Interaction

shared variables

can be read and written by any process (most interaction)

difficult to implement

difficult to reason about

interactive variables

can be read by any process, written by only one process (some interaction)

easier to implement

easier to reason about

boundary variables

can be read and written by only one process (least interaction)

but initial value can be seen by all processes

easiest to implement

easiest to reason about

Interactive Variables

boundary variable $\quad$ **var** $a{:}\ T\cdot\ S \ = \ \exists a, a'{:}\ T\cdot\ S$

interactive variable $\quad$ **ivar** $x{:}\ T\cdot\ S \ = \ \exists x{:}\ time\to T\cdot\ S$

The value of variable x at time t is $x\,t$

But sometimes we write x for $x\,t$, x' for $x\,t'$, x'' for $x\,t''$, ...

$a{:}{=}\ a{+}x$

is really

$a{:}{=}\ a + x\,t$

Most laws still work but not the Substitution Law

Interactive Variables

suppose boundary a , b ; interactive x , y ; time t

$$ok \quad = \quad a'=a \;\land\; b'=b \;\land\; t'=t$$

$$x'=x \;\land\; y'=y \quad \text{means} \quad x\,t' = x\,t \;\land\; y\,t' = y\,t$$

Interactive Variables

suppose boundary a , b ; interactive x , y ; time t

$ok \;=\; a'=a \;\land\; b'=b \;\land\; t'=t$

$a{:=}\,e \;=\; a'=e \;\land\; b'=b \;\land\; t'=t$

$x{:=}\,e \;=\; a'=a \;\land\; b'=b \;\land\; x'=e \;\land\; (\forall t''\cdot\, t{\le}t''{\le}t' \Rightarrow y''=y)$

$\land\;\; t' = t+$(the time required to evaluate and store e)

$P.\,Q \;=\; \exists a'', b'', t''\cdot$ (substitute a'', b'', t'' for a', b', t' in P)

$\land$ (substitute a'', b'', t'' for a, b, t in Q)

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

$\land$ (substitute t_Q for t' in Q)

$\land\;\; t' = max\ t_P\ t_Q$

$\land\;\; (\forall t''\cdot\, t_P{\le}t''{\le}t' \Rightarrow xt''=x(t_P))$ — interactive variables of P

$\land\;\; (\forall t''\cdot\, t_Q{\le}t''{\le}t' \Rightarrow yt''=y(t_Q))$ — interactive variables of Q

Interactive Variables

example boundary a , b ; interactive x , y ; extended integer time t

$(x{:=}\ 2.\ \ x{:=}\ x{+}y.\ \ x{:=}\ x{+}y) \parallel (y{:=}\ 3.\ \ y{:=}\ x{+}y)$ $\qquad$ x left, y right, a left, b right

$$= \quad (a'{=}a \wedge xt'{=}2 \wedge t'{=}t{+}1.\ \ a'{=}a \wedge xt'{=}\ xt{+}yt \wedge t'{=}t{+}1.\ \ a'{=}a \wedge xt'{=}\ xt{+}yt \wedge t'{=}t{+}1)$$

$$\parallel\ (b'{=}b \wedge yt'{=}3 \wedge t'{=}t{+}1.\ \ b'{=}b \wedge yt'{=}\ xt{+}yt \wedge t'{=}t{+}1)$$

$$= \quad (a'{=}a \wedge x(t{+}1){=}2 \wedge x(t{+}2){=}\ x(t{+}1){+}y(t{+}1) \wedge x(t{+}3){=}\ x(t{+}2){+}y(t{+}2) \wedge t'{=}t{+}3)$$

$$\parallel\ (b'{=}b \wedge y(t{+}1){=}3 \wedge y(t{+}2){=}\ x(t{+}1){+}y(t{+}1) \wedge t'{=}t{+}2)$$

$$= \quad x(t{+}1){=}2 \wedge x(t{+}2){=}\ x(t{+}1){+}y(t{+}1) \wedge x(t{+}3){=}\ x(t{+}2){+}y(t{+}2)$$

$$\wedge\ y(t{+}1){=}3 \wedge y(t{+}2){=}\ x(t{+}1){+}y(t{+}1) \wedge y(t{+}3){=}y(t{+}2)$$

$$\wedge\ a'{=}a \wedge b'{=}b \wedge t'{=}t{+}3$$

$$= \quad x(t{+}1){=}2 \wedge x(t{+}2){=}5 \wedge x(t{+}3){=}10 \wedge y(t{+}1){=}3 \wedge y(t{+}2){=}y(t{+}3){=}5 \wedge a'{=}a \wedge b'{=}b \wedge t'{=}t{+}3$$

Thermostat

$thermometer \parallel control \parallel thermostat \parallel burner$

inputs to the thermostat:

- real *temperature* , which comes from the thermometer and indicates the actual temperature.
- real *desired* , which comes from the control and indicates the desired temperature.
- binary *flame* , which comes from a flame sensor in the burner and indicates whether there is a flame.

outputs of the thermostat:

- binary *gas* ; assigning it $\top$ turns the gas on and $\bot$ turns the gas off.
- binary *spark* ; assigning it $\top$ causes sparks for the purpose of igniting the gas.

Heat is wanted when the actual temperature falls ε below the desired temperature, and not wanted when the actual temperature rises ε above the desired temperature, where ε is small enough to be unnoticeable, but large enough to prevent rapid oscillation. To obtain heat, the spark should be applied to the gas for at least 1 second to give it a chance to ignite and to allow the flame to become stable. But a safety regulation states that the gas must not remain on and unlit for more than 3 seconds. Another regulation says that when the gas is shut off, it must not be turned on again for at least 20 seconds to allow any accumulated gas to clear. And finally, the gas burner must respond to its inputs within 1 second.

$$thermostat \;=\; (gas{:=}\bot \parallel spark{:=}\bot).\; GasOff$$

$$\begin{aligned}GasOff \;=\;\; & \textbf{if } temperature < desired - \varepsilon\\ & \textbf{then } (gas{:=}\top \parallel spark{:=}\top \parallel t' \ge t{+}1) \;\wedge\; t' \le t{+}3.\; spark{:=}\bot.\; GasOn\\ & \textbf{else } ((\textbf{frame } gas, spark\cdot ok) \parallel t'{\ge}t) \;\wedge\; t' \le t{+}1.\; GasOff \textbf{ fi}\end{aligned}$$

$$\begin{aligned}GasOn \;=\;\; & \textbf{if } temperature < desired + \varepsilon \;\wedge\; flame\\ & \textbf{then } ((\textbf{frame } gas, spark\cdot ok) \parallel t'{\ge}t) \;\wedge\; t' \le t{+}1.\; GasOn\\ & \textbf{else } (gas{:=}\bot \parallel (\textbf{frame } spark\cdot ok) \parallel t' \ge t{+}20) \;\wedge\; t' \le t{+}21.\; GasOff \textbf{ fi}\end{aligned}$$

Communication Channels

Channel c is described by

message script	$\mathbb{M}c$	string constant
time script	$\mathbb{T}c$	string constant
read cursor	rc	extended natural variable
write cursor	wc	extended natural variable

$$\mathbb{M} \;=\; 6 \;;\; 4 \;;\; 7 \;;\; 1 \;;\; 0 \;;\; 3 \;;\; 8 \;;\; 9 \;;\; 2 \;;\; 5 \;;\; ...$$

$$\mathbb{T} \;=\; 3 \;;\; 5 \;;\; 5 \;;\; \underset{\underset{r}{\uparrow}}{20} \;;\; 25 \;;\; \underset{\underset{w}{\uparrow}}{28} \;;\; 31 \;;\; 31 \;;\; 45 \;;\; 48 \;;\; ...$$

Input and Output

$$c!\,e \;=\; \mathbb{M}_w = e \;\wedge\; \mathbb{T}_w = t \;\wedge\; (w := w+1)$$

$$c! \;=\; \mathbb{T}_w = t \;\wedge\; (w := w+1)$$

$$c? \;=\; r := r+1$$

$$c \;=\; \mathbb{M}_{r-1}$$

$$\surd c \;=\; \mathbb{T}_r \le t$$

$$\begin{array}{lcccccccccccccccccccccc}
\mathbb{M} & = & 6 & ; & 4 & ; & 7 & ; & 1 & ; & 0 & ; & 3 & ; & 8 & ; & 9 & ; & 2 & ; & 5 & ; & \ldots \\
\mathbb{T} & = & 3 & ; & 5 & ; & 5 & ; & 20 & ; & 25 & ; & 28 & ; & 31 & ; & 31 & ; & 45 & ; & 48 & ; & \ldots \\
 & & & & & & & \uparrow & & & & \uparrow & & & & & & & & & & & \\
 & & & & & & & r & & & & w & & & & & & & & & & &
\end{array}$$

Input and Output

$c!\,e \;=\; \mathbb{M}_w = e \;\wedge\; \mathbb{T}_w = t \;\wedge\; (w{:=}\,w{+}1)$

$c! \;=\; \mathbb{T}_w = t \;\wedge\; (w{:=}\,w{+}1)$

$c? \;=\; r{:=}\,r{+}1$

$c \;=\; \mathbb{M}_{r-1}$

$\surd c \;=\; \mathbb{T}_r \le t$

if $\surd key$

then *key*?.

if *key*=“y”

then *screen*! “If you wish.”

else *screen*! “Not if you don't want.” **fi**

else *screen*! “Well?” **fi**

Input and Output

Repeatedly input numbers from channel c , and output their doubles on channel d .

$$S \;=\; \forall n: nat\cdot \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n}$$

$$S \;\Leftarrow\; c?.\ d!\ 2{\times}c.\ S$$

proof

$$c?.\ d!\ 2{\times}c.\ S$$

$= \quad rc{:=}\ rc{+}1.\ \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc-1} \wedge (wd{:=}\ wd{+}1).\ S$

$= \quad \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc} \wedge \forall n: nat\cdot \mathbb{M}d_{wd+1+n} = 2 \times \mathbb{M}c_{rc+1+n}$

$= \quad \forall n: nat\cdot \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n}$

$= \quad S$

Communication Timing

real time need to know implementation

transit time input and output take time 0

communication transit takes time 1

input $c?$	becomes	$t := max\ t\ (\mathbb{T}c_{rc} + 1).\ c?$
check $\surd c$	becomes	$\mathbb{T}c_{rc} + 1 \le t$

Communication Timing

$$W \;=\; t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ c?$$

$\quad\;=\;$ wait (if necessary) for input and then read it

$$W \;\Leftarrow\; \textbf{if}\ \surd c\ \textbf{then}\ c?\ \textbf{else}\ t{:=}\ t{+}1.\ \ W\ \textbf{fi}$$

proof

$\textbf{if}\ \surd c\ \textbf{then}\ c?\ \textbf{else}\ t{:=}\ t{+}1.\ \ W\ \textbf{fi}$

$=\quad \textbf{if}\ \mathbb{T}_r + 1 \le t\ \textbf{then}\ c?\ \textbf{else}\ t{:=}\ t{+}1.\ \ t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ c?\ \textbf{fi}$

$=\quad \textbf{if}\ \mathbb{T}_r + 1 \le t\ \textbf{then}\ t{:=}\ t.\ \ c?\ \textbf{else}\ t{:=}\ max\ (t{+}1)\ (\mathbb{T}_r + 1).\ \ c?\ \textbf{fi}$

$=\quad \textbf{if}\ \mathbb{T}_r + 1 \le t\ \textbf{then}\ t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ c?\ \textbf{else}\ t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ c?\ \textbf{fi}$

$=\quad W$

Recursive Communication

$$dbl \;=\; c?.\; d!\, 2\times c.\; t{:=}\, t{+}1.\; dbl$$

weakest solution

$$\forall n{:}\, nat\cdot\ \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n} \;\wedge\; \mathbb{T}d_{wd+n} = t{+}n$$

strongest implementable solution

$$(\forall n{:}\, nat\cdot\ \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n} \;\wedge\; \mathbb{T}d_{wd+n} = t{+}n)$$

$$\wedge\;\; rc'{=}wd'{=}t'{=}\infty \;\wedge\; wc'{=}wc \;\wedge\; rd'{=}rd$$

strongest solution

$$\bot$$

$$\forall n{:}\, nat\cdot\ \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n} \;\wedge\; \mathbb{T}d_{wd+n}{=}\, t{+}n \;\;\Leftarrow\;\; dbl$$

$$dbl \;\Leftarrow\; c?.\; d!\, 2\times c.\; t{:=}\, t{+}1.\; dbl$$

Recursive Construction

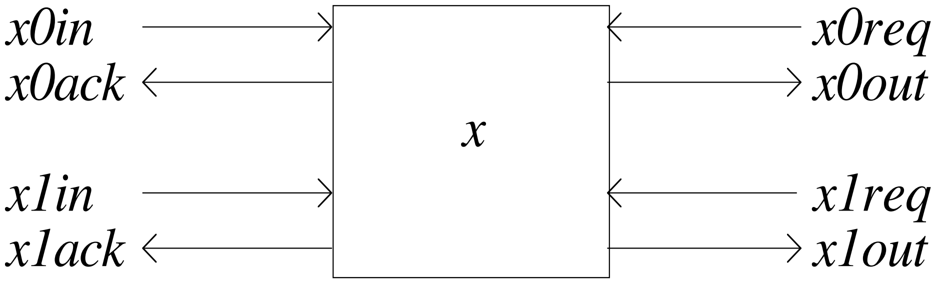
$$dbl_0 \;=\; \top$$

$$\begin{aligned}
dbl_1 \;&=\; c?.\;\; d!\; 2{\times}c.\; t{:=}\; t{+}1.\;\; dbl_0 \\
&=\; rc{:=}\; rc{+}1.\;\;\; \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc-1} \wedge \mathbb{T}d_{wd} = t \;\; \wedge (wd{:=}\; wd{+}1).\;\;\; t{:=}\; t{+}1.\;\;\; \top \\
&=\; \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc} \;\wedge\; \mathbb{T}d_{wd} = t
\end{aligned}$$

$$\begin{aligned}
dbl_2 \;&=\; c?.\;\; d!\; 2{\times}c.\;\; t{:=}\; t{+}1.\;\; dbl_1 \\
&=\; rc{:=}\; rc{+}1.\;\;\; \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc-1} \wedge \mathbb{T}d_{wd} = t \;\; \wedge (wd{:=}\; wd{+}1).\;\;\; t{:=}\; t{+}1. \\
&\;\; \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc} \;\wedge\; \mathbb{T}d_{wd} = t \\
&=\; \mathbb{M}d_{wd} = 2 \times \mathbb{M}c_{rc} \;\wedge\; \mathbb{T}d_{wd} = t \;\wedge\; \mathbb{M}d_{wd+1} = 2{\times}\mathbb{M}c_{rc+1} \wedge\; \mathbb{T}d_{wd+1} = t{+}1
\end{aligned}$$

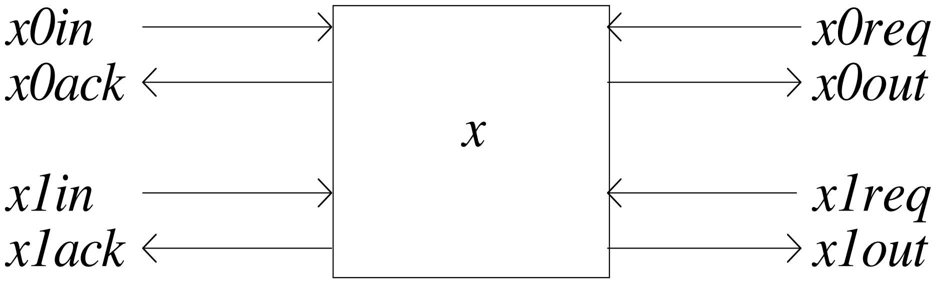
$$dbl_\infty \;=\; \forall n{:}\; nat\cdot\; \mathbb{M}d_{wd+n} = 2 \times \mathbb{M}c_{rc+n} \;\wedge\; \mathbb{T}d_{wd+n} = t{+}n$$

Monitor



$$
\begin{aligned}
monitor \;=\; & (\surd x0in \vee \mathbb{T}x0in_{rx0in} = m) \wedge (x0in?.\; x := x0in.\; x0ack!) \\
\vee\; & (\surd x1in \vee \mathbb{T}x1in_{rx1in} = m) \wedge (x1in?.\; x := x1in.\; x1ack!) \\
\vee\; & (\surd x0req \vee \mathbb{T}x0req_{rx0req} = m) \wedge (x0req?.\; x0out!\, x) \\
\vee\; & (\surd x1req \vee \mathbb{T}x1req_{rx1req} = m) \wedge (x1req?.\; x1out!\, x). \\
& monitor
\end{aligned}
$$

Monitor



$$\begin{aligned}
monitor \;\Leftarrow\;\; & \textbf{if } \surd x0in \textbf{ then } x0in?.\;\; x{:=}\; x0in.\;\; x0ack! \textbf{ else } ok \textbf{ fi}. \\
& \textbf{if } \surd x1in \textbf{ then } x1in?.\;\; x{:=}\; x1in.\;\; x1ack! \textbf{ else } ok \textbf{ fi}. \\
& \textbf{if } \surd x0req \textbf{ then } x0req?.\;\; x0out!\; x \textbf{ else } ok \textbf{ fi}. \\
& \textbf{if } \surd x1req \textbf{ then } x1req?.\;\; x1out!\; x \textbf{ else } ok \textbf{ fi}. \\
& t{:=}\; t{+}1.\;\; monitor
\end{aligned}$$

Communicating Processes

$$c!\,2 \;\|\; (c?.\;\; x{:=}\, c)$$

$$= \quad \mathbb{M}_w = 2 \;\wedge\; (w{:=}\, w{+}1) \,\|\, (r{:=}\, r{+}1.\;\; x{:=}\, \mathbb{M}_{r-1}\,)$$

$$= \quad \mathbb{M}_w = 2 \;\wedge\; w' = w{+}1 \;\wedge\; r' = r{+}1 \;\wedge\; x' = \mathbb{M}_r$$

$$c!\,1.\;\; (c!\,2 \;\|\; (c?.\;\; x{:=}\, c)).\;\; c?$$

channel declaration

$$\textbf{chan } c{:}\; T\cdot\; P$$

$$= \quad \exists \mathbb{M}c{:}\; \infty{*}T\cdot\; \exists \mathbb{T}c{:}\; \infty{*}xnat\cdot\; \textbf{var } rc\;, wc{:}\; xnat := 0\cdot\;\; P$$

ignoring time

chan c: *int*· c! 2 ‖ (c?. x:= c)

= $\exists \mathbb{M}: \infty^* int \cdot \exists \mathbb{T}: \infty^* xnat \cdot$ **var** $r, w: xnat := 0 \cdot$

$x' = \mathbb{M}_r \;\wedge\; \mathbb{M}_w = 2 \;\wedge\; r' = r+1 \;\wedge\; w' = w+1 \;\wedge\;$ (other variables unchanged)

= $\exists \mathbb{M}: \infty^* int \cdot \exists \mathbb{T}: \infty^* xnat \cdot$ **var** $r, w: xnat \cdot$

$x' = \mathbb{M}_0 \;\wedge\; \mathbb{M}_0 = 2 \;\wedge\; r'=1 \;\wedge\; w'=1 \;\wedge\;$ (other variables unchanged)

= $x'=2 \;\wedge\;$ (other variables unchanged)

= x:= 2

including time

chan c: *int*· c! 2 ‖ (t:= *max* t ($\mathbb{T}_r + 1$). c?. x:= c)

= $x'=2 \;\wedge\; t' = t+1 \;\wedge\;$ (other variables unchanged)

Deadlock

chan c: int· t:= $max\ t\ (\mathbb{T}_r + 1)$. c?. c! 5

$=$ $\exists \mathbb{M}: \infty{*}int\cdot\ \exists \mathbb{T}: \infty{*}xnat\cdot$ **var** $r, w: xnat := 0\cdot$

$t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ r{:=}\ r{+}1.\ \ \mathbb{M}_w = 5\ \wedge\ \mathbb{T}_w = t\ \wedge\ (w{:=}\ w{+}1)$

$=$ $\exists \mathbb{M}: \infty{*}int\cdot\ \exists \mathbb{T}: \infty{*}xnat\cdot\ \exists r, r', w, w': xnat\cdot$

$r{:=}\ 0.\ \ w{:=}\ 0.\ \ t{:=}\ max\ t\ (\mathbb{T}_r + 1).\ \ r{:=}\ r{+}1.$

$\mathbb{M}_w = 5\ \wedge\ \mathbb{T}_w = t\ \wedge\ r'{=}r\ \wedge\ w' = w{+}1\ \wedge\ t'{=}t$

$=$ $\exists \mathbb{M}: \infty{*}int\cdot\ \exists \mathbb{T}: \infty{*}xnat\cdot\ \exists r, r', w, w': xnat\cdot$

$\mathbb{M}_0 = 5\ \wedge\ \mathbb{T}_0 = max\ t\ (\mathbb{T}_0 + 1)\ \wedge\ r'{=}1\ \wedge\ w'{=}1\ \wedge\ t' = max\ t\ (\mathbb{T}_0 + 1)$

$=$ $t'{=}\infty$

Deadlock

chan c, d: $int \cdot\ (c?.\ d!\,6)\ \|\ (d?.\ c!\,7)$

chan c, d: $int \cdot\ (t{:=}\ max\ t\ (\mathbb{T}c_{rc}+1).\ c?.\ d!\,6)\ \|\ (t{:=}\ max\ t\ (\mathbb{T}d_{rd}+1).\ d?.\ c!\,7)$

$=\quad \exists \mathbb{M}c, \mathbb{M}d\colon \infty{*}int\cdot\ \exists \mathbb{T}c, \mathbb{T}d\colon \infty{*}xnat\cdot\ \exists rc, rc', wc, wc', rd, rd', wd, wd'\colon xnat\cdot$

$\quad \mathbb{M}d_0 = 6\ \wedge\ \mathbb{M}c_0 = 7\ \wedge\ rc' = wc' = rd' = wd' = 1$

$\wedge\quad \mathbb{T}c_0 = max\ t\ (\mathbb{T}d_0+1)\ \wedge\ \mathbb{T}d_0 = max\ t\ (\mathbb{T}c_0+1)$

$\wedge\quad t' = max\ (max\ t\ (\mathbb{T}d_0+1))\ (max\ t\ (\mathbb{T}c_0+1))$

$=\quad t'{=}\infty$

Power Series Multiplication

Input on channel a : $a_0\ a_1\ a_2 \ldots$ $\qquad A = a_0 + a_1{\times}x + a_2{\times}x^2 + \ldots$

Input on channel b : $b_0\ b_1\ b_2 \ldots$ $\qquad B = b_0 + b_1{\times}x + b_2{\times}x^2 + \ldots$

Output on channel c : $c_0\ c_1\ c_2 \ldots$ $\qquad C = c_0 + c_1{\times}x + c_2{\times}x^2 + \ldots$

$A_1 = a_1 + a_2{\times}x + a_3{\times}x^2 + \ldots$ $\qquad B_1 = b_1 + b_2{\times}x + b_3{\times}x^2 + \ldots$

$A_2 = a_2 + a_3{\times}x + a_4{\times}x^2 + \ldots$ $\qquad B_2 = b_2 + b_3{\times}x + b_4{\times}x^2 + \ldots$

$$C = A \times B = a_0{\times}b_0 + (a_0{\times}b_1 + a_1{\times}b_0){\times}x + (a_0{\times}B_2 + A_1{\times}B_1 + A_2{\times}b_0){\times}x^2$$

$\langle \mathbf{chan}\ c\!: rat \rightarrow C = A{\times}B \rangle\ c \quad \Leftarrow \quad (a? \parallel b?).\ \ c!\ a{\times}b.$

$\qquad \mathbf{var}\ a0\!: rat := a\cdot\ \ \mathbf{var}\ b0\!: rat := b\cdot\ \ \mathbf{chan}\ d\!: rat\cdot$

$\qquad\quad \langle \mathbf{chan}\ c\!: rat \rightarrow C = A{\times}B \rangle\ d$

$\qquad \parallel \quad ((a? \parallel b?).\ \ c!\ a0{\times}b + a{\times}b0.\ \ C = a0{\times}B + D + A{\times}b0)$

$C = a0{\times}B + D + A{\times}b0 \quad \Leftarrow \quad (a? \parallel b? \parallel d?).\ \ c!\ a0{\times}b + d + a{\times}b0.\ \ C = a0{\times}B + D + A{\times}b0$

Review

Binary Theory	laws	proof	
Number Theory	Character Theory		
Bunches	Sets	Strings	Lists
Functions	Quantifiers		
Specification	Refinement	exact precondition	exact postcondition
Program Development	Time Calculation	real time	recursive time
Space Calculation	maximum space	average space	
Scope	variable declaration	frame	
Data Structures	array element assignment		
Control Structures	**while**-loop	loop with **exit**	**for**-loop

Review

Time Dependence	wait		
Assertions	checking	backtracking	
Subprograms	function	procedure	
Probabilistic Programming	random number generator		
Functional Programming		refinement	timing
Recursive Data Definition	construction	induction	
Recursive Program Definition	construction	induction	
Theory Design and Implementation	data theory	program theory	
Data Transformation			
Independent Composition	sequential to parallel transformation		
Interactive Variables	Communication Channels		

Disjoint Composition

Independent composition $P||Q$ requires that P and Q have no variables in common, although each can make use of the initial values of the other's variables by making a private copy. An alternative, let's say disjoint composition, is to allow both P and Q to use all the variables with no restrictions, and then to choose disjoint sets of variables v and w and define

$$P\ |v|w|\ Q \ = \ (P.\ v'{=}v) \wedge (Q.\ w'{=}w)$$

(a) Prove that if P and Q are implementable specifications, then $P\ |v|w|\ Q$ is implementable.

Application Law $\langle v \rightarrow b\rangle\ a \ = \ $(substitute a for v in b)

Let the remaining variables (if any) be x .

Disjoint Composition

$$
\begin{array}{lll}
& P.\ v'=v & \text{expand dependent composition} \\
= & \exists v'', w'', x''\cdot \langle v', w', x' \rightarrow P\rangle\, v''\, w''\, x'' \ \land\ v'=v'' & \text{one-point } v'' \\
= & \exists w'', x''\cdot \langle v', w', x' \rightarrow P\rangle\, v'\, w''\, x'' & \text{rename } w'', x'' \text{ to } w', x' \\
= & \exists w', x'\cdot \langle v', w', x' \rightarrow P\rangle\, v'\, w'\, x' & \text{apply} \\
= & \exists w', x'\cdot P &
\end{array}
$$

$$
\begin{array}{ll}
& Q.\ w'=w \\
= & \exists v', x'\cdot Q
\end{array}
$$

$$P\ |v|w|\ Q \ = \ (P.\ v'=v) \land (Q.\ w'=w) \ = \ (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$$

Disjoint Composition

	$(P \ \|v\|w\| \ Q \text{ is implementable})$	definition of implementable
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot P \ \|v\|w\| \ Q$	use previous result
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \wedge (\exists v', x'\cdot Q)$	identity for x'
$=$	$\forall v, w, x\cdot \exists v', w'\cdot (\exists w', x'\cdot P) \wedge (\exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot \exists v'\cdot \exists w'\cdot (\exists w', x'\cdot P) \wedge (\exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot \exists v'\cdot (\exists w', x'\cdot P) \wedge (\exists w'\cdot \exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot (\exists v'\cdot \exists w', x'\cdot P) \wedge (\exists w'\cdot \exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot (\exists v', w', x'\cdot P) \wedge (\exists v', w', x'\cdot Q)$	splitting law
$=$	$(\forall v, w, x\cdot \exists v', w', x'\cdot P) \wedge (\forall v, w, x\cdot \exists v', w', x'\cdot Q)$	definition of implementable
$=$	$(P \text{ is implementable}) \wedge (Q \text{ is implementable})$	

Disjoint Composition

Independent composition $P||Q$ requires that P and Q have no variables in common, although each can make use of the initial values of the other's variables by making a private copy. An alternative, let's say disjoint composition, is to allow both P and Q to use all the variables with no restrictions, and then to choose disjoint sets of variables v and w and define

$$P\ |v|w|\ Q \;=\; (P.\ v'=v) \wedge (Q.\ w'=w)$$

(b) Describe how $P\ |v|w|\ Q$ can be executed.

Make a copy of all variables. Execute P using the original set of variables and in parallel execute Q using the copies. Then copy back from the copy w to the original w. Then throw away the copies.

Disjoint Composition

Independent composition $P||Q$ requires that P and Q have no variables in common, although each can make use of the initial values of the other's variables by making a private copy. An alternative, let's say disjoint composition, is to allow both P and Q to use all the variables with no restrictions, and then to choose disjoint sets of variables v and w and define

$$P\ |v|w|\ Q \;=\; (P.\ v'=v) \wedge (Q.\ w'=w)$$

(b) Describe how $P\ |v|w|\ Q$ can be executed.

$$P\ |v|w|\ Q \;\Leftarrow\; \textbf{var } cv:=v\cdot\ \textbf{var } cw:=w\cdot\ \textbf{var } cx:=x\cdot$$
$$(P \,\|\, \langle v, w, x, v', w', x' \to Q\rangle\ cv\ cw\ cx\ cv'\ cw'\ cx').\ \ w:=cw$$