

Generic Reasoning for Custom Concurrency Constructs

Mike Dodds, Suresh Jagannathan & Matthew Parkinson

Can we reason about custom-made concurrency constructs *without* custom-made logics?

This talk:

- `wait()` and `grant()` - pre-existing constructs for sequentialised access to resources.
- domain-specific behaviour: *chains* of threads with in-order access to resources, but out-of-order signalling.
- *different implementations* - we envision `wait()` and `grant()` as part of a library.

[*Quasi-Static Scheduling for Safe Futures*, Navabi, Zhang and Jagannathan, PPoPP'08.]

Results:

- an abstract specification for `wait()` and `grant()`.
- proofs that multiple implementations conform to our specification.
- we can reason about client correctness without requiring knowledge of specific implementations.

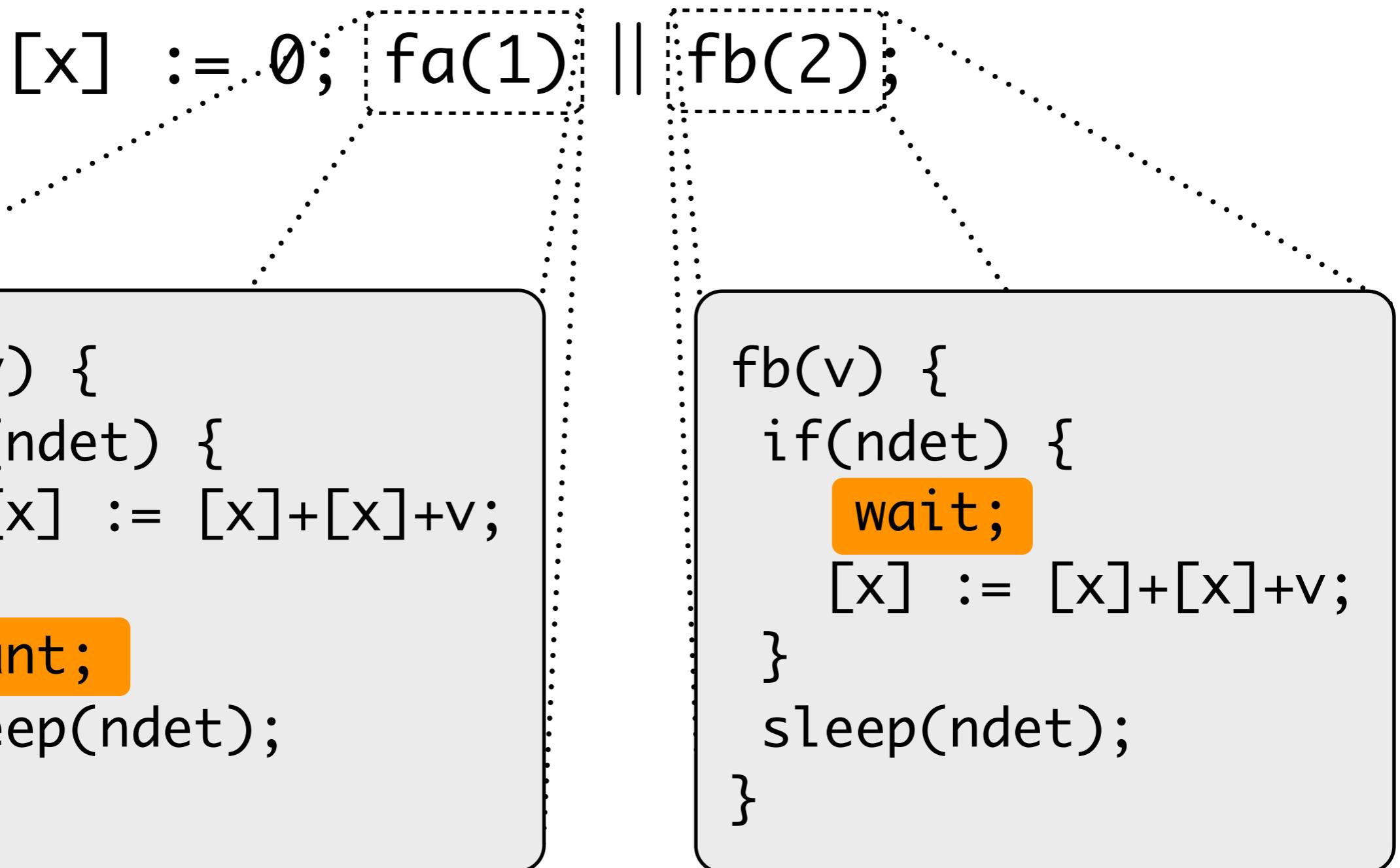
Paper: *Modular Reasoning for Deterministic Parallelism*,
Dodds, Jagannathan, Parkinson, POPL'11.

ordered resource transfer

```
[x] := 0; fa(1) || fb(2);
```

```
[x] := 0; fa(1) || fb(2);
```

```
fa(v) {  
    if(ndet) {  
        [x] := [x]+[x]+v;  
    }  
    grant;  
    sleep(ndet);  
}
```



Abstract specification

$\{ \text{emp} \}$ $i := \text{new}()$ $\{ \text{fut}(i, P) * \text{req}(i, P) \}$

$\{ \text{req}(i, P) * P \}$ $\text{grant}(i)$ $\{ \text{emp} \}$

$\{ \text{fut}(i, P) \}$ $\text{wait}(i)$ $\{ P \}$

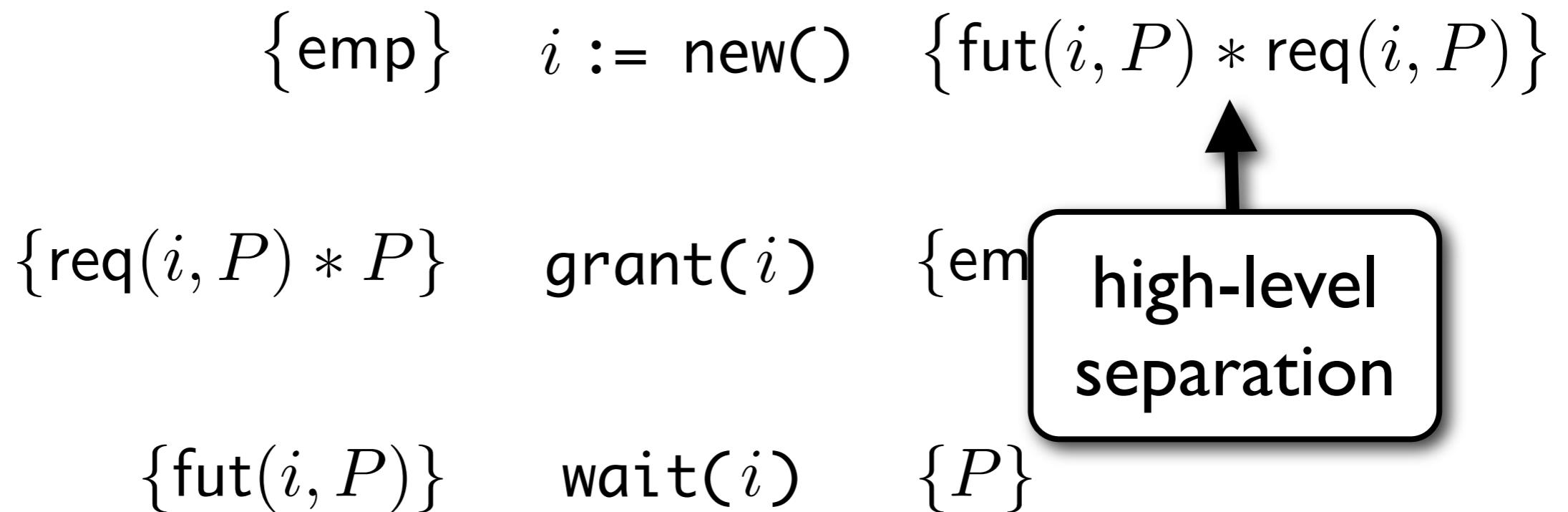
Abstract specification

$$\{ \text{emp} \} \quad i := \text{new}() \quad \{ \text{fut}(i, P) * \text{req}(i, P) \}$$
$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$\exists \text{fut. } \exists \text{req.}$

$$\left(\begin{array}{c} \{ \text{emp} \} \quad i := \text{new}() \quad \{ \text{fut}(i, P) * \text{req}(i, P) \} \\ \{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \} \\ \{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \} \end{array} \right)$$

Abstract specification



Abstract specification

$\{ \text{emp} \}$ $i := \text{new}()$ $\{ \text{fut}(i, P) * \text{req}(i, P) \}$

$\{ \text{req}(i, P) * P \}$ $\text{grant}(i)$ $\{ \text{emp} \}$

$\{ \text{fut}(i, P) \}$ $\text{wait}(i)$ $\{ P \}$

```
fa(1) {  
    if(ndet) {  
        [x] := [x]+[x]+1;  
    }  
    grant(i);  
    sleep(ndet);  
}  
  
fb(2) {  
    if(ndet) {  
        wait(i);  
        [x] := [x]+[x]+2;  
    }  
    sleep(ndet);  
}
```

```

fa(1) {
    if(ndet) {
        [x] := [x]+[x]+1;
    }
    grant(i);
    sleep(ndet);
}

fb(2) {
    if(ndet) {
        wait(i);
        [x] := [x]+[x]+2;
    }
    sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    [x] := [x]+[x]+1;
  }
  grant(i);
  sleep(ndet);
}

fb(2) {
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  grant(i);
  sleep(ndet);
}

fb(2) {
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  sleep(ndet);
}

fb(2) {
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
}

fb(2) {
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}

  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}

    [x] := [x]+[x]+1;

  }

  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }

  sleep(ndet);
  { emp }
}

fb(2) {
  if(ndet) {
    wait(i);

    [x] := [x]+[x]+2;

  }

  sleep(ndet);
}

{req(i, P) * P}   grant(i)   {emp}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  } emp
  {fut(i, P)}    wait(i)    {P}  }
fb(2) {
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  } emp
}

{fut(i, P)}    wait(i)    {P}  }

fb(2) {
  {fut(i, x ↦ {0, 1})}
  if(ndet) {
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}

```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  } emp

```

{fut(i, P)} **wait(i)** **{ P }**

```

fb(2) {
  {fut(i, x ↦ {0, 1})}
  if(ndet) {
    {fut(i, x ↦ {0, 1})}
    wait(i);
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}


```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  } emp

```

{fut(i, P)} **wait(i)** **{ P }**

```

fb(2) {
  {fut(i, x ↦ {0, 1})}
  if(ndet) {
    {fut(i, x ↦ {0, 1})}
    wait(i);
    {x ↦ {0, 1}}
    [x] := [x]+[x]+2;
  }
  sleep(ndet);
}


```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  } emp

```

$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$

```

fb(2) {
  {fut(i, x ↦ {0, 1})}
  if(ndet) {
    {fut(i, x ↦ {0, 1})}
    wait(i);
    {x ↦ {0, 1}}
    [x] := [x]+[x]+2;
  }
  {fut(i, x ↦ {0, 1})
   ∨ x ↦ {2, 4}}
  sleep(ndet);
}


```

```

fa(1) {
  {req(i, x ↦ {0, 1}) * x ↦ 0}
  if(ndet) {
    {req(i, x ↦ {0, 1}) * x ↦ 0}
    [x] := [x]+[x]+1;
  }
  {req(i, x ↦ {0, 1}) * x ↦ {0, 1}}
  grant(i);
  { emp }
  sleep(ndet);
  { emp }
  {fut(i, P)}    wait(i)    {P}

```

```

fb(2) {
  {fut(i, x ↦ {0, 1})}
  if(ndet) {
    {fut(i, x ↦ {0, 1})}
    wait(i);
    {x ↦ {0, 1}}
    [x] := [x]+[x]+2;
  }
  {fut(i, x ↦ {0, 1})
    ∨ x ↦ {2, 4}}
  sleep(ndet);
  {fut(i, x ↦ {0, 1})
    ∨ x ↦ {2, 4}}
}
```

simple
implementation

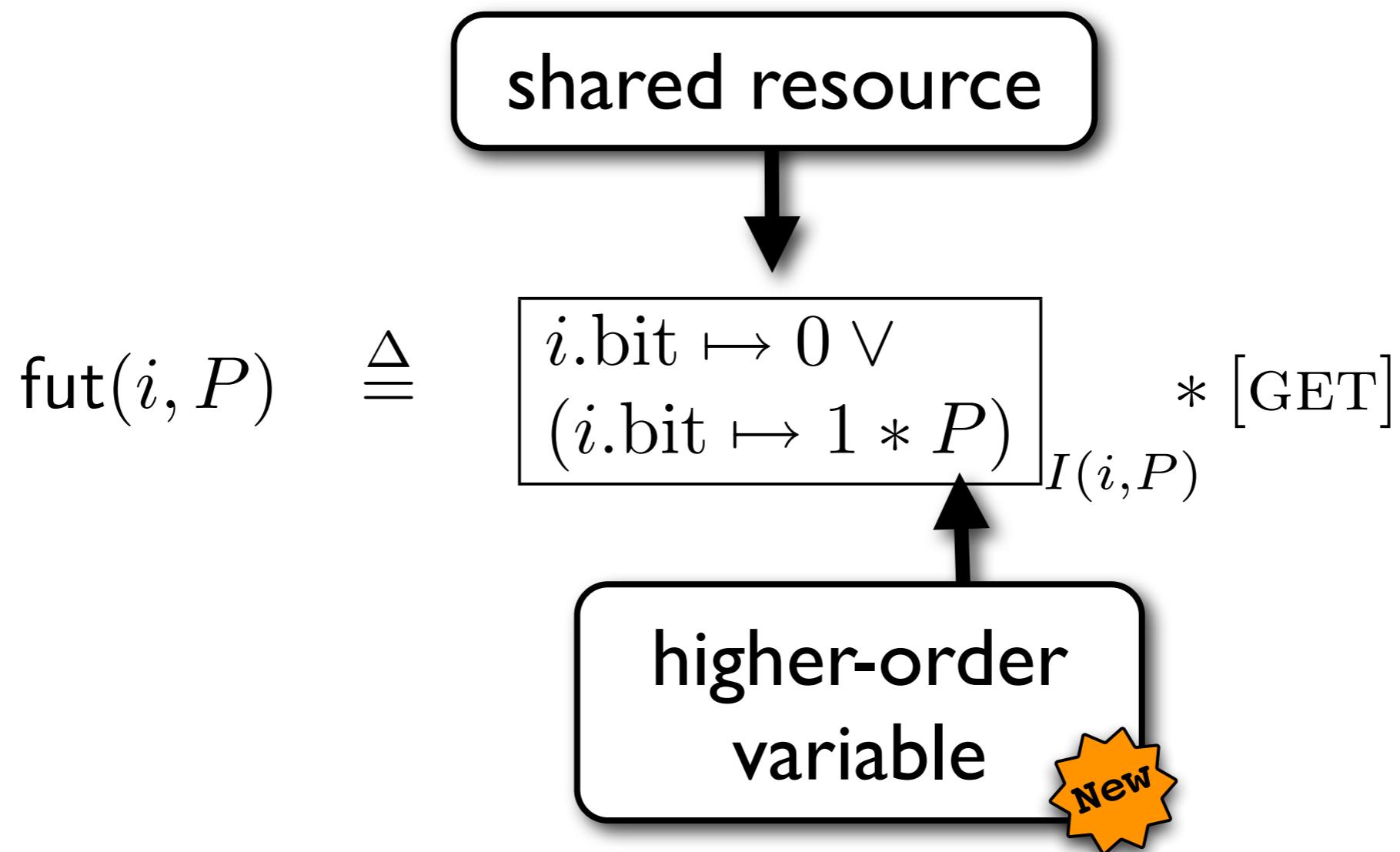
```
wait(i) {  
    while(i.bit != 1) {  
        skip;  
    }  
}  
  
grant(i) {  
    i.bit := 1;  
}
```

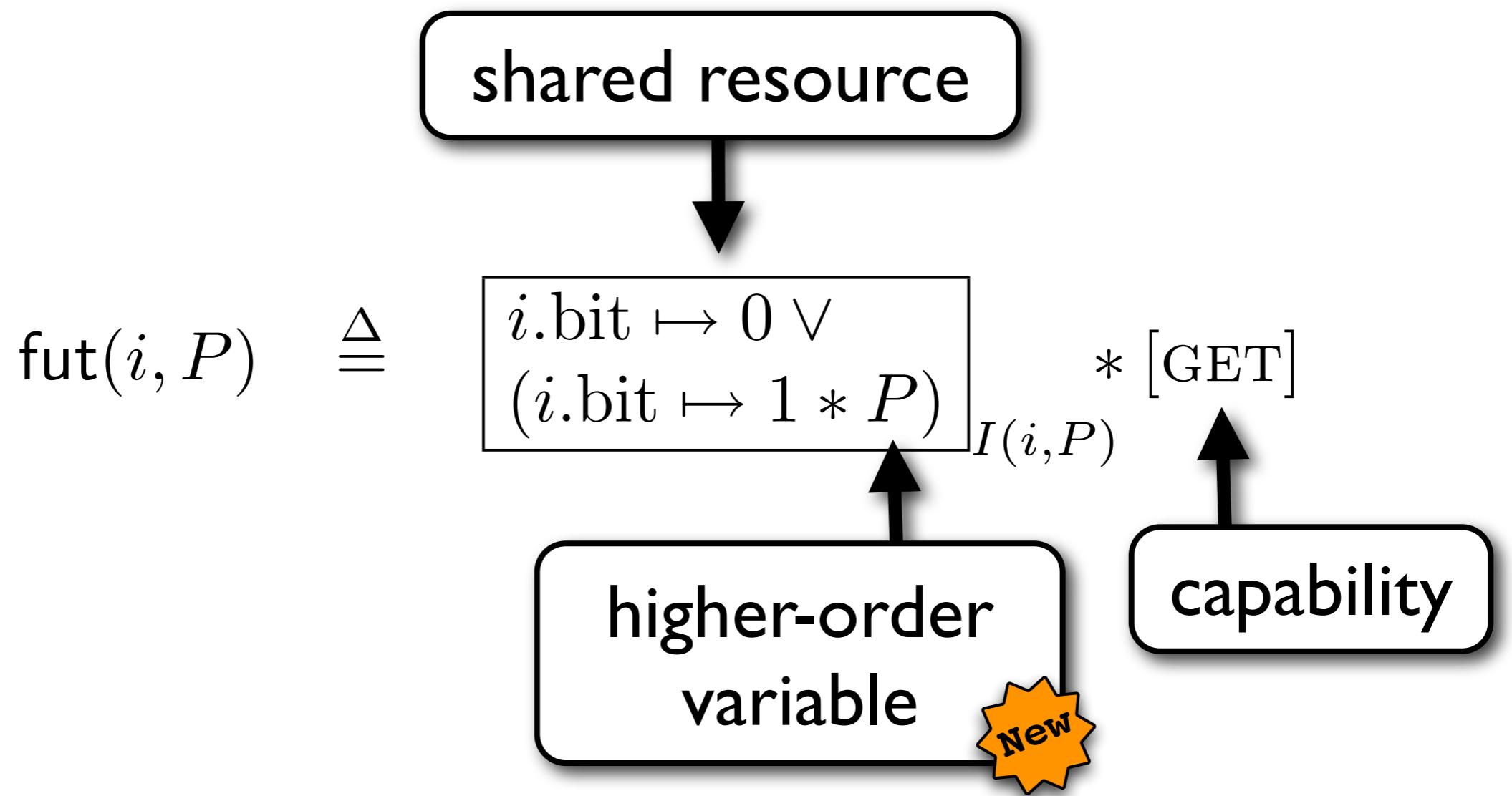
```
new() {  
    i := alloc(...);  
    i.bit := 0;  
    return i;  
}
```

$$\mathbf{fut}(i, P) \triangleq \boxed{i.\text{bit} \mapsto 0 \vee (i.\text{bit} \mapsto 1 * P)}_{I(i, P)} * [\text{GET}]$$

shared resource

$$\text{fut}(i, P) \triangleq \left[\begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i, P)} * [\text{GET}]$$





```
wait(i) {  
    while(i.bit != 1) {  
        skip;  
    }  
}
```

$\{\text{fut}(i, P)\}$ $\text{wait}(i)$ $\{P\}$

```
wait(i) {
```

```
    while(i.bit != 1) {
```

```
        skip;
```

```
}
```

```
}
```

$\{\text{fut}(i, P)\}$ $\text{wait}(i)$ $\{P\}$

```
wait(i) {  
    {fut( $i, P$ )}
```

```
while(i.bit != 1) {
```

```
skip;
```

```
}
```

```
}
```

{fut(i, P)} wait(i) { P }

```

wait(i) {
  {fut(i, P)}
   $\left\{ \left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$ 
  while(i.bit != 1) {
    skip;
  }
}

```

{fut}(i, P) **wait**(i) **{P}**

```

wait(i) {
  {fut(i, P)}

   $\left\{ \begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right|_{I(i,P)} * [\text{GET}] \right\}$ 

  while(i.bit != 1) {
    { $i.\text{bit} \mapsto 0 \vee (i.\text{bit} \mapsto 1 * P)$  }I(i,P)* [GET]
    skip;
  }

  }

```

{fut(*i*, *P*)} **wait**(*i*) {*P*}

```

wait(i) {
  {fut(i, P)}  

   $\left\{ \left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

  while(i.bit != 1) {
    { $\left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

    skip;  

  }  

  { $\left[ \begin{array}{l} i.\text{bit} \mapsto 1 * P \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

}

```

$\{\text{fut}(i, P)\}$ $\text{wait}(i)$ $\{P\}$

```

wait(i) {
  {fut(i, P)}  

   $\left\{ \left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

  while(i.bit != 1) {
    { $\left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

    skip;  

  }  

  { $\left[ \begin{array}{l} i.\text{bit} \mapsto 1 * P \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$   

  // move resource out of the shared area, using the capability.  

  { $\left[ \begin{array}{l} i.\text{bit} \mapsto 1 \end{array} \right]_{I(i,P)}^* [\text{GET}] * P \right\}$   

}

```

$\{fut(i, P)\}$ $wait(i)$ $\{P\}$

```

wait(i) {
  { fut(i, P) }
   $\left\{ \left[ \begin{array}{l} i.\text{bit} \mapsto 0 \vee \\ (i.\text{bit} \mapsto 1 * P) \end{array} \right]_{I(i,P)}^* [\text{GET}] \right\}$ 
  while(i.bit != 1) {
     $\left\{ \left[ i.\text{bit} \mapsto 0 \vee (i.\text{bit} \mapsto 1 * P) \right]_{I(i,P)}^* [\text{GET}] \right\}$ 
    skip;
  }
   $\left\{ \left[ i.\text{bit} \mapsto 1 * P \right]_{I(i,P)}^* [\text{GET}] \right\}$ 
  // move resource out of the shared area, using the capability.
   $\left\{ \left[ i.\text{bit} \mapsto 1 \right]_{I(i,P)}^* [\text{GET}] * P \right\}$ 
  // garbage collect.
   $\left\{ P \right\}$ 

```

$\{ \text{fut}(i, P) \}$ $\text{wait}(i)$ $\{ P \}$

out-of-order signalling

```
fa(v) {  
    if(ndet) {  
        [x] := [x]+[x]+v;  
    }  
    grant(i);  
    sleep(ndet);  
}
```

```
fb(v) {  
    if(ndet) {  
        wait(i);  
        [x] := [x]+[x]+v;  
    }  
    sleep(ndet);  
}
```

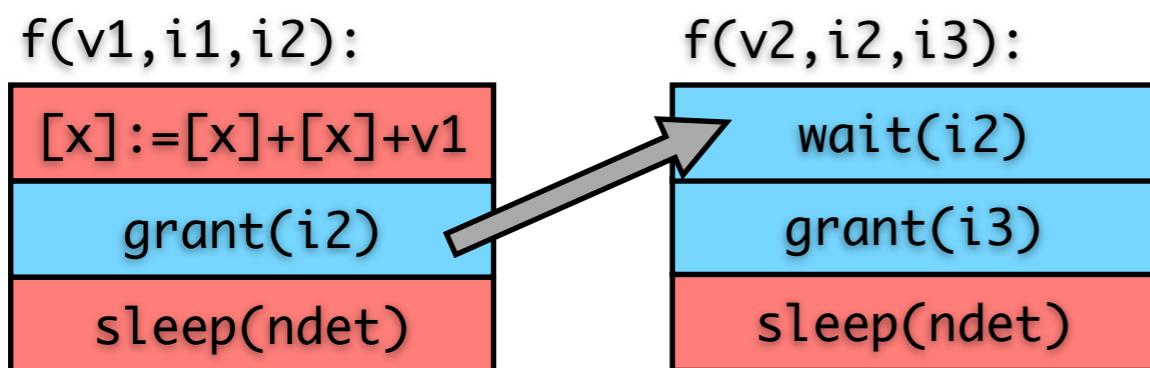
```
f(v,i,j) {  
    if(ndet) {  
        wait(i);  
        [x] := [x]+[x]+v;  
    }  
    grant(j);  
    sleep(ndet);  
}
```

```
[x] := 0;  
f(v1,i1,i2) || f(v2,i2,i3) || f(v3,i3,i4) ...
```

f(v1,i1,i2):

[x]:=[x]+[x]+v1
grant(i2)
sleep(ndet)

```
[x] := 0;  
f(v1,i1,i2) || f(v2,i2,i3) || f(v3,i3,i4) ...
```



$[x] := 0;$

$f(v_1, i_1, i_2) \parallel f(v_2, i_2, i_3) \parallel f(v_3, i_3, i_4) \dots$

$f(v_1, i_1, i_2):$

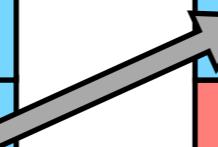
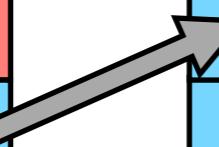
$[x]:=[x]+[x]+v_1$
$grant(i_2)$
$sleep(ndet)$

$f(v_2, i_2, i_3):$

$wait(i_2)$
$grant(i_3)$
$sleep(ndet)$

$f(v_3, i_3, i_4):$

$wait(i_3)$
$[x]:=[x]+[x]+v_3$
$grant(i_4)$
$sleep(ndet)$



$[x] := 0;$

$f(v_1, i_1, i_2) \parallel f(v_2, i_2, i_3) \parallel f(v_3, i_3, i_4) \dots$

$f(v_1, i_1, i_2):$

$[x]:=[x]+[x]+v_1$
grant(i_2)
sleep(ndet)

$f(v_2, i_2, i_3):$

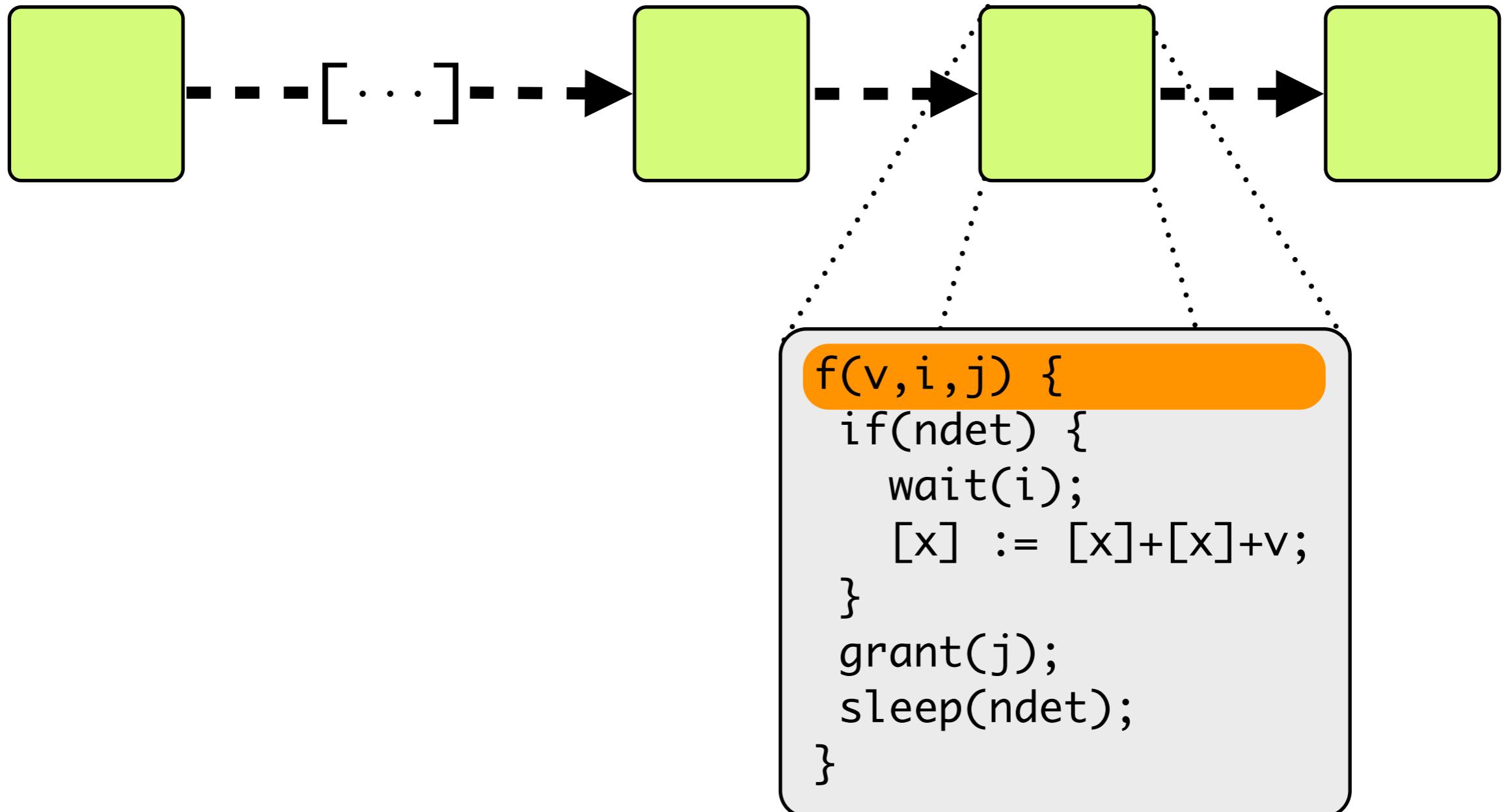
wait(i_2)
grant(i_3)
sleep(ndet)

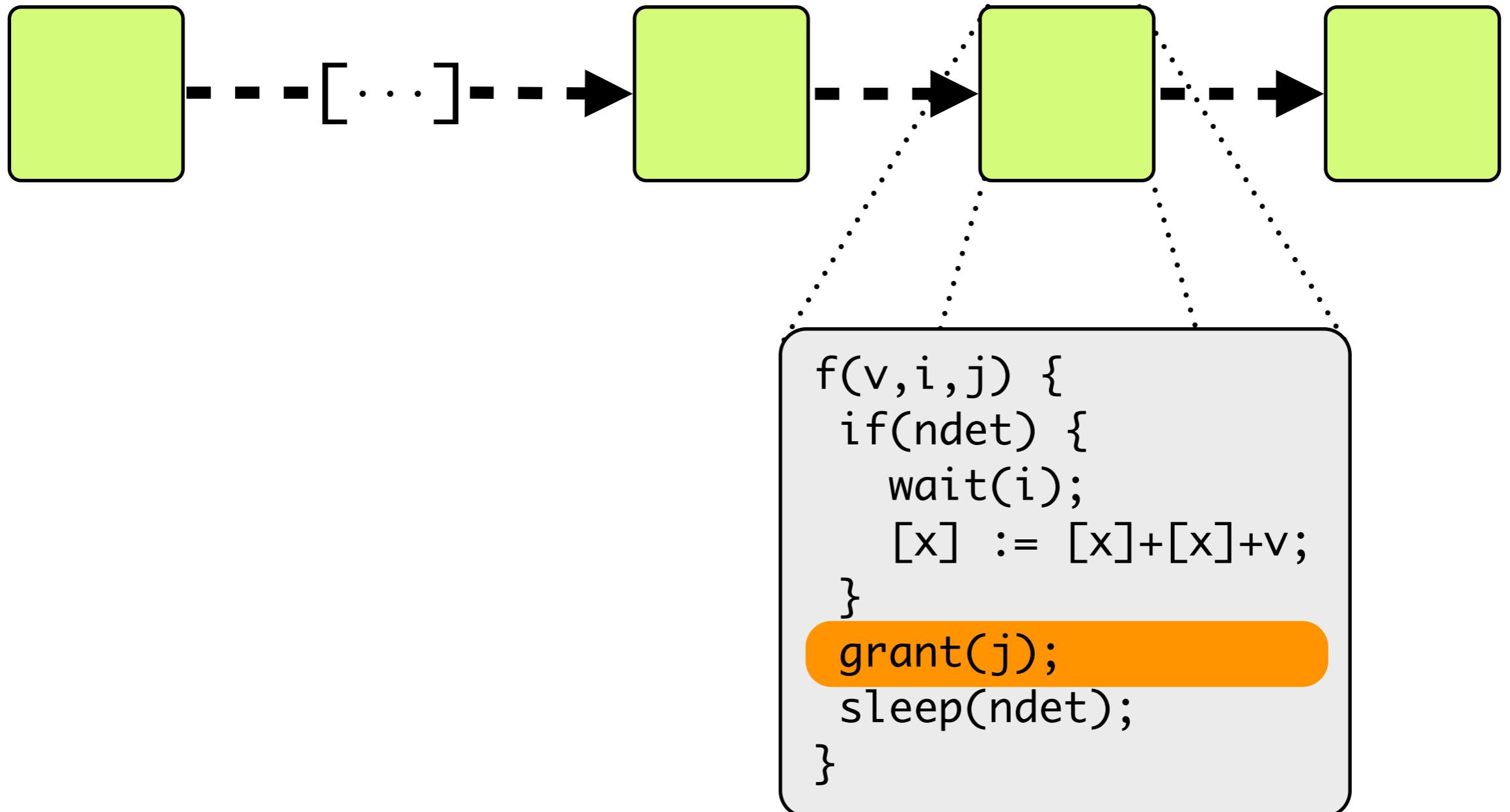
$f(v_3, i_3, i_4):$

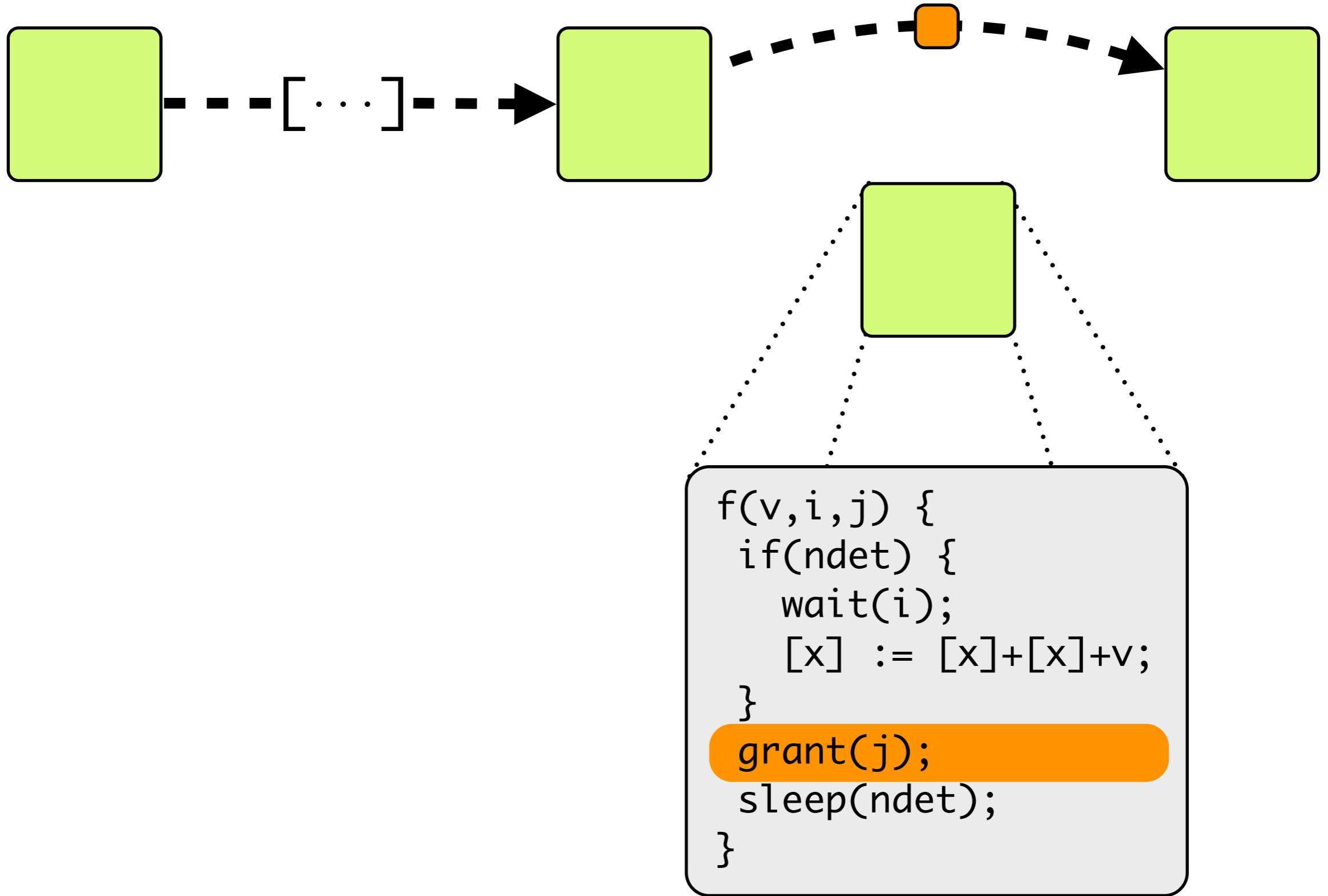
wait(i_3)
 $[x]:=[x]+[x]+v_3$
grant(i_4)
sleep(ndet)

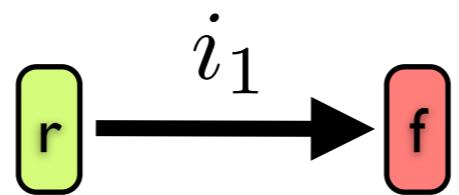
$f(v_4, i_4, i_5):$

wait(i_4)
 $[x]:=[x]+[x]+v_4$
grant(i_5)
sleep(ndet)

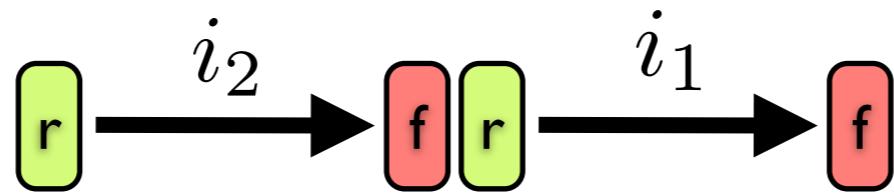






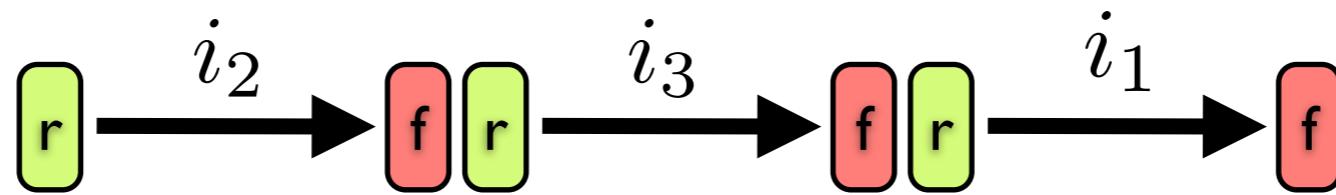


$i_1 := \text{new}(f);$



$i_1 := \text{new}();$

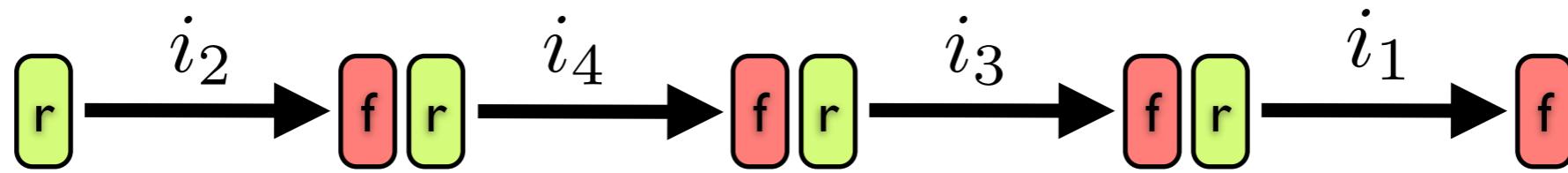
$i_2 := \text{split}(i_1);$



$i_1 := \text{new}();$

$i_2 := \text{split}(i_1);$

$i_3 := \text{split}(i_1);$



$i_1 := \text{new}();$

$i_2 := \text{split}(i_1);$

$i_3 := \text{split}(i_1);$

$i_4 := \text{split}(i_3);$

```
new() {
    i := alloc(...);
    i.prev := nil;
    i.bit := 0;
    return i;
}
```

```
grant(i) {
    i.bit := 1;
}
```

```
split(i) {
    n := alloc(...);
    n.bit := 0;
    n.prev := i.prev;
    i.prev := n;
    return n;
}
```

```
wait(i) {
    do {
        while(i.bit != 1) {
            skip;
        }
        i := i.prev;
    } while (i != nil)
}
```

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

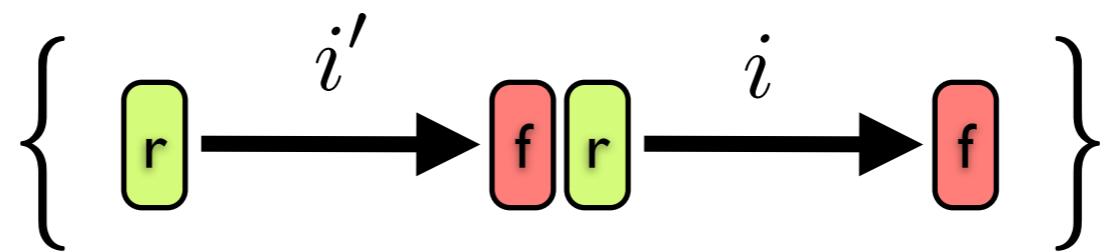
$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

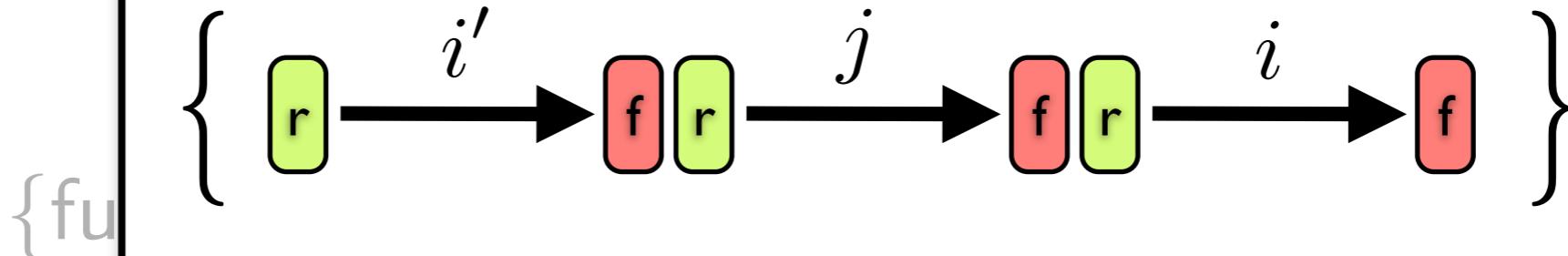
$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$\{ \text{req}(i, P) \}$



$\{ \text{req}(i, P) * \text{fut}(i, Q) * \text{pr}(i, R) \}$

$j := \text{split}(i);$



$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

$$\{ \text{emp} \} \quad i := \text{newO} \quad \left\{ \begin{array}{l} \text{fut}(i, P) * \text{req}(i, P) \\ * \text{pr}(i, \text{nil}) \end{array} \right\}$$

$$\{ \text{pr}(i, i') \} \quad j := \text{split}(i) \quad \left\{ \begin{array}{l} \text{pr}(i, j) * \text{pr}(j, i') * \\ \text{fut}(j, Q) * \text{req}(j, Q) \end{array} \right\}$$

$$\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{ \text{emp} \}$$

$$\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{ P \}$$

```

f(x,i,j) {
    if(ndet) {
        wait(i);
        [x] := [x]+[x]+v;
    }
}

```

```

grant(j);
sleep(ndet);
}

```

$\{ \text{req}(i, P) * P \} \text{ grant}(i) \{ \text{emp} \}$
 $\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \text{ grant}(i) \{ \text{emp} \}$
 $\{ \text{fut}(i, P) \} \text{ wait}(i) \{ P \}$

$\{\text{fut}(i, x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

if(ndet) {

 wait(i);

$[x] := [x] + [x] + v;$

}

grant(j);

$\{\text{req}(i, P) * P\} \text{ grant}(i) \{ \text{emp} \}$

sleep(ndet);

$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ * \text{pr}(i, i') \end{array} \right\} \text{ grant}(i) \{ \text{emp} \}$

$\{\text{fut}(i, P)\} \text{ wait}(i) \{ P \}$

$\{\text{fut}(i, x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

if(ndet) {

 wait(i);

 $\{x \mapsto _ * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

[x] := [x]+[x]+v;

}

grant(j);

 $\{\text{req}(i, P) * P\} \text{ grant}(i) \{ \text{emp} \}$

sleep(ndet);

$$\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ \quad * \text{pr}(i, i') \end{array} \right\} \text{ grant}(i) \{ \text{emp} \}$$
$$\{\text{fut}(i, P)\} \text{ wait}(i) \{ P \}$$

$\{\text{fut}(i, x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

if(ndet) {

 wait(i);

 $\{x \mapsto _ * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

[x] := [x]+[x]+v;

}

 $\{(\text{fut}(i, x \mapsto _) \vee x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

grant(j);

{req(i, P) * P} grant(i) {emp}

sleep(ndet);

{req(i, P) * fut(i', P)
 * pr(i, i')} grant(i) {emp}

{fut(i, P)} wait(i) {P}

$\{\text{fut}(i, x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

if(ndet) {

 wait(i);

 $\{x \mapsto _ * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

[x] := [x]+[x]+v;

}

 $\{(\text{fut}(i, x \mapsto _) \vee x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i)\}$

grant(j);

{emp}

sleep(ndet);

{req(i, P) * P} grant(i) {emp}

{req(i, P) * fut(i', P)
 * pr(i, i')} grant(i) {emp}

{fut(i, P)} wait(i) {P}

$\{ \text{fut}(i, x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i) \}$

if(ndet) {

 wait(i);

 $\{ x \mapsto _ * \text{req}(j, x \mapsto _) * \text{pr}(j, i) \}$

[x] := [x]+[x]+v;

}

 $\{ (\text{fut}(i, x \mapsto _) \vee x \mapsto _) * \text{req}(j, x \mapsto _) * \text{pr}(j, i) \}$

grant(j);

 $\{\text{emp}\}$

sleep(ndet);

 $\{\text{emp}\}$ $\{ \text{req}(i, P) * P \} \quad \text{grant}(i) \quad \{\text{emp}\}$ $\left\{ \begin{array}{l} \text{req}(i, P) * \text{fut}(i', P) \\ \quad * \text{pr}(i, i') \end{array} \right\} \quad \text{grant}(i) \quad \{\text{emp}\}$ $\{ \text{fut}(i, P) \} \quad \text{wait}(i) \quad \{P\}$

Results:

- specifications for `wait()` & `grant()` supporting out-of-order signalling.
- proofs against several implementations (see POPL paper).

Can we reason about custom-made concurrency constructs *without* custom-made logics?

Yes