Review

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

Review

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

Review

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

Review

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

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	Communication Channels	

Review

Time Dependence	**wait**	
Assertions	**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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	Communication Channels	

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	Communication Channels	

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	Communication Channels	

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	Communication Channels	

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	**sequential to concurrent transformation**	
Interactive Variables	Communication Channels	

Review

Time Dependence	wait	
Assertions	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		
Concurrent Composition	sequential to concurrent transformation	
Interactive Variables	**Communication Channels**	

Disjoint Composition

Concurrent 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

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

Disjoint Composition

Concurrent 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

$$P\ |v|w|\ Q \quad = \quad (P.\ \ v'=v) \land (Q.\ \ w'=w)$$

$$\uparrow \qquad\qquad \uparrow\ \ \uparrow$$

Disjoint Composition

Concurrent 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

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

↑ ↑

Disjoint Composition

Concurrent 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

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

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

Disjoint Composition

Concurrent 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\cdot b\rangle\, a \ = \ $ (substitute a for v in b)

Disjoint Composition

Concurrent 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

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

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

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

Let the remaining variables (if any) be x .

Disjoint Composition

$P. \; v'=v$

Disjoint Composition

$P.\ v'{=}v$ expand sequential composition

$= \quad \exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \land\ v'{=}v''$

Disjoint Composition

	$P.\ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \ \land\ \ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	

Disjoint Composition

	$P.\ \ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \ \land\ \ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'

Disjoint Composition

	$P.\ \ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \ \land\ \ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \underline{\langle v', w', x'\cdot P\rangle}\ v'\ w''\ x''$	rename w'', x'' to w', x'

Disjoint Composition

	$P.\ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \land\ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'
$=$	$\exists w', x'\cdot \langle v', w', x'\cdot P\rangle\ v'\ w'\ x'$	

Disjoint Composition

	$P.\ \ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \ \land\ \ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'
$=$	$\exists w', x'\cdot \langle v', w', x'\cdot P\rangle\ v'\ w'\ x'$	apply
$=$	$\exists w', x'\cdot P$	

Disjoint Composition

	$P.\ v'{=}v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \wedge\ v'{=}v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'
$=$	$\exists w', x'\cdot \langle v', w', x'\cdot P\rangle\ v'\ w'\ x'$	apply
$=$	$\exists w', x'\cdot P$	

	$Q.\ w'{=}w$	
$=$	$\exists v', x'\cdot Q$	

Disjoint Composition

	$P.\ v'=v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \wedge\ v'=v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'
$=$	$\exists w', x'\cdot \langle v', w', x'\cdot P\rangle\ v'\ w'\ x'$	apply
$=$	$\exists w', x'\cdot P$	

	$Q.\ w'=w$	
$=$	$\exists v', x'\cdot Q$	

$P\ |v|w|\ Q$

Disjoint Composition

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

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

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

Disjoint Composition

	$P.\ v'=v$	expand sequential composition
$=$	$\exists v'', w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v''\ w''\ x''\ \land\ v'=v''$	one-point v''
$=$	$\exists w'', x''\cdot \langle v', w', x'\cdot P\rangle\ v'\ w''\ x''$	rename w'', x'' to w', x'
$=$	$\exists w', x'\cdot \langle v', w', x'\cdot P\rangle\ v'\ w'\ x'$	apply
$=$	$\exists w', x'\cdot P$	

	$Q.\ w'=w$	
$=$	$\exists v', x'\cdot Q$	

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

Disjoint Composition

($P \; |v|w| \; Q$ is implementable)

Disjoint Composition

$(P\ |v|w|\ Q$ is implementable$)$ definition of implementable

$= \quad \forall v, w, x\cdot\ \exists v', w', x'\cdot\ P\ |v|w|\ Q$

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

Disjoint Composition

	$(P\ \|v\|w\|\ Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot P\ \|v\|w\|\ Q$	use previous result
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	

$\uparrow$

Disjoint Composition

	$(P\ \|v\|w\|\ Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x\cdot\ \exists v', w', x'\cdot\ P\ \|v\|w\|\ Q$	use previous result
$=$	$\forall v, w, x\cdot\ \exists v', w', x'\cdot\ (\exists w', x'\cdot\ P) \wedge (\exists v', x'\cdot\ Q)$	

Disjoint Composition

	$(P \mid v \mid w \mid Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x \cdot \exists v', w', x' \cdot P \mid v \mid w \mid Q$	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'
$=$	$\forall v, w, x \cdot \exists v', w' \cdot (\exists w', x' \cdot P) \wedge (\exists v', x' \cdot Q)$	

Disjoint Composition

	$(P \;|v|w|\; Q$ 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'
$=$	$\forall v, w, x\cdot\ \exists v', w'\cdot\ (\exists w', x'\cdot\ P) \wedge (\exists v', x'\cdot\ Q)$	

Disjoint Composition

$(P\ |v|w|\ Q$ is implementable$)$ — definition of implementable

$= \quad \forall v, w, x\cdot \exists v', w', x'\cdot P\ |v|w|\ Q$ — use previous result

$= \quad \forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$ — identity for x'

$= \quad \forall v, w, x\cdot \exists v', w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$

$= \quad \forall v, w, x\cdot \exists v'\cdot \exists w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$

Disjoint Composition

	$(P \|v\|w\| Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot P \|v\|w\| Q$	use previous result
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	identity for x'
$=$	$\forall v, w, x\cdot \exists v', w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot \exists v'\cdot \exists w'\cdot \underline{(\exists w', x'\cdot P) \land} (\exists v', x'\cdot Q)$	

Disjoint Composition

$(P\ |v|w|\ Q$ 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)

$= \forall v, w, x\cdot \exists v'\cdot (\exists w', x'\cdot P) \wedge (\exists w'\cdot \exists v', x'\cdot Q)$

Disjoint Composition

	$(P\ \|v\|w\|\ Q$ 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)
$=$	$\forall v, w, x\cdot \exists v'\cdot (\exists w', x'\cdot P)\ \underline{\wedge\ (\exists w'\cdot \exists v', x'\cdot Q)}$	

Disjoint Composition

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

Disjoint Composition

	$(\,P\,|v|w|\,Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot P\,|v|w|\,Q$	use previous result
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	identity for x'
$=$	$\forall v, w, x\cdot \exists v', w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot \exists v'\cdot \exists w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot \exists v'\cdot (\exists w', x'\cdot P) \land (\exists w'\cdot \exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot (\exists v'\cdot \exists w', x'\cdot P) \land (\exists w'\cdot \exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot (\exists v', w', x'\cdot P) \land (\exists v', w', x'\cdot Q)$	

Disjoint Composition

	$(P \,\|v\|w\|\, Q$ 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
$=$	$(\forall v, w, x\cdot \exists v', w', x'\cdot P) \wedge (\forall v, w, x\cdot \exists v', w', x'\cdot Q)$	

Disjoint Composition

	$(P \ \lvert v \rvert w \rvert\ Q$ is implementable$)$	definition of implementable
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot P \ \lvert v \rvert w \rvert\ Q$	use previous result
$=$	$\forall v, w, x\cdot \exists v', w', x'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	identity for x'
$=$	$\forall v, w, x\cdot \exists v', w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot \exists v'\cdot \exists w'\cdot (\exists w', x'\cdot P) \land (\exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot \exists v'\cdot (\exists w', x'\cdot P) \land (\exists w'\cdot \exists v', x'\cdot Q)$	distribution (factoring)
$=$	$\forall v, w, x\cdot (\exists v'\cdot \exists w', x'\cdot P) \land (\exists w'\cdot \exists v', x'\cdot Q)$	
$=$	$\forall v, w, x\cdot (\exists v', w', x'\cdot P) \land (\exists v', w', x'\cdot Q)$	splitting law
$=$	$(\forall v, w, x\cdot \exists v', w', x'\cdot P) \land (\forall v, w, x\cdot \exists v', w', x'\cdot Q)$	definition of implementable
$=$	$(P$ is implementable$) \land (Q$ is implementable$)$	

Disjoint Composition

Concurrent 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

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

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

Disjoint Composition

Concurrent 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

$$P\ |v|w|\ Q \quad = \quad (P.\ v'=v) \land (Q.\ w'=w)$$

(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

Concurrent 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

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

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

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