Higher-Order Calculus

Piotr Krzemiński

Wrocław, 4th June 2014

Introduction

- we start from $\mathbf{Ob_{1<:}}$
- structure rule for method update and variance annotations
- higher-order subtyping, based on Girard's $\mathbf{F}_\omega$
- finally we obtain $\mathbf{Ob}_{\omega<:\mu}$

Type operators

$$\lambda(X)\forall(Y <: X)Y$$

This is example of so-called *type operator*.
It's mapping from type X to the type $\forall(Y <: X)Y$.

Kinds

A structure of *kinds* is introduced to classify types and operators (collectively called *constructors*).

- the kind of all types is called Ty
- an operator from types to types has kind $Ty \Rightarrow Ty$
- higher-order operators can be expressed as well, such as $(Ty \Rightarrow Ty) \Rightarrow Ty$
- in general, kind $K \Rightarrow L$ is the kind of operators mapping kind K to kind L
- we write $A :: K$ to say that constructor A has kind K

Higher-order subtyping

Subtype relation is generalized to a higher-order: the *subconstructor* (or simply, *inclusion*) relation.

- on types, it reduces to ordinary subtyping
- on operators it is defined as pointwise inclusion:
  - $B <: B'$ holds at kind $K \Rightarrow L$ if for all A of kind K we have $B(A) <: B'(A)$ at kind L
- fully we write $A <: B :: K$ meaning the constructors A and B are both of kind K and A is included in B
- for example, $A <: Top$ is written in full as $A <: Top :: Ty$

Syntax

K, L ::=	kinds
Ty	types
$K \Rightarrow L$	operators from K to L
A, B ::=	constructors
X	constructor variable
Top	the biggest constructor at kind Ty
$[l_i \upsilon_i : B_i \ ^{i \in 1..n}]$	object type (l_i distinct, $\upsilon_i \in \{^{\circ}, ^{-}, ^{+}\}$)
$\forall (X <: A :: K) B$	bounded universal type
$\mu(X) A$	recursive type
$\lambda(X :: K) B$	operator
$B(A)$	operator application
a, b ::=	terms
x	variable
$[l = \varsigma(x_i : A_i) b_i \ ^{i \in 1..n}]$	object formation (l_i distinct)
$a.l$	method invocation
$a.l \Leftarrow \varsigma(x : A) b$	method update
$\lambda(X <: A :: K) b$	constructor abstraction
$b(A)$	constructor application
$fold(A, a)$	recursive fold
$unfold(a)$	recursive unfold

- $X <: A :: K$ is general form of bounds for constructors
- bounded universal types and constructor abstractions
- operators have restricted bounds (of the form $X :: K$) to simplify technical treatment of higher-order features
- we do not include primitive existential quantifiers nor function types

Example

Our initial example of an operator written in full as:

$$\lambda(X :: Ty)\forall(Y <: X :: Ty)Y :: Ty \Rightarrow Ty$$

Results

Results are described by the following grammar.

v	$::=$	results
	$[l_i{=}\varsigma(x_i{:}A_i)b_i\ {}^{i\in 1..n}]$	object result
	$\lambda(X{<:}A{::}K)b$	constructor abstraction result
	$fold(A,v)$	recursion result

Semantics (1/2)

(Red Object) (where $v \equiv [l_i = \varsigma(x_i{:}A_i)b_i \;^{i \in 1..n}]$)

$$\frac{}{\vdash v \leadsto v}$$

(Red Select) (where $v' \equiv [l_i = \varsigma(x_i{:}A_i)b_i\{x_i\} \;^{i \in 1..n}]$)

$$\frac{\vdash a \leadsto v' \qquad \vdash b_j\{\!\{v'\}\!\} \leadsto v \qquad j \in 1..n}{\vdash a.l_j \leadsto v}$$

(Red Update)

$$\frac{\vdash a \leadsto [l_i = \varsigma(x_i{:}A_i)b_i \;^{i \in 1..n}] \qquad j \in 1..n}{\vdash a.l_j \Leftarrow \varsigma(x{:}A)b \leadsto [l_j = \varsigma(x{:}A_j)b,\; l_i = \varsigma(x_i{:}A_i)b_i \;^{i \in (1..n)-\{j\}}]}$$

Semantics (2/2)

(Red Fun2::) (where $v \equiv \lambda(X<:A::K)b$)

$$\frac{}{\vdash v \leadsto v}$$

(Red Appl2::)

$$\frac{\vdash b \leadsto \lambda(X<:A::K)c\{X\} \qquad \vdash c\{\!\{A'\}\!\} \leadsto v}{\vdash b(A') \leadsto v}$$

(Red Fold)

$$\frac{\vdash b \leadsto v}{\vdash fold(A,b) \leadsto fold(A,v)}$$

(Red Unfold)

$$\frac{\vdash a \leadsto fold(A,v)}{\vdash unfold(a) \leadsto v}$$

Judgments

The type rules of $\mathbf{Ob}_{\omega<:\mu}$ are formulated in terms of following judgments.

$E \vdash \diamond$	E is an environment
$E \vdash K\ kind$	K is a kind
$E \vdash A :: K$	constructor A has kind K
$E \vdash A \leftrightarrow B :: K$	A and B are equivalent constructors of kind K
$E \vdash A <: B :: K$	A is a subconstructor of B, both of kind K
$E \vdash \upsilon A <: \upsilon' B$	A is a subtype of B according to variances υ and υ'
$E \vdash a : A$	a is a value of type A

- $E \vdash A \leftrightarrow B :: K$ needed because presence of operators entails computation at constructor level
- for example theory implies that $(\lambda(X :: Ty)\forall(Y <: X :: Ty)Y)(Top)$ equals $\forall(Y <: Top :: Ty)Y$
- there is no judgment for equivalence of terms

Notation

Several abbreviations are extensively used to omit some bounds and some kinds.

$\lceil Ty \rceil$	$\triangleq$	Top	
$\lceil K \Rightarrow L \rceil$	$\triangleq$	$\lambda(X{::}K)\lceil L \rceil$	
$X :: K$	$\triangleq$	$X <: \lceil K \rceil :: K$	(in environments and some binders)
$X <: A$	$\triangleq$	$X <: A :: Ty$	(in environments and some binders)
X	$\triangleq$	$X <: Top :: Ty$	(in environments and some binders)
$E \vdash A$	$\triangleq$	$E \vdash A :: Ty$	
$E \vdash A \leftrightarrow B$	$\triangleq$	$E \vdash A \leftrightarrow B :: Ty$	
$E \vdash A <: B$	$\triangleq$	$E \vdash A <: B :: Ty$	

- $\lceil K \rceil$ denotes maximum constructor at kind K

Environment & Kind formation

(Env $\emptyset$)
$$\frac{}{\emptyset \vdash \diamond}$$

(Env X<:)
$$\frac{E \vdash A :: K \qquad X \notin dom(E)}{E, X{<:}A{::}K \vdash \diamond}$$

(Env x)
$$\frac{E \vdash A \qquad x \notin dom(E)}{E, x{:}A \vdash \diamond}$$

(Kind Ty)
$$\frac{E \vdash \diamond}{E \vdash Ty\ kind}$$

(Kind $\Rightarrow$)
$$\frac{E \vdash K\ kind \qquad E \vdash L\ kind}{E \vdash K{\Rightarrow}L\ kind}$$

Constructor formation

(Con X)
$$\frac{E', X{<:}A{::}K, E'' \vdash \diamond}{E', X{<:}A{::}K, E'' \vdash X :: K}$$

(Con Top)
$$\frac{E \vdash \diamond}{E \vdash Top :: Ty}$$

(Con Object) (l_i distinct, $\upsilon_i \in \{^{\circ}, ^{-}, ^{+}\}$)
$$\frac{E \vdash B_i \quad \forall i \in 1..n}{E \vdash [l_i \upsilon_i{:}B_i{}^{i \in 1..n}]}$$

(Con All)
$$\frac{E, X{<:}A{::}K \vdash B}{E \vdash \forall(X{<:}A{::}K)B}$$

(Con Rec)
$$\frac{E, X \vdash A}{E \vdash \mu(X)A}$$

(Con Abs)
$$\frac{E, X{::}K \vdash B :: L}{E \vdash \lambda(X{::}K)B :: K{\Rightarrow}L}$$

(Con Appl)
$$\frac{E \vdash B :: K{\Rightarrow}L \quad E \vdash A :: K}{E \vdash B(A) :: L}$$

Constructor equivalence (1/3)

(Con Eq Symm)

$$\frac{E \vdash A \leftrightarrow B :: K}{E \vdash B \leftrightarrow A :: K}$$

(Con Eq Trans)

$$\frac{E \vdash A \leftrightarrow B :: K \qquad E \vdash B \leftrightarrow C :: K}{E \vdash A \leftrightarrow C :: K}$$

(Con Eq X)

$$\frac{E \vdash X :: K}{E \vdash X \leftrightarrow X :: K}$$

(Con Eq Top)

$$\frac{E \vdash \diamond}{E \vdash Top \leftrightarrow Top :: Ty}$$

Constructor equivalence (2/3)

(Con Eq Object) (l_i distinct, $\upsilon_i \in \{^o, ^-, ^+\}$)

$$\frac{E \vdash B_i \leftrightarrow B_i' \qquad \forall i \in 1..n}{E \vdash [l_i \upsilon_i : B_i \ ^{i \in 1..n}] \leftrightarrow [l_i \upsilon_i : B_i' \ ^{i \in 1..n}]}$$

(Con Eq All)

$$\frac{E \vdash A \leftrightarrow A' :: K \qquad E, X<:A::K \vdash B \leftrightarrow B'}{E \vdash \forall(X<:A::K)B \leftrightarrow \forall(X<:A'::K)B'}$$

(Con Eq Rec)

$$\frac{E, X \vdash B \leftrightarrow B'}{E \vdash \mu(X)B \leftrightarrow \mu(X)B'}$$

Constructor equivalence (3/3)

(Con Eq Abs)

$$\frac{E, X{::}K \vdash B \leftrightarrow B' :: L}{E \vdash \lambda(X{::}K)B \leftrightarrow \lambda(X{::}K)B' :: K{\Rightarrow}L}$$

(Con Eq Appl)

$$\frac{E \vdash B \leftrightarrow B' :: K{\Rightarrow}L \qquad E \vdash A \leftrightarrow A' :: K}{E \vdash B(A) \leftrightarrow B'(A') :: L}$$

(Con Eval Beta)

$$\frac{E, X{::}K \vdash B\{X\} :: L \qquad E \vdash A :: K}{E \vdash (\lambda(X{::}K)B\{X\})(A) \leftrightarrow B\{\!\{A\}\!\} :: L}$$

Constructor inclusion (1/3)

(Con Sub Refl)

$$\frac{E \vdash A \leftrightarrow B :: K}{E \vdash A <: B :: K}$$

(Con Sub Trans)

$$\frac{E \vdash A <: B :: K \qquad E \vdash B <: C :: K}{E \vdash A <: C :: K}$$

(Con Sub X)

$$\frac{E', X{<:}A{::}K, E'' \vdash \diamond}{E', X{<:}A{::}K, E'' \vdash X <: A :: K}$$

(Con Sub Top)

$$\frac{E \vdash A :: Ty}{E \vdash A <: Top :: Ty}$$

Constructor inclusion (2/3)

(Con Sub Object) (l_i distinct)

$$\frac{E \vdash \upsilon_i B_i <: \upsilon_i' B_i' \quad \forall i \in 1..n \qquad E \vdash B_i \quad \forall i \in n+1..n+m}{E \vdash [l_i \upsilon_i : B_i \ ^{i \in 1..n+m}] <: [l_i \upsilon_i' : B_i' \ ^{i \in 1..n}]}$$

(Con Sub All)

$$\frac{E \vdash A' <: A :: K \qquad E, X <: A' :: K \vdash B <: B'}{E \vdash \forall (X <: A :: K) B <: \forall (X <: A' :: K) B'}$$

(Con Sub Rec)

$$\frac{E \vdash \mu(X)A \qquad E \vdash \mu(Y)B \qquad E, Y, X <: Y \vdash A <: B}{E \vdash \mu(X)A <: \mu(Y)B}$$

Constructor inclusion (3/3)

(Con Sub Abs)

$$\frac{E, X{::}K \vdash B <: B' :: L}{E \vdash \lambda(X{::}K)B <: \lambda(X{::}K)B' :: K{\Rightarrow}L}$$

(Con Sub Appl)

$$\frac{E \vdash B <: B' :: K{\Rightarrow}L \qquad E \vdash A :: K}{E \vdash B(A) <: B'(A) :: L}$$

(Con Sub Invariant)

$$\frac{E \vdash B}{E \vdash {}^{o}B <: {}^{o}B}$$

(Con Sub Covariant)

$$\frac{E \vdash B <: B' \qquad \upsilon \in \{{}^{o}, {}^{+}\}}{E \vdash \upsilon B <: {}^{+}B'}$$

(Con Sub Contravariant)

$$\frac{E \vdash B' <: B \qquad \upsilon \in \{{}^{o}, {}^{-}\}}{E \vdash \upsilon B <: {}^{-}B'}$$

Term typing (1/2)

(Val Subsumption)

$$\frac{E \vdash a : A \qquad E \vdash A <: B}{E \vdash a : B}$$

(Val x)

$$\frac{E', x{:}A, E'' \vdash \diamond}{E', x{:}A, E'' \vdash x : A}$$

(Val Object)

$$\frac{E, x_i{:}A \vdash b_i : B_i \quad \forall i \in 1..n \qquad E \vdash A \leftrightarrow [l_i \upsilon_i{:}B_i{}^{i \in 1..n}]}{E \vdash [l_i = \varsigma(x_i{:}A)b_i{}^{i \in 1..n}] : A}$$

(Val Select)

$$\frac{E \vdash a : [l_i \upsilon_i{:}B_i{}^{i \in 1..n}] \qquad \upsilon_j \in \{{}^{o}, {}^{+}\} \qquad j \in 1..n}{E \vdash a.l_j : B_j}$$

Term typing (2/2)

(Val Update) (where $A \equiv [l_i \upsilon_i{:}B_i{}^{i\in 1..n}]$)

$$\frac{E \vdash C <: A \qquad E \vdash a : C \qquad E, x{:}C \vdash b : B_j \qquad \upsilon_j \in \{^{\circ}, ^{-}\} \qquad j \in 1..n}{E \vdash a.l_j \Leftarrow \varsigma(x{:}C)b : C}$$

(Val Fun2::)

$$\frac{E, X<:A::K \vdash b : B}{E \vdash \lambda(X<:A::K)b : \forall(X<:A::K)B}$$

(Val Appl2::)

$$\frac{E \vdash b : \forall(X<:A::K)B\{X\} \qquad E \vdash A' <: A :: K}{E \vdash b(A') : B\{\!\{A'\}\!\}}$$

(Val Fold)

$$\frac{E \vdash b : B\{\!\{A\}\!\} \qquad E \vdash A \leftrightarrow \mu(X)B\{X\}}{E \vdash fold(A,b) : A}$$

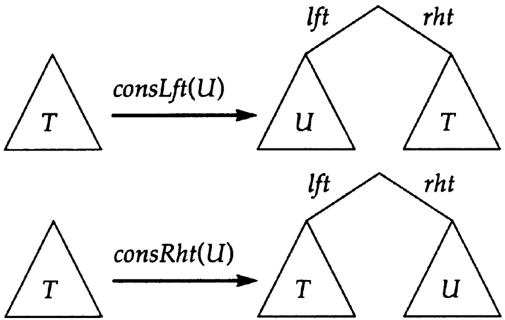
(Val Unfold) (where $A \equiv \mu(X)B\{X\}$)

$$\frac{E \vdash a : A}{E \vdash unfold(a) : B\{\!\{A\}\!\}}$$

General Self types

- so far treatment of Self types was restricted to covariant occurences only in order to keep subtyping and subsumption working smoothly
- we may want to give up certain subtyping properties in exchange for a treatment of general Self types (without the covariance restriction)
- it naturally involves higher-order constructions

Binary-Tree Objects



- generalization of example with numerals (which can be seen as unary trees)
- two distinct successor functions

Object-oriented binary trees

$$Bin \triangleq \mu(X)[isLeaf{:}Bool,\ lft{:}X,\ rht{:}X,\ consLft{:}X \to X,\ consRht{:}X \to X]$$

$$UBin \triangleq [isLeaf{:}Bool,\ lft{:}Bin,\ rht{:}Bin,\ consLft{:}Bin \to Bin,\ consRht{:}Bin \to Bin]$$

$$\begin{aligned}
leaf : Bin \triangleq\ & fold(Bin,\\
& \quad [isLeaf = true,\\
& \quad\ lft = \varsigma(self{:}UBin)\, self.lft,\\
& \quad\ rht = \varsigma(self{:}UBin)\, self.rht,\\
& \quad\ consLft = \varsigma(self{:}UBin)\, \lambda(lft{:}Bin)\\
& \quad\quad fold(Bin, ((self.isLeaf := false).lft := lft).rht := fold(Bin, self)),\\
& \quad\ consRht = \varsigma(self{:}UBin)\, \lambda(rht{:}Bin)\\
& \quad\quad fold(Bin, ((self.isLeaf := false).lft := fold(Bin, self)).rht := rht)]
\end{aligned}$$

- we write $a.l := b$ for $a.l \Leftarrow \varsigma(x : A)b$ when $x \notin FV(b)$
- tree with two leaves and joining root: $unfold(leaf).consLft(leaf) : Bin$

Binary-Tree Classes

We wish to define a class type *BinClass*, and a class *binClass* of a type *BinClass* that generates trees of type Bin, which methods can be inherited.

An inheriting class could, for example, generate trees with nodes containing natural numbers. Such tree would have type:

$$NatBin \triangleq \mu(X)[n{:}Nat,\ isLeaf{:}Bool,\ lft{:}X,\ rht{:}X,\ consLft{:}X\to X,\ consRht{:}X\to X]$$

Note that $NatBin <: Bin$ cannot hold, but we still aim to reuse, for example, the *consLft* binary method.

Notation

We introduce following notation in order to transform the type *Bin* into a type operator *BinOp*.

- Op is the kind of simple type operators; it stands for $Ty \Rightarrow Ty$
- $A <: B$ means that A is a suboperator of B; it stands for $A <: B :: Op$
- A^* is the fixpoint of the operator A; it stands for the type $\mu(X)A(X)$ where $A :: Op$

Object-oriented binary-tree operator

$$
\begin{aligned}
&BinOp :: Op \triangleq \\
&\quad \lambda(X)[isLeaf{:}Bool,\ lft{:}X,\ rht{:}X,\ consLft{:}X\to X,\ consRht{:}X\to X] \\
&Bin :: Ty \triangleq BinOp^{*} \\
&UBin :: Ty \triangleq BinOp(Bin)
\end{aligned}
$$

- the operator *BinOp* is the object protocol of *Bin*
- the type *Bin* can be recovered from *BinOp* by taking a fixpoint
- the type *UBin* is the unfolding of *Bin* obtained by applying *BinOp* to *Bin*

Object-oriented binary-tree class type

$$
\begin{aligned}
BinClass \triangleq\ & \\
& [new^{+}{:}Bin, \\
& \ \ isLeaf^{+}{:}\forall(X{<:}BinOp)\ X^{*}\to Bool, \\
& \ \ lft^{+}, rht^{+}{:}\forall(X{<:}BinOp)\ X^{*}\to X^{*}, \\
& \ \ consLft^{+}, consRht^{+}{:}\forall(X{<:}BinOp)\ X^{*}\to X^{*}\to X^{*}]
\end{aligned}
$$

- based on notion of parametric pre-methods
- types of pre-methods are quantified over the suboperators of *BinOp* instead of subtypes of *Bin*
- fixpoints introduced where appropriate to collapse operators down to types
- use of $^{+}$ guarantees that classes cannot be accidentally modified (although it's not necessary for the typing of classes)

Object-oriented binary-tree class

$$
\begin{aligned}
binClass : BinClass \;\triangleq\; [&new = \varsigma(z{:}BinClass)\, fold(Bin,\\
&\qquad [isLeaf = \varsigma(s{:}UBin)\, z.isLeaf(BinOp)(\,fold(Bin, s)),\\
&\qquad\; lft = \varsigma(s{:}UBin)\, z.lft(BinOp)(\,fold(Bin, s)),\\
&\qquad\; rht = \varsigma(s{:}UBin)\, z.rht(BinOp)(\,fold(Bin, s)),\\
&\qquad\; consLft = \varsigma(s{:}UBin)\, z.consLft(BinOp)(\,fold(Bin, s)),\\
&\qquad\; consRht = \varsigma(s{:}UBin)\, z.consRht(BinOp)(\,fold(Bin, s))]),\\
&isLeaf = \lambda(X{<:}BinOp)\, \lambda(self{:}X^*)\, true,\\
&lft = \lambda(X{<:}BinOp)\, \lambda(self{:}X^*)\, unfold(self\,).lft,\\
&rht = \lambda(X{<:}BinOp)\, \lambda(self{:}X^*)\, unfold(self\,).rht,\\
&consLft = \lambda(X{<:}BinOp)\, \lambda(rht{:}X^*)\, \lambda(lft{:}X^*)\\
&\qquad fold(X^*, ((unfold(rht).isLeaf := false).lft := lft).rht := rht),\\
&consRht = \lambda(X{<:}BinOp)\, \lambda(lft{:}X^*)\, \lambda(rht{:}X^*)\\
&\qquad fold(X^*, ((unfold(lft).isLeaf := false).lft := lft).rht := rht)]
\end{aligned}
$$

Inheritance of binary pre-method

We can verify that inheritance of binary pre-method is possible. For example for the *consLft* of type $\forall(X <: BinOp)X^* \to X^* \to X^*$, we have:

$$\forall(X <: BinOp)X^* \to X^* \to X^* \ <: \ \forall(X <: BinOp')X^* \to X^* \to X^*$$

for any $BinOp' <: BinOp$

A suboperator of *BinOp* is for example:

$$NatBinOp \equiv \lambda(X)[n : Nat, isLeaf : Bool, lft : X, rht : X, consLft : X \to X, consRht : X \to X]$$

Normal system

- our main purpose is to proove subject reduction theorem for $\mathbf{Ob}_{\omega<:\mu}$
- unfortunately, direct proofs on derivations rarely work i $\mathbf{Ob}_{\omega<:\mu}$, because the β rule *(Con Eval Beta)* introduces expressions of arbitrary shape
- we introduce a *normal system* which lacks that rule, but that is sound and complete with respect to $\mathbf{Ob}_{\omega<:\mu}$ over normal forms at the constructor level

Least upper bounds

- A^{nf} is notion of constructor A in normal form
- $lub_E(A)$ is notion of least upper bound of a constructor A in environment E, defined only for terms of the form $X(A_1)\ldots(A_n)$, for $n \geqslant 0$
- if the immediate bound of X in E is B, then $lub_E(X(A_1)\ldots(A_n)) = B(A_1)\ldots(A_n)$

Least upper bounds and normal forms

$$lub_{E,X<:A,E'}(X) \triangleq A$$
$$lub_E(B(A)) \triangleq lub_E(B)(A)$$
$$lub_E(A) \quad \text{undefined otherwise}$$

$$X^{nf} \triangleq X$$
$$Top^{nf} \triangleq Top$$
$$[l_i \upsilon_i{:}B_i{}^{i\in 1..n}]^{nf} \triangleq [l_i \upsilon_i{:}B_i^{nf}{}^{i\in 1..n}]$$
$$(\forall(X<:A::K)B)^{nf} \triangleq \forall(X<:A^{nf}::K)B^{nf}$$
$$(\mu(X)A)^{nf} \triangleq \mu(X)A^{nf}$$
$$(\lambda(X::K)B)^{nf} \triangleq \lambda(X::K)B^{nf}$$
$$(B(A))^{nf} \triangleq \text{if } B^{nf} \equiv \lambda(X::K)C\{X\} \text{ for some } X,K,C, \text{ then } (C\{\!\{A\}\!\})^{nf} \text{ else } B^{nf}(A^{nf})$$

$$\emptyset^{nf} \triangleq \emptyset$$
$$(E, X<:A::K)^{nf} \triangleq E^{nf}, X<:A^{nf}::K$$
$$(E, x{:}A)^{nf} \triangleq E^{nf}, x{:}A^{nf}$$

Judgments for the normal system

$E \vdash \diamond$	E is an environment
$E \vdash K\ kind$	K is a kind
$E \vdash A :: K$	constructor A has kind K
$E \vdash^{n} A <: B :: K$	A is a subconstructor of B, both of kind K
$E \vdash^{n} \upsilon A <: \upsilon' B$	A is a subtype of B according to variances υ and υ'

- judgments $E \vdash A <: B :: K$ and $E \vdash \upsilon A <: \upsilon' B$ of $\mathbf{Ob}_{\omega<:\mu}$ replaced with $E \vdash^{n} A <: B :: K$ and $E \vdash^{n} \upsilon A <: \upsilon' B$
- judgment $E \vdash A \leftrightarrow B :: K$ is dropped to avoid problems caused by β-equivalence
- remaining judgments unchanged

Normal constructor inclusion (1/2)

(NCon Sub Refl)
$$\frac{E \vdash A :: K}{E \vdash^{n} A <: A :: K}$$

(NCon Sub Trans)
$$\frac{E \vdash^{n} A <: B :: K \qquad E \vdash^{n} B <: C :: K}{E \vdash^{n} A <: C :: K}$$

(NCon Sub X)
$$\frac{E', X{<:}A{::}K, E'' \vdash^{n} A^{nf} <: B :: K}{E', X{<:}A{::}K, E'' \vdash^{n} X <: B :: K}$$

(NCon Sub Top)
$$\frac{E \vdash A :: Ty}{E \vdash^{n} A <: Top :: Ty}$$

(NCon Sub Abs)
$$\frac{E, X{::}K \vdash^{n} B <: B' :: L}{E \vdash^{n} \lambda(X{::}K)B <: \lambda(X{::}K)B' :: K{\Rightarrow}L}$$

(NCon Sub Appl) (when $lub_E(B(A))$ is defined)
$$\frac{E \vdash B :: K{\Rightarrow}L \qquad E \vdash A :: K \qquad E \vdash^{n} (lub_E(B(A)))^{nf} <: C :: L}{E \vdash^{n} B(A) <: C :: L}$$

Normal constructor inclusion (2/2)

(NCon Sub Object) (l_i distinct)

$$\frac{E \vdash^{n} \upsilon_i B_i <: \upsilon_i' B_i' \quad \forall i \in 1..n \qquad E \vdash B_i \quad \forall i \in n{+}1..n{+}m}{E \vdash^{n} [l_i \upsilon_i{:}B_i^{\ i\in 1..n+m}] <: [l_i \upsilon_i'{:}B_i'^{\ i\in 1..n}]}$$

(NCon Sub All)

$$\frac{E \vdash^{n} A' <: A :: K \qquad E, X{<:}A'{::}K \vdash^{n} B <: B'}{E \vdash^{n} \forall(X{<:}A{::}K)B <: \forall(X{<:}A'{::}K)B'}$$

(NCon Sub Rec)

$$\frac{E \vdash \mu(X)A \qquad E \vdash \mu(Y)B \qquad E, Y, X{<:}Y \vdash^{n} A <: B}{E \vdash^{n} \mu(X)A <: \mu(Y)B}$$

(NCon Sub Invariant)

$$\frac{E \vdash B}{E \vdash^{n} {}^{o}B <: {}^{o}B}$$

(NCon Sub Covariant)

$$\frac{E \vdash^{n} B <: B' \qquad \upsilon \in \{^{o}, ^{+}\}}{E \vdash^{n} \upsilon B <: {}^{+}B'}$$

(NCon Sub Contravariant)

$$\frac{E \vdash^{n} B' <: B \qquad \upsilon \in \{^{o}, ^{-}\}}{E \vdash^{n} \upsilon B <: {}^{-}B'}$$

Basic properties (1/3)

Lemma 20.5-2 (Equivalence to normal form)

If $E \vdash A :: K$, then $E \vdash A \leftrightarrow A^{nf} :: K$.

□

Lemma 20.5-3 (Substitution for unbounded type variables)

If $E, X{::}K, E'\{X\} \vdash^{n} \mathfrak{I}\{X\}$ and $E \vdash A :: K$, then $E, (E'\{\!\{A\}\!\})^{nf} \vdash^{n} (\mathfrak{I}\{\!\{A\}\!\})^{nf}$, where $\mathfrak{I} \equiv B <: B' :: K$ or $\mathfrak{I} \equiv \upsilon\, B <: \upsilon'\, B'$.

□

Lemma 20.5-4 (Normal inclusion of operator application)

If $E \vdash^{n} A <: B :: L{\Rightarrow}K$ and $E \vdash C :: L$, then $E \vdash^{n} (A\ C)^{nf} <: (B\ C)^{nf} :: K$.

□

Basic properties (2/3)

Lemma 20.5-5 (Soundness and completeness of normal system)

If $E \vdash^{\mathrm{n}} A <: B :: K$, then $E \vdash A <: B :: K$.

If $E \vdash^{\mathrm{n}} \upsilon\, B <: \upsilon'\, B'$, then $E \vdash \upsilon\, B <: \upsilon'\, B'$.

If $E \vdash A <: B :: K$, then $E^{nf} \vdash^{\mathrm{n}} A^{nf} <: B^{nf} :: K$.

If $E \vdash \upsilon\, B <: \upsilon'\, B'$, then $E^{nf} \vdash^{\mathrm{n}} \upsilon\, B^{nf} <: \upsilon'\, B'^{nf}$.

$\square$

Basic properties (3/3)

Lemma 20.5-6 (Structural subtyping)

(1) If $E \vdash Top <: C$, then $E \vdash C \leftrightarrow Top$.

(2) Let $E \vdash \forall(X{<:}D{::}L)C <: \forall(X{<:}D'{::}L')C'$.
Then $L \equiv L'$, $E \vdash D' <: D :: L$, and $E, X{<:}D'{::}L \vdash C <: C'$.

(3) Let $E \vdash [l_i \upsilon_i{:}B_i\ ^{i \in I}] <: [l_i \upsilon_i'{:}B_i'\ ^{i \in J}]$.

(1) $J \subseteq I$.

(2) If $\upsilon_j' \in \{^\circ, ^+\}$ for some $j \in J$, then $\upsilon_j \in \{^\circ, ^+\}$ and $E \vdash B_j <: B_j'$.

(3) If $\upsilon_j' \in \{^\circ, ^-\}$ for some $j \in J$, then $\upsilon_j \in \{^\circ, ^-\}$ and $E \vdash B_j' <: B_j$.

(4) Let $E \vdash \mu(X)B\{X\} <: \mu(X')B'\{X'\}$.
Then either $E, X \vdash B\{X\} \leftrightarrow B'\{X\}$, or $E, X', X{<:}X' \vdash B\{X\} <: B'\{X'\}$.

□

Subject reduction theorem

Theorem 20.6-1 (Subject reduction)

If $\emptyset \vdash a : A$ and $\vdash a \rightsquigarrow v$, then $\emptyset \vdash v : A$.

Proof

By induction on the derivation of $\vdash a \rightsquigarrow v$.

Questions?