Concurrent Composition

Concurrent Composition

Sequential Composition $P.Q$ (sequential execution)

Concurrent Composition

Sequential Composition $P.Q$ (sequential execution)

P and Q must have exactly the same state variables

Concurrent Composition

Sequential Composition $P.Q$ (sequential execution)

P and Q must have exactly the same state variables

Concurrent Composition $P \| Q$ (parallel execution)

Concurrent Composition

Sequential Composition $P.Q$ (sequential execution)

P and Q must have exactly the same state variables

Concurrent Composition $P \| Q$ (parallel execution)

P and Q must have completely different state variables

and the state variables of the composition are those of both P and Q

Concurrent Composition

Sequential Composition $P.Q$ (sequential execution)

P and Q must have exactly the same state variables

Concurrent Composition $P \| Q$ (parallel execution)

P and Q must have completely different state variables

and the state variables of the composition are those of both P and Q

Ignoring time and space variables

$$P \| Q \;=\; P \wedge Q$$

Concurrent Composition

example in integer variables x , y , and z

$$x := x+1 \parallel y := y+2$$

Concurrent Composition

example in integer variables x , y , and z

$x := x+1 \parallel y := y+2$ partition the variables:

Concurrent Composition

example in integer variables x , y , and z

$x := x+1 \parallel y := y+2$

partition the variables:

put x in left part, put y and z in right part

Concurrent Composition

example in integer variables x , y , and z

$x := x+1 \parallel y := y+2$ partition the variables:

put x in left part, put y and z in right part

$=$ $x' = x+1 \parallel y' = y+2 \wedge z'=z$

Concurrent Composition

example in integer variables x , y , and z

$x := x+1 \parallel y := y+2$ partition the variables:

put x in left part, put y and z in right part

$= \quad x' = x+1 \parallel y' = y+2 \wedge z'=z$

$= \quad x' = x+1 \wedge y' = y+2 \wedge z'=z$

Concurrent Composition

example in integer variables x , y , and z

$x := x+1 \parallel y := y+2$

partition the variables:

put x in left part, put y and z in right part

$= \quad x' = x+1 \parallel y' = y+2 \wedge z'=z$

$= \quad x' = x+1 \wedge y' = y+2 \wedge z'=z$

reasonable partition rule

If either x' or $x:=$ appears in a process specification, then x belongs to that process

(then neither x' nor $x:=$ can appear in the other process specification).

If neither x' nor $x:=$ appears at all, then x can be placed on either side of the partition.

Concurrent Composition

example in variables x , y , and z

$$x := y \parallel y := x$$

Concurrent Composition

example in variables x , y , and z

$x := y \parallel y := x$

partition: put x in left, y in right, z in either

Concurrent Composition

example in variables x , y , and z

$x := y \parallel y := x$ partition: put x in left, y in right, z in either

$=$ $x' = y \wedge y' = x \wedge z' = z$

Concurrent Composition

example in variables x , y , and z

$x := y \parallel y := x$ partition: put x in left, y in right, z in either

$=$ $x' = y \wedge y' = x \wedge z' = z$

implementation of a process makes a private copy of the initial value of a variable belonging to the other process if the other process contains an assignment to that variable

Concurrent Composition

example in binary variable b and integer variable x

$$b := x = x \;\|\; x := x + 1$$

Concurrent Composition

example in binary variable b and integer variable x

$b := x=x \parallel x := x+1$ replace $x=x$ by $\top$

$= \quad b := \top \parallel x := x+1$

Concurrent Composition

example in binary variable b and integer variable x

$b := x=x \parallel x := x+1$ replace $x=x$ by $\top$

$= \quad b := \top \parallel x := x+1$

example in integer variables x and y

$(x := x+1 \,.\; x := x-1) \parallel y := x$

Concurrent Composition

example in binary variable b and integer variable x

$b := x=x \parallel x := x+1$ replace $x=x$ by $\top$

$= \quad b := \top \parallel x := x+1$

example in integer variables x and y

$(x := x+1 \,.\; x := x-1) \parallel y := x$

$= \quad ok \parallel y := x$

Concurrent Composition

example in binary variable b and integer variable x

$b := x=x \parallel x := x+1$ replace $x=x$ by $\top$

$= \quad b := \top \parallel x := x+1$

example in integer variables x and y

$(x := x+1 \,.\; x := x-1) \parallel y := x$

$= \quad ok \parallel y := x$

$= \quad y := x$

Concurrent Composition

$(x := x+y.\ \ x := x \times y) \ \ \| \ \ (y := x-y.\ \ y := x/y)$

Concurrent Composition

$(x := x+y.\ \ x := x \times y) \ \|\ (y := x-y.\ \ y := x/y)$

Concurrent Composition

$(x := x+y.\ \ x := x \times y) \ \| \ (y := x-y.\ \ y := x/y)$

Concurrent Composition

$$(x := x+y.\ \ x := x \times y) \ \ \| \ \ (y := x-y.\ \ y := x/y)$$

You should have written

$$(x := x+y \ \| \ y := x-y).\ (x := x \times y \ \| \ y := x/y)$$

Concurrent Composition

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad \text{(substitute } tP \text{ for } t' \text{ in } P\text{)}$$

$$\wedge \ \text{(substitute } tQ \text{ for } t' \text{ in } Q\text{)}$$

$$\wedge \ t' = tP \uparrow tQ$$

Concurrent Composition

$P \| Q \;=\; \exists tP, tQ \cdot$ (substitute tP for t' in P)

$\rightarrow$ $\wedge$ (substitute tQ for t' in Q)

$\wedge\ t' = tP \uparrow tQ$

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$

$P \parallel Q \;=\; Q \parallel P$ symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ distributivity

$P \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \parallel Q \textbf{ else } P \parallel R \textbf{ fi}$ distributivity

$\textbf{if } b \textbf{ then } P \| Q \textbf{ else } R \| S \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \textbf{ else } R \textbf{ fi} \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } S \textbf{ fi}$ distributivity

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$ ←

$P \parallel Q \;=\; Q \parallel P$ symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ distributivity

$P \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \parallel Q \textbf{ else } P \parallel R \textbf{ fi}$ distributivity

$\textbf{if } b \textbf{ then } P \| Q \textbf{ else } R \| S \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \textbf{ else } R \textbf{ fi} \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } S \textbf{ fi}$ distributivity

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$

$P \parallel Q \;=\; Q \parallel P$ ← symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ distributivity

$P \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \parallel Q \textbf{ else } P \parallel R \textbf{ fi}$ distributivity

$\textbf{if } b \textbf{ then } P \| Q \textbf{ else } R \| S \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \textbf{ else } R \textbf{ fi} \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } S \textbf{ fi}$ distributivity

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$

$P \parallel Q \;=\; Q \parallel P$ symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ ← associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ distributivity

$P \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \parallel Q \textbf{ else } P \parallel R \textbf{ fi}$ distributivity

$\textbf{if } b \textbf{ then } P \| Q \textbf{ else } R \| S \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \textbf{ else } R \textbf{ fi} \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } S \textbf{ fi}$ distributivity

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$

$P \parallel Q \;=\; Q \parallel P$ — symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ — associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ ← — distributivity

$P \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } R \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \parallel Q \textbf{ else } P \parallel R \textbf{ fi}$ — distributivity

$\textbf{if } b \textbf{ then } P \| Q \textbf{ else } R \| S \textbf{ fi} \;=\; \textbf{if } b \textbf{ then } P \textbf{ else } R \textbf{ fi} \parallel \textbf{if } b \textbf{ then } Q \textbf{ else } S \textbf{ fi}$ — distributivity

Concurrent Composition

$$P \| Q \;=\; \exists tP, tQ \cdot \quad (\text{substitute } tP \text{ for } t' \text{ in } P)$$
$$\wedge\ (\text{substitute } tQ \text{ for } t' \text{ in } Q)$$
$$\wedge\ t' = tP \uparrow tQ$$

laws

$(x := e \parallel y := f).\ P \;=\; (\text{for } x \text{ substitute } e \text{ and concurrently for } y \text{ substitute } f \text{ in } P)$

$P \parallel Q \;=\; Q \parallel P$ symmetry

$P \parallel (Q \parallel R) \;=\; (P \parallel Q) \parallel R$ associativity

$P \parallel Q \vee R \;=\; (P \parallel Q) \vee (P \parallel R)$ distributivity

if b **then** Q **else** R **fi** $\;=\;$ **if** b **then** $P \parallel Q$ **else** $P \parallel R$ **fi** ← distributivity

if b **then** $P \| Q$ **else** $R \| S$ **fi** $\;=\;$ **if** b **then** P **else** R **fi** $\parallel$ **if** b **then** Q **else** S **fi** ← distributivity

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$$life \Leftarrow Socrates \parallel Plato$$

$$\begin{aligned} Socrates \Leftarrow\ & Sthinks.\ t := t + Sthinktime. \\ & Stalks.\ t := t + Stalktime. \\ & Socrates \end{aligned}$$

$$\begin{aligned} Plato \Leftarrow\ & Pthinks.\ t := t + Pthinktime. \\ & Ptalks.\ t := t + Ptalktime. \\ & Plato \end{aligned}$$

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow (Stalking := \bot \parallel Ptalking := \bot).\ (Socrates \parallel Plato)$

$Socrates \Leftarrow Sthinks.\ t := t + Sthinktime.$

if *Ptalking* **then** *ok* **else** $Stalking := \top.\ Stalks.\ t := t + Stalktime.\ Stalking := \bot$ **fi**

Socrates

$Plato \Leftarrow Pthinks.\ t := t + Pthinktime.$

if *Stalking* **then** *ok* **else** $Ptalking := \top.\ Ptalks.\ t := t + Ptalktime.\ Ptalking := \bot$ **fi**

Plato

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow bothThink.\ SocratesTalks \ntriangleleft PlatoTalks.\ life$

$bothThink \Leftarrow (Sthinks \parallel Pthinks).\ t{:=}\ t{+}boththinktime$

$SocratesTalks \Leftarrow (Stalks \parallel Pthinks).\ t{:=}\ t{+}Stalktime$

$PlatoTalks \Leftarrow (Ptalks \parallel Sthinks).\ t{:=}\ t{+}Ptalktime$

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow bothThink.\ SocratesTalks \mathbin{+} PlatoTalks.\ life$

$bothThink \Leftarrow (Sthinks \,||\, Pthinks).\ t := t+boththinktime$

$SocratesTalks \Leftarrow (Stalks \,||\, Pthinks).\ t := t+Stalktime$

$PlatoTalks \Leftarrow (Ptalks \,||\, Sthinks).\ t := t+Ptalktime$

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow bothThink.\ SocratesTalks \mathbin{+} PlatoTalks.\ life$

$bothThink \Leftarrow (Sthinks \parallel Pthinks).\ t := t+boththinktime$

$SocratesTalks \Leftarrow (Stalks \parallel Pthinks).\ t := t+Stalktime$ ←

$PlatoTalks \Leftarrow (Ptalks \parallel Sthinks).\ t := t+Ptalktime$ ←

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow bothThink.\ SocratesTalks \neq PlatoTalks.\ life$

$bothThink \Leftarrow (Sthinks \,||\, Pthinks).\ t := t+boththinktime$

$SocratesTalks \Leftarrow (Stalks \,||\, Pthinks).\ t := t+Stalktime$

$PlatoTalks \Leftarrow (Ptalks \,||\, Sthinks).\ t := t+Ptalktime$

$SocratesTalks \neq PlatoTalks \Leftarrow$

Talking Philosophers

Socrates and Plato are philosophers. They think for a while, and when one of them has an interesting thought, he says it aloud, and then goes back to thinking. But they must not talk at the same time. Write a program whose execution simulates the life of these philosophers.

$life \Leftarrow bothThink.\ SocratesTalks \neq PlatoTalks.\ life$

$bothThink \Leftarrow (Sthinks \parallel Pthinks).\ t := t+boththinktime$

$SocratesTalks \Leftarrow (Stalks \parallel Pthinks).\ t := t+Stalktime$

$PlatoTalks \Leftarrow (Ptalks \parallel Sthinks).\ t := t+Ptalktime$

$SocratesTalks \neq PlatoTalks \Leftarrow$ **if** *turn* **then** *SocratesTalks* **else** *PlatoTalks* **fi**

List Concurrency

List Concurrency

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j : 0,..\#L \cdot j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$$

List Concurrency

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j : 0,..\#L \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$$

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j : \text{(this part)} \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$$

List Concurrency

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon 0\,..\#L \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$$

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon \text{(this part)} \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$$

example find the maximum item in a nonempty list

List Concurrency

$$L\,i{:=}\,e \quad = \quad L'i{=}e \;\wedge\; (\forall j{:}\; 0,..\#L{\cdot}\; j{\neq}i \Rightarrow L'j = L\,j) \;\wedge\; x'{=}x \;\wedge\; y'{=}y \;\wedge\; \ldots$$

$$L\,i{:=}\,e \quad = \quad L'i{=}e \;\wedge\; (\forall j{:}\; (\text{this part}){\cdot}\; j{\neq}i \Rightarrow L'j = L\,j) \;\wedge\; x'{=}x \;\wedge\; \ldots$$

example find the maximum item in a nonempty list

$$\mathit{findmax} \quad = \quad \langle i, j{\cdot}\;\; i{<}j \Rightarrow L'\; i = \Uparrow(L\;[i;..j])\rangle$$

List Concurrency

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j{:}\ 0,..\#L \cdot\ j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$$

$$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j{:}\ \text{(this part)} \cdot\ j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

$$\mathit{findmax} \;=\; \langle i, j \cdot\ i < j \Rightarrow L'\ i = \Uparrow(L\ [i;..j]) \rangle$$

List Concurrency

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon 0\,..\#L\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon \text{(this part)}\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

$findmax \;=\; \langle i, j\cdot\; i<j \Rightarrow L'\,i = \Uparrow(L\,[i;..j])\rangle$

findmax i j $\Leftarrow$ **if** $j-i = 1$ **then** *ok*

else (*findmax i* (*div* (*i*+*j*) 2) $\|$ *findmax* (*div* (*i*+*j*) 2) *j*).

$L\,i := (L\,i)\uparrow(L\,(div\;(i+j)\;2))$ **fi**

List Concurrency

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon 0,..\#L\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon \text{(this part)}\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

$findmax \;=\; \langle i, j\cdot\; i<j \Rightarrow L'\,i = \Uparrow(L\,[i;..j])\rangle$

$findmax\;i\;j \;\Leftarrow\;$ **if** $j-i = 1$ **then** *ok* ←

else (*findmax i* (*div* (*i*+*j*) 2) ‖ *findmax* (*div* (*i*+*j*) 2) *j*).

$L\,i := (L\,i)\uparrow(L\,(div\;(i+j)\;2))$ **fi**

List Concurrency

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j: 0,..\#L \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j: \text{(this part)} \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

findmax $= \langle i, j \cdot\; i<j \Rightarrow L'\,i = \Uparrow(L\,[i;..j]) \rangle$

findmax i j $\Leftarrow$ **if** $j-i = 1$ **then** *ok*

$\rightarrow$ **else** (*findmax i* (*div* (*i*+*j*) 2) $\|$ *findmax* (*div* (*i*+*j*) 2) *j*).

$L\,i := (L\,i) \uparrow (L\,(div\;(i+j)\;2))$ **fi**

List Concurrency

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon 0,..\#L\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j\colon \text{(this part)}\cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

$findmax \;=\; \langle i, j\cdot\; i<j \Rightarrow L'\,i = \Uparrow(L\,[i;..j])\rangle$

findmax i j $\Leftarrow$ **if** $j-i = 1$ **then** *ok*

else (*findmax i* (*div* (*i*+*j*) 2) $\|$ *findmax* (*div* (*i*+*j*) 2) *j*).

$\rightarrow$ $L\,i := (L\,i)\uparrow(L\,(div\;(i+j)\;2))$ **fi**

List Concurrency

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j: 0,..\#L \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; y' = y \;\wedge\; \ldots$

$L\,i := e \quad = \quad L'i = e \;\wedge\; (\forall j: \text{(this part)} \cdot\; j \neq i \Rightarrow L'j = L\,j) \;\wedge\; x' = x \;\wedge\; \ldots$

example find the maximum item in a nonempty list

findmax 0 (#*L*) where

findmax $= \langle i, j \cdot\; i<j \Rightarrow L'\,i = \Uparrow(L\,[i;..j]) \rangle$

findmax i j $\Leftarrow$ **if** $j-i = 1$ **then** *ok*

else (*findmax i* (*div* (*i*+*j*) 2) $\|$ *findmax* (*div* (*i*+*j*) 2) *j*).

$L\,i := (L\,i) \uparrow (L\,(div\ (i+j)\ 2))$ **fi**

recursive time = *ceil* (*log* (*j*–*i*))