

# Transformation Priority Premises

From Nothing . . .



to Mutation



`{}`  $\rightarrow$  `nil`

no code at all

```
public String wrap(String s, int length) {  
    return null;  
}
```

code that employs `nil`

```
public String wrap(String s, int length){  
    return "";  
}
```

**nil** → **constant**

**nil**

```
public String wrap(String s) {  
    return "";  
}
```

**constant**

```
public String wrap(String s){  
    return "at least something";  
}
```

constant  $\rightarrow$  constant+



constant

```
public Food giveMeFood(Money money){
```

```
    return   
}
```

constant+

```
public Food giveMeFood(Money money){
```

```
    return  +   
}
```

constant → scalar

constant

```
public Food giveMeFood(Money money){
```

```
    return 
```

```
}
```

scalar

```
public Food giveMeFood(  ){
```

```
    return 
```







```
}
```

statements  $\rightarrow$  statements

statement

```
public Food giveMeFood(  ) {  
    return   
}
```

statements

```
public Food giveMeFood( ,  , ) {  
    return max (    )  
}
```

unconditional → conditional

unconditional

```
public Drink giveMeADrink(){
```

```
    return
```



```
}
```

conditional

```
public Drink giveMeADrink(){
```

```
    if ( TGIF ){
```

```
        return
```



```
    }
```

```
    return
```





```
}
```

scalar → array

scalar

```
public Food giveMeFood(){  
    return  +   
}
```

array

```
public Food giveMeFood(){  
    return [ ,  ]  
}
```



array → container

array

```
public Food[] giveMeFood(){
```

```
    return [   ]  
}
```

container

```
public FoodContainer giveMeFood(){
```

```
    return   
}
```

statement  $\rightarrow$  tail-recursion

statement



tail-recursion



statement  $\rightarrow$  loop

statement

if (! Saturday && ! Sunday) {

return work

}

loop

(for Day day:workday){

return work

}

statement → recursion

statement

```
if (! Saturday && ! Sunday) {  
    return work  
}
```

recursion



**expression → function**

**expression**

**putDoughInForm()**

**putFormInOven()**

**waitForAnHour()**

**function**

**bakeACake(){**

**putDoughInForm()**

**putFormInOven()**

**waitForAnHour()**

**}**

variable → mutation

variable

var animal =



mutation

animal =

