Chapter 8

Initial variables

We met initial variables briefly in Chapter 5, where they were necessary to specify the decrease of variants. In this chapter we study them further, presenting generalisations of earlier laws and definitions in order to take initial variables fully into account.

8.1 Simple specifications

We have seen that both initial variables and logical constants can be used to refer in a postcondition to a value based on the initial (rather than the final) state. Just which one is used in any particular situation is a matter of taste: the three commands

$$\begin{array}{ll} & x := x + 1 \;, \\ & x\colon [x = x_0 + 1] \\ \text{and} & [\![\ \textbf{con}\ X \ \cdot\ x\colon [x = X\ ,\ x = X + 1]\ ]\!] \end{array}$$

all increment x by 1. They are equal as programs.

That the first two are equal is a consequence of this abbreviation, relating assignments and specifications of a simple kind:

Abbreviation 8.1 <u>simple specification</u> For any relation $\odot$,

$$w\colon \odot E \quad = \quad w\colon [w \odot E_0]\,,$$

where E_0 is $E[w \backslash w_0]$.
□

As a special case we have that

$$w := E \quad = \quad w\colon [w = E_0] \;\;,$$

which explains the equality of $x := x + 1$ and x: $[x = x_0 + 1]$. A further specialisation, requiring E to contain no w, returns us to our earlier *simple specification* 1.7.

But there are many other uses of the idiom: the command $x\!: > x$, for example, increases x strictly. In the context of the declaration $n : \mathbb{N}$, the command $n\!: < n$ decreases n strictly, but not below 0. Abbreviation 8.1 allows us to write such assignments in abstract programs without losing the opportunity of refining them subsequently.

That latter example above is perhaps slightly surprising, and we should look at precisely how the declaration ensures that n remains non-negative. The simple answer is that in the context $n \in \mathbb{N}$ *nothing* can make n negative, not even the assignment $n := -1$. (Recall the discussion in Section 6.6.) Thus if we refined $n\!: < n$ to $n := n - 1$ in the context $n : \mathbb{N}$ — and it *is* a refinement — the assignment $n := n - 1$ would be miraculous, and would fail the type-checking. Could we refine $n\!: < n$ to $n := n \ominus 1$, for which type-checking would succeed? We cannot, for after using *simple specification* 8.1 to produce n: $[n < n_0]$, we would by *assignment* 5.2 have to show

$$n \in \mathbb{N} \quad \Rrightarrow \quad n \ominus 1 < n \ .$$

That we cannot do, because we do not have in particular that $0 \ominus 1 < 0$.

8.2 Initial variables precisely

The second equality of the previous section is a consequence of this abbreviation, which by using **con** makes our 0-subscript convention precise:

Abbreviation 8.2 <u>initial variable</u> Occurrences of 0-subscripted variables in the postcondition of a specification refer to values held by those variables in the *initial* state. Let x be any variable, probably occurring in the frame w. If X is a fresh name, and T is the type of x, then

$$\begin{array}{ll} & w\!: [pre\ ,\ post] \\ \widehat{=} & |[\ \textbf{con}\ X : T \bullet w\!: [pre \wedge x = X\ ,\ post[x_0 \backslash X]]\ ]|\ . \end{array}$$

□

The frame of a specification has so far been our only reference to initial values: those *not* in the frame are preserved. Now that initial variables allow us to be more explicit, we can give laws for manipulating the frame:

Law 8.3 <u>expand frame</u>

$$w\!: [pre\ ,\ post] \quad = \quad w, x\!: [pre\ ,\ post \wedge x = x_0]\ .$$

□

Note that Law 8.3 is an *equality*: the refinement goes both ways. The conjunct $x = x_0$ in the postcondition prevents x from changing; and so does omitting it from the frame. With Law 8.3 we can prove our earlier *contract frame* 5.4, as follows:

$$
\begin{array}{ll}
 & w, x\colon [pre\ ,\ post] \\
\sqsubseteq & \text{“\textit{strengthen postcondition} 5.1”} \\
 & w, x\colon [pre\ ,\ post \wedge x = x_0] \\
= & w, x\colon [pre\ ,\ post[x_0 \backslash x] \wedge x = x_0] \\
= & \text{“\textit{expand frame} 8.3 in reverse”} \\
 & w\colon [pre\ ,\ post[x_0 \backslash x]]\ .
\end{array}
$$

8.3 Sequential composition revisited

Initial variables bring also a danger: there are some laws in which they must be explicitly banned. Consider the following *incorrect* use of *sequential composition* 3.3:

$$
\begin{array}{lll}
 & x\colon [x = x_0 + 1] & \text{(i)} \\
\sqsubseteq ? & \text{“\textit{sequential composition} 3.3”} & \\
 & x\colon [x = 0]\,; & \\
 & x\colon [x = 0\ ,\ x = x_0 + 1] & \text{(ii)} \\
\sqsubseteq & x := 0;\ \ x := 1\ . &
\end{array}
$$

It is incorrect because at (i) the initial variable x_0 refers to the initial value of x, while at (ii) it has come to refer to the *intermediate* value of x, between the two commands.

That is why Law 3.3 carried a warning footnote: it may not be used when initial variables occur in *mid* or *post*.

The correct law for sequential composition, when dealing with initial variables, is the following:

Law 8.4 <u>sequential composition</u> For fresh constants X,

$$
\begin{array}{ll}
 & w, x\colon [pre\ ,\ post] \\
\sqsubseteq & \mathbf{con}\ X \bullet \\
 & x\colon [pre\ ,\ mid]\,; \\
 & w, x\colon [mid[x_0 \backslash X]\ ,\ post[x_0 \backslash X]]\,.
\end{array}
$$

The formula *mid* must not contain initial variables other than x_0.
□

Law 8.4 is considerably more complicated than *sequential composition* 3.3, and so should be reserved for cases in which its extra power is essential. Other alternatives are *sequential composition* B.2 (probably the most appropriate in general), *leading assignment* 8.5 and *following assignment* 3.5 (good for specific cases).

$$
\begin{array}{ll}
 & f, n\colon [n = n_0 + 1 \wedge f = f_0 \times n] \\
\sqsubseteq & \mathbf{con}\ N \bullet \\
 & n\colon [n = n_0 + 1]\,; \\
 & f, n\colon [n = N + 1\ ,\ n = N + 1 \wedge f = f_0 \times n] \\
\sqsubseteq & n := n + 1; \\
 & f := f \times n\ .
\end{array}
$$

Figure 8.1 Sequential composition with initial variables

The constraint of Law 8.4 ensures that $mid[x_0 \backslash X]$ contains no initial variables at all: they would not be meaningful in a precondition. (But see Exercise 8.2.) Figure 8.1 gives an example of using Law 8.4.

Returning to the example that began this section, using Law 8.4 we have

$$
\begin{array}{ll}
 & x\colon [x = x_0 + 1] \\
\sqsubseteq & \mathbf{con}\ X \bullet \\
 & x\colon [x = 0]\,; \\
 & x\colon [x = 0\ ,\ x = X + 1]\ .
\end{array}
$$

No longer can the second command be refined to $x := 1$. In fact, it cannot refine to *any* code, since the logical constant X cannot be eliminated from it. But that is not surprising: we do not expect to increment x by first setting it to 0.

For more examples of *sequential composition* 8.4 see Exercises 5.5 and 8.1.

8.4 Leading assignment

As an example of the use of the fuller form of sequential composition, we give here a law complementary to *following assignment* 3.5; now the assignment comes before, rather than after, the specification:

Law 8.5 <u>leading assignment</u> For any expression E,

$$
\begin{array}{ll}
 & w, x\colon [pre[x \backslash E]\ ,\ post[x_0 \backslash E_0]] \\
\sqsubseteq & x := E; \\
 & w, x\colon [pre\ ,\ post]\ .
\end{array}
$$

The expression E_0 abbreviates $E[w, x \backslash w_0, x_0]$.
□

With *sequential composition* 8.4, we can prove Law 8.5 as follows:

$$
\begin{array}{ll}
 & w, x\colon [pre[x\backslash E]\ ,\ post[x_0\backslash E_0]] \\
\sqsubseteq & \text{“definition of } E_0\text{”} \\
 & w, x\colon [pre[x\backslash E]\ ,\ post[x_0\backslash E[w, x\backslash w_0, x_0]]] \\
\sqsubseteq & \text{“\textit{sequential composition} 8.4”, } \mathbf{con}\ X \bullet \\
 & x\colon [pre[x\backslash E]\ ,\ pre \wedge x = E[x\backslash x_0]]\,; \qquad \text{(i)} \\
 & w, x\colon [pre \wedge x = E[x\backslash x_0][x_0\backslash X]\ , \\
 & \qquad post[x_0\backslash E[w, x\backslash w_0, x_0]][x_0\backslash X]] \qquad \text{(ii)} \\
\text{(i)}\ \sqsubseteq & x := E \\
\text{(ii)}\ \sqsubseteq & w, x\colon [pre \wedge x = E[x\backslash X]\ ,\ post[x_0\backslash E[w, x\backslash w_0, X]]] \\
\sqsubseteq & \text{“\textit{strengthen postcondition} 5.1, using } x_0 = E[x\backslash X][w_0\backslash w] \text{ from precondition”} \\
 & w, x\colon [pre \wedge x = E[x\backslash X]\ ,\ post] \\
\sqsubseteq & w, x\colon [pre\ ,\ post]\ .
\end{array}
$$

Our earlier version *leading assignment* 3.6 is a special case of the above, where we start with a simple specification derived from an assignment.

8.5 Exercises

Ex. 8.1 Assuming $x : \mathbb{R}$, prove this:

$$
\begin{array}{ll}
 & x\colon \left[x > 0\ ,\ x = 1/\sqrt{x_0}\right] \\
\sqsubseteq & x\colon \left[x \geq 0\ ,\ x = \sqrt{x_0}\right]; \\
 & x\colon [x \neq 0\ ,\ x = 1/x_0]\ .
\end{array}
$$

Hint: Use a stronger *mid* than $x = \sqrt{x_0}$.

Ex. 8.2 ♡ Suppose a specification $w\colon [pre\ ,\ post]$ refers to x_0 in the postcondition even though x is not in the frame. Why is that unnecessary? Use *expand frame* 8.3 to show that it is equal to $w\colon [pre\ ,\ post[x_0\backslash x]]$.

Ex. 8.3 The abbreviation *initial variable* 8.2 gives us this alternative to *expand frame* 8.3:

> *Law 8.6* <u>expand frame</u> For fresh constant X,
>
> $$
> \begin{array}{ll}
> & w\colon [pre\ ,\ post] \\
> \sqsubseteq & \mathbf{con}\ X \bullet \\
> & w, x\colon [pre \wedge x = X\ ,\ post \wedge x = X]\,.
> \end{array}
> $$
>
> □

Use *expand frame* 8.6 to show that

skip

$$\begin{array}{l} \sqsubseteq \ \mathbf{con}\ N \bullet \\ \quad n := n+1;\ \ n := n-1\ . \end{array}$$

Assume $n : \mathbb{N}$. *Hint*: Recall the formulation of **skip** on p.13.

Ex. 8.4 ♡ Prove these equalities:

$$\begin{array}{ll} \ w\colon [pre\ ,\ post] & \text{(i)} \\ =\ w\colon [pre\ ,\ (\exists\, w \bullet pre) \Rightarrow post] & \text{(ii)} \\ =\ w\colon [pre\ ,\ (\exists\, w \bullet pre) \wedge post]\ . & \text{(iii)} \end{array}$$

Hint: To prove equality, show (i) $\sqsubseteq$ (ii) $\sqsubseteq$ (iii) $\sqsubseteq$ (i).

Ex. 8.5 ♡ Write down the law resulting from *sequential composition* 8.4 in the special case that *mid* and *post* contain no x_0. In what way is the result more general than *sequential composition* 3.3?

Ex. 8.6 Assuming $n : \mathbb{N}$, use Definition 6.5 to show that the specification $n\colon [n \neq 0\ ,\ n < n_0]$ is feasible.

Ex. 8.7 ♡ Show for any frame w that

$$w\colon [\mathsf{true}\ ,\ \mathsf{false}] \quad = \quad :[\mathsf{true}\ ,\ \mathsf{false}]\ ,$$

and hence that **magic** need not mention its frame. *Hint*: Use *expand frame* 8.3, and recall the hint of Exercise 8.4.

Ex. 8.8 Repeat Exercise 8.7 for **abort**.

Ex. 8.9 Repeat Exercise 3.8, this time taking initial variables into account.