# 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.



## 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.



## 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* |
| ... |



## Notation

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

$$\mathrm{pre}\ \mathit{Operation} = \exists\, \mathit{State}' \bullet \mathit{Operation} \setminus \mathit{outputs}$$



## Example

$\mathrm{pre}\ \mathit{Purchase}_0$

$= \exists\, \mathit{BoxOffice}' \bullet \mathit{Purchase}_0$ [definition of pre ]

$= [\mathit{BoxOffice}; \boxed{s? : \mathit{Seat};}$ [definition of $\mathit{Purchase}_0$]

$\boxed{c? : \mathit{Customer}} \mid$

$\exists\, \mathit{seating}' : \mathbb{P}\, \mathit{Seat} \bullet$ This part is the BoxOffice Schema expansion

$\exists\, \mathit{sold}' : \mathit{Seat} \twoheadrightarrow \mathit{Customer} \bullet$

$\mathrm{dom}\, \mathit{sold}' \subseteq \mathit{seating}' \land$

$s? \in \mathit{seating} \setminus \mathrm{dom}\, \mathit{sold} \land$ This part is the Purchase Schema expansion

$\mathit{sold}' = \mathit{sold} \cup \{s? \mapsto c?\} \land$

$\mathit{seating}' = \mathit{seating}]$

## Example

pre $Purchase_0$

$= \exists\, BoxOffice' \bullet Purchase_0$ [definition of pre]

$= [BoxOffice;\ s? : Seat;$ [definition of $Purchase_0$]

$\quad c? : Customer \mid$

$\quad\quad \exists\, seating' : \mathbb{P}\, Seat \bullet$

$\quad\quad\quad \exists\, sold' : Seat \rightarrowtail Customer \bullet$

$\quad\quad\quad\quad \text{dom}\, sold' \subseteq seating' \wedge$

$\quad\quad\quad\quad s? \in seating \setminus \text{dom}\, sold \wedge$

$\quad\quad\quad\quad sold' = sold \cup \{s? \mapsto c?\} \wedge$

$\quad\quad\quad\quad seating' = seating]$



$= [BoxOffice;\ s? : Seat;$ [one-point rule, twice]

$\quad c? : Customer \mid$

$\quad\quad \cancel{\text{dom}(sold \cup \{s? \mapsto c?\}) \subseteq seating \wedge}$

$\quad\quad s? \in seating \setminus \text{dom}\, sold]$

$= [BoxOffice;\ s? : Seat;$ [property of 'dom']

$\quad c? : Customer \mid$

$\quad\quad s? \in seating \setminus \text{dom}\, sold]$



## 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:

$\exists\, State' \bullet StateInit \setminus outputs$

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



## Example

*BoxOfficeInit*

$BoxOffice'$

$allocation? : \mathbb{P}\, Seat$

---

$seating' = allocation?$

$sold' = \varnothing$



$\exists\, BoxOffice' \bullet BoxOfficeInit$

$\Leftrightarrow \exists\, BoxOffice' \bullet$ [definition of $BoxOfficeInit$]

$\quad [BoxOffice';\ allocation? : \mathbb{P}\, Seat \mid$

$\quad\quad seating' = allocation? \wedge$

$\quad\quad sold' = \varnothing]$

$\Leftrightarrow [allocation? : \mathbb{P}\, Seat \mid$ [schema quantification]

$\quad \exists\, BoxOffice' \bullet$

$\quad\quad seating' = allocation? \wedge$

$\quad\quad sold' = \varnothing]$

Refer to page 181 regarding this cancelation of BoxOffice and its quantification inside the predicate part.



$\Leftrightarrow [allocation? : \mathbb{P}\, Seat \mid$ [definition of $BoxOffice'$]

$\quad \exists\, seating' : \mathbb{P}\, Seat \bullet$

$\quad\quad \exists\, sold' : Seat \rightarrowtail Customer \bullet$

$\quad\quad\quad \text{dom}\, sold' \subseteq seating' \wedge$

$\quad\quad\quad seating' = allocation? \wedge$

$\quad\quad\quad sold' = \varnothing]$

$$
\begin{array}{ll}
\Leftrightarrow [\, allocation? : \mathbb{P}\, Seat \mid & \text{[one-point rule, twice]} \\
\quad allocation? \in \mathbb{P}\, Seat \land & \\
\quad \varnothing \in Seat \twoheadrightarrow Customer \land & \\
\quad \varnothing \subseteq allocation? \,] & \\
\Leftrightarrow [\, allocation? : \mathbb{P}\, Seat \,] & \text{[properties of sets]}
\end{array}
$$



## 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**

$count : \mathbb{N}$

---

$count \leq capacity$



**$Enter_0$**

$\Delta CarPark$

---

$count' = count + 1$

**$Exit_0$**

$\Delta CarPark$

---

$count' = count - 1$



$$
\begin{array}{ll}
\text{pre } Exit_0 & \\
\quad = \exists\, CarPark' \bullet Exit_0 & \text{[definition of } Exit_0\text{]} \\
\quad = [\, CarPark \mid & \text{[definition of } CarPark'\text{]} \\
\qquad \exists\, count' : \mathbb{N} \mid & \\
\qquad\quad count' \leq capacity \bullet & \\
\qquad\quad count' = count - 1 \,] & \\
\quad = [\, CarPark \mid count - 1 \in \mathbb{N} \,] & \text{[one-point rule]}
\end{array}
$$



Informed design:

**ExtraCar**

$\Xi CarPark$

$r! : Report$

---

$count = 0$

$r! = extra\_car$

$Exit \mathrel{\widehat{=}} Exit_0 \lor ExtraCar$

## A recipe for preconditions

Suppose that we wish to calculate the precondition of

$$\begin{array}{|l}\hline \textit{Operation} \\ \textit{Declaration} \\ \hline \textit{Predicate} \\ \hline\end{array}$$



## 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

$$\begin{array}{|l}\hline \textit{Before} \\ \hline \exists\, \textit{After} \bullet \\ \quad \textit{Predicate} \\ \hline\end{array}$$



## Question

Given the following schema definitions,

$$\begin{array}{|l}\hline S \\ a : \mathbb{N} \\ b : \mathbb{N} \\ \hline a \neq b \\ \hline\end{array} \qquad \begin{array}{|l}\hline T \\ S \\ c : \mathbb{N} \\ \hline b \neq c \\ \hline\end{array}$$



what is the precondition of the following operation?

$$\begin{array}{|l}\hline \textit{Increment} \\ \Delta T \\ in? : \mathbb{N} \\ out! : \mathbb{N} \\ \hline a' = a + in? \\ b' = b \\ c' = c \\ out! = c \\ \hline\end{array}$$

## Simplification

Suppose that we wished to simplify the precondition schema

$$\begin{array}{|l}\hline \textit{Before} \\ \hline \exists\, \textit{After} \bullet \\ \quad \textit{Predicate} \\ \hline \end{array}$$



## 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 $\textit{After}_1$ and $\textit{Predicate}_1$ are what remains of *After* and *Predicate*, then the precondition is now

$$\begin{array}{|l}\hline \textit{Before} \\ \hline \exists\, \textit{After}_1 \bullet \\ \quad \textit{Predicate}_1 \\ \hline \end{array}$$



## Question

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

$$\begin{array}{l} \exists\, out! : \mathbb{N};\ T' \bullet \\ \quad a' = a + in? \land \\ \quad b' = b \land \\ \quad c' = c \land \\ \quad out! = c \end{array}$$

## Disjunction

If

$$Op \mathrel{\widehat{=}} Op_1 \lor Op_2$$

then

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

Existential quantification distributes through disjunction



## Conjunction

In general, if

$$Op \mathrel{\widehat{=}} Op_1 \land Op_2$$

then

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

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



## Example

$$\begin{array}{|l}\textit{Success} \\ \hline r! : \textit{Response} \\ \hline r! = \textit{okay} \\ \hline \end{array}$$

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



Skip

## Free promotion

$$\exists\, \textit{Local}' \bullet \exists\, \textit{Global}' \bullet \textit{Promote} \quad \Leftrightarrow \quad \forall\, \textit{Local}' \bullet \exists\, \textit{Global}' \bullet \textit{Promote}$$



Leave for students to review later

## A useful result

$$\begin{array}{ll}
\text{pre } \textit{GOp} & \\
\Leftrightarrow \exists\, \textit{Global}' \bullet \textit{GOp} & [\text{definition of `pre'}] \\
\Leftrightarrow \exists\, \textit{Global}' \bullet \exists\, \Delta \textit{Local} \bullet \textit{Promote} \land \textit{LOp} & [\text{definition of } \textit{GOp}] \\
\Leftrightarrow \exists\, \Delta \textit{Local} \bullet \exists\, \textit{Global}' \bullet \textit{Promote} \land \textit{LOp} & [\text{property of } \exists]
\end{array}$$



$$\begin{array}{ll}
\Leftrightarrow \exists\, \textit{Local} \bullet \exists\, \textit{Local}' \bullet \exists\, \textit{Global}' \bullet \textit{Promote} \land \textit{LOp} & [\text{definition of } \Delta] \\
\Leftrightarrow \exists\, \textit{Local} \bullet (\exists\, \textit{Local}' \bullet \exists\, \textit{Global}' \bullet \textit{Promote}) \land (\exists\, \textit{Local}' \bullet \textit{LOp}) & [\text{lemma}] \\
\Leftrightarrow \exists\, \textit{Local} \bullet \text{pre } \textit{Promote} \land \text{pre } \textit{LOp} & [\text{definition of `pre', twice}]
\end{array}$$

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.



$$\frac{\exists L' \bullet \exists G' \bullet P}{\forall L' \bullet \exists G' \bullet P}\ [\text{free promotion}]$$

$$\frac{\dfrac{[\theta L' \in L]^{[1]}}{\exists G' \bullet P}\ [\forall\text{−elim}] \qquad [LOp]^{[1]}}{\exists G' \bullet P \land LOp}\ [G' \text{ not free in } LOp]$$

$$\frac{[\theta L' \in L]^{[1]}}{\exists L' \bullet \exists G' \bullet P \land LOp}\ [\exists\text{−intro}]$$

$$\frac{\exists L' \bullet LOp}{\exists L' \bullet \exists G' \bullet P \land LOp}\ [\exists\text{−elim}^{[1]}]$$



## Example

$$AssignIndex \mathrel{\widehat=} \exists \Delta Data \bullet AssignData \land Promote$$

$$\text{pre } AssignIndex =$$
$$\quad \exists Data \bullet \text{pre } AssignData \land \text{pre } Promote$$



## Question

What is the precondition of *AssignData*?

*AssignData*
$\Delta Data$
$new? : Value$
---
$value' = new?$



## Question

What is the precondition of *Promote*?

*Promote*
$\Delta Array$
$\Delta Data$
$index? : \mathbb{N}$
---
$index? \in \operatorname{dom} array$
$\{index?\} \mathbin{\lhd} array = \{index?\} \mathbin{\lhd} array'$
$array\ index? = \theta Data$
$array'\ index? = \theta Data'$



## Question

What is the precondition of

$$\exists \Delta Data \bullet Promote \land AssignData \qquad ?$$

## 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 |
|---|---|
| $InitBoxOffice$ | $true$ |
| $Purchase_0$ | $s? \in seating \setminus \operatorname{dom} sold$ |
| $NotAvailable$ | $s? \notin seating \setminus \operatorname{dom} sold$ |
| $Purchase$ | $true$ |
| $Return_0$ | $s? \mapsto c? \in sold$ |
| $NotPossible$ | $s? \mapsto c? \notin sold$ |
| $Return$ | $true$ |



## Summary

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