FbD — declaration

$e ::= \ldots$

$d ::= \text{Let } x = e \text{;;}$

$D ::= [d, \ldots]$

$\Delta ::= \{x \mapsto v, \ldots\}$

$e \Rightarrow v$

$\Delta \vdash D \Rightarrow \Delta$

$$\frac{\Delta = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \quad e[v_1/x_1][v_2/x_2]\cdots[v_n/x_n] \Rightarrow v \quad \Delta\{x \mapsto v\} \vdash D \Rightarrow \Delta'}{\Delta \vdash [\text{Let } x = e\text{;;}] @ D \Rightarrow \Delta'}$$

$$\frac{}{\Delta \vdash [] \Rightarrow \Delta}$$

C: T F b D S R M A

C++: E S T F b D S R M A O *

$\frac{}{\tau <: \tau}$ $\qquad$ $\frac{\tau_1' <: \tau_1 \quad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}$ $\qquad$ $\tau_1 <: \tau_2$

I can use $\tau_1$ anywhere that $\tau_2$ is expected.

```
dog*[] dogs = new dog[4];
animal*[] a = dogs;
cat* c = new Cat();
a[0] = c;
```

dog*[]
{
  get : dog*[] → int → dog*
  set : dog*[] → int → dog* → unit
}

animal*[]
{
  get : animal*[] → int → animal*
  set : animal*[] → int → animal* → unit
}

List is mutable

✗ `List<Animal> lst = new List<Dog>();`

`List<? extends Animal> lst = new ArrayList<Dog>();`

```
void foo(Animal* a) {...}
void foo(Dog* d) {...}

foo(x);
```