Essence of DOT calculus

System D<:

Syntax

x, y, z	**Variable**	$S, T, U ::=$	**Type**
$v ::=$	**Value**	$\top$	top type
$\{A = T\}$	type tag	$\bot$	bottom type
$\lambda(x{:}T)t$	lambda	$\{A : S..T\}$	type declaration
$s, t, u ::=$	**Term**	$x.A$	type projection
x	variable	$\forall(x{:}S)T$	dependent function
v	value		
$x\ y$	application		
$\textbf{let}\ x = t\ \textbf{in}\ u$	let		

Evaluation $\boxed{t \longrightarrow t}$

$$
\begin{array}{rcll}
\textbf{let}\ x = v\ \textbf{in}\ e[x\ y] & \longrightarrow & \textbf{let}\ x = v\ \textbf{in}\ e[[z := y]t] & \textbf{if}\ v = \lambda(z{:}T)t \\
\textbf{let}\ x = y\ \textbf{in}\ t & \longrightarrow & [x := y]t & \\
\textbf{let}\ x = \textbf{let}\ y = s\ \textbf{in}\ t\ \textbf{in}\ u & \longrightarrow & \textbf{let}\ y = s\ \textbf{in}\ \textbf{let}\ x = t\ \textbf{in}\ u & \\
e[t] & \longrightarrow & e[u] & \textbf{if}\ t \longrightarrow u \\
 & & \textbf{where}\ e ::= [\,]\ |\ \textbf{let}\ x = [\,]\ \textbf{in}\ t\ |\ \textbf{let}\ x = v\ \textbf{in}\ e &
\end{array}
$$

```
Syntax

  x,y,z              Variable
  v ::=              Value
       {A = T}         type tag
       λ(x:T)t         lambda
  s,t,u ::=          Term
    x                  variable
    v                  value
    x y                application ここで変数しか使えない
    let x = t in u     let

  S,T,U ::=          Type
    ⊤                   top type
    ⊥                   bottom type
    {A:S..T}            type declaration
    x.A.                type projection
    ∀(x:S)T             dependent function

Evaluation t -> t

let x = v in e[x y] ---> let x = v in e[[z := y]t]   If v = λ(z:T)t
let x = y in t ---> [x := y]t
let x = let y = s in t in u ---> let y = s in let x = t in u
e[t] ---> e[u] if t ---> u
                    where e ::= []  let x = [] in t | let x = v in e
```

System D<:

Type Assignment $\boxed{\Gamma \vdash t : T}$

$$\Gamma,\, x : T,\, \Gamma' \vdash x : T \qquad (\textsc{Var})$$

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash T <: U}{\Gamma \vdash t : U} \qquad (\textsc{Sub})$$

$$\frac{\Gamma,\, x : T \vdash t : U \quad x \notin \mathrm{fv}(T)}{\Gamma \vdash \lambda(x{:}T)t : \forall(x{:}T)U} \qquad (\textsc{All-I})$$

$$\frac{\Gamma \vdash x : \forall(z{:}S)T \quad \Gamma \vdash y : S}{\Gamma \vdash x\, y : [z := y]T} \qquad (\textsc{All-E})$$

$$\frac{\Gamma \vdash t : T \quad \Gamma,\, x : T \vdash u : U \quad x \notin \mathrm{fv}(U)}{\Gamma \vdash \mathbf{let}\ x = t\ \mathbf{in}\ u : U} \qquad (\textsc{Let})$$

$$\Gamma \vdash \{A = T\} : \{A : T..T\} \qquad (\textsc{Typ-I})$$

Subtyping $\boxed{\Gamma \vdash T <: T}$

$$\Gamma \vdash T <: \top \qquad (\textsc{Top})$$

$$\Gamma \vdash \bot <: T \qquad (\textsc{Bot})$$

$$\Gamma \vdash T <: T \qquad (\textsc{Refl})$$

$$\frac{\Gamma \vdash S <: T \quad \Gamma \vdash T <: U}{\Gamma \vdash S <: U} \qquad (\textsc{Trans})$$

$$\frac{\Gamma \vdash x : \{A : S..T\}}{\Gamma \vdash S <: x.A} \qquad (\textsc{<:-Sel})$$

$$\frac{\Gamma \vdash x : \{A : S..T\}}{\Gamma \vdash x.A <: T} \qquad (\textsc{Sel-<:})$$

$$\frac{\Gamma \vdash S_2 <: S_1 \quad \Gamma,\, x : S_2 \vdash T_1 <: T_2}{\Gamma \vdash \forall(x{:}S_1)T_1 <: \forall(x{:}S_2)T_2} \qquad (\textsc{All-<:-All})$$

$$\frac{\Gamma \vdash S_2 <: S_1 \quad \Gamma \vdash T_1 <: T_2}{\Gamma \vdash \{A : S_1..T_1\} <: \{A : S_2..T_2\}} \qquad (\textsc{Typ-<:-Typ})$$

Fig. 1. System $\mathrm{D}_{<:}$

```
Type Assignment Γ ⊢ t : T

Γ, x:T, Γ' ⊢ x : T              (Var)

Γ ⊢ t : T   Γ ⊢ T <: U
------------------------------  (Sub)
Γ ⊢ t : U

Γ, x:T ⊢ t : U   x ∉ fv(T)

------------------------------  (All-I)
Γ ⊢ λ(x : T)t : ∀(x : T)U


Γ ⊢ x:∀(z : S)T   Γ ⊢ y : S
------------------------------  (All-E)
Γ ⊢ x y : [x := y]T

Γ ⊢ t : T   Γ, x : T ⊢ u : U
x ∉ fv(T)
------------------------------  (Let)
Γ ⊢ let x = t in u : U

Γ ⊢ {A=T} : {A:T..T}            (Typ-I)

Subtyping Γ ⊢ T <: T

Γ ⊢ T <: ⊤ (Top)
Γ ⊢ ⊥ <: T (Bot)
Γ ⊢ T <: T (Refl)

Γ ⊢ S <: T   Γ ⊢ T <: U
------------------------------  (Trans)
Γ ⊢ S <: U

Γ ⊢ x : {A:S..T}
------------------------------  (<:-Sel)
Γ ⊢ S <: x.A

Γ ⊢ x : {A:S..T}
------------------------------  (Sel-<:)
Γ ⊢ x.A <: T

Γ ⊢ S2 <: S1
Γ, x : S2 ⊢ T1 <: T2
------------------------------  (All-<:-All)
Γ ⊢ ∀(x:S1)T1 <: ∀(x:S2)T2


Γ ⊢ S2 <: S1   Γ ⊢ T1 <: T2
------------------------------  (Typ-<:-Typ)
Γ ⊢ {A:S1..T1} <: {A:S2..T2}
```

System D<:

```
Γ, x:T, Γ' ⊢ x : T              % (Var)

Γ ⊢ t : T, Γ ⊢ T <: U
--%---------------------------- (Sub)
Γ ⊢ t : U.

Γ, x:T ⊢ t : U, /+ x ∈ fv(T)
--%---------------------------- (All-I)
Γ ⊢ λ(x : T)t : ∀(x : T)U.

Γ ⊢ x:∀(z : S)T   Γ ⊢ y : S
--%---------------------------- (All-E)
Γ ⊢ x y : [x := y]T.

Γ ⊢ t : T   Γ, x : T ⊢ u : U,
/+ x ∈ fv(T)
--%---------------------------- (Let)
Γ ⊢ x y : [x := y]T.

Γ ⊢ {A=T} : {A:T..T}.          (Typ-I)

% Subtyping Γ ⊢ T <: T

Γ ⊢ T <: ⊤. %(Top)
Γ ⊢ ⊥ <: T. %(Bot)

Γ ⊢ T <: T. %(Refl)
/*
Γ ⊢ S <: T, Γ ⊢ T <: U
--%---------------------------- (Trans)
Γ ⊢ S <: U.

Γ ⊢ x : {A:S..T}
--%---------------------------- (<:-Sel)
Γ ⊢ S <: x.>A.

Γ ⊢ x : {A:S..T}
--%---------------------------- (Sel-<:)
Γ ⊢ x.A <: T.

Γ ⊢ S2 <: S1,
Γ, x : S2 ⊢ T1 <: T2
--%---------------------------- (All-<:-All)
Γ ⊢ ∀(x:S1)T1 <: ∀(x:S2)T2.

Γ ⊢ S2 <: S1, Γ ⊢ T1 <: T2
--%---------------------------- (Typ-<:-Typ)
Γ ⊢ {A:S1..T1} <: {A:S2..T2}.
```


```
Γ, x:T, Γ' ⊢ x : T              % (Var)

Γ ⊢ t : T, Γ ⊢ T <: U
--%---------------------------- (Sub)
Γ ⊢ t : U.

Γ, x:T ⊢ t : U, /+ x ∈ fv(T)
--%---------------------------- (All-I)
Γ ⊢ λ(x : T)t : ∀(x : T)U.

Γ ⊢ x:∀(z : S)T   Γ ⊢ y : S
--%---------------------------- (All-E)
Γ ⊢ x y : [x := y]T.

Γ ⊢ t : T   Γ, x : T ⊢ u : U,
/+ x ∈ fv(T)
--%---------------------------- (Let)
Γ ⊢ x y : [x := y]T.

Γ ⊢ {A=T} : {A:T..T}.          (Typ-I)

% Subtyping Γ ⊢ T <: T

Γ ⊢ T <: ⊤. %(Top)
Γ ⊢ ⊥ <: T. %(Bot)

Γ ⊢ T <: T. %(Refl)
/*
Γ ⊢ S <: T, Γ ⊢ T <: U
--%---------------------------- (Trans)
Γ ⊢ S <: U.

Γ ⊢ x : {A:S..T}
--%---------------------------- (<:-Sel)
Γ ⊢ S <: x.>A.

Γ ⊢ x : {A:S..T}
--%---------------------------- (Sel-<:)
Γ ⊢ x.A <: T.

Γ ⊢ S2 <: S1,
Γ, x : S2 ⊢ T1 <: T2
--%---------------------------- (All-<:-All)
Γ ⊢ ∀(x:S1)T1 <: ∀(x:S2)T2.

Γ ⊢ S2 <: S1, Γ ⊢ T1 <: T2
--%---------------------------- (Typ-<:-Typ)
Γ ⊢ {A:S1..T1} <: {A:S2..T2}.
```

$\Gamma \vdash x\ y : [x := y]T.$

$\Gamma \vdash \{A=T\} : \{A:T..T\}.$ (Typ-I)

% Subtyping $\Gamma \vdash T <: T$

$\Gamma \vdash T <: \top.$ %(Top)

$\Gamma \vdash \bot <: T.$ %(Bot)

$\Gamma \vdash T <: T.$ %(Refl)

/*

$\Gamma \vdash S <: T,\ \Gamma \vdash T <: U$

--%------------------------------ (Trans)

$\Gamma \vdash S <: U.$

A正規化

$$
\begin{array}{rcl}
M & ::= & V \\
 & & \mid (\textbf{let}\ (x\ M_1)\ M_2) \\
 & & \mid (\textbf{if0}\ M_1\ M_2\ M_3) \\
 & & \mid (M\ M_1\ \ldots\ M_n) \\
 & & \mid (O\ M_1\ \ldots\ M_n) \\
 & & \\
V & ::= & c \mid x \mid (\lambda x_1 \ldots x_n.M)
\end{array}
\qquad
\begin{array}{rcl}
V & \in & \mathit{Values} \\
c & \in & \mathit{Constants} \\
x & \in & \mathit{Variables} \\
O & \in & \mathit{Primitive\ Operations}
\end{array}
$$

Figure 1: Abstract Syntax of Core Scheme (CS)

Semantics: Let $M \in CS$,

$$eval_d(M) = c \quad \text{if} \quad \langle M, \emptyset, \textbf{stop}\rangle \longmapsto^* \langle \textbf{stop}, c\rangle.$$

Data Specifications:

$$
\begin{array}{rclll}
S & \in & State_d & = & CS \times Env_d \times Cont_d \mid Cont_d \times Value_d & \text{(machine states)} \\
E & \in & Env_d & = & Variables \multimap Value_d & \text{(environments)} \\
V^* & \in & Value_d & = & c \mid \langle \textbf{cl}\ x_1 \ldots x_n, M, E\rangle & \text{(machine values)} \\
K & \in & Cont_d & = & \textbf{stop} \mid \langle \textbf{ap}\ \langle \ldots, V^*, \bullet, M, \ldots\rangle, E, K\rangle \mid \langle \textbf{lt}\ x, M, E, K\rangle & \text{(continuations)} \\
 & & & & \mid \langle \textbf{if}\ M_1, M_2, E, K\rangle \mid \langle \textbf{pr}\ O, \langle \ldots, V^*, \bullet, M, \ldots\rangle, E, K\rangle &
\end{array}
$$

Transition Rules:

$$
\begin{array}{rcl}
\langle V, E, K\rangle & \longmapsto & \langle K, \gamma(V, E)\rangle \\
\langle (\textbf{let}\ (x\ M_1)\ M_2), E, K\rangle & \longmapsto & \langle M_1, E, \langle \textbf{lt}\ x, M_2, E, K\rangle\rangle \\
\langle (\textbf{if0}\ M_1\ M_2\ M_3), E, K\rangle & \longmapsto & \langle M_1, E, \langle \textbf{if}\ M_2, M_3, E, K\rangle\rangle \\
\langle (M\ M_1\ \ldots\ M_n), E, K\rangle & \longmapsto & \langle M, E, \langle \textbf{ap}\ \langle \bullet, M_1, \ldots, M_n\rangle, E, K\rangle\rangle \\
\langle (O\ M_1\ M_2\ \ldots\ M_n), E, K\rangle & \longmapsto & \langle M_1, E, \langle \textbf{pr}\ O, \langle \bullet, M_2, \ldots, M_n\rangle, E, K\rangle\rangle \\
 & & \\
\langle \langle \textbf{lt}\ x, M, E, K\rangle, V^*\rangle & \longmapsto & \langle M, E[x := V^*], K\rangle \\
\langle \langle \textbf{if}\ M_1, M_2, E, K\rangle, 0\rangle & \longmapsto & \langle M_1, E, K\rangle \\
\langle \langle \textbf{if}\ M_1, M_2, E, K\rangle, V^*\rangle & \longmapsto & \langle M_2, E, K\rangle \quad \text{where } V^* \neq 0 \\
\langle \langle \textbf{ap}\ \langle \ldots, V_i^*, \bullet, M, \ldots\rangle, E, K\rangle, V_{i+1}^*\rangle & \longmapsto & \langle M, E, \langle \textbf{ap}\ \langle \ldots, V_i^*, V_{i+1}^*, \bullet, \ldots\rangle, E, K\rangle\rangle \\
\langle \langle \textbf{ap}\ V^*, V_1^*, \ldots, \bullet\rangle, E, K\rangle, V_n^*\rangle & \longmapsto & \langle M', E'[x_1 := V_1^*, \ldots, x_n := V_n^*], K\rangle \quad \text{if } V^* = \langle \textbf{cl}\ x_1 \ldots x_n, M', E'\rangle \\
\langle \langle \textbf{pr}\ O, \langle \ldots, V_i^*, \bullet, M, \ldots\rangle, E, K\rangle, V_{i+1}^*\rangle & \longmapsto & \langle M, E, \langle \textbf{pr}\ O, \langle \ldots, V_i^*, V_{i+1}^*, \bullet, \ldots\rangle, E, K\rangle\rangle \\
\langle \langle \textbf{pr}\ O, \langle V_1^*, \ldots, \bullet\rangle, E, K\rangle, V_n^*\rangle & \longmapsto & \langle K, \delta(O, V_1^*, \ldots, V_n^*)\rangle \quad \text{if } \delta(O, V_1^*, \ldots, V_n^*) \text{ is defined}
\end{array}
$$

Converting syntactic values to machine values:

$$
\begin{array}{rcl}
\gamma(c, E) & = & c \\
\gamma(x, E) & = & E(x) \\
\gamma((\lambda x_1 \ldots x_n.M), E) & = & \langle \textbf{cl}\ x_1 \ldots x_n, M, E\rangle
\end{array}
$$

Figure 2: The CEK-machine

A正規形の論文

The Essence of Compiling with Continuations - Flanagan, Sabry, Duba, Felleisen (ResearchIndex)という論文で解説されています。

Figure 1: Abstract Syntax of Core Scheme (CS)

```
M := V
   | (let (X M1) M1)    V ∈ Values
   | (if0 M1 M2 M3)     c ∈ Constants
   | (M M1 ... Mn)      x ∈ Variables
   | (O M1 ... Mn)      O ∈ Primitive Operation

V ::= c | x | (λx1...xn.M)
```

コアのSchemeの構文はLISPの方言なので、カッコで囲まれた文法で、値かlet,if,関数適用があり、値は定数か、変数か、多引数のλ抽象があるということですね。

Translation Rules: 変換規則

<V,E,K> |---> <K,γ(V,E)> 値は、継続Kと値環境から得られた変数になるのかな。

γ(c, E)= c 定数ならそのまま

γ(x, E)= E(x) 変数なら環境から値を取り出す

γ((λx1...xn.M), E) = <cl x1...xn,M,E> λ抽象なら、

A正規化

Syntax

```
  x,y,z              Variable
  v ::=              Value
       {A = T}         type tag
       λ(x:T)t         lambda
  s,t,u ::=          Term
    x                  variable
    v                  value
    x y                application ここで変数しか使えない
    let x = t in u     let

  S,T,U ::=          Type
    ⊤                   top type
    ⊥                   bottom type
    {A:S..T}            type declaration
    x.A.                type projection
    ∀(x:S)T             dependent function
```

Evaluation t -> t

```
let x = v in e[x y] ---> let x = v in e[[z := y]t]   If v = λ(z:T)t
let x = y in t ---> [x := y]t
let x = let y = s in t in u ---> let y = s in let x = t in u
e[t] ---> e[u] if t ---> u
                    where e ::= []  let x = [] in t | let x = v in e
```

A正規形の論文

The Essence of Compiling with Continuations - Flanagan, Sabry, Duba, Felleisen (ResearchIndex)という論文で解説されています。

```
v({_=_}).
v(λ(_->_)).

genid(E,E_,X) :- E_ is E + 1, format(atom(X),'.x~d',[E]).

anf_trans(E,X,E,Hole,Hole,X) :- atom(X).
anf_trans(E,V,E_,Hole,let(X=V,Hole),X) :- v(V), genid(E,E_,X).
anf_trans(E,S$T,E_,Hole,let(Z=Hole2,Hole),Z):-
  anf_trans(E,S,Hole1,E1,S_,X),
  anf_trans(E1,T,Hole2,E2,T_,Y),
  Hole1=T_,Hole2=X$Y,
  genid(E2,E_,Z).
anf_trans(E,let(X=T,U),Hole,E_,let(X=T_,U_),Y) :-
  anf_trans(E,T,Hole1,E1,T_,X),Hole1= let(X=T_,U_),
  anf_trans(E1,U,Hole,E_,U_,Y).
```

anf_trans述語は、環境E, 項Tを受け取り穴(Hole)と変換式、対応する変数名を返します。

holeは、継続式を置き換えるための入れ物で、束縛のないProlog変数です。返却された項の中には、必ず穴が空いているので穴に変数を設定することで、継続の式を設定できます。

```
s,t,u := Term

x   variable

v   value

x y                   application

Llet x = t in u    let
```


```
s,t,u := Term

x   variable

v   value

x y
application
```


$s, t, u ::=$	**Term**
x	variable
v	value
$x\ y$	application
$\mathbf{let}\ x = t\ \mathbf{in}\ u$	let
$x.A$	type projection
$\forall(x\!:\!S)T$	dependent function

Evaluation $\boxed{t \longrightarrow t}$

$$
\begin{array}{rcll}
\mathbf{let}\ x = v\ \mathbf{in}\ e[x\ y] & \longrightarrow & \mathbf{let}\ x = v\ \mathbf{in}\ e[[z := y]t] & \mathbf{if}\ v = \lambda(z\!:\!T)t \\
\mathbf{let}\ x = y\ \mathbf{in}\ t & \longrightarrow & [x := y]t & \\
\mathbf{let}\ x = \mathbf{let}\ y = s\ \mathbf{in}\ t\ \mathbf{in}\ u & \longrightarrow & \mathbf{let}\ y = s\ \mathbf{in}\ \mathbf{let}\ x = t\ \mathbf{in}\ u & \\
e[t] & \longrightarrow & e[u] & \mathbf{if}\ t \longrightarrow u \\
 & & \mathbf{where}\ e ::= [\,]\ |\ \mathbf{let}\ x = [\,]\ \mathbf{in}\ t\ |\ \mathbf{let}\ x = v\ \mathbf{in}\ e &
\end{array}
$$

Type Assignment $\boxed{\Gamma \vdash t : T}$

$$\Gamma,\ x : T,\ \Gamma' \vdash x : T \quad (\text{VAR})$$

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash T <: U}{\Gamma \vdash t : U} \quad (\text{SUB})$$

$$\frac{\Gamma,\ x : T \vdash t : U \quad x \notin fv(T)}{\Gamma \vdash \lambda(x\!:\!T)t : \forall(x\!:\!T)U} \quad (\text{ALL-I})$$

$$\frac{\Gamma \vdash x : \forall(z\!:\!S)T \quad \Gamma \vdash y : S}{\Gamma \vdash x\ y : [z := y]T} \quad (\text{ALL-E})$$

$$\frac{\Gamma \vdash t : T \quad \Gamma,\ x : T \vdash u : U \quad x \notin fv(U)}{\Gamma \vdash \mathbf{let}\ x = t\ \mathbf{in}\ u : U} \quad (\text{LET})$$

$$\Gamma \vdash \{A = T\} : \{A : T..T\} \quad (\text{TYP-I})$$

Subtyping $\boxed{\Gamma \vdash T <: T}$

$$\Gamma \vdash T <: \top \quad (\text{TOP})$$

$$\Gamma \vdash \bot <: T \quad (\text{BOT})$$

$$\Gamma \vdash T <: T \quad (\text{REFL})$$

$$\frac{\Gamma \vdash S <: T \quad \Gamma \vdash T <: U}{\Gamma \vdash S <: U} \quad (\text{TRANS})$$

$$\frac{\Gamma \vdash x : \{A : S..T\}}{\Gamma \vdash S <: x.A} \quad (<:\text{-SEL})$$

$$\frac{\Gamma \vdash x : \{A : S..T\}}{\Gamma \vdash x.A <: T} \quad (\text{SEL-}<:)$$

$$\Gamma \vdash S_2 <: S_1$$