**Assignment-Simply typed λ-calculus**

**Student ID: u5323288**

**Name: Mingyuan Cui**

**Question 1 (10 marks) Find a most general unifier for the set of pairs**

E= {(α → γ → δ, α`→ β`), (α → β → γ, β`→ α`)}

**Show all the steps of the unification algorithm shown in lectures.**

**Answer:**

E= {(α → γ → δ, α`→ β`), (α → β → γ, β`→ α`)}

U= {} ↓ 4.c

E= {(α, α`), (γ → δ, β`), (α → β → γ, β`→ α`)}

U= {α := α`} ↓ 4.b

E= {(γ → δ, β`), (α` → β → γ, β`→ α`)}

U= {α := α`} ↓ 4.c

E= {(γ → δ, β`), (α`, β`), (β → γ, α`)}

U= {α := β`, α`:=β`} ↓ 4.b

E= {(γ → δ, β`), (β → γ, β`)}

U= {α := γ → δ, α`:= γ → δ, β`:=γ → δ } ↓ 4.b

E= {( β → γ, γ → δ)}

U= {α := γ → δ, α`:= γ → δ ,β`:=γ → δ } ↓ 4.c

E= {( β, γ), (γ, δ) }

U= {α := γ → δ, α`:= γ → δ , β`:=γ → δ , β :=γ } ↓ 4.b

E= { (γ, δ) }

U= {α := δ → δ, α`:= δ → δ , β`:= δ → δ, β := δ, γ :=δ } ↓ 4.b

E={}

The most general unifier is U={ α := δ → δ, α`:= δ → δ , β`:= δ → δ, β := δ, γ :=δ }

U(α → γ → δ, α`→ β`)=

(δ → δ)→ δ → δ, (δ → δ) → δ → δ =

U(α → β → γ, β`→ α`)

**Question 2 (25 marks) For each term below, use the Principal Type Algorithm (PT) to determine whether it is typeable in λCu. If it is typeable, give a principal type and a principal deduction. Show all the steps used in applying the PT Algorithm and in finding any necessary unifiers.**

**Answer:**

**(a) λfλz.f (f z)**

First: f z

Δf:

→axiom

Δz:

→axiom

FV(f)={f}

FV(z)={z}

(1)

(2)γγ

E={ , γ→ }

U={ := γ→ }↓ 4.b

E={}

(1\*) f: γ→

(2\*) z: γ

U(

Δ(f z):

γ

Then: f (f z)

Δf:

→axiom (rename)

Δ(f z):

γ

FV(f)={f}

FV(f z)={z,f}

(1)

(2)γ

E={,), ()}

U={:=}↓ 4.b

E={()}

U={:=}↓ 4.c

E={()}

U={:=}↓ 4.b

E={}

U={:=}↓ 4.b

E={}

(1\*)

(2\*)

U(

Δf (f z):

Then: λz.f (f z)

z∈FV(f (f z))

Δf (f z):

Δ λz.f (f z):

→intro

Finally: λfλz.f (f z)

f∈FV(λz.f (f z))

Δ λz.f (f z):

Δ λfλz.f (f z):

→intro

**(b) λxλy.x**

Fisrt: λy.x

y∉FV(x)

Δx:

→axiom

Δλy.x:

→intro

Then: λxλy.x

x∈FV(λy.x)

Δλy.x:

Δλxλy.x:

→intro

**(c) (λfλz.f (f z)) (λxλy.x)**

P:= λfλz.f (f z)

Q:= λxλy.x

M:= P Q

ΔP:

→intro

ΔQ:

→intro

FV(P)={}

FV(Q)={}

(1)

(2)

E={}

U={}↓ 4.c

E={}

U={}↓ 4.c

E={}

U ={}↓ 4.b

E={}

↓

FAIL

**Question 3 (15 marks) One way of encoding natural numbers in λ-calculus is via the so-called Church numerals. A number n is encoded as the λ-term λfλx.fnx, where fnx means f applied n times to x.**

**For example, 0 is represented as λfλx.x; 1 is represented as λfλx.f x, 2 is represented as λfλx.f (f x) and so on.**

**Write C1 = λfλx.f x, C2 = λfλx.f (f x), C3 = λfλx.f (f (f x)), etc. Thus Ci f = fi for any i.**

**Since fmn = (fm)n, we can define multiplication of these Church numerals by (T for "times")**

**T Cm Cn f = Cm(Cnf); that is, T = λmλnλf.m(n f)**

**Likewise, since fm+n(x) = fm(fn(x)) we can define addition of Church numerals by (P for "plus")**

**P Cm Cn f x = Cm f (Cn f x) that is, P = λmλnλfλx.mf (n f x)**

**Since T and P are both functions which take two Church numerals as arguments and produce a third Church numeral:**

**(a) Find the principal types for T and for P (using the definitions of T and P above)**

**Answer:**

(1) T = λmλnλf.m(n f)

Δn:

→axiom

Δf:

→axiom

Δ n f:

Δ m(n f):

Δ λf.m(n f):

→intro

Δ λnλf.m(n f):

→intro

Δ T:

→intro

(2) P = λmλnλfλx.mf (n f x)

Δn:

→axiom

Δf:

→axiom

Δx:

→axiom

Δ n f:

Δ n f x:

Δ m:

→axiom

Δ m f :

Δ m f (n f x):

Δ λx.mf (n f x):

→intro

Δλfλx.mf (n f x):

→intro

Δλnλfλx.mf (n f x):

→intro

ΔP:

→intro

**(b) Can the types of T and of P be unified to a common type? What is it?**

**Answer:**

E={} (Rename the variables)

U={}↓ 4.c

E={}

U={}↓ 4.c

E={}

U={}↓ 4.c

E={}

U={}↓ 4.c

E={}

U={}↓ 4.c

E={}

U={}↓ 4.b

E={}

U={}↓ 4.b

E={}

U={}↓ 4.b

E={}

U={}↓ 4.c

E={}

U={}↓ 2

E={}

U={}↓ 4.b

E={}

U={}↓ 4.b

E={}

U={→}

U(P)=U(T)=))→)→

**(c) How does this compare with the type you might expect T and P to have, given that they are binary operators on Church numerals? Why?**

**Answer:**

Ci = λfλx.f i (x),

Δf:

→axiom

Δx:

→axiom

Δ f x:

Δ f (f x) :

Δ f i (x):

f i (x):

Thus we can conclude that any church numeral has a type: for some

T Cm Cn = Cp for some m,n and p

T Cm Cn:

We can derive from PT algorithm that T has a principle type:

(

And it`s the same for P.

This type is derived from the definition and the type in (b) is derived from PT Algorithm.

It is possible that T and P`s expected type satisfies the common type in (b) as long as and :=.

That is, they can be unified.

**Question 4 (10 marks) Consider the term (λx.x) x**

**(a) In this term, identify the free and bound occurrences of x**

**Answer:**

The first x is bound and binding, the second x is bound and the rightmost x is free.

**(b) Give a typing derivation for this term (Hint: you may need to use α-equivalence)**

**Answer:**

(λx.x) x≡ (λy.y) x

P:= λy.y, Q:=x M:=P Q

y ∈FV(y)

Δy:

Δ λy.y:

→intro

Δx:

→axiom

FV(λy.y)={}

FV(x)={x}

(1)

(2)

E={()}

U={}↓ 4.c

E={ , ), , )}

U={:= }↓ 4.b

E={ , )}

U={:=:= }↓ 4.b

E={}

(1\*) λy.y:

(2\*)x:

U(

Δ(λy.y) x:

**Question 5 (20 marks) Assuming the substitution lemma,**

**(a) Prove that the basic -reduction step, (λx.M) N→β M[x := N], preserves the type: that is, if (λx.M) N : then M[x := N] :**

**Answer:**

Assume that Γ is some typing context

Γ(λx.M) N :

(λx.M) N is not a variable, that is, (λx.M) N : is not in Γ and can only be inferred by rule elim.

Thus Γλx.M: and ΓN: for some (By inversion of elim)

Again

Γ ,x:λx.M : for some

λx.M is not a variable, that is λx.M: is not in Γ and can only be inferred by rule intro.

Thus x: and M:, and = (By inversion of intro)

So x and N has the same type

Given that Γ ,x:M: and Γ N:,

then Γ:

And this is exactly what we want to show.

**(b) From this, prove that any single step -reduction of a term (including where the reduction takes place at a subterm of the term) preserves the type of the term**

**Answer:**

Assume M can be reduced to M` in more than one step.

Induction start: the basic -reduction step, (λx.M) N : then M`=M[x := N] : , it has already been proved in (a) (Basic case)

Induction step: Assume that, the first n steps -reduction of a term preserves the type of the term, that is , if M can be reduced to M` in n steps, M`: (Hypothesis)

(1) if M` is in form of (λx.M1) M2

then, M` can be reduced to M``≡M1[x:=M2] in one more step and M``:, with the same type as M` and M according to the basic case

(2) if M` is in form of M1 M2 and M1 can be reduced to M1`in the next step and

M1: ,M2 : for some and

We can conclude that = for some (By inversion of rule elim)

And that =, for M`≡ M1 M2:

M1` : according to the basic case

Then M1` M2: according to rule elim.

And M` can be reduced to M``≡M1` M2 in onemore step reduction according to the rule shown below:

(3) if M` is in form of M1 M2 and M2 can be reduced to M2`in the next step and

M1: ,M2 : for some and

We can conclude that = for some (By inversion of rule elim)

And that =, for M`≡ M1 M2:

M2` : according to the basic case

Then M1 M2`: according to rule elim.

And M` can be reduced to M``≡M1 M2`in onemore step reduction according to the rule shown below:

Thus M` can be reduced to M`` in one more step and M``:, with the same type as M` and M.

(4) if M` is in form of λx.M1 : and M1 can be reduced to M1` in the next step and

x: , M1 : for some and

We can conclude that (By rule intro)

And that M1`: according to the basic case.

M` can be reduced to M``≡ λx.M1``in onemore step reduction according to the rule shown below:

Thus M` can be reduced to M`` in one more step and M``:, with the same type as M` and M.

To sum up, the first n+1 steps -reduction of a term preserves the type of the term, that is, if M can be reduced to M`` in n+1 steps, M``:

Thus,any single step -reduction of a term preserves the type of the term.

**Question 6 (20 marks) We had an example in lectures of M β N, where N is typeable but M is not.**

**In that example, the explanation was that M is an abstraction which ignored its argument, and N is an untypeable argument. Here is an example with a different explanation of how this can happen.**

**(a) Show that (λx.x x) (λy. y) is not typeable.**

**Answer:**

Δx:

→axiom

FV(x)=x

(1)

(2)(Rename)

E={ }

U={:= }↓ 4.b

E={ }

↓

FAIL

Thus x x is untypeable and (λx.x x) (λy. y) is untypeable.

**(b) Find its λ-normal form.**

**Answer:**

(λx.x x) (λy. y) β (λy. y) (λy. y)β λy. y

**(c) Show that its λ-normal form is typeable.**

**Answer:**

Δy:

→axiom

y ∈FV(y)

Δ λy.y:

→intro

**(d) Can you explain how** β**-reduction changes an untypeable term to a typeable one in this case?**

**Answer:**

In our case, the term is untypeable due to the form of M M which is untypeable.

Assume that M:

The term M M can only possibly apply the rule elim due to its form but these two subterms have the same type which makes the term M M untypeable

If reductions can eliminate such kind of form, it may be typeable after reductions.

Recall the process of reductions

(λx.x x) (λy. y) β (λy. y) (λy. y)β λy. y

After the first reduction, the term (λy. y) (λy. y) is still preserve the form M M, thus still untypeable.

After the second reduction, the term λy. y is typeable because the M M form has been eliminated.