

## Preconditions

### Analysis

We may wish to show that

- the requirements are consistent: the constraint part of the state schema is satisfiable
- each operation is applied within its domain: the effect of the operation is properly defined whenever it is used

In each case, it is enough to consider preconditions.

### Precondition schemas

A precondition schema is a schema that characterises the combinations of before states and inputs for which the effect of an operation is defined.

State
inputs
...

### Preconditions

The precondition of an operation is that constraint which is necessary and sufficient for the operation to be defined: that is, for an after state to exist.

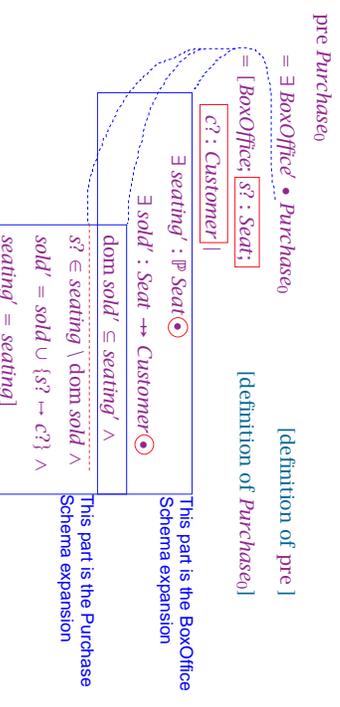
The nature of the after state does not concern us; neither do the outputs of the operation. The precondition will take the form of a constraint upon the combination of the before state and the inputs.

### Notation

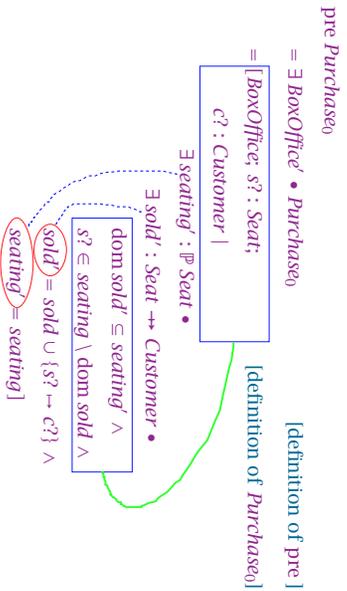
If the schema *Operation* describes an operation upon *State*, with a list of *outputs*, then we write *pre Operation* to denote its precondition.

$$\text{pre Operation} = \exists \text{State}' \bullet \text{Operation} \setminus \text{outputs}$$

### Example



### Example



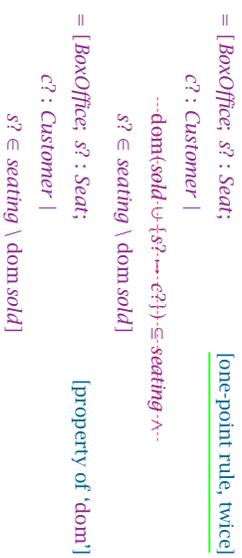
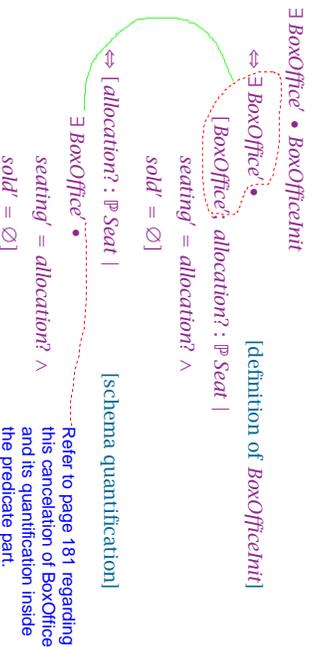
### Initialisation

The operation of initialisation is a special case: there is no before state, although there may be inputs:

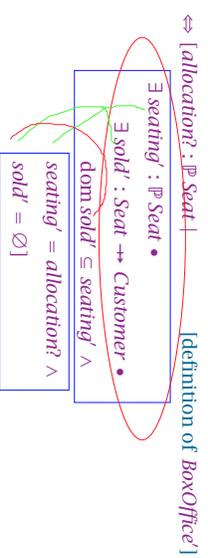
The statement that initialisation is possible is sometimes called the initialisation theorem:

$\exists \text{State}' \bullet \text{StateInit} \setminus \text{outputs}$

What does this last part say? (...see pg 203)



### Example



$\Leftrightarrow [allocation? : \mathbb{P} Seat \mid [one\text{-point rule, twice}]$   
 $allocation? \in \mathbb{P} Seat \wedge$   
 $\emptyset \in Seat \leftrightarrow Customer \wedge$   
 $\emptyset \subseteq allocation?]$   
 $\Leftrightarrow [allocation? : \mathbb{P} Seat] \quad [properties\ of\ sets]$

**Explicit vs implicit preconditions**

There is a minor advantage to be gained by concentrating upon what an operation is supposed to do, and calculating its precondition later.

Even where an explicit precondition has been included, the calculation provides for a degree of cross-checking.

**Example**

$capacity : \mathbb{N}$   
 $capacity > 0$

$CarPark$   
 $count : \mathbb{N}$   
 $count \leq capacity$

$Enter_0$   
 $\Delta CarPark$   
 $count' = count + 1$

$Exit_0$   
 $\Delta CarPark$   
 $count' = count - 1$

$pre\ Exit_0$   
 $= \exists CarPark' \bullet Exit_0$   
 $= [CarPark \mid$   
 $\exists count' : \mathbb{N} \mid$   
 $count \leq capacity \bullet$   
 $count' = count - 1]$   
 $= [CarPark \mid count - 1 \in \mathbb{N}]$   
 $[definition\ of\ Exit_0]$   
 $[definition\ of\ CarPark']$   
 $[one\text{-point rule}]$

**Informed design:**

$ExtraCar$   
 $\exists CarPark$   
 $r! : Report$   
 $count = 0$   
 $r! = extra\_car$

$Exit \hat{=} Exit_0 \vee ExtraCar$

**A recipe for preconditions**

Suppose that we wish to calculate the precondition of

Operation
Declaration
Predicate

**Step One**

Take the various clauses of *Declaration* and assemble them to make three new declarations:

- *Before* introducing only inputs and before components (unprimed state components);
- *After* introducing only outputs and after components (primed state components);
- *Mixed* consisting of the remaining clauses.

**Step Two**

If *Mixed* is not an empty declaration, expand every schema mentioned in *Mixed*: add all input and before components to *Before*; add all output and after components to *After*.

As there may be several levels of schema inclusion, repeat this step until there are no clauses left in *Mixed*.

**Step Three**

The precondition of *Operation* is then

<i>Before</i>
$\exists$ <i>After</i>
<i>Predicate</i>

**Question**

Given the following schema definitions,

S
$a : \mathbb{N}$
$b : \mathbb{N}$
$a \neq b$

T
S
$c : \mathbb{N}$
$b \neq c$

what is the precondition of the following operation?

<i>Increment</i>
$\Delta T$
$in? : \mathbb{N}$
$out! : \mathbb{N}$
$a' = a + in?$
$b' = b$
$c' = c$
$out! = c$

**Simplification**

Suppose that we wished to simplify the precondition schema

<i>Before</i>
$\exists$ <i>After</i> •
<i>Predicate</i>

**Step Four**

Expand any schemas in *After* that contain equations identifying outputs or after components.

**Step Five**

Expand any schemas in *After* that refer to outputs or after components for which we already have equations.

**Step Six**

If *Predicate* contains an equation identifying a component declared in *After*, then use the one-point rule to eliminate that component.

Repeat this step as many times as possible.

**Step Seven**

If *After*<sub>1</sub> and *Predicate*<sub>1</sub> are what remains of *After* and *Predicate*, then the precondition is now

<i>Before</i>
$\exists$ <i>After</i> <sub>1</sub> •
<i>Predicate</i> <sub>1</sub>

**Question**

How may we simplify the predicate part of *pre Increment*?

$\exists$  *out*! :  $\mathbb{N}$ ;  $T'$  •

$a' = a + in?$   $\wedge$

$b' = b$   $\wedge$

$c' = c$   $\wedge$

*out*! = *c*

**Disjunction**

If

$$Op \hat{=} Op_1 \vee Op_2$$

then

$$\text{pre } Op = \text{pre } Op_1 \vee \text{pre } Op_2$$

Existential quantification distributes through disjunction

**Example**

$$\text{pre } (\text{Purchase}_0 \wedge \text{Success}) = \text{pre } \text{Purchase}_0$$

Leave for students to review later

**A useful result**pre  $GOp$ 

$$\Leftrightarrow \exists \text{Global}' \bullet \quad \text{[definition of 'pre']}$$

 $GOp$ 

$$\Leftrightarrow \exists \text{Global}' \bullet \quad \text{[definition of } GOp]$$

 $\exists \Delta \text{Local} \bullet \text{Promote} \wedge LOP$ 

$$\Leftrightarrow \exists \Delta \text{Local} \bullet \quad \text{[property of } \exists]$$

 $\exists \text{Global}' \bullet \text{Promote} \wedge LOP$ **Conjunction**

In general, if

$$Op \hat{=} Op_1 \wedge Op_2$$

then

$$\text{pre } Op \neq \text{pre } Op_1 \wedge \text{pre } Op_2$$

However, it may be that the declarations introduce disjoint sets of variables...

**Skip****Free promotion**

$$\exists \text{Local}' \bullet \quad \Leftrightarrow \forall \text{Local}' \bullet$$

$$\exists \text{Global}' \bullet \text{Promote} \quad \exists \text{Global}' \bullet \text{Promote}$$

$$\Leftrightarrow \exists \text{Local} \bullet \quad \text{[definition of } \Delta]$$

$$\exists \text{Local}' \bullet \exists \text{Global}' \bullet \text{Promote} \wedge LOP$$

$$\Leftrightarrow \exists \text{Local} \bullet \quad \text{[lemma]}$$

$$(\exists \text{Local}' \bullet \exists \text{Global}' \bullet \text{Promote}) \wedge (\exists \text{Local}' \bullet LOP)$$

$$\Leftrightarrow \exists \text{Local} \bullet \quad \text{[definition of 'pre', twice]}$$

$$\text{pre } \text{Promote} \wedge \text{pre } LOP$$

Skip

**Lemma**

The equivalence labelled 'lemma' is easily proved in the forward direction. A proof in the other direction ( $\Leftarrow$ ) requires the free promotion property.

We abbreviate *Local*, *Global*, and *Promote* to  $L$ ,  $G$ , and  $P$ , respectively.

$$\frac{\frac{\exists L' \bullet \exists G' \bullet P}{\forall L' \bullet \exists G' \bullet P} \text{ [free promotion]}}{\frac{[\theta L' \in L]^{[1]}}{\exists G' \bullet P} \text{ [V-elim]}}{\frac{[\theta L' \in L]^{[1]}}{\exists G' \bullet P \wedge LOP} \text{ [IOP]^{[1]}}} \text{ [G' not free in LOP]}} \text{ [I-intro]}$$

$$\frac{\frac{[\theta L' \in L]^{[1]}}{\exists L' \bullet LOP} \text{ [I-intro]}}{\exists L' \bullet \exists G' \bullet P \wedge LOP} \text{ [I-intro]}$$

$$\frac{\exists L' \bullet LOP}{\exists L' \bullet \exists G' \bullet P \wedge LOP} \text{ [I-elim]^{[1]}}$$

**Example**

$$\text{AssignIndex} \hat{=} \exists \Delta \text{Data} \bullet \text{AssignData} \wedge \text{Promote}$$

$$\text{pre AssignIndex} =$$

$$\exists \text{Data} \bullet \text{pre AssignData} \wedge \text{pre Promote}$$
**Question**

What is the precondition of *AssignData*?

<i>AssignData</i>
$\Delta \text{Data}$
$\text{new?} : \text{Value}$
$\text{value}' = \text{new?}$

**Question**

What is the precondition of *Promote*?

<i>Promote</i>
$\Delta \text{Array}$
$\Delta \text{Data}$
$\text{index?} : \mathbb{N}$
$\text{index?} \in \text{dom array}'$
$\{\text{index?}\} \triangleleft \text{array}' = \{\text{index?}\} \triangleleft \text{array}'$
$\text{array}' \text{index?} = \theta \text{Data}$
$\text{array}' \text{index?} = \theta \text{Data}'$

**Question**

What is the precondition of

$$\exists \Delta \text{Data} \bullet \text{Promote} \wedge \text{AssignData} \quad ?$$

## Results

It is often useful to tabulate the results of our analysis; against each operation, we record the predicate that characterises its precondition.

We should check that the predicates are a correct reflection of our expectations: that each operation schema is exactly as prescriptive as it should be.

## Example

Operation schema	Precondition
<i>InitBoxOffice</i>	<i>true</i>
<i>Purchase<sub>0</sub></i>	$s? \in \text{seating} \setminus \text{dom sold}$
<i>NotAvailable</i>	$s? \notin \text{seating} \setminus \text{dom sold}$
<i>Purchase</i>	<i>true</i>
<i>Return<sub>0</sub></i>	$s? \mapsto c? \in \text{sold}$
<i>NotPossible</i>	$s? \mapsto c? \notin \text{sold}$
<i>Return</i>	<i>true</i>

## Summary

- preconditions
- pre *Schema*
- initialisation
- calculation and simplification
- disjunction
- promotion