) CHAPTER TWO

) Basic Transformation

roduction

This chapter uses the proof rules developed in the previous chapter to derive a set of basic transformations of the extended language. These include transformations for adding and removing assertions, which can be used to verify properties of programs, together with simple manipulations and simplifications. These basic transformations will be used extensively in later chapters in the course of deriving the more complex (and less intuitively obvious) transformations. The ultimate aim is to develop a "toolkit" of transformation rules and techniques which will enable the derivation of algorithms from specifications and the analysis of programs in order to derive their specification, or to verify thatCa program meets its specification. This means that, for practical purposes, there will be no need to work directly with the weakest precondition formulae.

ASSERTIONS

Assertions give information about the context in which they occur and thus make it easier to give a correct refinement for a component of a program. In this section we will introduce some rules to enable us to introduce assertions into programs and thus "migrate" information about the program to its various components. They will also enable us to establish "global invariants" - assertions which are preserved throughout the execution of the program.

The transformations for introducing and removing assertions can be used to prove all of the results from Hoare's axiomatic basis for programming. The equivalent of proving **{P};S;{Q}** (which means "if **P** holds before the execution of **S** and **S** terminates then **Q** will hold on termination") would be to prove **{P};S** $\leqslant$ **{P};S;{Q}** in our system. We can do much more that this with our system, we can prove that two programs are equivalent, prove that a program is guaranteed to terminate as well as proving that a program implements its specification. The first set of examples are taken from [Back 80]:

Lemma: **Induction Rule for Loops:**
Let Δ be a countable set of sentences for **L**.
If $\Delta \vdash$**{P};do** $\mathbf{B}_1 \rightarrow \mathbf{S}_1 \, \square \, \ldots \, \square \, \mathbf{B}_m \rightarrow \mathbf{S}_m$ **od**$^n \leqslant$ **S** for $\mathbf{n} < \omega$
then $\Delta \vdash$**{P};do** $\mathbf{B}_1 \rightarrow \mathbf{S}_1 \, \square \, \ldots \, \square \, \mathbf{B}_m \rightarrow \mathbf{S}_m$ **od** $\leqslant$ **S**
Proof: From the inference rule for recursion and the inference rule for infinite disjunction.

Example 1 Assertion Weakening:
If $\Delta \vdash \mathbf{P} \Rightarrow \mathbf{P'}$ then $\Delta \vdash \{\mathbf{P}\} \leqslant \{\mathbf{P'}\}$ follows by computing the weakest preconditions.
$\Delta \vdash \mathbf{P} \Rightarrow \mathbf{true}$ so we have $\Delta \vdash \{\mathbf{P}\} \leqslant \mathbf{skip}$, since $\{\mathbf{true}\} = \mathbf{skip}$ so we can always remove an assertion.

Example 2 Inserting Assertions:
If $\Delta \vdash \mathbf{P} \Rightarrow \mathbf{WP(S,Q)}$ then $\Delta \vdash \{\mathbf{P}\};\mathbf{S} \leqslant \{\mathbf{P}\};\mathbf{S};\{\mathbf{Q}\}$.
$\Delta \vdash \mathbf{x{:=}t} \;\approx\; \mathbf{x{:=}t};\ \{\mathbf{x{=}t}\}$
These follow directly from the weakest preconditions.

Example 3:
$\Delta \vdash \{\mathbf{P}\};\underline{\mathbf{if}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{fi}} \;\approx\; \{\mathbf{P}\};\underline{\mathbf{if}}\ \mathbf{B}_1 \rightarrow \{\mathbf{P} \wedge \mathbf{B}_1\};\mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \{\mathbf{P} \wedge \mathbf{B}_n\};\mathbf{S}_n\ \underline{\mathbf{fi}}$
and $\Delta \vdash \underline{\mathbf{if}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{fi}};\{\mathbf{Q}\} \;\approx\; \underline{\mathbf{if}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1;\{\mathbf{Q}\} \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n;\{\mathbf{Q}\}\ \underline{\mathbf{fi}}$
These can be used to carry an invariant through an **if** statement.
Cor: $\Delta \vdash \{\mathbf{P}\};\ \underline{\mathbf{if}}\ \mathbf{B}\ \underline{\mathbf{then}}\ \mathbf{S}_1\ \underline{\mathbf{else}}\ \mathbf{S}_2\ \underline{\mathbf{fi}} \;\approx\; \{\mathbf{P}\};\ \underline{\mathbf{if}}\ \mathbf{B}\ \underline{\mathbf{then}}\ \{\mathbf{B} \wedge \mathbf{P}\};\ \mathbf{S}_1\ \underline{\mathbf{else}}\ \{\neg\mathbf{B} \wedge \mathbf{P}\};\ \mathbf{S}_2\ \underline{\mathbf{fi}}$

Example 4:
$\Delta \vdash \underline{\mathbf{while}}\ \mathbf{B}\ \underline{\mathbf{do}}\ \mathbf{S}\ \underline{\mathbf{od}} \;\approx\; \underline{\mathbf{while}}\ \mathbf{B}\ \underline{\mathbf{do}}\ \mathbf{S}\ \underline{\mathbf{od}};\ \{\neg\mathbf{B}\}$
Proof: Prove for the **n**th truncation by induction on **n** and use the induction rule for iteration.

Example 5:
If $\Delta \vdash \{\mathbf{P} \wedge \mathbf{B}_i\};\mathbf{S}_i \;\approx\; \{\mathbf{P} \wedge \mathbf{B}_i\};\mathbf{S}_i;\{\mathbf{P}\}$ then:
$\Delta \vdash \{\mathbf{P}\};\underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{od}} \;\approx\; \{\mathbf{P}\};\underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \{\mathbf{P} \wedge \mathbf{B}_1\};\mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \{\mathbf{P} \wedge \mathbf{B}_n\};\mathbf{S}_n\ \underline{\mathbf{od}}$
Proof: Prove for each case $\underline{\mathbf{do}}\text{-}\underline{\mathbf{od}}^m$ by induction and use the induction rule for loops.

Example 6: If $\underline{\mathbf{x}} \cap \mathbf{var(P)} = \varnothing$ then
$\Delta \vdash \{\mathbf{P}\};\underline{\mathbf{begin}}\ \mathbf{x{:}S}\ \underline{\mathbf{end}};\{\mathbf{Q}\} \;\approx\; \{\mathbf{P}\};\underline{\mathbf{begin}}\ \mathbf{x}{:}\{\mathbf{P}\};\mathbf{S};\{\mathbf{Q}\}\ \underline{\mathbf{end}};\{\mathbf{Q}\}$

Example 7:
$\Delta \vdash \underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{od}} \;\approx\; \underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{od}};\ \{\neg\mathbf{B}_1 \wedge \neg\mathbf{B}_2 \wedge \ldots \wedge \neg\mathbf{B}_n\}$
Proof: Follows from Example 5.

Example 8:
If $\{\mathbf{P} \wedge \mathbf{B}_i\};\mathbf{S}_i \;\approx\; \{\mathbf{P} \wedge \mathbf{B}_i\};\mathbf{S}_i;\{\mathbf{P}\}$ then:
$\Delta \vdash \{\mathbf{P}\};\underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{od}} \;\approx\; \{\mathbf{P}\};\underline{\mathbf{do}}\ \mathbf{B}_1 \rightarrow \mathbf{S}_1 \,\square \ldots \square\, \mathbf{B}_n \rightarrow \mathbf{S}_n\ \underline{\mathbf{od}};\{\mathbf{P}\}$
Proof: By example 5 and Example 3.

Example 9:
From the Deduction Theorem we can prove:

$$\Delta \cup \{\mathbf{P}\} \vdash \mathbf{S}_1 \leqslant \mathbf{S}_2 \text{ iff } \Delta \vdash \mathbf{\{P\};S}_1 \leqslant \mathbf{S}_2$$

since $\Delta \cup \{\mathbf{P}\} \vdash \mathbf{WP(S_1,G(w))} \Rightarrow \mathbf{WP(S_2,G(w))}$
iff $\Delta \vdash \mathbf{P} \Rightarrow \big(\mathbf{WP(S_1,G(w))} \Rightarrow \mathbf{WP(S_2,G(w))}\big)$ (by the Deduction Theorem)
iff $\Delta \vdash \big(\mathbf{P} \ \wedge \ \mathbf{WP(S_1,G(w))}\big) \Rightarrow \mathbf{WP(S_2,G(w))}$
iff $\Delta \vdash \mathbf{WP(\{P\};S_1,G(w))} \Rightarrow \mathbf{WP(S_2,G(w))}$
iff $\Delta \vdash \mathbf{\{P\};S_1} \leqslant \mathbf{S_2}$.
Hence as a corollary:
$\Delta \cup \{\mathbf{P}\} \vdash \mathbf{S}_1 \ \approx \ \mathbf{S}_2$ iff $\Delta \vdash \mathbf{\{P\};S}_1 \ \approx \ \mathbf{\{P\};S}_2$.

Abstract Data Types

Often in the development of a program it is important to be able to change the data representation used, an obvious example is where an abstract data type has been used which needs to be represented by data types which have already been implemented. For example we may use a "stack" variable in the higher-level representations of a program which we wish to implement using an array or a linked list. There are also many cases where the right choice of date representation can make a program much more efficient, we will give some examples later. The general technique involves the following stages:

(i) Add "ghost variables" to the program (these will become the concrete variables which will replace the abstract variables). These variables are assigned to at each point where the abstract variables are assigned, but as yet their values are not tested. The assignments to the ghost variables are made in such a way that the relationship between the abstract and concrete variables is maintained.

(ii) Add assertions before each assignment to the abstract variables which describe the relationship between the abstract and concrete variables.

(iii) Replace all references to the abstract variables by references to the concrete variables. Note that this includes the references to the abstract variables which occur in the assignments to concrete variables.

(iv) Now the abstract variables are assigned to but never tested; they have become ghost variables. So remove the abstract variables to give a program expressed entirely in terms of the concrete variables.

We take a different approach than that of Back in [Back 80]. His approach requires that each individual abstract assignment is replaced by a set of statements involving only the concrete variables. We believe that our approach allows more flexibility in the way data types are represented (see chapters 8 and 9 for examples of our technique in action).

SIMPLIFICATIONS

The following basic transformations are used extensively in the proofs of the more complex transformation. They are also used in restructuring a program, and in putting a program in the right form for applying more complex transformations.

Prune Conditional:

$\Delta \vdash \mathbf{\{B\};\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}} \approx \mathbf{\{B\};S_1}$

$\Delta \vdash \mathbf{\{\neg B\};\underline{if}\ B\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}} \approx \mathbf{\{\neg B\};S_2}$

$\Delta \vdash \mathbf{\underline{if}\ B\ \underline{then}\ S\ \underline{else}\ S\ \underline{fi}} \approx \mathbf{S}$ (also called "Splitting a Tautology")

Proof: The proofs follow directly from the weakest precondition for **if** eg:

$\mathbf{WP(\underline{if}\ B\ \underline{then}\ S\ \underline{else}\ S\ \underline{fi},\ G(w))} \iff \big(\mathbf{B \Rightarrow WP(S,G(w))}\big) \wedge \big(\mathbf{\neg B \Rightarrow WP(S,G(w))}\big)$
$\iff \mathbf{WP(S,G(w))}$.

Since **skip={true}** we also have:

$\Delta \vdash \mathbf{\underline{if}\ true\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}} \approx \mathbf{S_1}$

$\Delta \vdash \mathbf{\underline{if}\ false\ \underline{then}\ S_1\ \underline{else}\ S_2\ \underline{fi}} \approx \mathbf{S_2}$

A generalisation of this is:

$\Delta \vdash \mathbf{\{\neg B_n\};\ \underline{if}\ B_1 \to S_1 \Box \ldots \Box B_n \to S_n\ \underline{fi}} \approx \mathbf{\{\neg B_n\};\ \underline{if}\ B_1 \to S_1 \Box \ldots \Box B_{n-1} \to S_{n-1}\ \underline{fi}}$

where we use the convention **if fi** $\approx$ **abort**.

Proof: $\mathbf{WP(\{\neg B_n\};\ \underline{if}\ B_1 \to S_1 \Box \ldots \Box B_n \to S_n\ \underline{fi},\ G(w))}$

$\iff \mathbf{\neg B_n} \wedge \big(\mathbf{B_1 \vee \ldots \vee B_n}\big) \wedge \big(\mathbf{B_1 \Rightarrow WP(S_1,G(w))}\big) \wedge \ldots \wedge \big(\mathbf{B_n \Rightarrow WP(S_n,G(w))}\big)$

$\iff \mathbf{\neg B_n} \wedge \big(\mathbf{B_1 \vee \ldots \vee B_{n-1}}\big) \wedge \big(\mathbf{B_1 \Rightarrow WP(S_1,G(w))}\big) \wedge \ldots \wedge \big(\mathbf{\neg B_n \vee WP(S_n,G(w))}\big)$

$\iff \mathbf{\neg B_n} \wedge \big(\mathbf{B_1 \vee \ldots \vee B_{n-1}}\big) \wedge \big(\mathbf{B_1 \Rightarrow WP(S_1,G(w))}\big) \wedge \ldots \wedge \big(\mathbf{B_{n-1} \Rightarrow WP(S_{n-1},G(w))}\big)$

since $\mathbf{a} \wedge \big(\mathbf{a \vee b}\big) \iff \mathbf{a}$

$\iff \mathbf{WP(\{\neg B_n\};\ \underline{if}\ B_1 \to S_1 \Box \ldots \Box B_{n-1} \to S_{n-1}\ \underline{fi},\ G(w))}$.

Lemma: Proof by case analysis:

We have the result

$\Delta \vdash \mathbf{\{P\};S} \approx \mathbf{\{P\};\ \underline{if}\ Q\ \underline{then}\ \{P \wedge Q\};S\ \underline{else}\ \{P \wedge \neg Q\};S\ \underline{fi}}$

so if we can prove:

$\mathbf{\{P,Q\}} \cup \Delta \vdash \mathbf{S} \approx \mathbf{S'}$ and $\mathbf{\{P,\neg Q\}} \cup \Delta \vdash \mathbf{S} \approx \mathbf{S'}$ then we can prove

$\Delta \vdash \mathbf{\{P\};S} \approx \mathbf{\{P\};\ \underline{if}\ Q\ \underline{then}\ S'\ \underline{else}\ S'\ \underline{fi}}$ (by above and replacement) $\approx \mathbf{\{P\};S'}$.

This easily generalises to division into more cases.

Reorder Conditional:

$\Delta \vdash$ **if B then S_1 else S_2 fi** $\approx$ **if $\neg$B then S_2 else S_1 fi**

Proof: Follows from the weakest precondition for **if**.

A corrollary is: $\Delta \vdash$ **if B then skip else S fi** $\approx$ **if $\neg$B then S fi**

Split Conditional:

$\Delta \vdash$ **if $B_1 \vee B_2$ then S_1 else S_2 fi** $\approx$ **if B_1 then S_1 else if B_2 then S_1 else S_2 fi fi**

$\Delta \vdash$ **if $B_1 \wedge B_2$ then S_1 else S_2 fi** $\approx$ **if B_1 then if B_2 then S_1 else S_2 fi else S_2 fi**

Proof: By case analysis on $\mathbf{B}_1$ and $\mathbf{B}_2$ and pruning the conditional.

Assignment Elimination/Insertion:

For any variable **x** and term **t**:

$\Delta \vdash$ **{x=t};x:=t** $\approx$ **{x=t}**

Proof: WP({x=t};x:=t, R)

$\iff$ **x=t** $\wedge$ **WP(x:=t,R)** $\iff$ **x=t** $\wedge$ **R[t/x]**

$\iff$ **x=t** $\wedge$ **R** (by an Axiom of equality) $\iff$ **WP({x=t}, R)**

Assignment Merging/Splitting:

For any variable **x** and terms $\mathbf{t}_1$ and $\mathbf{t}_2$:

$$\Delta \vdash \mathbf{x{:=}t_1;\ x{:=}t_2} \approx \mathbf{x{:=}t_2[t_1/x]}$$

For example: **x:=x+1; x:=x−1** $\approx$ **x:=(x−1)[x+1/x]** $\approx$ **x:=(x+1)−1**

$\approx$ **x:=x** $\approx$ **{x=x}; x:=x** $\approx$ **skip**. (by the previous example)

Proof: $\mathbf{WP(x{:=}t_1;\ x{:=}t_2,\ R)}$

$\iff \mathbf{WP(x{:=}t_1,\ WP(x{:=}t_2,\ R))} \iff \mathbf{WP(x{:=}t_1,\ R[t_2/x])}$

$\iff \mathbf{R[t_2/x][t_1/x]} \iff \mathbf{R[t_2[t_1/x]/x]}$ since the only free **x**'s in $\mathbf{R[t_2/x]}$ are those in $\mathbf{t}_2$.

$\iff \mathbf{WP(x{:=}t_2[t_1/x],R)}$

Lemma: If the variable **m** is constant i S (ie is not assigned to) and all the variables of the term **t** are constant in **S** then **WP(S,R)[t/m]** $\iff$ **WP(S[t/m]**,Rt/m])

Proof: By induction on the structure of **S**, using the order relation on structure given above.

Subsumption:

If the variable **m** is constant in **S:V $\to$ W** and all the variables of the term **t** are constant in **S** and **m$\notin$var(t)** then:

$$\Delta \vdash \underline{\textbf{begin}}\ \mathbf{m{:=}t{:}\ S}\ \underline{\textbf{end}} \approx \mathbf{S[t/m]}$$

ie one can replace **m** by **t** in **S** and remove the variable **m** from the program. This is especially

valuable when the term is a single variable or if there is only one use of **m** (in the case where **t** is a constant this is called scalar propagation).
Proof: Let $\mathbf{S'}$ = **begin** **m:=t: S** **end** and let **w** be a list of the variables in **W**.
Note that $\mathbf{m \notin W}$.
$\mathbf{WP(S', G(w))} \iff \forall \mathbf{m.WP(m{:=}t;\ S,\ G(w))}$
For the assignment **m:=t** we have : $\mathbf{WP(m{:=}t,R)} \iff \mathbf{R[t/m]}$

So $\mathbf{WP(S',G(w))} \iff \forall \mathbf{m.WP(m{:=}t;\ S;\ \langle\rangle/\langle m\rangle.true,\ G(w))}$
$\iff \forall \mathbf{m.WP(S;\ \langle\rangle/\langle m\rangle.true,\ G(w))[t/m]} \iff \mathbf{WP(S;\langle\rangle/\langle m\rangle.true,\ G(w))[t/m]}$
since $\mathbf{m \notin var(t)}$ so **m** does not occur free.
$\iff \mathbf{WP(S,G(w))[t/m]} \iff \mathbf{WP(S[t/m],G(w)[t/m])}$ by Lemma above
$\iff \mathbf{WP(S[t/m],G(w))}$ since **m** does not occur in **R**.

MANIPULATION

Exportation of Independent Conditions:

This transformation provides one way in which a complex atomic description can be analysed into an **if** statement and two (or more) simpler atomic description. This it is a kind of "factoring" operation.
If no variable in **x** occurs in the formulae **P** and **Q** (ie $\underline{\mathbf{x}} \cap (\mathbf{var(P)} \cup \mathbf{var(Q)}) = \varnothing$) then:
$\Delta \cup \{\mathbf{P \wedge Q} \Rightarrow (\exists \mathbf{x.P'} \iff \exists \mathbf{x.Q'})\} \vdash$
$\mathbf{x/y.((P \wedge P') \vee (Q \wedge Q'))} \approx$ **if** $\mathbf{P} \rightarrow \mathbf{x/y.P'} \ \square\ \mathbf{Q} \rightarrow \mathbf{x/y.Q'}$ **fi**
Proof: This relies on the following Lemma:

Lemma: If $\underline{\mathbf{x}} \cap \mathbf{var(B)} = \varnothing$ then $\Delta \vdash \mathbf{\{B\};\ x/y.Q} \approx \mathbf{x/y.(Q \wedge B)}$.
Proof: $\mathbf{WP(\{B\};\ x/y.Q,\ G(w))}$
$\iff \mathbf{B} \wedge (\exists \mathbf{x.Q} \wedge \forall \mathbf{x.(Q \Rightarrow G(w))})$
$\iff \exists \mathbf{x.Q} \wedge \forall \mathbf{x.(B} \wedge \mathbf{(Q \Rightarrow G(w)))}$
$\iff \exists \mathbf{x.Q} \wedge \forall \mathbf{x.(B} \wedge (\neg \mathbf{Q} \vee \mathbf{G(w)}))$
$\iff \exists \mathbf{x.Q} \wedge \forall \mathbf{x.(B} \wedge (\neg \mathbf{Q} \vee \neg \mathbf{B} \vee \mathbf{G(w)}))$
$\iff \exists \mathbf{x.Q} \wedge \forall \mathbf{x.(B} \wedge \mathbf{(Q \wedge B)} \Rightarrow \mathbf{G(w)}))$
$\iff \exists \mathbf{x.Q} \wedge \mathbf{B} \wedge \forall \mathbf{x.((Q \wedge B)} \Rightarrow \mathbf{G(w)}))$
$\iff \exists \mathbf{x.(Q \wedge B)} \wedge \forall \mathbf{x.((Q \wedge B)} \Rightarrow \mathbf{G(w)}))$
$\iff \mathbf{WP(x/y.(Q \wedge B),\ G(w))}$ as required.

Proof of Theorem: By case analysis:
<u>Case</u> (i): $\neg\mathbf{Q}$:
$\mathbf{x/y.}\big((\mathbf{P} \wedge \mathbf{P'}) \ \vee \ (\mathbf{Q} \wedge \mathbf{Q'})\big)$
$\quad \approx \ \mathbf{x/y.}\big(\neg\mathbf{Q} \ \wedge \ (((\mathbf{P} \wedge \mathbf{P'}) \ \vee \ (\mathbf{Q} \wedge \mathbf{Q'}))\big)$ by Lemma.
$\quad \approx \ \mathbf{x/y.}\big(\neg\mathbf{Q} \ \wedge \ \mathbf{P} \ \wedge \ \mathbf{P'}\big)$
$\quad \approx \ \{\neg\mathbf{Q} \ \wedge \ \mathbf{P}\}\mathbf{;\ x/y.P'}$ by Lemma.
$\underline{\mathbf{if}}\ \mathbf{P} \ \rightarrow \ \mathbf{x/y.P'} \ \square \ \mathbf{Q} \ \rightarrow \ \mathbf{x/y.Q'}\ \underline{\mathbf{fi}}$
$\quad \approx \ \underline{\mathbf{if}}\ \mathbf{P} \ \rightarrow \ \mathbf{x/y.P'}\ \underline{\mathbf{fi}}$ by $\underline{\mathbf{if}}$ pruning.
$\quad \approx \ \{\mathbf{P}\}\mathbf{;\ x/y.P'}$
$\quad \approx \ \{\neg\mathbf{Q} \ \wedge \ \mathbf{P}\}\mathbf{;\ x/y.P'}$.

<u>Case</u> (ii): $\neg\mathbf{P}$: This is similar to Case (i).

<u>Case</u> (iii): $\mathbf{P} \wedge \mathbf{Q}$:
$\mathbf{WP(x/y.}\big((\mathbf{P} \wedge \mathbf{P'}) \ \vee \ (\mathbf{Q} \wedge \mathbf{Q'})\big)\mathbf{,\ G(w))}$
$\quad \iff \big((\mathbf{P} \ \wedge \ \exists\mathbf{x.P'}) \ \vee \ (\mathbf{Q} \ \wedge \ \exists\mathbf{x.Q'})\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{P} \wedge \mathbf{P'} \Rightarrow \mathbf{G(w)}\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{Q} \wedge \mathbf{Q'} \Rightarrow \mathbf{G(w)}\big)$
$\quad \iff \big((\mathbf{P} \ \wedge \ \exists\mathbf{x.P'}) \ \vee \ (\mathbf{Q} \ \wedge \ \exists\mathbf{x.Q'})\big) \ \wedge \ \mathbf{P} \Rightarrow \forall\mathbf{x.}\big(\mathbf{P'} \Rightarrow \mathbf{G(w)}\big) \ \wedge \ \mathbf{Q} \Rightarrow \forall\mathbf{x.}\big(\mathbf{Q'} \Rightarrow \mathbf{G(w)}\big)$
$\quad \iff \big(\exists\mathbf{x.P'} \ \vee \ \exists\mathbf{x.Q'}\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{P'} \Rightarrow \mathbf{G(w)}\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{Q'} \Rightarrow \mathbf{G(w)}\big)$
$\mathbf{WP(}\underline{\mathbf{if}}\ \mathbf{P} \ \rightarrow \ \mathbf{x/y.P'} \ \square \ \mathbf{Q} \ \rightarrow \ \mathbf{x/y.Q'}\ \underline{\mathbf{fi}}\mathbf{,\ G(w))}$
$\quad \iff \mathbf{P} \Rightarrow \big(\exists\mathbf{x.P'} \ \wedge \ \forall\mathbf{x.}\big(\mathbf{P'} \Rightarrow \mathbf{G(w)}\big)\big) \ \wedge \ \mathbf{Q} \Rightarrow \big(\exists\mathbf{x.Q'} \ \wedge \ \forall\mathbf{x.}\big(\mathbf{Q'} \Rightarrow \mathbf{G(w)}\big)\big)$
$\quad \iff \big(\exists\mathbf{x.P'} \ \wedge \ \exists\mathbf{x.Q'}\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{P'} \Rightarrow \mathbf{G(w)}\big) \ \wedge \ \forall\mathbf{x.}\big(\mathbf{Q'} \Rightarrow \mathbf{G(w)}\big)$
Now $\mathbf{P} \wedge \mathbf{Q} \Rightarrow \big(\exists\mathbf{x.P'} \ \iff \ \exists\mathbf{x.Q'}\big)$ By premise.
$\qquad \Rightarrow \big(\exists\mathbf{x.P'} \ \wedge \ \exists\mathbf{x.Q'}\big) \ \iff \ \big(\exists\mathbf{x.P'} \ \vee \ \exists\mathbf{x.Q'}\big)$ so the sides are equivalent.

<u>Case</u> (iv): $\neg\mathbf{P} \ \wedge \ \neg\mathbf{Q}$: Both sides are **abort**.

Note: The only place we used the premise was in Case (iii) where we used the fact that $\big(\exists\mathbf{x.P'} \ \iff \ \exists\mathbf{x.Q'}\big) \Rightarrow \big(\exists\mathbf{x.P'} \ \vee \ \exists\mathbf{x.Q'}\big) \ \Rightarrow \ \big(\exists\mathbf{x.P'} \ \wedge \ \exists\mathbf{x.Q'}\big)$.
The RHS of this is in fact equivalent to our premise since $\mathbf{a} \vee \mathbf{b} \Rightarrow \mathbf{a} \wedge \mathbf{b} \ \iff \ \neg(\mathbf{a} \vee \mathbf{b}) \ \vee \ (\mathbf{a} \wedge \mathbf{b}) \ \iff$ $(\neg\mathbf{a} \ \vee \ \mathbf{b}) \ \wedge \ (\neg\mathbf{b} \ \vee \ \mathbf{a}) \ \iff \ (\mathbf{a} \Rightarrow \mathbf{b}) \ \wedge \ (\mathbf{b} \Rightarrow \mathbf{a}) \ \iff \ (\mathbf{a} \iff \mathbf{b})$.

<u>Dead</u> <u>Variable</u> <u>Elimination:</u>
If the only assignments in **T** are to the variables in the list **a** then:
(a) $\Delta \vdash$ **<u>begin</u> a: S;T <u>end</u>** $\leqslant$ **<u>begin</u> a: S <u>end</u>**
(b) $\{\mathbf{WP(T,true)}\} \cup \Delta \vdash$ **<u>begin</u> a: S;T <u>end</u>** $\approx$ **<u>begin</u> a: S <u>end</u>**
This is frequently used for adding and removing "ghost variables".

Proof: First we prove $\Delta \vdash \{\mathbf{B}\};\mathbf{T} \approx \mathbf{T};\{\mathbf{B}\}$ if $\underline{\mathbf{x}}\cap\mathbf{var(B)}=\varnothing$.
Proof is by induction on the structure of **T**, we need to prove:

$$\mathbf{WP(\{B\};T,R)} \iff \mathbf{WP(T;\{B\},R)}$$
$$\text{ie } \mathbf{WP(T,R\wedge B)} \iff \mathbf{WP(T,R)\wedge B}$$

(i) $\mathbf{WP(x/y.Q,R\wedge B)} \iff \exists\mathbf{x.Q} \wedge \forall\mathbf{x.}(\mathbf{Q}\Rightarrow(\mathbf{R}\wedge\mathbf{B}))$
All the variables of **x** must be in **a** since **T** only assigns to variables in **a**. So none of the variables in **x** are free in **B**.

$$\exists\mathbf{x.Q} \wedge \forall\mathbf{x.}(\mathbf{Q}\Rightarrow(\mathbf{R}\wedge\mathbf{B})) \iff \exists\mathbf{x.Q} \wedge \forall\mathbf{x.}(\neg\mathbf{Q}\vee(\mathbf{R}\wedge\mathbf{B}))$$
$$\iff \exists\mathbf{x.Q} \wedge \forall\mathbf{x.}(\mathbf{Q}\Rightarrow\mathbf{R}) \wedge \mathbf{B} \text{ since } \underline{\mathbf{x}}\cap\mathbf{var(B)}=\varnothing$$
$$\iff \mathbf{WP(T,R)} \wedge \mathbf{B} \text{ as required.}$$

(ii) -(iv) $\mathbf{WP(S_1;S_2, R)}$, $\mathbf{WP(\underline{oneof}\ S_1 \vee S_2\ \underline{foeno}, R)}$ and $\mathbf{WP(\underline{if}\ Q\ \underline{then}\ S_1\ \underline{else}\ S_2\underline{fi}, R)}$
These are trivial applications of the induction hypothesis.

(v) $\mathbf{WP(\underline{proc}\ X \equiv S_1.,R\wedge B)} \iff \bigvee_{n<\omega}\mathbf{WP(\underline{proc}\ X \equiv S_1.^{n}\ ,R\wedge B)}$
$$\iff \bigvee_{n<\omega}(\mathbf{WP(\underline{proc}\ X \equiv S_1.^{n}\ ,R)\wedge B})$$
$$\iff \bigvee_{n<\omega}\mathbf{WP(\underline{proc}\ X \equiv S_1.^{n}\ ,R)} \wedge \mathbf{B}$$
$$\iff \mathbf{WP(\underline{proc}\ X \equiv S_1., R)} \wedge \mathbf{B}$$

which proves the result.

To prove (a) we see that if $\underline{\mathbf{x}}\cap\mathbf{var(R)}=\varnothing$ then
$\mathbf{WP(T,R)} \iff \mathbf{WP(T,R\wedge R)} \iff \mathbf{R\wedge WP(T,R)} \Rightarrow \mathbf{R}$
so $\mathbf{T;\langle\rangle/\langle a\rangle.true} \leqslant \mathbf{\langle\rangle/\langle a\rangle.true}$
so **<u>begin</u> a: S;T <u>end</u>** $\leqslant$ **<u>begin</u> a: S <u>end</u>** as required.

To prove (b), we need to prove $\mathbf{R} \iff \mathbf{WP(T,R)}$. We have $\mathbf{WP(T,true)}$ so
$\mathbf{R} \iff \mathbf{WP(T,true)\wedge R} \iff \mathbf{WP(\{R\};T,true)} \iff \mathbf{WP(T;\{R\},true)} \iff \mathbf{WP(T,R\wedge true)}$
$\iff \mathbf{WP(T,R)}$
so $\mathbf{T;\langle\rangle/\langle a\rangle.true} \approx \mathbf{\langle\rangle/\langle a\rangle.true}$
so **<u>begin</u> a: S;T <u>end</u>** $\approx$ **<u>begin</u> a: S <u>end</u>** as required.

<u>Dead Assignment Elimination</u>

If the variable **x** is only assigned to and never accessed in **S** then:

$\Delta \vdash$ **<u>begin</u> x:S <u>end</u>** $\leqslant \mathbf{S'}$ where $\mathbf{S'}$ is **S** with all assignments to **x** replaced by **skip**.

This is a consequence of the equivalence:

$\Delta \vdash$ **<u>begin</u> x:S <u>end</u>** $\approx$ **<u>begin</u> x:S[{∃x.Q};⟨⟩/y.true / x/y.Q] <u>end</u>**

Here we have replaced each assignment $\mathbf{x/y.Q}$ to $\mathbf{x}$ by an assertion: $\mathbf{\{\exists x.Q\}}$ and an atomic description: $\langle\rangle\mathbf{/y.true}$.
The proof is by induction on the structure of $\mathbf{S}$ with the base step: $\mathbf{S{=}x/y.Q}$

$\mathbf{WP(x/y.Q;\ \langle\rangle/x.true,\ R)}$
$\iff \mathbf{WP(x/y.Q,\ R)}$
$\iff \exists\mathbf{x.Q} \wedge \forall\mathbf{x.}\big(\mathbf{Q{\Rightarrow}R}\big) \iff \exists\mathbf{x.Q} \wedge \big(\forall\mathbf{x.}\big(\neg\mathbf{Q}\big) \vee \mathbf{R}\big)$ since $\mathbf{x}\notin\mathbf{vars(R)}$
$\iff \exists\mathbf{x.Q} \wedge \mathbf{R}$
$\iff \mathbf{WP(\{Ex.Q\},\ R)}$

Induction steps **oneof** $\mathbf{S_1} \vee \mathbf{S_2}$ **foeno** and **if** **B** **then** $\mathbf{S_1}$ **else** $\mathbf{S_2}$ **fi** are trivial.

$\mathbf{WP(S_1;S_2,\ R)} \iff \mathbf{WP(S_1,\ WP(S_2,\ R))}$
$\iff \mathbf{WP(S_1,\ WP(S_2[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ R))}$

$\mathbf{x}$ is never accessed and does nor occur in $\mathbf{R}$ so the formula $\mathbf{WP(S_2[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ R)}$ will contain no free occurrences of $\mathbf{x}$. So we can use the induction hypothesis again to get:

$\iff \mathbf{WP(S_1[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ WP(S_2[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ R))}$
$\iff \mathbf{WP(S_1[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q];\ S_2[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ R)}$
$\iff \mathbf{WP((S_1;S_2)[\{\exists x.Q\};\langle\rangle/y.true\ /\ x/y.Q],\ R)}$ as required.

The other cases $(\mathbf{B{*}S_1})$ and $(\mu\mathbf{X.S_1})$ are trivial.

This transformation will be used extensively in adding and removing variables which change the data representation of a program. See for example our derivation of the Schorr-Waite graph-marking algorithm.

Lemma: The Invariance Lemma.
Let $\mathbf{invar(B,S)} =_{DF} \big(\mathbf{B} \wedge \mathbf{WP(S,true)} \Rightarrow \mathbf{WP(S,B)}\big) \wedge \big(\neg\mathbf{B} \wedge \mathbf{WP(S,true)} \Rightarrow \mathbf{WP(S,\neg B)}\big)$
Then $\{\mathbf{invar(B,S)}\} \cup \Delta \vdash \mathbf{S;\{B\}} \approx \mathbf{\{B\};S}$ and $\{\mathbf{invar(B,S)}\} \cup \Delta \vdash \mathbf{S;\{\neg B\}} \approx \mathbf{\{\neg B\};S}$.

The statement $\mathbf{invar(B,S)}$ says that $\mathbf{B}$ is invariant over $\mathbf{S}$ and the result states that we can assume $\mathbf{B}$ holds before $\mathbf{S}$ if it holds afterwards and conversely (and similarly for $\neg\mathbf{B}$). This Lemma shows that if we can insert a condition into the postcondition of a **WP** without weakening it then we can insert the same condition as an assertion after the statement.

Proof: $\mathbf{WP(S;\{B\},R)} \iff \mathbf{WP(S,WP(\{B\},R))} \iff \mathbf{WP(S,B \wedge R)}$
$\iff \mathbf{WP(S,B)} \wedge \mathbf{WP(S,R)}$

$\mathbf{R{\Rightarrow}true}$ so $\mathbf{WP(S,R){\Rightarrow}WP(S,true)}$ So from $\mathbf{WP(S,B)} \wedge \mathbf{WP(S,R)}$ we can deduce $\mathbf{B}$ since if $\neg\mathbf{B}$ holds then the second term of $\mathbf{invar(B,S)}$ gives $\mathbf{WP(S,\neg B)}$ which contradicts $\mathbf{WP(S,B)}$ since $\mathbf{WP(S,\neg B)} \wedge \mathbf{WP(S,B)} \iff \mathbf{WP(S,\neg B \wedge B)} \iff \mathbf{WP(S,false)} \iff \mathbf{false}$.

So $\mathbf{WP(S;\{B\},R)} \Rightarrow \mathbf{B} \wedge \mathbf{WP(S,R)}$
$\iff \mathbf{WP(\{B\};S,R)}$.
Conversely $\mathbf{WP(\{B\};S,R)} \iff \mathbf{B} \wedge \mathbf{WP(S,R)} \Rightarrow \mathbf{WP(S,B)} \wedge \mathbf{WP(S,R)} \iff \mathbf{WP(S;\{B\},R)}$ since $\mathbf{WP(S,R)} \Rightarrow \mathbf{WP(S,true)}$ and the first term of **invar(B,S)** then gives **WP(S,B)**.

Back Expansion of a Conditional:
If formula **B** is invariant over statement **S** then:
$\Delta \vdash$ **S; if B then TC else FC fi** $\approx$ **if B then S; TC else S; FC fi**
Proof: S; if B then TC else FC fi
$\approx$ **if B then {B}; S; if B then TC else FC fi**
else {¬B}; S; if B then TC else FC fi fi by splitting a tautology.
$\approx$ **if B then S; {B}; if B then TC else FC fi**
else S; {¬B}; if B then TC else FC fi fi by Invariance Lemma.
$\approx$ **if B then S; TC else S; FC fi**. by Prune Conditional.

Forward Expansion:
$\Delta \vdash$ **if B then TC else FC fi; S** $\approx$ **if B then TC; S else FC; S fi**
Proof: WP(if B then TC else FC fi; S, R)
$\iff$ **WP(if B then TC else FC fi, WP(S,R))**
$\iff$ **(B⇒WP(TC, WP(S,R)))** $\wedge$ **(¬B⇒WP(FC, WP(S,R)))**
$\iff$ **(B⇒WP(TC;S, R))** $\wedge$ **(¬B⇒WP(FC;S, R))**
$\iff$ **WP(if B then TC; S else FC; S fi, R)**.
These two transformations are often used to replace two copies of a statement by a single copy. More generally they are used to move components of a program through the structure of the program. See the "absorption" transformations for examples.

Often the first or last iteration of a loop may be a special case and so the loop can be simplified if one iteration is taken out of the loop:

Loop Unrolling:
$\Delta \vdash$ **while B do S od** $\approx$ **if B then S; while B do S od fi**
Proof: Let **DO=while B do S od**, $\mathbf{DO}^n$ =**while B do S od**^{n}.
For any $\mathbf{n} < \omega$, $\mathbf{DO}^{n+1} \approx$ **if B then S;**$\mathbf{DO}^n$ **fi**
$\leqslant$ **if B then S;DO fi** by induction rule for iteration.
So **DO** $\leqslant$ **if B then S;DO fi** by induction rule for iteration.

Conversely $\mathbf{WP(\underline{if}\ B\ \underline{then}\ S;DO\ \underline{fi},\ R)} \iff \big(\mathbf{B\Rightarrow WP(S;DO,R)}\big) \wedge \big(\mathbf{\neg B\Rightarrow WP(skip,R)}\big)$

$\iff \big(\mathbf{B\Rightarrow WP(S,WP(DO,R))}\big) \wedge \big(\mathbf{\neg B\Rightarrow R}\big)$

$\iff \big(\mathbf{B\Rightarrow WP(S,}\bigvee_{n<\omega}\mathbf{WP(DO^n,R))}\big) \wedge \big(\mathbf{\neg B\Rightarrow R}\big)$

$\mathbf{DO}^n \leqslant \mathbf{DO}^{n+1}$ so $\mathbf{WP(DO^n,R)\Rightarrow WP(DO^{n+1},R)}$ for all $\mathbf{n}<\omega$.

So by continuity of **WP**: $\mathbf{WP(S,}\bigvee_{n<\omega}\mathbf{WP(DO^n,R))} \Rightarrow \bigvee_{n<\omega}\mathbf{WP(S,WP(DO^n,R))}$

So $\mathbf{WP(\underline{if}\ B\ \underline{then}\ S;DO\ \underline{fi},\ R)} \Rightarrow \big(\mathbf{B\Rightarrow}\bigvee_{n<\omega}\mathbf{WP(S,WP(DO^n,R))}\big) \wedge \big(\mathbf{\neg B\Rightarrow R}\big)$

$\iff \bigvee_{n<\omega}\Big(\big(\mathbf{B\Rightarrow WP(S,WP(DO^n,R))}\big) \wedge \big(\mathbf{\neg B\Rightarrow R}\big)\Big)$

$\iff \bigvee_{n<\omega}\mathbf{WP(\ \underline{if}\ B\ \underline{then}\ S;DO^n\ \underline{fi},\ R)}$

$\iff \bigvee_{n<\omega}\mathbf{WP(DO^{n+1},R)}$

$\iff \mathbf{WP(DO,R)}$

Hence **<u>if</u> B <u>then</u> S;DO <u>fi</u>** $\leqslant$ **DO** and the result is proved.

<u>Recursion</u> <u>Unfolding:</u>

$\Delta \vdash \mathbf{\underline{proc}\ X \equiv S.} \approx \mathbf{S[\underline{proc}\ X \equiv S./X]}$

Proof: The proof uses the induction rule for recursion. For any $\mathbf{n}<\omega$:

$\mathbf{\underline{proc}\ X \equiv S.}^{n+1} = \mathbf{S[\underline{proc}\ X \equiv S.}^n\mathbf{/X]} \leqslant \mathbf{S[\underline{proc}\ X \equiv S./X]}$ by induction rule for recursion.

So $\mathbf{\underline{proc}\ X \equiv S.} \leqslant \mathbf{S[\underline{proc}\ X \equiv S./X]}$.

Conversely $\mathbf{WP(S[\underline{proc}\ X \equiv S./X],\ R)}$

$\iff \mathbf{WP(S,R)[WP(\underline{proc}\ X \equiv S.,R)/X]}$ (since $\mathbf{WP(X,R)=X}$)

$\iff \mathbf{WP(S,R)[}\bigvee_{n<\omega}\mathbf{WP(\underline{proc}\ X \equiv S.^n,R)/X]}$

By induction on the structure of **S** and using the continuity of **WP** we can prove

$\mathbf{WP(S,R)[}\bigvee_{n<\omega}\mathbf{P}_n\mathbf{/X]} \Rightarrow \bigvee_{n<\omega}\mathbf{WP(S,R)[P}_n\mathbf{/X]}$ provided $\mathbf{P}_n \leqslant \mathbf{P}_{n+1}$ for all $\mathbf{n}<\omega$. (see below).

$\mathbf{\underline{proc}\ X \equiv S.}^n \leqslant \mathbf{\underline{proc}\ X \equiv S.}^{n+1}$ for all $\mathbf{n}<\omega$ so

$\mathbf{WP(S[\underline{proc}\ X \equiv S./X],\ R)} \Rightarrow \bigvee_{n<\omega}\mathbf{WP(S,R)[WP(\underline{proc}\ X \equiv S.^n,R)/X]}$

$\iff \bigvee_{n<\omega}\mathbf{WP(\underline{proc}\ X \equiv S.^{n+1},R)}$

$\iff \mathbf{WP(\underline{proc}\ X \equiv S.,R)}$

Hence $\mathbf{S[\underline{proc}\ X \equiv S./X]} \leqslant \mathbf{\underline{proc}\ X \equiv S.}$ and the result is proved.

In this transformation we replace <u>every</u> recursive call in the body of the procedure by a copy of the recursive procedure. Later we will prove a transformation ("selective unfolding") which allows us to select a subset of the recursive calls which are unfolded if a given condition is satisfied.

<u>Cor:</u> <u>Loop</u> <u>Unrolling:</u>

$\Delta \vdash$ **<u>while</u> B <u>do</u> S <u>od</u>** $\approx$ **<u>if</u> B <u>then</u> S; <u>while</u> B <u>do</u> S <u>od</u> <u>fi</u>**

Lemma: If $\mathbf{P}_n \Rightarrow \mathbf{P}_{n+1}$ for all $\mathbf{n} < \omega$ then $\mathbf{WP(S,R)}[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}] \Rightarrow \bigvee_{n<\omega}\mathbf{WP(S,R)}[\mathbf{P}_n/\mathbf{X}]$.
Proof: By induction on the structure of **S**, using a lexical order of:
(i) Depth of recursion nesting.
(ii) Length of program text.

Case(i): **X** does not appear in **S**.
$\mathbf{WP(S,R)}[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}] \iff \mathbf{WP(S,R)} \iff \bigvee_{n<\omega}\mathbf{WP(S,R)} \iff \bigvee_{n<\omega}\mathbf{WP(S,R)}[\mathbf{P}_n/\mathbf{X}]$.

Case (ii): **S=X**
$\mathbf{WP(X,R)}[\bigvee_{n<\omega}\mathbf{P}_n,/\mathbf{X}] \iff \mathbf{X}[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}] \iff \bigvee_{n<\omega}\mathbf{P}_n \iff \bigvee_{n<\omega}\mathbf{WP(X,R)}[\mathbf{P}_n/\mathbf{X}]$.

Case (iii): $\mathbf{S}=\mathbf{S}_1;\mathbf{S}_2$
$\mathbf{WP(S_1;S_2,R)}[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}] \iff \mathbf{WP(S_1,\ WP(S_2,R)}[\bigvee_{m<\omega}\mathbf{P}_m/\mathbf{X}])[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}]$
$\iff \bigvee_{n<\omega}\mathbf{WP(S_1,}\ \bigvee_{m<\omega}\mathbf{WP(S_2,R)}[\mathbf{P}_m/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}]$
$\iff \bigvee_{n,m<\omega}\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_m/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}]$ (by continuity of **WP**)
$\iff \bigvee_{n<\omega}\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_n/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}]$
To prove $\bigvee_{n,m<\omega}\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_m/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}] \Rightarrow \bigvee_{n<\omega}\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_n/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}]$
we use the fact that $(\mathbf{P}\Rightarrow\mathbf{Q}) \Rightarrow (\mathbf{WP(S,P)}\Rightarrow\mathbf{WP(S,Q)})$
hence if $\mathbf{k}\geqslant\mathbf{max(n,m)}$ then

$\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_m/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}] \Rightarrow \mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_k/\mathbf{X}])[\mathbf{P}_k/\mathbf{X}]$ So
$\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_m/\mathbf{X}])[\mathbf{P}_n/\mathbf{X}] \Rightarrow \bigvee_{k<\omega}\mathbf{WP(S_1,\ WP(S_2,R)}[\mathbf{P}_k/\mathbf{X}])[\mathbf{P}_k/\mathbf{X}]$ for $\mathbf{n,m} < \omega$.
The other implication is trivial.

Hence $\mathbf{WP(S_1;S_2,R)}[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}]$
$\iff \bigvee_{n<\omega}\mathbf{WP(S_1,\ WP(S_2,R))}[\mathbf{P}_n/\mathbf{X}] \iff \bigvee_{n<\omega}\mathbf{WP(S_1;S_2,R)}[\mathbf{P}_n/\mathbf{X}]$.

Case (iv): **S=** **oneof** $\mathbf{S}_1 \square \mathbf{S}_2$ **foeno** and

Case (v): **S=** **if** **B** **then** $\mathbf{S}_1$ **else** $\mathbf{S}_2$ **fi** follow directly from the induction hypothesis.

Case (vi): **S=** **proc** $\mathbf{X} \equiv \mathbf{S}_1$**.**
$\mathbf{WP(}$ **proc** $\mathbf{X} \equiv \mathbf{S}_1.\ ,\mathbf{R})[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}] \iff \bigvee_{m<\omega}\mathbf{WP(}$ **proc** $\mathbf{X} \equiv \mathbf{S}_1.^m\ ,\mathbf{R})[\bigvee_{n<\omega}\mathbf{P}_n/\mathbf{X}]$
$\iff \bigvee_{m<\omega}\bigvee_{n<\omega}\mathbf{WP(}$ **proc** $\mathbf{X} \equiv \mathbf{S}_1.^m\ ,\mathbf{R})[\mathbf{P}_n/\mathbf{X}]$ by induction hypothesis
$\iff \bigvee_{n<\omega}\bigvee_{m<\omega}\mathbf{WP(}$ **proc** $\mathbf{X} \equiv \mathbf{S}_1.^m\ ,\mathbf{R})[\mathbf{P}_n/\mathbf{X}]$
$\iff \bigvee_{n<\omega}\mathbf{WP(}$ **proc** $\mathbf{X} \equiv \mathbf{S}_1.\ ,\mathbf{R})[\mathbf{P}_n/\mathbf{X}]$ as required.

This proves the result.

Loop first case:

$\Delta \vdash$ **for i:=b to f step s do S od**
  $\approx$ **if b$\leqslant$f then S[b/i]; for i:=b+s to f step s do S od fi**

Proof: **for i:=b to f step s do S od** $\approx$ **begin i:=b: while i$\leqslant$f do S;i:=i+s od end**
  $\approx$ **begin i:=b: if i$\leqslant$f then S; i:=i+s; while i$\leqslant$f do S;i:=i+s od fi end** (by Loop Unrolling)

Use Subsumption in reverse on **i:=i+s; while i$\leqslant$f do S;i:=i+s od** to get
  $\approx$ **begin i:=b: if i$\leqslant$f then S; begin i$'$:=i: i$'$:=i$'$+s;**
    while i$'$ $\leqslant$f do S[i$'$/i]; i$'$:=i$'$+s od end fi end
  $\approx$ **if b$\leqslant$f then S[b/i]; begin i$'$:=b: i$'$:=i$'$+s;** (by subsumption)
    while i$'$ $\leqslant$f do S[i$'$/i]; i$'$:=i$'$+s od end fi

since **S** does not assign to **i** or any variable of **b** by definition of a **for** loop.
  $\approx$ **if b$\leqslant$f then S[b/i]; begin i$'$:=b+s;** (by assignment merging)
    while i$'$ $\leqslant$f do S[i$'$/i]; i$'$:=i$'$+s od end fi

Replacing **i$'$** by **i** throughout (since **i** no longer occurs) and writing as a **for** statement gives:
  $\approx$ **if b$\leqslant$f then S[b/i]; for i:=b+s to f step s do S od fi**.

Loop Last Case:

If in addition to the "+" function we have a "−" function with the property that $(\mathbf{x}+\mathbf{y})-\mathbf{y}=\mathbf{y}$ then we can use this in the following transformation which unrolls the last step of a **for** loop:

$\Delta \vdash$ **for i:=b to f step s do S od**
  $\approx$ **if b$\leqslant$f then begin i:=b: while i$\leqslant$f−s do S; i:=i+s od; S end fi**

Proof: Let **FOR**=**for i:=b to f step s do S od**
  FOR $\approx$ **if b$\leqslant$f then {b$\leqslant$f};FOR else {b>f};FOR fi** (by splitting a tautology)

Case(a): Assume **b>f**. Then:

FOR $\approx$ **begin i:=b: {i=b $\wedge$ b>f}; while i$\leqslant$f do S;i:=i+s od end**
  $\approx$ **begin i:=b: {i>f}; while i$\leqslant$f do S;i:=i+s od end**
  $\approx$ **begin i:=b: skip end**
  $\approx$ **skip**.

Case(b): Assume **b$\leqslant$f**. Then:

FOR $\approx$ **begin i:=b: {i=b $\wedge$ b$\leqslant$f}; while i$\leqslant$f do S;i:=i+s od end**
  $\approx$ **begin i:=b: {i$\leqslant$f}; while i$\leqslant$f do S;i:=i+s od end**

WP({i$\leqslant$f}; while i$\leqslant$f do S; i:=i+s od, R)
  $\iff$ **i$\leqslant$f** $\wedge$ $\bigvee_{n<\omega}$**WP(while i$\leqslant$f do S;i:=i+s od^n, R)**

$\iff \bigvee_{n<\omega}\big(\mathbf{i\leqslant f} \wedge \mathbf{WP(\underline{while}\ i\leqslant f\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n},\ R)}\big)$

Claim: $\big(\mathbf{i\leqslant f}\big) \wedge \mathbf{WP(\underline{while}\ i\leqslant f\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n},\ R)}$

$\iff \big(\mathbf{i\leqslant f}\big) \wedge \mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1};\ S;i{:=}i{+}s,\ R)}$

ie $\Delta \vdash \mathbf{\{i\leqslant f\};\underline{while}\ i\leqslant f\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n}} \approx \mathbf{\{i\leqslant f\};\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1};\ S;i{:=}i{+}s}$

Proof of claim: Let $\mathbf{DO}^{n} = \mathbf{\underline{while}\ i\leqslant f\ \underline{do}\ S;\ i{:=}i{+}s\ \underline{od}^{n}}$. Use induction on **n**.

For **n=1**: $\mathbf{\{i\leqslant f\};DO^{1}} \approx \mathbf{\{i\leqslant f\};\ \underline{if}\ i\leqslant f\ \underline{then}\ S;\ abort\ \underline{fi}} \approx \mathbf{abort} \approx \mathbf{abort;S;i{:=}i{+}s}$

Induction step: suppose result holds for **n**.

Let $\mathbf{DO'}^{n} = \mathbf{\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;\ i{:=}i{+}s\ \underline{od}^{n-1};\ S;i{:=}i{+}s}$

$\mathbf{\{i\leqslant f\};DO^{n+1}} \approx \mathbf{\{i\leqslant f\};\ \underline{if}\ i\leqslant f\ \underline{then}\ S;i{:=}i{+}s;\ DO^{n}\ \underline{fi}}$

$\approx \mathbf{\{i\leqslant f\};\ S;i{:=}i{+}s;\ DO^{n}}$

Case (i): $\mathbf{i{+}s\leqslant f}$ initially. Then we have:

$\mathbf{\{i\leqslant f\};DO^{n+1}} \approx \mathbf{\{i\leqslant f\};S;i{:=}i{+}s;\ \{i\leqslant f\};\ DO^{n}}$

since **S** does not assign to **i** (the control variable of a **<u>for</u>** loop).

$\approx \mathbf{\{i\leqslant f\};S;i{:=}i{+}s;\ \{i\leqslant f\};\ \underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1};\ S;i{:=}i{+}s}$

by induction hypothesis.

$\approx \mathbf{\{i\leqslant f\};\underline{if}\ i\leqslant f{-}s\ \underline{then}\ S;i{:=}i{+}s;}$

$\mathbf{\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1}\ \underline{fi};\ S;i{:=}i{+}s}$

since $\mathbf{i\leqslant f{-}s}$ initially.

$\approx \mathbf{\{i\leqslant f\};\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n};\ S;i{:=}i{+}s}$ as required.

Case (ii): $\mathbf{i{+}s>f}$ initially ie $\mathbf{f{-}s<i}$ initially. Then we have:

$\mathbf{\{i\leqslant f\};DO^{n+1}} \approx \mathbf{\{i\leqslant f\};\ S;i{:=}i{+}s;\ \{i>f\};\ DO^{n}}$

$\approx \mathbf{\{i\leqslant f\};\ S;i{:=}i{+}s;\ \{i>f\};\ skip}$

$\approx \mathbf{\{i\leqslant f\};\ \underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n};\ S;i{:=}i{+}s}$

which proves the claim.

So $\bigvee_{n<\omega}\big(\mathbf{i\leqslant f} \wedge \mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n},\ R)}\big)$

$\iff \bigvee_{n<\omega}\big(\mathbf{i\leqslant f} \wedge \mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1};\ S;i{:=}i{+}s,\ R)}\big)$

$\iff \mathbf{i\leqslant f} \wedge \bigvee_{n<\omega}\mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od}^{n-1},\ WP(S;i{:=}i{+}s,\ R))}$

$\iff \mathbf{i\leqslant f} \wedge \mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od},\ WP(S;i{:=}i{+}s,\ R))}$

$\iff \mathbf{i\leqslant f} \wedge \mathbf{WP(\underline{while}\ i\leqslant f{-}s\ \underline{do}\ S;i{:=}i{+}s\ \underline{od};\ S;i{:=}i{+}s,\ R)}$

Hence $\{\mathbf{b}>\mathbf{f}\}$;**FOR** $\approx$ **begin i:=b: {i⩽f−s}; while i⩽f−s do S;i:=i+s od; S;i:=i+s end**
Putting the two cases (a) and (b) together gives
FOR $\approx$ **if b⩽f then begin i:=b: while i⩽f−s do S;i:=i+s od S;i:=i+s end fi**

Now **WP(i:=i+s,R)** $\iff$ **R[i+s/i]** $\iff$ **R**
if **R** is a condition on the final state, since the final state space does not include **i** so **i**$\notin$**var(R)**.
Hence **FOR** $\approx$ **if b⩽f then begin i:=b: while i⩽f−s do S;i:=i+s od S end fi** as required, where the final assignment to **i** has been removed by dead variable elimination.

Loop middle case:
If **m** is a term and **S** does not assign to any variables of **m** then:
$\{\mathbf{m}\leqslant\mathbf{f}\} \cup \Delta \vdash$ **for i:=b to f step s do S od**
$\approx$ **begin i:=b: while i⩽m do {i⩽m}; S;i:=i+s od;**
while i⩽f do {i>m}; S;i:=i+s od end
Proof: for i:=b to f step s do S od $\approx$ **begin i:=b: while i⩽f do S;i:=i+s od end**
Let $\mathbf{S}' = \mathbf{S;i{:=}i+s}$. We need to prove
$\Delta \vdash$ **while i⩽f do S′ od** $\approx$ **while i⩽m do S′ od; while i⩽f do S′ od**.
In fact this is a special case of the more general result:

Lemma: Loop Merging: If $\mathbf{B}_1 \Rightarrow \mathbf{B}_2$ then:
$\Delta \vdash$ **while** $\mathbf{B}_2$ **do S′ od** $\approx$ **while** $\mathbf{B}_1$ **do S′ od; while** $\mathbf{B}_2$ **do S′ od**.

In our case we have $\mathbf{B}_1 \iff \mathbf{i}\leqslant\mathbf{m}$ and $\mathbf{B}_2 \iff \mathbf{i}\leqslant\mathbf{f}$ from which $\mathbf{i}\leqslant\mathbf{m} \Rightarrow \mathbf{i}\leqslant\mathbf{f}$ follows from $\mathbf{m}\leqslant\mathbf{f}$.
Proof: (of Lemma) **WP(while** $\mathbf{B}_1$ **do S′ od; while** $\mathbf{B}_2$ **do S′ od, R)**
$\iff$ **WP(while** $\mathbf{B}_1$ **do S′ od, WP(while** $\mathbf{B}_2$ **do S′ od, R))**
$\iff \bigvee_{n<\omega}$**WP(while** $\mathbf{B}_1$ **do S′** $\mathbf{od}^n$, $\bigvee_{k<\omega}$**WP(while** $\mathbf{B}_2$ **do S′** $\mathbf{od}^k$**, R))**
$\iff \bigvee_{n,k<\omega}$**WP(while** $\mathbf{B}_1$ **do S′** $\mathbf{od}^n$**, WP(while** $\mathbf{B}_2$ **do S′** $\mathbf{od}^k$**, R))**
$\iff \bigvee_{n,k<\omega}$**WP(while** $\mathbf{B}_1$ **do S′** $\mathbf{od}^n$**; while** $\mathbf{B}_2$ **do S′** $\mathbf{od}^k$**, R)**

Claim: for $\mathbf{n}<\omega$ there exists $\mathbf{k}<\omega$ such that
$\Delta \vdash \mathbf{DO}^n \leqslant$ **while** $\mathbf{B}_1$ **do S′** $\mathbf{od}^k$**; while** $\mathbf{B}_2$ **do S′** $\mathbf{od}^k$

Proof of claim: use induction on **n**, result is trivial for **n=0**.
$\mathbf{DO}^{n+1} \approx$ **if** $\mathbf{B}_1$ **then S′;** $\mathbf{DO}^n$ **fi**
$\leqslant$ **if** $\mathbf{B}_1$ **then S′; while** $\mathbf{B}_1$ **do S′** $\mathbf{od}^k$**; while** $\mathbf{B}_2$ **do S′** $\mathbf{od}^k$ **fi**
for some **k** (by induction hypothesis).

Now consider cases on $\mathbf{B}_1$ and $\mathbf{B}_2$ to show:

$\mathbf{DO}^{n+1} \leqslant$ **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ $\mathbf{od}^{k+1}$; **while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ $\mathbf{od}^{k+1}$

and the claim is proved.

Hence: $\mathbf{DO}^{n+1} \leqslant$ **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ **od; while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ **od** by induction rule for iteration

DO $\leqslant$ **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ **od; while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ **od** (*) by induction rule again

Claim: for **n**,**k**$< \omega$ there exists **l**$< \omega$ such that

while $\mathbf{B}_1$ **do** $\mathbf{S'}$ $\mathbf{od}^n$; **while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ $\mathbf{od}^k \leqslant \mathbf{DO}^l$.

Proof of claim: by induction on **n**:

while $\mathbf{B}_1$ **do** $\mathbf{S'}$ $\mathbf{od}^{n+1}$; **while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ $\mathbf{od}^k$

$\approx$ **if** $\mathbf{B}_1$ **then** $\mathbf{S'}$; **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ $\mathbf{od}^n$ **fi**; $\mathbf{DO}^k$

Consider cases on $\mathbf{B}_1$.

So **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ $\mathbf{od}^n$; **while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ $\mathbf{od}^k \leqslant$ **DO** for all **n**,**k**$< \omega$.

So **while** $\mathbf{B}_1$ **do** $\mathbf{S'}$ **od; while** $\mathbf{B}_2$ **do** $\mathbf{S'}$ **od** $\leqslant$ **DO** by the general induction rule for loops.

Combining this with (*) above proves the Lemma.

Loop elimination:

If **S** does not assign to any variables of a term **m** and $\bigvee_{n<\omega}\mathbf{m}=\mathbf{s}_n$ (with $\mathbf{s}_n$ as above) then:

$\{\mathbf{b}\leqslant\mathbf{m}\leqslant\mathbf{f} \wedge (\mathbf{B} \iff \mathbf{i}=\mathbf{m})\} \cup \Delta \vdash$ **for i:=b by s to f do if B then S fi od** $\leqslant$ **S[m/i]**

Proof: Let **FOR**=**for i:=b step s to f do S od**.

From the last transformation we get:

FOR $\approx$ **begin i:=b: while i**$\leqslant$**m do if i=m then S fi; i:=i+s od; {i>m};**

while i$\leqslant$**f do if i=m then S fi; i:=i+s od** **end**

From the last transformation again: (since **m**$-$**s**$<$**m**)

FOR $\approx$ **begin i:=b: while i**$\leqslant$**m**$-$**s do {i<m}; if i=m then S fi; i:=i+s od;**

while i$\leqslant$**m do if i=m then S fi; i:=i+s od; {i>m};**

while i$\leqslant$**f do if i=m then S fi; i:=i+s od** **end**

≈ **begin i:=b: while i⩽m−s do i:=i+s; {i⩽m} od; {m−s<i⩽m};**
while i⩽m do if i=m then S fi; i:=i+s od; {i>m};
while i⩽f do if i=m then S fi; i:=i+s od end

The first loop must terminate since **FOR** terminates. The invariant $\bigvee_{k<\omega}\mathbf{i}=\mathbf{s}_k$ is set up and maintained over the first loop so after the first loop we have:

$\mathbf{s}_{n-1} < \mathbf{s}_k \leqslant \mathbf{s}_n$

So $\mathbf{s}_{n-1} < \mathbf{s}_k \leqslant \mathbf{s}_n$ so $(\mathbf{n}{-}\mathbf{1}) < \mathbf{k} \leqslant \mathbf{n}$ so **k=n** ie **i=m**.

Also the condition **i>m** is true at the beginning of the third loop and is maintained by it so we can move it inside. We get:

≈ **begin i:=m: while i⩽m do if i=m then S fi; i:=i+s od;**
while i⩽f do {i>m}; if i=m then S fi; i:=i+s od end

≈ **begin i:=m: if i⩽m then if i=m then S fi; i:=i+s; {i>m};**
while i⩽m do if i=m then S fi; i:=i+s od; (by loop unrolling)
while i⩽f do {i>m}; i:=i+s od end

≈ **begin i:=m: S;i:=i+s**
while i⩽f do i:=i+s od end

≈ **begin i:=m: S end** by dead variable elimination (the loop must terminate).

≈ **S[m/i]** by subsumption.