vdash M_1 : (\tau \rightarrow \sigma)$ and $\Gamma' \vdash M_2 : \tau$. Thus $\Gamma' \vdash (M_1 M_2) : \sigma$.

Case 3. $M : \sigma$ is $(\lambda x.M_1) : (\sigma_1 \rightarrow \sigma_2)$ and it follows directly from $\Gamma, x : \sigma_1 \vdash M_1 : \sigma_2$. by the convention concerning bounded variables, one may assume that the variable x does not occur in the domain of Γ'. Therefore by the Induction Hypothesis one has $\Gamma', x : \sigma_1 \vdash M_1 : \sigma_2$ and thus

$$\Gamma' \vdash (\lambda x.M_1) : (\sigma_1 \rightarrow \sigma_2).$$

(ii) By induction on derivation of $M : \sigma$. We prove only the case that $M : \sigma$ is $(\lambda x.M_1) : (\sigma_1 \rightarrow \sigma_2)$ and follows directly from the assumption $\Gamma, x : \sigma_1 \vdash M_1 : \sigma_2$.
Let $y \in FV(\lambda x.M_1)$, then $y \in FV(M_1)$ and $y \not\equiv x$. By the Induction Hypothesis one has $y \in dom(\Gamma, x : \sigma_1)$ and hence $y \in dom\,\Gamma$.

(iii) By induction on the derivation of $M : \sigma$. We only treat the case that $M : \sigma$ is $(M_1 M_2) : \sigma$ and follows directly from

$$\Gamma \vdash M_1 : (\tau \rightarrow \sigma) \quad \text{and} \quad \Gamma \vdash M_2 : \tau$$

for some Γ, τ. By the Induction Hypothesis one has

$$\Gamma \mid FV(M_1) \vdash M_1 : (\tau \rightarrow \sigma) \quad \text{and} \quad \Gamma \mid FV(M_2) \vdash M_2 : \tau.$$

As $FV(M_1 M_2) = FV(M_1) \cup FV(M_2)$, by (i) one has that

$$\Gamma \mid FV(M_1 M_2) \vdash M_1 : (\tau \rightarrow \sigma) \quad \text{and} \quad \Gamma \mid FV(M_1 M_2) \vdash M_2 : \tau$$

and hence $\Gamma \mid FV(M_1 M_2) \vdash (M_1 M_2) : \sigma$.

Now we show how terms of a certain form get typed. It gives us new insight, among other things we can show that certain terms have no types.

Generation lemma for $\lambda\!\rightarrow$-Curry

(i) $\Gamma \mid\!- x : \sigma \Rightarrow (x : \sigma) \in \Gamma$

(ii) $\Gamma \mid\!- (MN) : \tau \Rightarrow \exists \sigma [\Gamma \mid\!- M : (\sigma \rightarrow \tau) \,\&\, \Gamma \mid\!- N : \sigma]$

(iii) $\Gamma \mid\!- (\lambda x.M) : \rho \Rightarrow \exists \sigma, \tau [\Gamma, x : \sigma \mid\!- M : \tau \,\&\, \rho \equiv (\sigma \rightarrow \tau)]$

Proof. By induction on the length of derivation.

Typability of subterms in $\lambda\!\rightarrow$-Curry

Let M' be a subterm of M. Then

$$\Gamma \mid\!- M : \sigma \Rightarrow \Gamma' \mid\!- M' : \sigma'$$

for some Γ' and σ'.

In other words: if M has a type which means that for some $\Gamma \mid\!- M : \sigma$ then every subterm M' of M has a type as well. Note that the subterm may be typed from a different basis.

Proof. By induction on the complexity of M.

Substitution lemma for $\lambda\!\rightarrow$-Curry.

(i) $\Gamma \mid\!- M : \sigma \Rightarrow \Gamma[\alpha := \tau] \mid\!- M : \sigma[\alpha := \tau]$

where α is a type variable.

(ii) If $\Gamma, x : \sigma \mid\!- M : \tau$ and $\Gamma \mid\!- N : \sigma$, then $\Gamma \mid\!- M[x := N] : \tau$

Proof.

(i) By induction on derivation of $\Gamma \mid\!- M : \sigma$.

(ii) By induction on generation of $\Gamma, x : \sigma \mid\!- M : \tau$

Subject reduction theorem for $\lambda\!\rightarrow$-Curry

Let $M \twoheadrightarrow_\beta M'$. Then

$$\Gamma \mid\!- M : \sigma \;\Rightarrow\; \Gamma \mid\!- M' : \sigma.$$

Proof.

By induction on generation of $\twoheadrightarrow_\beta$ using Generation lemma and Substitution lemma. We shall treat the prime case $M \rightarrow_\beta M'$. Assume that $M \equiv (\lambda x.P)Q$, $M' \equiv P[x := Q]$ and

$$\Gamma \mid\!- (\lambda x.P)Q : \sigma$$

Then it follows by the Generation lemma that for some τ one has

$$\Gamma \mid\!- (\lambda x.P) : (\tau \rightarrow \sigma) \quad \text{and} \quad \Gamma \mid\!- Q : \tau$$

Using once more the Generation lemma, we get

$$\Gamma, x:\tau \mid\!- P:\sigma \quad \text{and} \quad \Gamma \mid\!- Q:\tau$$

And therefore by the Substitution lemma, we have

$$\Gamma \mid\!- P[x := Q]:\sigma$$

Using once more the Generation lemma, we get

$$\Gamma, x:\tau \mid\!- P:\sigma \quad \text{and} \quad \Gamma \mid\!- Q:\tau$$

and therefore by the Substitution lemma, we have

$$\Gamma \mid\!- P[x := Q]:\sigma$$

Exercises.

Let $\mathbf{I} = (\lambda x.x)$, $\mathbf{K} = (\lambda xy.x)$ and $\mathbf{S} = (\lambda xyz.xy(yz))$.
Show that for all $\sigma, \tau, \rho \in \mathbf{T}$, one has

$(a) \mid\!- \mathbf{S} : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)$

$(b) \mid\!- \mathbf{SK} : (\sigma \to \tau) \to \sigma \to \sigma$

$(c) \mid\!- \mathbf{KI} : (\tau \to \sigma \to \sigma)$

Type assignments.

(a) $\mathbf{S} \equiv \lambda xyz.xz(yz)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x:(\sigma\to\tau\to\rho)^3 \quad z:\sigma^1}{(yz):\tau} \quad \dfrac{x:(\sigma\to\tau\to\rho)^3 \quad y:(\sigma\to\tau)^2}{(xz):(\tau\to\rho)}}{(xz)(yz):\rho}}{\lambda z.(xz)(yz):(\sigma\to\rho)^1}}{\lambda yz.(xz)(yz):(\sigma\to\tau)\to(\sigma\to\rho)^2}}{\lambda xyz.(xz)(yz):(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to(\sigma\to\rho)^3}$$

(b) $\mathbf{SK} \equiv (\lambda xyz.xz(yz))(\lambda xy.x) \equiv (\lambda xyz.xz(yz))(\lambda uv.u)$

We have

$$\dfrac{\dfrac{u:\sigma^2 \quad v:\tau^1}{\lambda v.u:(\tau\to\sigma)^1}}{\lambda uv:(\sigma\to\tau\to\sigma)^2}$$

by $\to$–introduction

Hence

$$\mid\!-\mathbf{K} \equiv \lambda xy.x : (\sigma \to \tau \to \sigma)$$

It follows from (a) and $\to$–elimination that

$$\mid\!-\mathbf{SK} : (\sigma \to \tau) \to \sigma \to \sigma.$$

The set of Typable terms is not closed under expansion.

We have $|\!-\mathbf{I}:(\sigma\to\sigma)$, but $|\!\not-\mathbf{KI}(\lambda x.xx):(\sigma\to\sigma)$.

A stronger failure of subject expansion has shown van Bakel.

Observation. (van Bakel 1991)

There are $M, M' \in \Lambda$ and $\sigma,\sigma' \in \mathbf{T}$ such that $M \twoheadrightarrow_\beta M'$ and

$$|\!- M:\sigma,$$
$$|\!- M':\sigma',$$

but

$$|\!\not- M':\sigma.$$

Proof. Take $M \equiv \lambda xy.y$, $M' \equiv \mathbf{SK}$, $\sigma \equiv \alpha\to(\beta\to\beta)$ and $\sigma' \equiv (\beta\to\alpha)\to(\beta\to\beta)$.

Then use the fact that $|\!\not- \mathbf{SK}:\alpha\to(\beta\to\beta)$.

Exercises.

Show that

(a) $|\!\not- \mathbf{SK}:(\tau\to\sigma\to\sigma)$ for all σ,τ.

(b) $(\lambda x.xx)$ and $\mathbf{KI}(\lambda x.xx)$ have no type in $\lambda\to$.

Proof.

(a) If $|\!-\mathbf{SK} \equiv (\lambda xyz.xz(yz))(\lambda xy.x):(\tau\to\sigma\to\sigma)$

Then by the generation lemma

$\exists\rho\,[|\!-(\lambda xyz.xz(yz)):\rho\to(\tau\to\sigma\to\sigma)\ \&\ |\!-(\lambda xy.x):\rho]$

using the Generation lemma again, we obtain

$\exists\rho\exists\mu,\nu\,[x:\mu$

$|\!-(\lambda yz.xz(yz)):\nu\ \&\ \rho\to(\tau\to\sigma\to\sigma)\equiv(\mu\to\nu)\&\ |\!-(\lambda xy.x):\rho]$

It follows that

$\mu\equiv\rho\ \&\ \upsilon\equiv(\tau\to\sigma\to\sigma)$

hence

$\exists\rho\,[x:\rho\,|\!-(\lambda yz.xz(yz)):(\tau\to\sigma\to\sigma)\&\,|\!-(\lambda xy.x):\rho]$.

using the Generation lemma repeatedly, we have

$\exists\rho\exists\varepsilon,\varphi\,[x:\rho, y:\varepsilon$

$|\!-(\lambda z.xz(yz)):\varphi\ \&\ (\varepsilon\to\varphi)\equiv(\tau\to\sigma\to\sigma)\&\ |\!-(\lambda xy.x):\rho]$

thus

$\varepsilon \equiv \tau \,\&\, \varphi \equiv \sigma \to \sigma$

hence, we have

$\exists \rho\, [x:\rho, y:\tau \mid -(\lambda z.xz(yz)):\tau \,\&\, \mid -(\lambda xy.x):\rho]$

now

$\exists \rho\, [x:\rho, y:\tau \mid -(\lambda z.xz(yz)):\tau \,\&\, \mid -(\lambda xy.x):(\rho \to \tau \to \rho)]$

a contradiction.

(b) Assume that for some $\Gamma, \sigma,$ holds $\Gamma \mid -(\lambda x.xx):\sigma.$ It follows from the Generation lemma that

$$\exists \rho, \tau\, [\Gamma, x:\rho \mid - xx:\tau \,\&\, (\rho \to \tau) \equiv \sigma]$$

then

$$\exists \rho, \tau\, \exists \zeta\, [\Gamma, x:\rho \mid - xx:\tau \,\&\, x:(\zeta \to \tau) \,\&\, x:\zeta]$$

hence

$$\exists \rho, \tau\, \exists \zeta\, [\Gamma, x:\rho \mid - x:(\zeta \to \tau) \,\&\, x:\zeta \,\&\, (\rho \to \tau) \equiv \sigma]$$

$$\exists \rho, \tau\, [\Gamma, x:\rho \mid - x:(\rho \to \tau) \,\&\, (\rho \to \tau) \equiv \sigma]$$

$$\exists \rho, \tau\, [\Gamma, x:\rho \mid - x:(\rho \to \tau)]$$

a contradiction. Thus $(\lambda x.xx)$ has no type in $\lambda \to$–Curry.

$\mathbf{KI}(\lambda x.xx)$ has no type by the Typability of subterms lemma.

The system $\lambda \to$–Church A digression.

At the first sight, the main difference between the Curry systems of type assignment and Church typing systems consists in the fact that in the Curry system, the bounded variables are typed implicitely by the system while in the Church typing system, the bounded variables are typed explicitely.

But there is in it more than that:

one has

$$\mid -_{Curry} (\lambda x.x):(\sigma \to \sigma) \qquad \text{for every type } \sigma,$$

while

$$\mid -_{Church} (\lambda x:\sigma.x):(\sigma \to \sigma).$$

The term $(\lambda x.x)$ is annotated in the Church sustem by ‚:σ', in fact it is not a lambda term in the strict sense.

The intuitive meaning is that $(\lambda x{:}\sigma.x)$ takes the argument x from the domain of the type σ. The explicit mention of types in terms make it possible the *type checking* i.e. to decide whether a term has a certain type. For some Curry systems this question is undecidable.

Definition. (pseudoterms)

Let $\mathbf{T}$ be some set of types. The set of $\mathbf{T}$-*annotated* λ-*terms* (also called *pseudoterms*), denoted by $\Lambda_{\mathbf{T}}$, is defined as follows:

$$\Lambda_{\mathbf{T}} = V \mid \Lambda_{\mathbf{T}} \Lambda_{\mathbf{T}} \mid \lambda x:\mathbf{T}.\Lambda_{\mathbf{T}}$$

where V is the set of term variables.

The same syntactic conventions are used for Λ_T as for Λ, e.g.

$$\lambda x_1 : \sigma_1 \cdots \lambda x_n : \sigma_n . M \equiv (\lambda x_1 : \sigma_1 (\lambda x_2 : \sigma_2 \cdots (\lambda x_n : \sigma_n . M) \cdots)) \equiv \lambda \vec{x} : \vec{\sigma} . M$$

Remark.

Several systems of typed λ-calculi *`a la* Church resemble to the Curry systems of type assignments since they consist of a choice of the set $\mathbf{T}$ of types and of an assignment of types $\sigma \in \mathbf{T}$ to terms $M \in \Lambda_T$.

However, this is not the case in all systems *`a la* Church. In some such systems the sets of terms and types are defined simultaneously.

Anyway, for $\lambda \rightarrow$–Church the separate definition of the sets of types and and (pseudo)terms is possible and one may have the same set of types $\mathbf{T} = \text{Type}(\lambda \rightarrow)$ as for $\lambda \rightarrow$–Curry.

Definition.

The typed lambda calculus $\lambda \rightarrow$–Church consists of

(i) *the set of types* $\mathbf{T} = \text{Type}(\lambda \rightarrow)$ defined by

$$\mathbf{T} = \mathbf{V} \mid \mathbf{T} \rightarrow \mathbf{T}$$

where $\mathbf{V}$ is the set of type variables.

(ii) *statements* of the form $M : \sigma$ with $M \in \Lambda_T$ and $\sigma \in \mathbf{T}$.

(iii) *bases* which are again sets of statements with only distinct (term) variables as subjects.

(iv) *axioms* and *rules* $\lambda \rightarrow$–Church

(axiom) $\quad \Gamma \mid\!- x : \sigma \quad$ if $(x : \sigma) \in \Gamma$

($\rightarrow$-elimination) $\quad \dfrac{\Gamma \mid\!- M : (\sigma \rightarrow \tau) \quad \Gamma \mid\!- N : \sigma}{\Gamma \mid\!- (MN) : \tau}$

($\rightarrow$-introduction) $\quad \dfrac{\Gamma, x : \sigma \mid\!- M : \tau}{\Gamma \mid\!- (\lambda x : \sigma . M) : (\sigma \rightarrow \tau)}$

Definition.

A statement $M : \sigma$ is *derivable* from the basis Γ, notation $\Gamma \mid\!- M : \sigma$, if $M : \sigma$ can be produced using the above axioms and rules.

As we have seen, derivations can be given in several styles. We will not repeat it here, although we slightly prefere the Gentzen (natural deduction) style.

Definition.

The set of *legal* $\lambda \rightarrow$–terms, denoted by $\Lambda(\lambda \rightarrow)$, is defined by

$$\Lambda(\lambda \rightarrow) = \{M \in \Lambda_T \mid \exists \Gamma, \sigma \quad \Gamma \mid\!- M : \sigma\}.$$

To refer specifically to $\lambda \rightarrow$–Church, one uses the notation

$$\Gamma \mid\!-_{\lambda \rightarrow \text{–Church}} M : \sigma.$$

If there is little danger of ambiguity one uses also

$$\mid\!-_{\lambda \rightarrow}, \quad \mid\!-_{\text{Church}} \quad \text{or} \quad \mid\!-.$$

Exercises.

(a) $|-(\lambda x:\sigma.\, x):(\sigma\to\sigma)$

(b) $|-(\lambda x:\sigma\,\lambda y:\tau.\, x):(\sigma\to\tau\to\sigma)$

(c) $|-(\lambda y:\tau.\, x):(\tau\to\sigma)$

Similarly as for the type-free theory, one can define reduction and conversion on the set of pseudoterms Λ_T.

Definition.

The binary relations $\to_\beta$, $\to\!\!>_\beta$ and $=_\beta$, denoting *one-step β-reduction, many-steps β-reduction* and *β-convertibility* on Λ_T. respectively, are generated by the contraction rule

$$(\lambda x:\sigma.\, M)N \to M[x := N].$$

Examples.

(a) $(\lambda x:\sigma.\, x)(\lambda y:\tau.\, yy) \to_\beta (\lambda y:\tau.\, yy)$

(b) $(\lambda x:\sigma\,\lambda y:\tau.\, xy)(\lambda z:\pi.\, z) \to\!\!>_\beta (\lambda y:\tau.\, y)$

(c) $(\lambda x:\sigma\,\lambda y:\tau.\, zy)(\lambda z:\pi.\, z)z \to\!\!>_\beta (zz)$

Remarks.

(i) It can be shown that the Church-Rosser theorem for $\to\!\!>_\beta$ also holds on Λ_T.

(ii) The following results for $\lambda\to$–Church are essentially the same as the corresponding propositions for $\lambda\to$–Curry.

Basis lemma for $\lambda\to$–Church.

Let Γ be a basis, we have

(i) If $\Gamma', \Gamma\subseteq\Gamma'$ is another basis, then $\Gamma|-M:\sigma \Rightarrow \Gamma'|-M:\sigma$.

(ii) $\Gamma|-M:\sigma \Rightarrow FV(M)\subseteq \mathrm{dom}(\Gamma)$.

(iii) $\Gamma|-M:\sigma \Rightarrow \Gamma|FV(M)|-M:\sigma$.

Generation lemma for $\lambda\to$–Church.

(i) $\Gamma|-x:\sigma \Rightarrow (x:\sigma)\in\Gamma$.

(ii) $\Gamma|-MN:\tau \Rightarrow \exists\sigma[\Gamma|-M:(\sigma\to\tau)\,\&\,\Gamma|-N:\sigma]$.

(iii) $\Gamma|-(\lambda x:\sigma.M):\rho \Rightarrow \exists\tau[\rho=(\sigma\to\tau)\,\&\,\Gamma,x:\sigma|-M:\tau]$.

Typability of subterms in $\lambda\to$–Church.

If M' is a subterm of M and M has a type i.e. if $\Gamma|-_{Church} M:\sigma$ for some Γ and σ, then M' has a type as well, i.e. $\Gamma'|-_{Church} M':\sigma'$ for some Γ' and σ'.

Substitution lemma for $\lambda\to$–Church.

(i) $\Gamma|-M:\sigma \Rightarrow \Gamma[\alpha:=\tau]|-M[\alpha:=\tau]:\sigma[\alpha:=\tau]$.

(ii) Suppose $\Gamma, x:\sigma|-M:\tau$ and $\Gamma|-N:\sigma$.
Then $\Gamma|-M[x:=N]:\tau$,

where α, x are a type and a term variable respectively and σ, τ are types.

Subject reduction Theorem for $\lambda\!\rightarrow$–Church.

Let $M \twoheadrightarrow_\beta M'$. Then $\Gamma \mid\!- M : \sigma \Rightarrow \Gamma \mid\!- M' : \sigma$.

Remark.

This theorem implies that the set of legal expressions is closed under reduction. It is not closed under expansion and conversion.

Take $\mathbf{I} =_\beta \mathbf{KI\Omega}$ annotated with appropriate types. It follows from the Typability of subterms lemma that $\mathbf{KI\Omega}$ has no type.

On the other hand convertible *legal* terms have the same type with respect to a given basis.

Lemma on uniqueness of types for $\lambda\!\rightarrow$–Church.

(i) Let $\Gamma \mid\!- M : \sigma$ and $\Gamma \mid\!- M : \sigma'$. Then $\sigma \equiv \sigma'$.

(ii) Let $\Gamma \mid\!- M : \sigma, \Gamma \mid\!- M' : \sigma'$ and $M =_\beta M'$. Then $\sigma = \sigma'$.

Proof.

(i) By induction on the structure of M.

(ii) Use the Church-Rosser Theorem for Λ_T, the subject reduction theorem for $\lambda\!\rightarrow$–Church, and (i).

We have seen that this proposition does not hold for $\lambda\!\rightarrow$–Curry.

Relating the Curry and Church systems

For typed lambda calculi that can be described in both ways *á la* Curry and *á la* Church, often a simple relations can be defined between the two versions. We shall show it for the simplest calculus $\lambda\!\rightarrow$.

Definition

There is a „forgetful“ mapping $|.| : \Lambda_T \rightarrow \Lambda$ defined as follows

$$|x| \equiv x$$
$$|MN| \equiv |M||N|$$
$$|\lambda x : \sigma . M| \equiv \lambda x . |M|$$

The mapping just erases all annotations of a term in Λ_T.

The following results show that legal pseudoterms in the Church version of $\lambda\!\rightarrow$ ´project´ to legal terms in the Curry version of $\lambda\!\rightarrow$.
On the other hand, legal terms in $\lambda\!\rightarrow$–Curry can be ´lifted´ to legal terms in $\lambda\!\rightarrow$–Church.

Theorem.

(i) (projection) Let $M \in \Lambda_T$. Then

$$\Gamma \mid\!-_{Church} M : \sigma \Rightarrow \Gamma \mid\!-_{Curry} |M| : \sigma.$$

(ii) (lifting) Let $M \in \Lambda$. Then

$$\Gamma \mid\!-_{Curry} M : \sigma \Rightarrow \exists M' \in \Lambda_T [\Gamma \mid\!-_{Church} M' : \sigma \;\&\; |M'| \equiv M].$$

Proof.

By induction on the derivation of the respective type assignment.

Corollary.

For an arbitrary type $\sigma \in \mathbf{T}$, we have

σ *is inhabited in* $\lambda\!\!\rightarrow$*-Curry* $\Leftrightarrow$ σ *is inhabited in* $\lambda\!\!\rightarrow$*-Church.*

Bőhm trees and Approximation. A digression.

To the rule **A** we need to introduce Bőhm trees which are a kind of ´infinite normal forms´.

Lemma.

Each $M \in \Lambda$ is in one of the following forms.

(i) $M \equiv \lambda x_1 \ldots x_n . y N_1 \ldots N_m$, with $n, m \geq 0$, and y a variable.

(ii) $M \equiv \lambda x_1 \ldots x_n . (\lambda y . N_0) N_1 \ldots N_m$, with $n \geq 0, m \geq 1$.

Proof.

By the definition a λ-term is either a variable, or of the form of application PQ or an abstraction $\lambda x.P$. We have to analyze three cases:

(a) if M is a variable, then M is of the form (i) with $n = m = 0$.

(b) if M is an application, then

$M \equiv P_0 P_1 \ldots P_m$ with P_0 not an application. Then M is of the form (i) or (ii) with $n = 0$, depending on whether P_0 is a variable (giving (i)) or an abstraction giving (ii).

(c) if $M \equiv \lambda x.P$, where $P \equiv \lambda x_1 \lambda x_2 \ldots \lambda x_k . Q, k \geq 0$ and Q is not an abstraction. Then Q is a variable or an application and it follows from the Induction hypothesis that Q is in one of forms (i) or (ii) for $n = 0$. Adding the prefix $\lambda x \lambda x_1 \lambda x_2 \ldots \lambda x_k$ does not change the form.

The following definition deals with the two forms of λ-terms from the above lemma.

Definition. (head normal form, head redex)

(i) A λ-term M is in *head normal form* (hnf) if M is in the form (i) of the above lemma. In that case y is called the *head variable* of M.

(ii) We say that M *has an head normal form* if there is N in hnf such that $M =_\beta N$.

(iii) If M is in the form (ii), we call $(\lambda y.N_0)N_1$ *the head redex* of M.

Lemma. **(convertibility of head normal forms)**

If $M =_\beta M'$ and

M has hnf $M_1 \equiv \lambda x_1 \ldots x_n.y\, N_1 \ldots N_m$,

M' has hnf $M_1' \equiv \lambda x_1 \ldots x_{n'}.y'\, N_1' \ldots N_m'$,

then $n = n', y \equiv y', m = m'$ and $N_1 =_\beta N'_1, \ldots, N_m =_\beta N'_m$.

Proof.

By the Church-Rosser theorem M_1 and M'_1 have a common reduct L. But then the only possibility is that

$$L \equiv \lambda x_1 \ldots x_{n''}.y''N''_1 \ldots N''_{m''}$$

where

$$n = n'' = n', y = y'' = y', m = m'' = m' \text{ and } N_1 =_\beta N''_1 =_\beta N'_1, \ldots$$

λ⊥ – calculus a digression.

Definition.

λ⊥ – calculus is the extension of the lambda calculus defined as follows. One of the (term) variables is selected for use as a constant and is given the name ⊥.

(i) two contraction rules are added:

$$\lambda x.\bot \to \bot$$
$$\bot M \to \bot$$

(ii) A $\beta\bot$ – *normal form* is such that it cannot be $\beta\bot$ – reduced.

We are going to introduce the notion of Böhm tree. The definition is not complete, because it does not specify the ordering of the direct successors of a node. However this ordering is displayed in the pictures of the trees. This suffices for our purposes.

The precise definition of the order can be found in (Barendregt 1984).

Definition.

Let $M \in \Lambda$. The Böhm tree of M, denoted by BT(M), is the labelled tree defined as follows

$$BT(M) = \begin{cases} \lambda x_1 \ldots x_n . y \;\; (BT(N_1) \;\; \ldots \;\; BT(N_m)) & \text{if the hnf of M is } \lambda x_1 \ldots x_n . y N_1 \ldots N_m \\ \bot & \text{if M has no hnf} \end{cases}$$

Böhm trees for $\lambda\bot-$ calculus are defined under condition that a $\lambda\bot-$ term $\lambda x_1 \ldots x_n . y N_1 \ldots N_m$ is in $\beta\bot-$ head normal form only if $y \not\equiv \bot$ or $n = m = 0$.

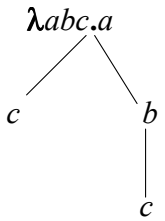
Note that if M has β-hnf, then M has a $\beta\bot-$ hnf, too. This is because an hnf $\lambda x_1 \ldots x_n . y N_1 \ldots N_m$ is also $\beta\bot-$ hnf unless $y = \bot$. If it is the case, then

$$\lambda x_1 \ldots x_n . y N_1 \ldots N_m \twoheadrightarrow_{\beta\bot} \bot$$

and hence M has a $\beta\bot-$ hnf.

Examples.

(a) $BT(\lambda abc.ac(bc)) =$



(b) $BT((\lambda x.xx)(\lambda x.xx)) = \bot$

(c) Recall that the Fixed point operator $\mathbf{Y}$ is defined by

$$\mathbf{Y} \equiv \lambda f . \underbrace{(\lambda x . f(xx))}_{\omega_f} \underbrace{(\lambda x . f(xx))}_{\omega_f}.$$

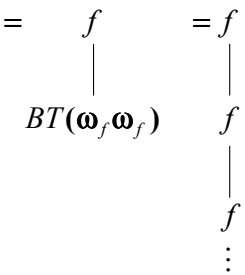
It follows that $\mathbf{Y} \equiv \lambda f . \omega_f \omega_f = \lambda f . f(\omega_f \omega_f)$ and we have

$BT(\mathbf{Y}) = \lambda f . f$

$BT(\omega_f \omega_f)$

Now $\omega_f \omega_f = f(\omega_f \omega_f)$

thus $BT(\omega_f \omega_f) =$



Hence

$BT(\mathbf{Y}) = \lambda f.f$

f

f

f

$\vdots$

Remark.

The definition of the Böhm tree is not an inductive definition of $BT(M)$, although it seems to be according to presented examples. The terms $N_1, \ldots, N_m$ in the tail of an hnf of M may be more complex than the term M itself. [Barendregt 1984, Chapter 10]

Lemma. (Correctnost of the definition of Böhm trees)

(i) Böhm trees are well defined,

(ii) $M =_\beta N \Rightarrow BT(M) = BT(N)$.

Proof.

The definition is correct as it is independent of the choice of the head normal forms. This and (ii) follows from the lemma on convertibility of hnfs.

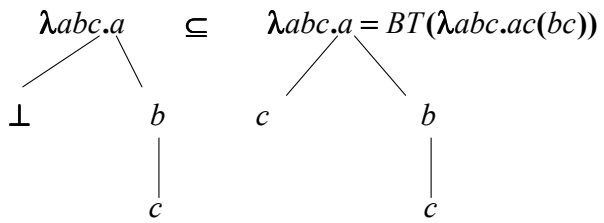
Definition. Approximate normal forms.

(i) Let A and B be Böhm trees of some $\lambda\perp-$ terms. We say that A is *included* in B and write $A \subseteq B$, if A results from B by cutting of some subtrees, leaving an empty subtree $\perp$.

(ii) Let P, Q be $\lambda\perp-$terms. We say that P *approximates* Q and write $P \subseteq Q$, if $BT(P) \subseteq BT(Q)$.

(iii) Let P be a $\lambda\perp-$term. The set of *approximate normal forms* (anf's) of P, is defined as follows

$$A(P) = \{Q \subseteq P \mid Q \text{ is a } \beta\perp-\text{nf}\}.$$



Example.

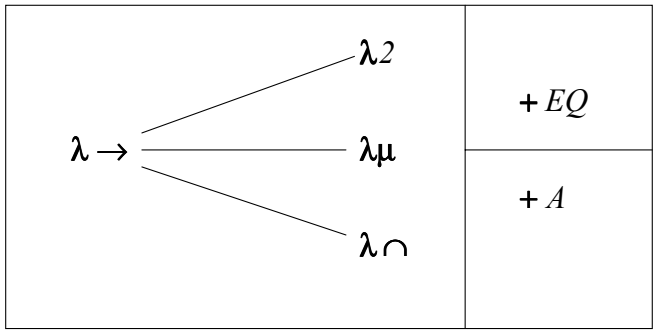
The set of approximate normal forms for the fixed point operator $\mathbf{Y}$ is

$$A(\mathbf{Y}) = \{\perp, \quad \lambda f.f\perp, \quad \lambda f.f^2\perp, \quad \lambda f.f^3\perp, \quad \ldots \}$$

Typing á la Curry

The basic system $\lambda\!\rightarrow\!$–Curry can be extended in various ways to stronger systems by adding new types and by adding new rules. Some of the new rules are related to combinatorial properties of the trees representing the terms.

The systems to be discussed are $\lambda\!\rightarrow, \lambda 2, \lambda\mu$ and $\lambda\cap$. To each of these can be added one of the extra derivation rules *EQ* and ***A.***



The systems á la Curry

Definition. Rules of equality (EQ) and approximation (A).

(i) The *equality rule* EQ

$$\frac{M:\sigma \qquad M =_\beta N}{N:\sigma}$$

(ii) The *approximation rules* **A**

$$\frac{\Gamma \mid\!- P:\sigma \text{ for all } P \in \mathbf{A}(M)}{\Gamma \mid\!- M:\sigma} \qquad \frac{}{\Gamma \mid\!- \bot:\sigma}$$

Remark. (Side conditions)

Note that in these rules the assumptions $M =_\beta N$ and $P \in \mathsf{A}(M)$ are not type assignments. We call them *side conditions*. The last rule states that $\bot$ has any type.

Notation.

Let λ- be any of the systems $\lambda\!\rightarrow, \lambda 2, \lambda\mu$ or $\lambda\cap$. We denote by

(i) $\lambda\!-^{+}$, the system $\lambda\!-$ extended by the rule EQ.

(ii) $\lambda\!-\mathbf{A}$, the system $\lambda\!-$ extended by the rule **A**.

So for example

$$\lambda 2^{+} = \lambda 2 + EQ \quad \text{and} \quad \lambda\mu\mathbf{A} = \lambda\mu + \mathbf{A}.$$

Examples.

(a) One has

$$|-_{\lambda\to^+} (\lambda pq.(\lambda r.p)(qp)) : (\sigma \to \tau \to \sigma)$$

It follows from the equality $\lambda pq.(\lambda r.p)(qp) = \lambda pq.p$
Note that this statement is not provable in general in $\lambda\to$ itself. The term has in $\lambda\to$ only types of the form $\sigma \to (\sigma \to \tau) \to \sigma$, as follows from the generation lemma.

(b) Let $\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ be the fixed point operator. Then

$$|-_{\lambda\to\mathsf{A}} \mathbf{Y} : ((\sigma \to \sigma) \to \sigma)$$

Indeed, the set of approximate normal forms of $\mathbf{Y}$ is

$$\{\bot, \lambda f.f\bot, \lambda f.f^2\bot, \ldots \lambda f.f^n\bot, \ldots\}$$

And all these terms have type $((\sigma \to \sigma) \to \sigma)$. Again, this statement is not derivable in $\lambda\to$ itself. (In $\lambda\to$ all typable terms have a normal form as we shall see later on.)

We are going to show that the rule **A** is stronger than the rule EQ.

Proposition.

Let λ- be one of the systems $\lambda 2, \lambda\mu$ or $\lambda\cap$ of type assignments. In all systems λ-**A**, we have

(i) $\Gamma |- M : \sigma$ and $P \in \mathbf{A}(M) \Rightarrow \Gamma |- P : \sigma$

(ii) Let $BT(M) = BT(M')$. Then

$$\Gamma |- M : \sigma \Rightarrow \Gamma |- M' : \sigma$$

(iii) Let $M =_\beta M'$. Then

$$\Gamma |- M : \sigma \Rightarrow \Gamma |- M' : \sigma$$

Note that (iii) is the rule EQ.

Proof.

(i) If P is an approximate normal form of M, then P results from BT(M) by replacing some subtrees by $\bot$ and writing the result as a λ-term by one of the rules **A**. Therefore P has the same type as M. (see an Example below).

(ii) Suppose $BT(M) = BT(M')$. Then $\mathbf{A}(M) = \mathbf{A}(M')$, and, consequently

$$\Gamma |- M : \sigma \Rightarrow (\forall P \in \mathbf{A}(M) = \mathbf{A}(M'))[\Gamma |- P : \sigma], \text{ by (i)},$$

$$\Rightarrow \Gamma |- M' : \sigma, \text{ by rule } \mathbf{A}.$$

(iii) If $M =_\beta M'$, then $BT(M) = BT(M')$, by the lemma on the correctness of the definition of Böhm trees. The result then follows by (ii).

Example.

Let $M \equiv \mathbf{Y}$, the fixed point combinator and let $P \equiv \lambda f.f(f\bot)$ be an approximant. We have

$$\vdash \mathbf{Y} : (\sigma \to \sigma) \to \sigma$$

By choosing σ as a type for $\bot$, one obtains

$$\vdash P : (\sigma \to \sigma) \to \sigma$$

System λ2

- Polymorphic typed lambda calculus
- Second-order typed lambda calculus
- Second-order polymorphic typed λ-calculus
- System F
  - Girard (1972)
  - Reynolds (1974)

Motivation.

(i) Usually these names refer to λ2-Church, we shall introduce the Curry version of λ2 to discuss the Church version later.

(ii) The idea of polymorphism: while in $\lambda\!\to$ we have

$$(\lambda x.x) : (\alpha \to \alpha)$$

for arbitrary (type) variable α (and for arbitrary type σ as well), one stipulates in λ2

$$(\lambda x.x) : (\forall \alpha.(\alpha \to \alpha))$$

to indicate that $\lambda x.x$ has all types $\sigma \to \sigma$ or that the type of $\lambda x.x$ depends uniformly on α.

As we shall see later, the mechanism is rather powerful.

Definition. The set of types of λ2.

The set $\mathbf{T} = \text{Type}(\lambda 2)$ is defined by the following abstract grammar

$$\mathbf{T} = \mathbf{V} \mid \mathbf{T} \to \mathbf{T} \mid \forall\, \mathbf{V}\, \mathbf{T}$$

Notation.

(i) The parentheses by quantifiers cumulate to the right, so we have $\forall \alpha_1 \ldots \alpha_n.\sigma$ as a shorthand for $(\forall \alpha_1 (\forall \alpha_2 \ldots (\forall \alpha_n (\sigma)) \ldots))$

(ii) If there are no parentheses, $\forall$ binds more strongly than $\to$.

Hence

$$\forall \alpha\, \sigma \to \tau \equiv (\forall \alpha\, \sigma) \to \tau, \text{ but } \forall \alpha.\sigma \to \tau \equiv \forall \alpha(\sigma \to \tau).$$

Definition. Type assignment in λ2-Curry.

$\lambda 2$

(start rule)	$\dfrac{(x:\sigma)\in\Gamma}{\Gamma\mid -x:\sigma}$
($\to$ elimination)	$\dfrac{\Gamma\mid -M:(\sigma\to\tau)\qquad \Gamma\mid -N:\sigma}{\Gamma\mid -(MN):\tau}$
($\to$ introduction)	$\dfrac{\Gamma,x:\sigma\mid -M:\tau}{\Gamma\mid -(\lambda x.M):(\sigma\to\tau)}$
($\forall-$elimination)	$\dfrac{\Gamma\mid -M:(\forall\alpha\,\sigma)}{\Gamma\mid -M:(\sigma[\alpha:=\tau])}$
($\forall-$introduction)	$\dfrac{\Gamma\mid -M:\sigma}{\Gamma\mid -M:(\forall\alpha.\sigma)}$

Exercises.

(a) $\mid\!- (\lambda x.x) \quad : \quad (\forall\alpha.\alpha\to\alpha)$

(b) $\mid\!- (\lambda xy.y) \quad : \quad (\forall\alpha\beta.\alpha\to\beta\to\alpha)$

(c) $\mid\!- (\lambda fx.f^n x) \quad : \quad (\forall\alpha.(\alpha\to\alpha)\to\alpha\to\alpha)$

(d) $\mid\!- (\lambda x.xx) \quad : \quad (\forall\beta.\forall\alpha\alpha\to\beta)$

(e) $\mid\!- (\lambda x.xx) \quad : \quad (\forall\beta.\forall\alpha\alpha\to(\beta\to\beta))$

(f) $\mid\!- (\lambda x.xx) \quad : \quad (\forall\alpha\alpha)\to(\forall\alpha\alpha)$

Remarks.

(i) Exercise (c) shows that the Church numerals $c_n \equiv (\lambda fx.f^n x)$ have type $(\forall\alpha.(\alpha\to\alpha)\to\alpha\to\alpha)$ which is sometimes called "*polynat*".

(ii) One reason for the strength of $\lambda 2$ is that the Church numerals may be used as iterators for functions of types $\sigma\to\sigma$ for arbitrary σ and not only for functions of a fixed type $\alpha\to\alpha$.

(iii) We shall show later that the typable terms in $\lambda 2$ have a normal form, in fact they are *strongly normalizing.*

The system λμ

- It is the system of *recursive types.*
- The recursive types come together with an equivalence relation $\approx$.
- The type assignment rules consist of the rules of $\lambda\to$ and the following rule

$$\frac{\Gamma\mid -M:\sigma\qquad \sigma\approx\sigma'}{\Gamma\mid -M:\sigma'}$$

Motivation

A typical example of a recursive type is a type σ_0

$$\sigma_0 \approx \sigma_0 \to \sigma_0 \qquad (1)$$

This particular type can be used to type arbitrary terms $M \in \Lambda$.
As an example, we shall show that $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$
has σ_0 as a type

$$x:\sigma_0 \mid\!- x:\sigma_0 \to \sigma_0$$
$$x:\sigma_0 \mid\!- xx:\sigma_0$$
$$\mid\!- \lambda x.xx:\sigma_0 \to \sigma_0$$
$$\mid\!- \lambda x.xx:\sigma_0$$
$$\mid\!- \underbrace{(\lambda x.xx)(\lambda x.xx)}_{\Omega}:\sigma_0$$

Here is the proof of the same statement in a natural deduction setting

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\sigma_0}\,1}{x:\sigma_0 \to \sigma_0} \qquad \dfrac{}{x:\sigma_0}}{(xx):\sigma_0}}{(\lambda x.xx):\sigma_0 \to \sigma_0}\,1 \qquad \dfrac{}{(\lambda x.xx):\sigma_0}}{\underbrace{(\lambda x.xx)(\lambda x.xx)}_{\Omega}:\sigma_0}$$

Remarks.

(i) The equation (1) is similar to a recursive domain equation

$$D \cong [D \to D]$$

that enables to interpret elements of Λ in denotational semantics.

(ii) In order to construct a type σ_0 satisfying (1), there is an operator μ such that putting $\sigma_0 \equiv \mu\alpha.\alpha \to \alpha$ implies (1).

Definition. The set T=Type(λμ), trees of types of λμ.

(i) The set of types of $\lambda\mu$, $\mathbf{T}$ = Type($\lambda\mu$), is defined by the following abstract grammar.

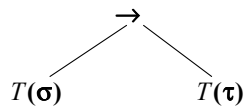
$$\mathbf{T} = \mathbf{V} \mid \mathbf{T} \to \mathbf{T} \mid \mu\mathbf{V}.\mathbf{T}$$

where $\mathbf{V}$ is the set of type variables.

(ii) Let $\sigma \in \mathbf{T}$ be a type. The *tree of* σ, $T(\sigma)$ is defined by induction on the structure of σ as follows:

$T(\alpha) = \alpha$ if α is a type variable

$T(\sigma \to \tau) =$



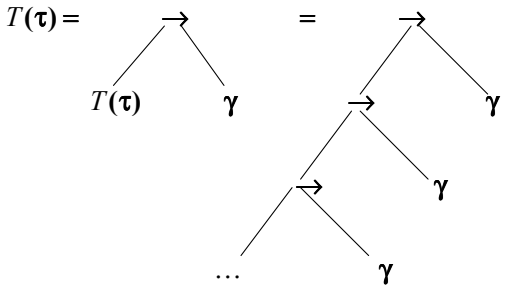
$$T(\mu\alpha.\sigma) = \begin{cases} \bot & \text{If } \sigma \equiv \mu\beta_1 \ldots \beta_n.\alpha \text{ for some } n \geq 0 \\ T(\sigma[\alpha := \mu\alpha.\sigma]) & \text{else} \end{cases}$$

(iii) The equivalence relation $\approx$ on trees is defined as follows:

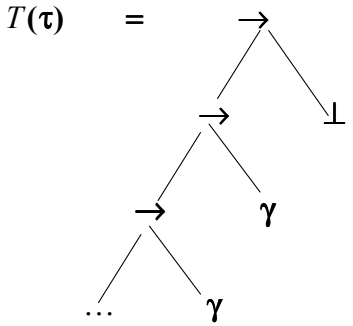
$$\sigma \approx \tau \Leftrightarrow T(\sigma) = T(\tau)$$

Exercises.

(a) Assume $\tau \equiv \mu\alpha.\alpha \to \gamma$, then



(b) Assume $\tau \equiv (\mu\alpha.\alpha \to \gamma) \to \mu\delta\mu\beta.\beta$, then



(c) $(\mu\alpha.\alpha \to \gamma) \approx (\mu\alpha(\alpha \to \gamma) \to \gamma)$.

(d) $\mu\alpha.\sigma \approx \sigma[\alpha := \mu\alpha.\sigma]$ for all σ, even if $\sigma \equiv \mu\beta\alpha$.

Definition.

The type assignment system $\lambda\mu$ is defined by the natural deduction system presented in the following picture

$\lambda\mu$

(start rule) $\dfrac{(x:\sigma)\in\Gamma}{\Gamma\mid- x:\sigma}$

$(\rightarrow-$elimination) $\dfrac{\Gamma\mid- M:(\sigma\rightarrow\tau)\qquad\Gamma\mid- N:\sigma}{\Gamma\mid-(MN):\tau}$

$(\rightarrow-$introduction) $\dfrac{\Gamma,x:\sigma\mid- M:\tau}{\Gamma\mid-(\lambda x.M):(\sigma\rightarrow\tau)}$

$(\approx-$rule) $\dfrac{\Gamma\mid- M:\sigma\qquad\sigma\approx\tau}{\Gamma\mid- M:\tau}$

Proposition. (Coppo 1985)

For an arbitrary type σ, we have in $\lambda\mu$

(i) $\mid-\mathbf{Y}:(\sigma\rightarrow\sigma)\rightarrow\sigma$

(ii) $\mid-\Omega:\sigma$

Proof.

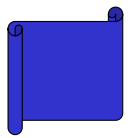
(i) If we put $\tau\equiv\mu\alpha.\alpha\rightarrow\alpha$, then $\tau\approx\tau\rightarrow\sigma$. We will derive

$$\mathbf{Y}\equiv\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)):(\sigma\rightarrow\sigma)\rightarrow\sigma$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\tau}\,1}{x:\tau\rightarrow\sigma\qquad x:\tau}}{f:\sigma\rightarrow\sigma\qquad xx:\sigma}}{f(xx):\sigma}}{\lambda x.f(xx):\tau\rightarrow\sigma}\,1}{\dfrac{\lambda x.f(xx):\tau\rightarrow\sigma\qquad\lambda x.f(xx):\tau}{(\lambda x.f(xx))(\lambda x.f(xx)):\sigma}}$$

$$\overline{\mathbf{Y}\equiv\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)):(\sigma\rightarrow\sigma)\rightarrow\sigma}\;2$$

(ii) Note that $\mathbf{YI}\rightarrow\!>_{\beta}\Omega$ the result follows from the fact that $\mathbf{I}\equiv(\lambda x.x):(\sigma\rightarrow\sigma)$, $(\rightarrow-$elimination) and the subject reduction theorem. It is possible to prove (ii) directly.



The system $\lambda\cap$

- It is called the system of intersection types or Torino system.
- Barendregt, Coppo, Dezani, Honsell and Longo (1981 - 1987)
- The system makes it possible that a (term) variable x has exactly two types σ and τ at the same time.

The set of types of the system $\lambda\cap$ comes together with a preorder on the set of types.

Definition. The set of types.

(i) The set of types $\mathbf{T} = \mathrm{Type}(\lambda\cap)$, is defined by an abstract grammar as follows:

$$\mathbf{T} = \mathbf{V} \mid \mathbf{T} \to \mathbf{T} \mid \mathbf{T} \cap \mathbf{T}$$

where $\mathbf{V}$ is the set of type variables.

(ii) We select one of the type variables as a constant and name it as ω.

Definition. The preorder on T.

(i) The relation $\leq$ is defined on **T** by the following axioms and rules

$\sigma \leq \sigma$

$\sigma \leq \tau, \tau \leq \rho \Rightarrow \sigma \leq \rho$

$\sigma \leq \omega$

$\omega \leq \omega \to \omega$

$(\sigma \to \rho) \cap (\sigma \to \tau) \leq (\sigma \to (\rho \cap \tau))$

$\sigma \cap \tau \leq \sigma, \sigma \cap \tau \leq \tau$

$\sigma \leq \tau, \sigma \leq \rho \Rightarrow \sigma \leq \sigma \cap \tau$

$\sigma \leq \sigma', \tau \leq \tau' \Rightarrow \sigma' \to \tau \leq \sigma \to \tau'$

(ii) $\sigma \lessgtr \tau \Leftrightarrow (\sigma \leq \tau \,\&\, \tau \leq \sigma)$

Exercises.

(a) $\omega \lessgtr (\omega \to \omega)$

(b) $((\sigma \to \tau) \cap (\sigma' \to \tau)) \leq ((\sigma \cap \sigma') \to \tau)$

Exercises.

(a) $\omega \leftrightarrows (\omega \to \omega)$

(b) $((\sigma \to \tau) \cap (\sigma' \to \tau)) \leq ((\sigma \cap \sigma') \to \tau)$

Proof.

(a) obvious

(b) We know $\sigma \cap \sigma' \leq \sigma$ thus

$$\sigma \to \tau \leq ((\sigma \cap \sigma') \to \tau) \qquad (1)$$

trivially

$$(\sigma \to \tau) \cap (\sigma' \to \tau) \leq (\sigma \to \tau) \qquad (2)$$

Then (b) follows from (1) and (2) by transitivity.

Definition. The system of type assignment $\lambda\cap$.

(start rule) $$\frac{(x:\sigma)\in\Gamma}{\Gamma \mid\!- x:\sigma}$$

($\to$–elimination) $$\frac{\Gamma \mid\!- M:(\sigma\to\tau) \qquad \Gamma \mid\!- N:\sigma}{\Gamma \mid\!- (MN):\tau}$$

($\to$–introduction) $$\frac{\Gamma, x:\sigma \mid\!- M:\tau}{\Gamma \mid\!- (\lambda x.M):(\sigma\to\tau)}$$

($\cap$– elimination) $$\frac{\Gamma \mid\!- M:(\sigma\cap\tau)}{\Gamma \mid\!- M:\sigma \qquad \Gamma \mid\!- M:\tau}$$

($\cap$– introduction) $$\frac{\Gamma \mid\!- M:\sigma \qquad \Gamma \mid\!- M:\tau}{\Gamma \mid\!- M:(\sigma\cap\tau)}$$

(ω– introduction) $$\frac{}{\Gamma \mid\!- M:\omega}$$

($\leq$–rule) $$\frac{\Gamma \mid\!- M:\sigma \qquad \sigma\leq\tau}{\Gamma \mid\!- M:\tau}$$

Exercises.

(a) $\mid\!- \lambda x.xx : ((\sigma \to \tau)\cap) \to \tau$

(b) $\mid\!- \Omega : \omega$

(c) $\mid\!- (\lambda pq.(\lambda r.p)(qp)) : (\sigma \to (\tau \to \sigma))$.

Proof.

(a)
$$\dfrac{\dfrac{\dfrac{\cancel{x:(\sigma\to\tau)\cap\sigma}\,1}{x:\sigma\to\tau} \qquad \dfrac{\cancel{x:(\sigma\to\tau)\cap\sigma}\,1}{x:\sigma}}{(xx):\tau}}{(\lambda x.xx):((\sigma\to\tau)\cap\sigma)\to\tau}\,1$$

(b) Obvious, it can be shown that M has no head normal form iff ω is the only possible type for M. (Barendregt 1983)

(c)
$$\dfrac{\dfrac{\dfrac{\dfrac{\cancel{q:\tau}\,2 \quad \cancel{p:\sigma}\,3 \quad \cancel{r:\omega}\,1}{(\lambda r.p):(\omega\to\sigma)}\,1 \qquad \dfrac{}{(qp):\omega}}{(\lambda r.p)(qp):\sigma}}{(\lambda q.(\lambda r.p)(qp)):(\tau\to\sigma)}\,2}{(\lambda pq.(\lambda r.p)(qp)):(\sigma\to(\tau\to\sigma))}\,3$$

Combining the systems á la Curry

(i) there are some variants of the system $\lambda\cap$. in one of them the rule (axiom) that assigns ω to any term.

(ii) The systems $\lambda\!\to, \lambda 2, \lambda\mu$ and $\lambda\cap$ are all extensions of $\lambda\!\to$. They can be combined into other systems, an extreme case is $\lambda 2\mu\cap$ which includes all these systems. It can be extended by cartesian products and direct sums in order to fall into the cartesian closed category.

Basic properties.

The Curry systems $\lambda\!\to, \lambda 2, \lambda\mu$ and $\lambda\cap$ enjoy several properties

Common to all systems:
- Basis lemma
- Subterm lemma
- Substitution lemma
- Subject reduction Theorem

Each system has a proper variant
- Generation lemma

- Strong normalization does not hold for all

In the following $\vdash$ refers to one of Curry systems $\lambda\!\to, \lambda 2, \lambda\mu$ and $\lambda\cap$.

The following three common properties are proved in the same way as we have done for $\lambda\!\to$.

Basis lemma for the Curry systems.

Let Γ be a basis.

(i) If $\Gamma', \Gamma \subseteq \Gamma'$ is another basis, then $\Gamma \vdash M : \sigma \Rightarrow \Gamma' \vdash M : \sigma$

(ii) $\Gamma \vdash M : \sigma \Rightarrow FV(M) \subseteq \mathrm{dom}(\Gamma)$

(iii) $\Gamma \vdash M : \sigma \Rightarrow \Gamma \mid FV(M) \vdash M : \sigma$

Subterm lemma for the Curry systems.

Let M' be a subterm of M. Then

$$\Gamma \vdash M : \sigma \Rightarrow \Gamma' \vdash M' : \sigma'$$

For some Γ' and σ'.

Substitution lemma for the Curry systems.

(i) $\Gamma \vdash M : \sigma \Rightarrow \Gamma[\alpha := \tau] \vdash M : \sigma[\alpha := \tau]$

(ii) $(\Gamma, x : \sigma \vdash M : \tau \,\&\, \Gamma \vdash N : \sigma) \Rightarrow \Gamma \vdash M[x := N] : \tau$

Exercise.

Show that for each of the systems $\lambda\!\to, \lambda 2, \lambda\mu$ and $\lambda\cap$ one has

$$\not\vdash K : (\alpha \to \alpha)$$

in that system.

Subject reduction and subject conversion.

Subject reduction

$$\Gamma \vdash M : \sigma \text{ and } M \twoheadrightarrow_\beta M' \Rightarrow \Gamma \vdash M' : \sigma$$

holds for the main systems of type assignment á la Curry, namely $\lambda\!\to, \lambda 2, \lambda\mu$ and $\lambda\cap$, with or without the additional rules EQ and **A.**

Subject conversion

$$\Gamma \vdash M : \sigma \text{ and } M =_\beta M' \Rightarrow \Gamma \vdash M' : \sigma$$

Holds only for the systems including $\lambda\cap$ or rule **A** or if the rule EQ is included.

Subject reduction.

We have already proved the Subject reduction theorem for the basic system $\lambda\!\to$ and we are going to prove it for $\lambda 2$. We need some definitions and throughout the proof $\boldsymbol{T}$ = Type($\lambda 2$).

Definition.

(i) Write $\sigma > \tau$ if one of the following conditions is satisfied

$$\tau \equiv \forall\alpha.\sigma, \text{ for some } \sigma,$$

or

$$\sigma \equiv \forall\alpha.\sigma_1 \;\&$$
$$\tau \equiv \sigma_1[\alpha := \pi], \text{ for some } \pi$$

(ii) The relation $\geq$ is the reflexive and transitive closure of $>$.

(iii) A map $o : \mathbf{T} \to \mathbf{T}$ is defined as follows

$$\alpha^o = \alpha \quad \text{if } \alpha \text{ is a type variable}$$
$$(\sigma \to \tau)^o = \sigma \to \tau$$
$$(\forall\alpha.\sigma)^o = \sigma^o$$

Remark.

Note that the $\forall$-introduction and $\forall$-elimination rules are the only ones in which the subject does not change. Several instances of these rules may be applied consecutively, giving

$$\begin{array}{c} M : \sigma \\ \vdots \\ \vdots \\ \hline M : \tau \end{array}$$

In this case $\sigma \geq \tau$. By this reasoning, one obtains the following:

Lemma.

Let $\sigma \geq \tau$ an assume that no free type variable in σ occurs in Γ. Then

$$\Gamma \mid\!- M : \sigma \Rightarrow \Gamma \mid\!- M : \tau$$

Proof.

Suppose $\Gamma \mid\!- M : \sigma$ and $\sigma \geq \tau$. Then $\sigma \equiv \sigma_1 > \cdots > \sigma_n \equiv \tau$ for some $\sigma_1, \cdots, \sigma_n$. if necessary, by renaming some bound type variables, we may assume that for $i, 1 \leq i < n$, we have

$$\sigma_{i+1} \equiv \forall \alpha.\sigma_1 \Rightarrow \alpha \notin FV(\Gamma)$$

By the definition of the relation $>$ and the rules of $\lambda 2$, it follows that we have

$$\Gamma \mid\!- M : \sigma_i \Rightarrow \Gamma \mid\!- M : \sigma_{i+1}$$

for all $i < n$. Hence $\Gamma \mid\!- M : \sigma_n \equiv \tau$.

Generation lemma for λ2-Curry.

(i) $\Gamma \mid\!- x : \sigma \Rightarrow \exists \sigma' \geq \sigma((x : \sigma) \in \Gamma$

(ii) $\Gamma \mid\!- (MN) : \tau \Rightarrow \exists \sigma \exists \tau' \geq \tau\, [\Gamma \mid\!- M : \sigma \rightarrow \tau' \;\&\; \Gamma \mid\!- N : \sigma]$

(iii) $\Gamma \mid\!- (\lambda x.M) : \rho \Rightarrow \exists \sigma \exists \tau\, [\Gamma, x : \sigma \mid\!- M : \tau \;\&\; \sigma \rightarrow \tau \geq \rho]$

Proof.

By induction on derivations.

Lemma on preorder of types.

(i) Given types σ, τ, there exists a type τ' such that

$$(\sigma[\alpha := \tau])^o \equiv \sigma^o[\alpha := \tau']$$

(ii) $$\sigma_1 \geq \sigma_2 \Rightarrow \exists \alpha \exists \tau\, (\sigma_2^o \equiv \sigma_1^o[\alpha := \tau]$$

(iii) $$(\sigma \rightarrow \rho) \geq (\sigma' \rightarrow \rho') \Rightarrow \exists \vec{\alpha} \exists \vec{\tau}\, (\sigma' \rightarrow \rho') \equiv (\sigma \rightarrow \rho)[\vec{\alpha} := \vec{\tau}]$$

Proof.

(i) By induction on the structure of σ.

(ii) It suffices to prove it for $\sigma_1 \geq \sigma_2$. We have to consider two cases

Case 1. $\sigma_2 \equiv \forall \alpha.\sigma_1$. Then $\sigma_2^o \equiv \sigma_1^o$.

Case 2. $\sigma_1 \equiv \forall \alpha.\rho$ and $\sigma_2 \equiv \rho[\alpha := \tau]$

Then by (i) we have $\sigma_2^o \equiv \rho^o[\alpha := \tau'] \equiv \sigma_1^o[\alpha := \tau']$

(iii) By (ii), we have

$$(\sigma' \rightarrow \rho') \equiv (\sigma' \rightarrow \rho')^o \equiv (\sigma \rightarrow \rho)^o[\vec{\alpha} := \vec{\tau}] \equiv (\sigma \rightarrow \rho)[\vec{\alpha} \rightarrow \vec{\tau}]$$

Subject reduction theorem for λ2-Curry.

If $M \twoheadrightarrow_\beta M'$, then we have

$$\Gamma \mid\!- M : \sigma \Rightarrow \Gamma \mid\!- M' : \sigma$$

Proof.

By induction on the derivation of $M \twoheadrightarrow_\beta M'$. We will treat only the case of β-reduction i.e. the case that $M \equiv (\lambda x.P)Q$ and $M' \equiv P[x := Q]$.

By the generation lemma, we obtain

$$\Gamma \mid\!- ((\lambda x.P)Q) : \sigma$$
$$\Rightarrow \exists \rho\, \exists \sigma' \geq \sigma\, [\Gamma \mid\!- (\lambda x.P) : (\rho \to \sigma') \;\&\; \Gamma \mid\!- Q : \rho]$$
$$\Rightarrow \exists \rho'\, \exists \sigma'' \geq \sigma\, [\Gamma, x : \rho' \mid\!- P : \sigma'' \;\&\; (\rho' \to \sigma'') \geq (\rho \to \sigma') \;\&\; \Gamma \mid\!- Q : \rho]$$

From (iii) of the lemma on preorder of types, it follows

$$(\rho \to \sigma') \equiv (\rho' \to \sigma'')[\alpha := \tau]$$

and hence by (i) of the the Substitution lemma

$$\Rightarrow \Gamma, x : \rho \mid\!- P : \sigma', \Gamma \mid\!- Q : \rho \text{ and } \sigma' \geq \sigma$$
$$\Rightarrow \Gamma \mid\!- P[x := Q] : \sigma' \text{ and } \sigma' \geq \sigma \text{ by (ii) of the Substitution lemma}$$
$$\Rightarrow \Gamma \mid\!- P[x := Q] : \sigma \text{ by the lemma stating that assignment of a bigger type to a term implies the assignment of a smaller type.}$$

Subject reduction theorem for λμ.

Let $M \twoheadrightarrow_\beta M'$, then for λμ one has

$$\Gamma \mid\!- M : \sigma \Rightarrow \Gamma \mid\!- M' : \sigma$$

Proof.

The proof of the subject reduction theorem for λμ is somewhat easier than that for λ2. It follows similar steps but using the relation $\approx$ insted of $\geq$.

Remark.

The subject reduction theorem holds also for $\lambda\cap$. This system is also closed under the rule EQ as we will show later on. We will see that in the systems $\lambda\!\rightarrow, \lambda 2$ and $\lambda\mu$. and λ-**A** the subject conversion theorem holds. This is not so for $\lambda\cap$.

Example. What makes $\lambda\cap$ closed under β-expansion.

Let

$M \equiv (\lambda x.P)Q$ be the redex and $M' \equiv P[x := Q]$ its contractum.

To show that β-expansion holds for this pair assume that

$$\Gamma \mid -_{\lambda\cap} M' : \sigma.$$

Now Q occurs $n \geq 0$ times in M', each occurence having its proper type τ_i for $1 \leq i \leq n$. Define

$$\tau \equiv \begin{cases} \tau_1 \cap \cdots \cap \tau_n & \text{if } n > 0 \\ \omega & \text{if } n = 0 \end{cases}$$

Then $\Gamma \mid - Q : \tau$

$\Gamma, x : \tau \mid - P : \sigma$

Hence $\Gamma \mid - (\lambda x.P) : (\tau \rightarrow \sigma)$ and $\Gamma \mid - (\lambda x : P)Q : \sigma.$

In $\lambda\!\rightarrow, \lambda 2$ and $\lambda\mu$. it is not guaranteed that there is a common type for the different occurrences of Q. Note that the type ω is essential in case when Q has no occurrence in P[x:= Q].

Subject conversion theorem for $\lambda\cap$.

Let $M =_\beta M'$, then for $\lambda\cap$ one has

$$\Gamma \mid - M : \sigma \;\Rightarrow\; \Gamma \mid - M' : \sigma$$

Without proof

Strong normalization

Definition

A lambda term M is called *strongly normalizing* iff all reduction sequences starting with M terminate.

KIK is strongly normalizing, while **KIΩ** is not.

We are going to show that every term typable in $\lambda\!\rightarrow$ and $\lambda 2$ is strongly normalizing. This is not true for $\lambda\mu$ and $\lambda\cap$ since in these systems, all terms are typable.

We start with the proof of strong normalization for $\lambda\!\rightarrow$.

Definition.

(i) $SN = \{M \in \Lambda \mid M \text{ is strongly normalizing}\}$

(ii) Let $A, B \subseteq \Lambda$. Define a subset $A \rightarrow B$ of Λ as follows:

$$A \rightarrow B = \{F \in A \mid \forall a \in A(Fa \in B)\}$$

(iii) For every $\sigma \in \text{Type}(\lambda\!\rightarrow)$, we define a set $\|\sigma\| \subseteq \Lambda$ as follows

$$\|\alpha\| = SN \quad \text{if } \alpha \text{ is a type variable}$$

$$\|\sigma \rightarrow \tau\| = \|\sigma\| \rightarrow \|\tau\|$$

Definition.

(i) We call a subset $X \subseteq SN$ saturated if

(a) $(\forall n \geq 0)(\forall R_1, \ldots, R_n \in SN)[x\vec{R} \in X]$

where x is any term variable.

(b) $(\forall n \geq 0)(\forall R_1, \ldots, R_n \in SN)(\forall Q \in SN)$
$[P[x := Q]\vec{R} \in X \Rightarrow (\lambda x.P)Q\vec{R} \in X]$

(ii) $SAT = \{X \subseteq \Lambda \mid X \text{ is saturated}\}$

Note that saturated sets are non-empty, as they contain all term variables, and that they are closed under a particular type of expansion.

Lemma on saturated sets.

(i) $SN \in SAT$

(ii) $A, B \in SAT \Rightarrow A \rightarrow B \in SAT$

(iii) Let $\{A_i\}_{i \in I}$ be a collection of members of SAT, then
$\bigcap_{i \in I} A_i \in SAT$

(iv) For all $\sigma \in \text{Type}(\lambda\!\rightarrow)$ one has $\|\sigma\| \in \text{SAT}$

Proof.

(i) Obviously $SN \subseteq SN$ and it satisfies the condition (a). As to the condition (b), suppose

$$P[x := Q]\vec{R} \in SN \;\&\; Q, \vec{R} \in SN \qquad (1)$$

We claim that also

$$(\lambda x.P)Q\vec{R} \in SN \qquad (2)$$

Note that the reductions inside P, Q or the $\vec{R}$ must terminate since these terms are strongly normalising by assumption. The term

$$P[x := Q]$$

is a subterm of a term in SN by (1) hence it is itself in SN and, consequently, P is i n SN. So after finitely many reduction steps applied to the term in (2), we obtain

$$(\lambda x.P')Q'\vec{R}' \;\&\; P \to\!>_{\beta} P' \text{ etcetera}$$

Then the contraction of $(\lambda x.P')Q'\vec{R}'$ gives

$$P'[x := Q']\vec{R}' \qquad (3)$$

This is a reduct of $P[x := Q]\vec{R}$ and since this term is SN, then (3) and the term $(\lambda x.P)Q$ are SN.

(ii) Let $A, B \in SAT$. Then by definition $x \in A$ for all variables x.

Hence

$$F \in A \to B \Rightarrow Fx \in B$$
$$\Rightarrow Fx \in SN$$
$$\Rightarrow F \in SN$$

So indeed $A \to B \subseteq SN$. We prove the condition (i) (a) of saturation, let $\vec{R} \in SN$. We must show for a variable x that $x\vec{R} \in A \to B$ which means

$$\forall Q \in A\,(x\vec{R}Q \in B)$$

which is true since $A \subseteq SN$ and B is saturated.

(iii) Similarly

(iv) By induction on the generation of σ, using (i) and (ii).

In order to prove the key Soundness Theorem, we need the folowing

Definition.

(i) A *valuation* in Λ is a map $\rho : V \to \Lambda$, where V is the set of term variables.

(ii) Let ρ be a valuation in Λ. We define

$$\|M\|_{\rho} = M[x_1 := \rho(x_1), \ldots, x_n := \rho(x_n)]$$

where $\vec{x} = x_1, \ldots, x_n$ is the set of free variables in M.

(iii) Let ρ be a valuation in Λ. We say that ρ satisfies $M : \sigma$ and write $\rho \models M : \sigma$, if $\|M\|_{\rho} \in \|\sigma\|$.

If Γ is a basis, we say that ρ *satisfies* Γ and write $\rho \models \Gamma$, if $\rho \models x : \sigma$ for all $(x : \sigma) \in \Gamma$.

(iv) A basis Γ *satisfies* $M : \sigma$ and write $\Gamma \models M : \sigma$, if

$$\forall \rho\,[\rho \models \Gamma \Rightarrow \rho \models M : \sigma]$$

(iii) Let ρ be a valuation in Λ. We say that ρ satisfies $M : \sigma$ and write $\rho \models M : \sigma$, if $\|M\|_\rho \in \|\sigma\|$.

If Γ is a basis, we say that ρ *satisfies* Γ and write $\rho \models \Gamma$, if $\rho \models x : \sigma$ for all $(x : \sigma) \in \Gamma$.

(iv) A basis Γ *satisfies* $M : \sigma$ and write $\Gamma \models M : \sigma$, if

$$\forall \rho [\rho \models \Gamma \Rightarrow \rho \models M : \sigma]$$

Soundness Theorem.

$$\Gamma \vdash_{\lambda\to} M : \sigma \Rightarrow \Gamma \models M : \sigma$$

Proof.

By induction on derivation of $M : \sigma$.

Case 1. If $M \equiv x$ and $\Gamma \vdash M : \sigma$ follows from $(x : \sigma) \in \Gamma$, then trivially $\Gamma \models x : \sigma$.

Case 2. If $M \equiv M_1 M_2$ and $\Gamma \vdash M : \sigma$ is a direct consequence of $\Gamma \vdash M_1 : \tau \to \sigma$ and $\Gamma \vdash M_2 : \tau$, In order to show $\rho \models M_1 M_2 : \sigma$, We suppose $\rho \models \Gamma$. Then $\rho \models M_1 : \tau \to \sigma$ and $\rho \models M_2 : \tau$ which means

$$\|M_1\|_\rho \in \|\tau \to \sigma\| = \|\tau\| \to \|\sigma\| \text{ and } \|M_2\|_\rho \in \|\tau\|$$

But then

$$\|M_1 M_2\|_\rho = \|M_1\|_\rho \|M_2\|_\rho \in \|\sigma\| \text{ which means } \rho \models M_1 M_2 : \sigma$$

Case 3. Let $M \equiv \lambda x.M', \Gamma \vdash M : \sigma$ and let $\sigma \equiv \sigma_1 \to \sigma_2$ be a direct consequence of $\Gamma, x : \sigma_1 \vdash M' : \sigma_2$.

By the induction hypothesis, we have

$$\Gamma, x : \sigma_1 \models M' : \sigma_2 \qquad (1)$$

In order to show $\rho \models \lambda x.M' : \sigma_1 \to \sigma_2$, suppose $\rho \models \Gamma$. We have to show

$$\|\lambda x.M'\|_\rho N \in \|\sigma_2\| \text{ for all } N \in \|\sigma_1\|$$

Let $N \in \|\sigma_1\|$. Then $\rho(x := N) \models \Gamma, x : \sigma_1$, and hence

$$\|M'\|_{\rho(x:=N)} \in \|\sigma_2\|$$

by (1).

Since

$$\begin{aligned} \|\lambda x.M'\|_\rho N &\equiv (\lambda x.M')[\bar{y} := \rho(y)]N \\ &\to_\beta M'[\bar{y} := \rho(\bar{y}), x = N] \\ &\equiv \|M'\|_{\rho(x:=N)} \end{aligned}$$

it follows from the saturation of $\|\sigma_2\|$ that $\|\lambda x.M'\|_\rho N \in \|\sigma_2\|$.

Strong normalization theorem for $\lambda\rightarrow$ -Curry

Suppose

$$\Gamma \vdash_{\lambda\rightarrow} M : \sigma$$

then M is strongly normalizing.

Proof.

Suppose $\Gamma \vdash M : \sigma$. Then $\Gamma \models M : \sigma$ according to the Soundness Theorem. If we put $\rho_0(x) = x$ for all x, then $\rho_0 \models \Gamma$. Note that $x \in \|\tau\|$ since $\|\tau\|$ is saturated.

Therefore $\rho_0 \models M : \sigma$, hence $M \equiv \|M\|_{\rho_0} \in \|\sigma\| \subseteq SN$.

Proof.

Suppose $\Gamma \vdash M : \sigma$. Then $\Gamma \models M : \sigma$ according to the Soundness Theorem. If we put $\rho_0(x) = x$ for all x, then $\rho_0 \models \Gamma$. Note that $x \in \|\tau\|$ since $\|\tau\|$ is saturated.

Therefore $\rho_0 \models M : \sigma$, hence $M \equiv \|M\|_{\rho_0} \in \|\sigma\| \subseteq SN$.

Remark.

A simple generalization of the method proves the Strong Normalization Theorem for λ2.

Definition.

(i) A valuation in SAT is a map

$$\xi : \mathbf{V} \rightarrow SAT$$

Where $\mathbf{V}$ is the set of type variables.

(ii) Given a valuation ξ in SAT one defines a set $\|\sigma\|_\xi \subseteq \Lambda$ for every type σ in $\lambda 2$ as follows:

$$\|\alpha\|_\xi = \xi(\alpha), \text{ where } \alpha \in \mathbf{V}$$

$$\|\sigma \rightarrow \tau\|_\xi = \|\sigma\|_\xi \rightarrow \|\tau\|_\xi$$

$$\|\forall \alpha.\sigma\|_\xi = \bigcap_{X \in SAT} \|\sigma\|_{\xi(\alpha := X)}$$

Lemma.

Given a valuation ξ in SAT and a type σ in $\lambda 2$, then $\|\sigma\|_\xi \in SAT$.

Proof.

As the proof of (iv) in lemma on saturated sets using the fact that SAT is closed under arbitrary intersections.

Definition.

Let ρ be a valuation in Λ and ξ be a valuation in SAT.

(i) We write $\rho,\xi \models M:\sigma$ iff $\|M\|_\rho \in \|\sigma\|_\xi$

(ii) If Γ is a basis, we write

$$\rho,\xi \models \Gamma \text{ iff } \rho,\xi \models x:\sigma \text{ for all } x:\sigma \text{ in } \Gamma$$

(iii) We write

$$\Gamma \models M:\sigma \text{ iff } \forall \rho,\xi\,[\rho,\xi \models \Gamma \Rightarrow \rho,\xi \models M:\sigma]$$

Soundness Theorem for $\lambda 2$.

$$\Gamma \vdash_{\lambda 2} M:\sigma \;\Rightarrow\; \Gamma \models M:\sigma$$

Soundness Theorem for $\lambda 2$.

$$\Gamma \vdash_{\lambda 2} M:\sigma \;\Rightarrow\; \Gamma \models M:\sigma$$

Proof.

By induction on the derivation of $\Gamma \vdash M:\sigma$ as in the proof of Soundness Theorem for $\lambda\!\rightarrow$. There are two more cases corresponding to $\forall$–rules.

Case 4. $\Gamma \vdash M:\sigma$ where $\sigma \equiv \sigma_0[\alpha := \tau]$ is a direct consequence of $\Gamma \vdash M:\forall\alpha.\sigma_0$. By the Induction Hypothesis, we have

$$\Gamma \models M:\forall\alpha.\sigma_0 \qquad (I)$$

In order to show

$$\rho,\xi \models M:\sigma_0[\alpha := \tau]$$

suppose

$$\rho,\xi \models \Gamma.$$

It follows from (1) that

$$\|M\|_\rho \in \|\forall\alpha.\sigma_0\|_\xi = \bigcap_{X \in SAT} \|\sigma_0\|_{\xi(\alpha:=X)}$$

Hence

$$\|M\|_\rho \in \|\sigma_0\|_{\xi(\alpha:=\|\tau\|_\xi)}$$

By induction on $\sigma_0 \in \text{Type}(\lambda 2)$ (some care is needed in case $\sigma_0 \equiv \forall\beta.\tau_0$)

we prove

$$\|\sigma_0\|_{\xi(\alpha:=\|\tau\|_\xi)} = \|\sigma_0[\alpha := \tau]\|$$

which completes the proof of the Case 4.

Case 5. Let $\Gamma \mid\!- M : \sigma$ with $\sigma \equiv \forall\alpha.\sigma_0$ and $\alpha \notin FV(\Gamma)$ is a direct consequence of $\Gamma \mid\!- M : \sigma_0$. By the Induction Hypothesis, we have

$$\Gamma \mid\!= M : \sigma_0 \qquad (2)$$

In order to show $\rho, \xi \mid\!= M : \forall\alpha.\sigma_0$, we suppose $\rho, \xi \mid\!= \Gamma$. Since $\alpha \notin FV(\Gamma)$,

we have $\rho, \xi(\alpha := X) \mid\!= \Gamma$ for all $X \in SAT$. Therefore

$$\|M\|_\rho \in \|\sigma_0\|_{\xi(\alpha:=X)} \text{ for all } X \in SAT$$

It follows from (2) that

$$\|M\|_\rho \in \|\forall\alpha.\sigma_0\|_\xi$$

Hence

$$\rho, \xi \mid\!= M : \underbrace{\forall\alpha.\sigma_0}_{\sigma}$$

Strong Normalization Theorem for λ2-Curry

$\Gamma \mid\!-_{\lambda 2} M : \sigma \;\Rightarrow\; M$ is strongly normalizing

Strong Normalization Theorem for λ2-Curry

$\Gamma \mid\!-_{\lambda 2} M : \sigma \;\Rightarrow\; M$ is strongly normalizing

Proof of the theorem is similar to the proof of Strong Normalizing Theorem for $\lambda \rightarrow$–Curry.

Decidability of type assignment.

Note that for arbitrary base $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ one has

$$\Gamma \vdash M : \sigma \quad \Leftrightarrow \quad \vdash (\lambda x_1 \ldots \lambda x_n . M) : (\sigma_1 \to \ldots \sigma_n \to \sigma)$$

Consequently, analysing the type assignment, we may assume that the base is always empty. Typical questions are

- Given M and σ, does it hold $\vdash M : \sigma$?
- Given M, does there exist a σ such that $\vdash M : \sigma$?
- Given σ, does there exist an M such that $\vdash M : \sigma$?

These three problems are called *type checking, typability* and *inhabitation* respectively and we shall denote them $M : \sigma ?$, $M : ?$, and $? : \sigma$.

We shall examine the decidability of these three problems for the various systems of type assignments. The results can be summarized in the following table

	$M:\sigma?$	$M:?$	$?:\sigma$
$\lambda\to$	yes	yes	yes
$\lambda 2$	??	??	no
$\lambda\mu$	yes	yes, always	yes, always
$\lambda\cap$	no	yes, always	??
$\lambda\to^+$	no	no	yes
$\lambda 2^+$	no	no	no
$\lambda\mu^+$	no	yes,always	yes, always
$\lambda\to A$	no	no	yes, always
$\lambda 2A$	no	no	yes, always
$\lambda\mu A$	no	yes, always	yes, always
$\lambda\cap A$	no	yes, always	yes, always

We first show decidability of the three questions for $\lambda\to$–Curry. In what follows $\mathbf{T}$ denotes Type($\lambda\to$) and $\vdash$ denotes $\vdash_{\lambda\to Curry}$.

We define some operations on the set of types: substitutor, unifier, unification and some concepts concerning type assignments: pricipal pair and principal type.

Definition.

(i) A *substitutor* is an operation $* : \mathbf{T} \to \mathbf{T}$ on the set of types such that

$$*(\sigma \to \tau) \equiv *(\sigma) \to *(\tau)$$

Notation.

a) We write σ^* for $*(\sigma)$.

b) In cases that are of interest for us, a substitutor $*$ has a finite support, which means that for all but finitely many type variables α one has $\alpha^* \equiv \alpha$, the support of $*$ being $\sup(*) = \{\alpha \mid \alpha^* \not\equiv \alpha\}$.

In that case we write

$$*(\sigma) = \sigma[\alpha_1 := \alpha_1^*, \ldots, \alpha_n := \alpha_n^*]$$

Where $\{\alpha_1, \ldots, \alpha_n\}$ is the support of $*$. We also write

$$* = [\alpha_1 := \alpha_1^*, \ldots, \alpha_n := \alpha_n^*].$$

Definition. Unifiers.

(i) Let σ and τ be two types. A *unifier* for σ and τ is a substitutor $*$ such that $\sigma^* \equiv \tau^*$.

(ii) The substitutor $*$ is a most general unifier for σ and τ if

(a) $*$ is a unifier for σ and τ

(b) if $*_1$ an arbitrary unifier for σ and τ, then there is a substitutor $*_2$ such that $*_1 \equiv *_2 \circ *$.

(iii) Let $E = \{\sigma_1 = \tau_1, \ldots, \sigma_n = \tau_n\}$ be a finite set of equations between types. The equations do not need to be valid. A *unifier* for E is a substitutor $*$ such that $\sigma_1^* = \tau_1^*, \ldots, \sigma_n^* = \tau_n^*$. In that case one writes $* \models E$. Similarly one defines the notion of a most general unifier for E.

Examples.

The types

$$\beta \to (\alpha \to \beta) \qquad (\gamma \to \gamma) \to \delta$$

have unfiers

$$* = [\beta := \gamma \to \gamma, \delta := \alpha \to (\gamma \to \gamma)]$$
$$*_1 = [\beta := \gamma \to \gamma, \alpha := \varepsilon \to \varepsilon, \delta := (\varepsilon \to \varepsilon) \to (\gamma \to \gamma)]$$

The unifier $*$ is most general, the unifier $*_1$ is not.

Definition. Variants.

The type σ is a variant of the type τ if there are substitutors $*_1, *_2$ such that

$$\sigma = \tau^{*_1} \quad \text{and} \quad \tau = \sigma^{*_2}$$

Examples.

$\alpha \to \beta \to \beta$ and $\gamma \to \delta \to \delta$ are variants of each other

$\alpha \to \beta \to \beta$ is not a variant of $\alpha \to \beta \to \alpha$

Note that if $*_1$ and $*_2$ are two most general unifiers of types σ and τ then σ^{*_1} and σ^{*_2} are variants of each other and similarly for τ.

Unification Theorem.

(i) There is a recursive function U with input (after coding) a pair of types and with output which is either a substitutor or fail such that

$$U(\sigma,\tau) = \begin{cases} \text{a most general unifier for } \sigma \text{ and } \tau \\ \qquad \text{if } \sigma \text{ and } \tau \text{ have a unifier} \\ \text{fail} \qquad \text{if } \sigma \text{ and } \tau \text{ have no unifier} \end{cases}$$

(ii) There is a recursive function U with input (after coding) finite sets of equations between types and with output either a substitutor or fail such that

$$U(E) = \begin{cases} \text{a most general unifier for E} \\ \qquad \text{if E has a unifier} \\ \text{fail} \qquad \text{if E has no unifier} \end{cases}$$

Proof.

Note that $\sigma_1 \to \sigma_2 \equiv \tau_1 \to \tau_2 \quad \Leftrightarrow \quad \sigma_1 \equiv \tau_1 \;\&\; \sigma_2 \equiv \tau_2$

(i) Define $U(\sigma,\tau)$ by the following recursive loop with case distinction.

$$U(\alpha,\tau) = \begin{cases} [\alpha := \tau] & \text{if } \alpha \notin FV(\tau) \\ Id \quad \text{(the identity)} & \text{if } \tau = \alpha \\ \text{fail} & \text{else} \end{cases}$$

$$U(\sigma_1 \to \sigma_2, \alpha) = U(\alpha, \sigma_1 \to \sigma_2)$$

$$U(\sigma_1 \to \sigma_2, \tau_1 \to \tau_2) = U(\sigma_1^{U(\sigma_2,\tau_2)}, \tau_1^{U(\sigma_2,\tau_2)}) \circ U(\sigma_2, \tau_2)$$

Where the last expression is considered to be fail if one of its parts is.

Proof.

(i) By induction on the lexicografic order of pairs of natural numbers defined as follows:

$\#_{\text{var}}(\sigma,\tau) =$ the number of variables in $\sigma \to \tau$ and $\#_{\to}(\sigma,\tau) =$ the number of arrows in $\sigma \to \tau$.

By induction on pairs $(\#_{\text{var}}(\sigma,\tau), \#_{\to}(\sigma,\tau))$ ordered lexicographically, one can show that $U(\sigma,\tau)$ is always defined. Moreover U satisfies the specification.

(ii) If $E = \{\sigma_1 = \tau_1, \ldots, \sigma_n = \tau_n\}$ then define $U(E) = U(\sigma,\tau)$,

Where

$\sigma = \sigma_1 \to \cdots \to \sigma_n$ and $\tau = \tau_1 \to \cdots \to \tau_n$.

Proposition on substitutors and equations.

Let Γ be a basis, $M \in \Lambda$ a term and $\sigma \in \mathbf{T}$ a type such that $FV(M) \subseteq \text{dom}(\Gamma)$. Then there is a finite set of equations $E = E(\Gamma, M, \sigma)$ such that for all substitutors $*$ one has

$$*\models E(\Gamma,M,\sigma) \;\Rightarrow\; \Gamma^* \vdash M : \sigma^* \qquad (1)$$

$$\Gamma^* \vdash M : \sigma^* \;\Rightarrow\; *_1 \models E(\Gamma,M,\sigma) \qquad (2)$$

For some $*_1$ such that $*$ and $*_1$ have the same effect on type variables in Γ and σ.

Proof.

Define the set $E(\Gamma,M,\sigma)$ by induction on the structure of M:

$$E(\Gamma,x,\sigma) = \{\sigma = \Gamma(x)\}$$

$$E(\Gamma,MN,\sigma) = E(\Gamma,M,\alpha \to \sigma) \cup E(\Gamma,N,\alpha)$$
where α is a fresh variable

$$E(\Gamma,\lambda x.M,\sigma) = E(\Gamma \cup \{x:\sigma\},M,\beta) \cup \{\alpha \to \beta = \sigma\}$$
where α, β are fresh variables

By induction on M one can show (using generation lemma) that (1) and (2) hold.

Definition. Principal pair, principal type.

(i) Let $M \in \Lambda$. Then (Γ,σ) is a *principal pair* (pp) for M if

(*1*) $\Gamma \vdash M : \sigma$

(*2*) $\Gamma' \vdash M : \sigma' \Rightarrow \exists * [\Gamma^* \subseteq \Gamma' \ \& \ \sigma^* \equiv \sigma']$

Here $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}^* = \{x_1 : \sigma_1^*, \ldots, x_n : \sigma_n^*\}$.

(ii) Let $M \in \Lambda$ be closed. Then σ is a *principal type* (pt) for M if

(*1*) $\vdash M : \sigma$

(*2*) $\vdash M : \sigma' \Rightarrow \exists * [\sigma^* \equiv \sigma']$

Remarks.

Note that if (Γ,σ) is a principal pair for M, then every variant (Γ',σ') of (Γ,σ), in the obvious sense, is a principal pair for M.

Conversely, if (Γ,σ) and (Γ',σ') are both principal pairs for M, then (Γ',σ') is a variant of (Γ,σ).

Moreover, if (Γ,σ) is a principal pair for M, then $FV(M) = \mathrm{dom}(\Gamma)$.

Principal Type Theorem for $\lambda\!\to$–Curry.

(i) There exists (after coding) a recursive function pp such that

$$pp(M) = \begin{cases} (\Gamma,\sigma) & \text{a principal pair for } M \text{ if a type for } M \text{ exists} \\ \text{fail} & \text{if } M \text{ has no type} \end{cases}$$

(ii) There exists (after coding) a recursive function pt such that for closed terms M one has

$$pt(M) = \begin{cases} \sigma & \text{a principal type for } M \quad \text{if a type for } M \text{ exists} \\ \text{fail} & \text{if } M \text{ has no type} \end{cases}$$

Proof.

(i) Let $FV(M) = \{x_1, \ldots, x_n\}$ and define $\Gamma_0 = \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$.

If we put $\sigma_0 = \beta$, we have

$$\begin{aligned} M \text{ has a type} &\Leftrightarrow \exists \Gamma \exists \sigma \ \Gamma \vdash M : \sigma \\ &\Leftrightarrow \exists * \ \Gamma_0^* \vdash M : \sigma_0^* \\ &\Leftrightarrow \exists * \quad * \models E(\Gamma_0, M, \sigma_0) \end{aligned}$$

Define

$$pp(M) = \begin{cases} (\Gamma_0^*, \sigma_0^*) & \text{if } U(E(\Gamma_0, M, \sigma_0)) = * \\ \text{fail} & \text{if } U(E(\Gamma_0, M, \sigma_0)) = \text{fail} \end{cases}$$

Then $pp(M)$ satisfies the statement (i) of the theorem. Indeed, if M has a type, then $U(E(\Gamma_0, M, \sigma_0)) = *$ and $\Gamma_0^* \vdash M : \sigma_0^*$ follows from (i) in the proposition on substitutors and equations.

To show that (Γ_0^*, σ_0^*) is a principal pair, suppose that also $\Gamma' \vdash M : \sigma'$.

Let $\tilde{\Gamma} = \Gamma' | FV(M)$, write $\tilde{\Gamma} = \Gamma_0^*$ and $\sigma' = \sigma_0^{*_0}$. Then also

$$\Gamma_0^{*_0} \vdash M : \sigma_0^{*_0}$$

Hence by (ii) in the proposition on substitutors and equations, there is $*_1$, acting in the same way as $*_0$ on Γ_0, σ_0 such that $*_1 \models E(\Gamma_0, M, \sigma_0)$.

By the Unification Theorem $*$ is a most general unifier, hence there is a $*_2$ such that $*_1 = *_2 \circ *$.

Now

$$(\Gamma_0^*)^{*_2} = \Gamma_0^{*_1} = \Gamma_0^{*_0} = \tilde{\Gamma} = \Gamma'$$

and

$$(\sigma_0^*)^{*_2} = \sigma_0^{*_1} = \sigma_0^{*_0} = \sigma'$$

This completes the case when M has a type.

If M has no type, then there is no substitutor satisfying $E(\Gamma_0, M, \sigma_0)$,

hence

$$U(E(\Gamma_0, M, \sigma_0)) = \text{fail} = pp(M)$$

(ii) Let M be closed and $pp(M) = (\Gamma, \sigma)$. Then $\Gamma = \emptyset$ and we can put

$$pt(M) = \sigma$$

Proof.

(a) Type checking: given M and σ, we have

$$\vdash M : \sigma \quad \Leftrightarrow \quad \exists * [\sigma = pt(M)^*]$$

This is decidable by a *pattern matching* algorithm similar to the unification algorithm.

(b) Typability: given M, then M has a type iff $pt(M) \neq \text{fail}$.

Decidability of the inhabitation problem for $\lambda \to -$ is shown equivalent to provability of σ in the minimal intuitionistic proposition calculus PROP with only $\to$ as connective and σ considered as an element of PROP. Using finite Kripke models it can be shown that provability of σ is decidable.

Theorem.

the inhabitation problem for $\lambda \to -$, that is $\exists M \in \Lambda \vdash_{\lambda\to} M : \sigma$ is a decidable property of σ.

Decidability of the inhabitation problem for $\lambda\!\rightarrow\!-$ is shown equivalent to provability of σ in the minimal intuitionistic proposition calculus PROP with only $\rightarrow$ as connective and σ considered as an element of PROP. Using finite Kripke models it can be shown that provability of σ is decidable.

Theorem.

the inhabitation problem for $\lambda\!\rightarrow$, that is $\exists M \in \Lambda \,|-_{\lambda\rightarrow} M : \sigma$ is a decidable property of σ.

Proof.

σ is inhabited in $\lambda\!\rightarrow$-Curry $\Leftrightarrow \sigma$ is inhabited in $\lambda\!\rightarrow$-Church
$\Leftrightarrow \sigma$ is provable in PROP

Now, we consider $\lambda 2$. The question whether type checking and typability is open. There is only a result showing that the problem of typability in $\lambda 2$ can be reduced to that of type checking.

Proposition.

$\{(M : \sigma) \,|\, |-_{\lambda 2} M : \sigma\}$ is decidable $\Rightarrow \{M \,|\, \exists \sigma \,|-_{\lambda 2} M : \sigma\}$ is decidable

Proof.

One has

$$\exists \sigma \,|- M : \sigma \Leftrightarrow |- (\lambda xy.y) M : (\alpha \rightarrow \alpha)$$

The implication $\Rightarrow$ is obvious, since

$$|- (\lambda xy.y) : (\sigma \rightarrow \alpha \rightarrow \alpha) \quad \text{for all } \sigma$$

The other implication follows from the lemma on typability of subterms.

Theorem.

The inhabitation problem for $\lambda 2$ is undecidable.

Proof.

As for $\lambda\!\rightarrow$, one can show the first equivalence

σ is inhabited in $\lambda 2-$Curry $\Leftrightarrow \sigma$ is inhabited in $\lambda 2-$Church
$\Leftrightarrow \sigma$ is provable in PROP2

where PROP2 is the constructive second-order propositional calculus. Löb (1976) proved that the last property is undecidable.

Theorem.

For $\lambda\mu$ one has the following:

(i) Type checking is decidable.

(ii) Typability is trivially decidable, we showed that every λ-term has a type.

(iii) The inhabitation problem for $\lambda\mu$ is trivially decidable: all types are inhabited.

Proof.

(i) Use the same method as for $\lambda\!\rightarrow$ and the fact that $T(\sigma) = T(\tau)$ is decidable.

(ii) In a motivation example for λμ, we have shown that every λ-term has a type σ_0, where $\sigma_0 = \mu\alpha.\alpha \rightarrow \alpha$.

(iii) All types are inhabited by the term Ω.

Lemma. Systems with subject conversion

Let λ- be a system of type assignment satisfying subject conversion i.e.

$$\Gamma \mid\!-_{\lambda-} M : \sigma \;\&\; M =_\beta N \Rightarrow \Gamma \mid\!-_{\lambda-} N : \sigma$$

(i) Suppose that some closed terms have the type $\alpha \rightarrow \alpha$ and others not. Then the problem of type checking is undecidable.

(ii) Suppose that some terms have a type and others not. Then the problem of typability is undecidable.

Proof.

(i) If the set $\{(M,\sigma) \mid\mid \; \mid\!- M : \sigma\}$ is decidable, then so is the set

$$\{M \mid\mid \; \mid\!- M : \alpha \rightarrow \alpha\}.$$

This set is by assumption closed under $=$ and non-trivial, hence by the Scott´s theorem is not recursive, a contradiction.

(ii) Similarly.

Proposition.

For $\lambda\cap$ one has the following

(i) Type checking problem is undecidable.

(ii) Typability is trivially decidable: all terms have a type.

Proof.

(i) Using the subject conversion for $\lambda\cap$, the statement (i) of the previous lemma applies and there are facts

$$\mid\!- \mathbf{I} : \alpha \rightarrow \alpha \quad \text{and} \quad \mid\!\neq \mathbf{K} : \alpha \rightarrow \alpha$$

(ii) For all M one has $M : \omega$.

It is not known whether inhabitation in $\lambda\cap$ is decidable.

Lemma on reduction.

Let $\lambda-$ be one of the systems á la Curry. Then we have

(i) $\Gamma \mid -_{\lambda-^+} M:\sigma \;\Leftrightarrow\; \exists M'[M \twoheadrightarrow_\beta M' \,\&\, \Gamma \mid -_{\lambda-} M':\sigma]$

(ii) σ is ihabited in $\lambda-^+ \;\Leftrightarrow\;$ σ is inhabited in $\lambda-$

Proof.

(i)$(\Leftarrow)$ is trivial since $M \twoheadrightarrow_\beta M'$ implies $M =_\beta M'$.

$(\Rightarrow)$ by induction on the derivation of $M:\sigma$.

The only interesting case is when the last applied rule is an application of rule EQ. Suppose

$$\frac{M_1:\sigma \qquad M_1 =_\beta M}{M:\sigma}$$

The induction hypothesis says that there is M_1' such that $M_1 \twoheadrightarrow_\beta M_1'$ and one has $\Gamma \mid -_{\lambda-} M_1':\sigma$. By the Church-Rosser theorem, M_1' and M have a common reduct, say M'. But by the subject reduction theorem, we have $\Gamma \mid -_{\lambda-} M':\sigma$ and the proof is complete.

(ii) By (i).

Proposition. The systems $\lambda-^+$.

For the systems $\lambda-^+$ one has the following:

(i) Type checking is undecidable

(ii) Typability is undecidable for $\lambda\!\rightarrow^+$ and $\lambda 2^+$, but trivially decidable for $\lambda\mu^+$ and $\lambda\cap^+$.

(iii) The status of the inhabitation problem is the same for both $\lambda-^+$ and $\lambda-$.

Proof.

(i) Subject conversion holds for the systems $\lambda-^+$ by definition. In all systems $\mathbf{I}:\alpha\rightarrow\alpha$. It follows from (i) of the lemma on reduction and the fact that $\mid\!\neq \mathbf{K}:\alpha\rightarrow\alpha$ that type checking is undecidable by (i) of the lemma on systems $\lambda-$.

(ii) We have already shown that terms without a normal form have no type in $\lambda\!\rightarrow$ and $\lambda 2$. Hence by the reduction lemma these terms have no type in $\lambda\!\rightarrow^+$ or $\lambda 2^+$. Since for these systems there are terms that have a type by (ii) of lemma on systems with subject conversion the undecidability of typability for $\lambda\!\rightarrow$ and $\lambda 2$ follows.

(iii) By (ii) of the reduction lemma.

Lemma. Typing of normal forms.

Let M be a term in normal form. Then

$$\mid -_{\lambda-A} M:\sigma \;\Rightarrow\; \mid -_{\lambda-} M:\sigma$$

Lemma. Typing of normal forms.

Let M be a term in normal form. Then

$$\vdash_{\lambda-\mathbf{A}} M:\sigma \;\Rightarrow\; \vdash_{\lambda-} M:\sigma$$

Proof.

By induction on the given derivation, using the fact that $M \in \mathbf{A}(M)$.

Proposition. Systems $\lambda-\mathbf{A}$.

For systems $\lambda-\mathbf{A}$, we have the following:

(i) The problem of type checking is undecidable for the systems $\lambda\to\mathbf{A}$, $\lambda 2\mathbf{A}$, $\lambda\mu\mathbf{A}$ and $\lambda\cap\mathbf{A}$.

(ii) The problem of typability is undecidable for the systems $\lambda\to\mathbf{A}$, and $\lambda 2\mathbf{A}$, but it is trivially decidable for the systems $\lambda\mu\mathbf{A}$ and $\lambda\cap\mathbf{A}$ (all terms are typable).

(iii) The problem of inhabitation is trivially decidable for all four systems includin the rule $\mathbf{A}$ (all types are inhabited).

Proof.

(i) By lemma on typing normal forms and the fact that $\vdash \mathbf{K}:\alpha\to\alpha$ in all four basic Curry systems and (i) of subject conversion systems lemma, we get undecidability.

(ii) similarly.

(iii) The inhabitation problem becomes trivial: in all four systems one has $\vdash \mathbf{\Omega}:\sigma$ for all types σ. This follows from the facts that $\vdash_{\lambda\to\mathbf{A}} \mathbf{Y}:((\sigma\to\sigma)\to\sigma)$, $\mathbf{YI}=_\beta \mathbf{\Omega}$ and $\lambda-\mathbf{A}$ is closed under the rule EQ.