

Lucius Gregory Meredith

RChain Research



New and improved for-comprehension

for(
$$ptrn_{11} \leftarrow src_{11} \& ... \& ptrn_{1n} \leftarrow src_{1n};$$
...
$$ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn};$$
){P}

where src ::= $x \mid x$?! | x!?(a_1 , ..., a_k) and '&' replaces the old meaning of ';'



New and improved for-comprehension desugared

```
[for(
       ptrn_{11} \leftarrow x_{11}!?(a_1, ..., a_k) \& ... \& ptrn_{1n} \leftarrow src_{1n};
...
      ptrn<sub>m1</sub> ← src<sub>m1</sub> & ... & ptrn<sub>mn</sub> ← src<sub>mn</sub>;
new r<sub>11</sub> in
x_{11}!(a_1, ..., a_k, *r_{11})
   | [for(ptrn<sub>11</sub> \leftarrow r<sub>11</sub> & ... & ptrn<sub>1n</sub> \leftarrow src<sub>1n</sub>;
        ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn};
       ){P}]
```

removing send/recv's: x₁₁!?(a₁, ..., a_k)



New and improved for-comprehension desugared

```
[for(
       ptrn_{11} \leftarrow x_{11}?! \& ... \& ptrn_{1n} \leftarrow src_{1n};
       ptrn<sub>m1</sub> ← src<sub>m1</sub> & ... & ptrn<sub>mn</sub> ← src<sub>mn</sub>;
[for(
       (ptrn_{11}, r) \leftarrow x_{11} \& ... \& ptrn_{1n} \leftarrow src_{1n};
       ptrn<sub>m1</sub> ← src<sub>m1</sub> & ... & ptrn<sub>mn</sub> ← src<sub>mn</sub>;
      ){ r!() | P}]
```

removing recv/send's: x₁₁?!

where r is fresh for the whole context

RChain Research



New and improved for-comprehension desugared

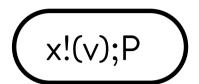
```
[for(
       ptrn_{11} \leftarrow x_{11} \& ... \& ptrn_{1n} \leftarrow x_{1n};
       ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn};
       ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn}
){P}]
for(
      ptrn_{11} \leftarrow x_{11} \& ... \& ptrn_{1n} \leftarrow x_{1n}
){
    [for(
           ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn};
           ptrn_{m1} \leftarrow src_{m1} \& ... \& ptrn_{mn} \leftarrow src_{mn};
      ){P}]
```

removing;'s

RChain Research



sequential output



Allows for sequences of sends



sequential send expressions desugared

$$[x!(v);P] = \text{new r in } x!((v,*r)) \mid \text{for(} _ \leftarrow r){ [P] }$$

removing;'s



let expressions

let
$$ptrn_1 \leftarrow v_1; ...; ptrn_m \leftarrow v_m in P$$

let $ptrn_1 \leftarrow v_1 \& ... \& ptrn_m \leftarrow v_m in P$

These provide immutable variables much like Scala's

val
$$x = v$$
; P



let expressions desugared

```
[let ptrn_1 \leftarrow v_1; ...; ptrn_n \leftarrow v_n in P]

= new x_1 in

x_1!(v_1)

| for(ptrn_1 \leftarrow x_1){

[let ptrn_2 \leftarrow v_2; ...; ptrn_n \leftarrow v_n in P]

}
```

removing;'s



let expressions desugared

[let ptrn₁ ← v₁ & ... & ptrn_n ← v_n in P]
= new x₁ ... x_n in

$$x_1!(v_1) | ... | x_n!(v_n)$$

| for(ptrn₁ ← x₁ & ... & ptrn_n ← x_n){ [P] }

removing &'s