Programming Languages: Imperative Program Construction 3. Quantifications

Shin-Cheng Mu

Autumn Term, 2022

Materials of this part of the course are adapted from Gries and Schneider [GS93]. Axioms and theorems are numbered according to the book. Some additional rules are taken from Kaldewaij [Kal90].

1 Synatx and Interpretation of Quantification

Summation, Dummy Variables

- We have all seen *quantifed* expressions like this: $\sum_{i=1}^{n} e$,
  - which denotes $e[i\backslash 1] + e[i\backslash 2] + \ldots e[i\backslash n]$.
  - Example: $\sum_{i=1}^{3} i^2 = 1^2 + 2^2 + 3^2$.
- Note that the variable i is a *dummy variable* (虛擬變數). It is different from an ordinary variable — its value is not drawn from the state. Instead it is a "local" variable.
- Name of the dummy variable does not matter. E.g. $\sum_{i=1}^{3} i^2 = \sum_{j=1}^{3} j^2$
- The usual notation for quantifiers is confusing at times, however.
  - $1 + \sum_{n=1}^{3} n^2$ is 15.
  - Should $\sum_{n=1}^{3} n^2 + 1$ be 15 or 17?
  - Should $\sum_{n=1}^{3} 1 + n^2$ be 17 or $3 + n^2$?
- Similar problems occurs in conventional notations for logic.
  - It is sometimes not clear whether $\forall x.P\ x \wedge Q$ denotes $(\forall x.P\ x) \wedge Q$ or $(\forall x.P\ x \wedge Q)$.

A Linear Notation

Instead of $\sum_{i=1}^{n} e$, we use a linear notation:

$$\langle\, \Sigma i : 1 \leqslant i \leqslant n : e \,\rangle$$

for several reasons:

- it is clearer that Σi declares a dummy variable i.
- The parentheses makes the *scope* of i clear.
- You can write more general ranges:
  - $\langle\, \Sigma i : 1 \leqslant i \leqslant 7 \wedge \mathit{even}\ i : i \,\rangle = 2 + 4 + 6$,
  - $\langle\, \Sigma i : 1 \leqslant i \leqslant 7 \wedge \mathit{odd}\ i : 2 \times i \,\rangle = 2 \times 1 + 2 \times 3 + 2 \times 5 + 2 \times 7$.
- And it extends easily to more variables:
  - $\langle\, \Sigma i, j : 1 \leqslant i \leqslant 2 \wedge 3 \leqslant j \leqslant 4 : i^j \,\rangle = 1^3 + 1^4 + 2^3 + 2^4$.

This notation is sometimes called the *Eindhoven notation*, named after the university where many advocates came from.

A review of this choice of notation (alone with some others) was given by Dijkstra [Dij00].

1.1 A General Notation for Quantified Expressions

Generalizing to Other Operators

- Let $\star$ be any binary operator that is
  - **symmetric**: $b \star c = c \star b$, and
  - **associative**: $(b \star c) \star d = b \star (c \star d)$, and has an
  - **identity** u: $u \star b = b = b \star u$.

- We allow the general *quantification* (量詞, 量化句) over $\star$:

  $$\langle \star x, y : R : P \rangle$$

  It informally means "for all the x, y such that R is $True$, collect all the P and apply $\star$ to them.".

  - Variables x and y are distinct. They are called the *bound variables*, or the *dummies*, of the quantification. There may be one or more dummies.
  - Note that x and y should be restricted by their types. For this course, we assume that their types can be inferred by the context.
  - R: an boolean expression, the *range* of the quantification. When it is omitted, as in $\langle \star x :: P \rangle$, we mean $R = True$.
  - P: an expression, the *body* of the quantification. The type of the result of the quantification is the type of P.

Examples

$$\langle +i : 0 \leqslant i < 4 : i \times 8 \rangle =$$
$$0 \times 8 + 1 \times 8 + 2 \times 8 + 3 \times 8$$
$$\langle \times i : 0 \leqslant i < 3 : i + (i+1) \rangle =$$
$$(0+1) \times (1+2) \times (2+3)$$
$$\langle \wedge i : 0 \leqslant i < 2 : i \times d \neq 6 \rangle =$$
$$0 \times d \neq 6 \wedge 1 \times d \neq 6$$
$$\langle \vee i : 0 \leqslant i < 21 : b[i] = 0 \rangle =$$
$$b[0] = 0 \vee \ldots \vee b[20] = 0$$

Conventions

To relate to more familiar symbols, we bow to the convention and write

$$\langle +x : R : P \rangle \quad \text{as} \quad \langle \Sigma x : R : P \rangle$$
$$\langle \times x : R : P \rangle \quad \text{as} \quad \langle \Pi x : R : P \rangle$$
$$\langle \vee x : R : P \rangle \quad \text{as} \quad \langle \exists x : R : P \rangle$$
$$\langle \wedge x : R : P \rangle \quad \text{as} \quad \langle \forall x : R : P \rangle$$

1.2 Scope

Free v.s. Bound Variables

- Consider $\langle \forall i :: x \times i = 0 \rangle$, which asserts that x multiplied by any integer equals 0. The value of this expression depends on x, which is in the state, but not i: writing $\langle \forall j :: x \times j = 0 \rangle$ means the same thing.
- Occurrences of x in such expression are said to be *free*.
  - All occurrences of a variable in an expression without quantifications are free. For example, all variables in expressions in Chapter 3 are free.
- The scope of the dummy i is the range and body of the expression.
- Occurrences of i in the scope are said to be *bound*.

Free v.s. Bound Occurrences

- Note that being free or bound is not a property of not variables, but *occurrences* of variables.
- In $i > 0 \vee \langle \forall i : 0 \leqslant i : x \times i = 0 \rangle$, the leftmost occurrence of i (in $i > 0$) is free.
- The variable i is used in two different ways. The first (i.e. free) occurrence of i refer to a different variable than the other (i.e. bound) occurrences.
- The expression is equivalent to $i > 0 \vee \langle \forall j : 0 \leqslant j : x \times j = 0 \rangle$.
- Similar to local variables in programming languages. Algol 60 (in 1960) was the first programming language to introduce full scoping rules for local variables, while such scoping rules were used in logic dating back to 1885, by Charles Sanders Peirce.

Free Occurrences, Formally

Formal definitions of free and bound occurrences are rather tedious. Let us try.

(8.9) **Definition**

1. The occurrence of i in the expression i is free.
2. If an occurrence of i is free in E, the same occurrence is also free in (E), in $f(\ldots, E, \ldots)$, and in $\langle \star x : E : F \rangle$ and in $\langle \star x : F : E \rangle$ if i is not x.

(8.9)' **Definition** $occurs(v, e)$ is $True$ iff. v occurs free at least once in e.

In general, both v and e could be sets. In that case, $occurs(v, e)$ means at least one variable in v occurs free at least once in e.

Bound Occurrences, Formally

(8.10) **Definition**

1. Let an occurrence of i be free in E. That occurrence of i is *bound* in $\langle \star i : E : F \rangle$ or $\langle \star i : F : E \rangle$.

2. If an occurrence of i is bound in E, the same occurrence is also bound (to the same dummy) in (E), in $f(\ldots, E, \ldots)$, $\langle \star x : E : F \rangle$ and in $\langle \star x : F : E \rangle$.

Examples

Consider the equation:

$$i + j + \langle \Sigma i : 1 \leqslant i \leqslant 10 : b[i]^j \rangle + \langle \Sigma i : 1 \leqslant i \leqslant 10 : \langle \Sigma j : 1 \leqslant j \leqslant 10 : c[i,j] \rangle \rangle$$

1.3 Textual Substitution Revisited

(8.11) Provided that $\neg occurs(y, \{x, F\})$,

$$\langle \star y : R : P \rangle [x \backslash F] = \langle \star y : R[x \backslash F] : P[x \backslash F] \rangle$$

- The caveat means that if y occurs free in x or F, it has to be replaced by a fresh dummy variable (using (8.21)) before we can perform the substitution.

- In a sense, bound occurrences are "protected" from alien substitutions. Their names are replaced, and thus never touched by an alien substitution.

Examples

$$\langle \Sigma x : 1 \leqslant x \leqslant 2 : y \rangle [y \backslash y + z] = \langle \Sigma x : 1 \leqslant x \leqslant 2 : y + z \rangle$$
$$\langle \Sigma i : 0 \leqslant i < n : b[i] = n \rangle [n \backslash m] = \langle \Sigma i : 0 \leqslant i < m : b[i] = m \rangle$$
$$\langle \Sigma y : 0 \leqslant y < n : b[y] = n \rangle [n \backslash y] = \langle \Sigma j : 0 \leqslant j < y : b[j] = y \rangle$$
$$\langle \Sigma y : 0 \leqslant y < n : b[y] = n \rangle [y \backslash m] = \langle \Sigma j : 0 \leqslant j < n : b[j] = n \rangle$$

2 Rules About Quantification

- Since $x + x = 2 \times x$, we would expect this to be true:

$$\langle \Sigma x : R : x + x \rangle = \langle \Sigma x : R : 2 \times x \rangle$$

- However, the current Leibniz rule does not allow us to prove the quality. In the attempt below:

$$\frac{x + x = 2 \times x}{\langle \Sigma x : R : z \rangle [z \backslash x + x] = \langle \Sigma x : R : z \rangle [z \backslash 2 \times x]}$$

  - Since x is protected in (8.11), the conclusion simplifies to $\langle \Sigma y : \ldots : x + x \rangle = \langle \Sigma y : \ldots : 2 \times x \rangle$.

Additional Leibniz Rule

The following additional rules allow substitution of equals for equals in the range and body of a quantification:s

(8.12) **Leibniz**:

$$\frac{P = Q}{\langle \star x : E[z \backslash P] : S \rangle = \langle \star x : E[z \backslash Q] : S \rangle}$$

$$\frac{R \Rightarrow P = Q}{\langle \star x : R : E[z \backslash P] \rangle = \langle \star x : R : E[z \backslash Q] \rangle}$$

Defining Axioms

(8.13) **Axiom, Empty range** :
$\langle \star x : False : P \rangle = u$,
where u is the identity of $\star$

(8.14) **Axiom, One-point rule** :
$\langle \star x : x = E : P \rangle = P[x \backslash E]$,
provided that $\neg occurs(x, E)$

Example of one-point rule:

$$\langle \Sigma x : x = 3 : x^2 \rangle = 3^2$$

Distributivity

(8.15) **Axiom, Distributivity** :
$\langle \star x : R : P \rangle \star \langle \star x : R : Q \rangle = \langle \star x : R : P \star Q \rangle$,
provided that $P, Q : Bool$ or R is finite

Example of distributivity:

$$\langle \Sigma i : i^2 < 9 : i^2 \rangle + \langle \Sigma i : i^2 < 9 : i^3 \rangle = \langle \Sigma i : i^2 < 9 : i^2 + i^3 \rangle$$

For an non-example showing why R must be finite when the type of P and Q are not (e.g. $P, Q : Int$), consider:

$$
\begin{array}{ll}
& 0 \\
= & \langle\, \Sigma i : 0 \leqslant i : 0 \,\rangle \\
= & \quad \{\ 0 = i + (-i)\ \} \\
& \langle\, \Sigma i : 0 \leqslant i : i + (-i) \,\rangle \\
= & \quad \{\ \text{distributivity without restriction}\ \} \\
& \langle\, \Sigma i : 0 \leqslant i : i \,\rangle + \langle\, \Sigma i : 0 \leqslant i : -i \,\rangle
\end{array}
$$

The last line is undefined because both sums are infinite.

2.1 Range Split

(8.16) **Axiom, Range split** :
$\langle\, \star x : R \vee S : P \,\rangle = \langle\, \star x : R : P \,\rangle \star \langle\, \star x : S : P \,\rangle$,
provided $R \wedge S \equiv \mathit{False}$ and
$P : \mathit{Bool}$ or R and S are finite

The restriction that $R \wedge S \equiv \mathit{False}$ ensures that an operand that satisfies both R and S is not accumulated twice in the RHS.

For the more general case, we may add the repeated operands to the RHS:

(8.17) **Axiom, Range split** :
$\langle\, \star x : R \vee S : P \,\rangle \star \langle\, \star x : R \wedge S : P \,\rangle =$
$\langle\, \star x : R : P \,\rangle \star \langle\, \star x : S : P \,\rangle$,
provided $P : \mathit{Bool}$ or R and S are finite

If $\star$ is idempotent, that is $e \star e = e$ for all e, it does not matter how many times e is accumulated.

(8.18) **Axiom, Range split for idempotent $\star$** :
$\langle\, \star x : R \vee S : P \,\rangle =$
$\langle\, \star x : R : P \,\rangle \star \langle\, \star x : S : P \,\rangle$,
provided that $\star$ is idempotent

2.2 Manipulating Dummies

Nested quantifications with the same operator can be interchanged:

(8.19) **Axiom, Interchange of dummies** :
$\langle\, \star x : R : \langle\, \star y : Q : P \,\rangle \,\rangle =$
$\langle\, \star y : Q : \langle\, \star x : R : P \,\rangle \,\rangle$,
provided that $\star$ is idempotent, or
R and Q are finite,
$\neg \mathit{occurs}(y, R)$, and $\neg \mathit{occurs}(x, Q)$

How a single quantification over a list of dummies can be viewed as a nested quantification:

(8.20) **Axiom, Nesting** :
$\langle\, \star x, y : R \wedge Q : P \,\rangle = \langle\, \star x : R : \langle\, \star y : Q : P \,\rangle \,\rangle$,
provided $\neg \mathit{occurs}(y, R)$

A dummy can be replaced by a fresh dummy.

(8.21) **Axiom, Renaming** :
$\langle\, \star x : R : P \,\rangle = \langle\, \star y : R[x \backslash y] : P[x \backslash y] \,\rangle$,
provided $\neg \mathit{occurs}(y, \{R, P\})$

The restrictions with $\neg \mathit{occur}$ in axioms (8.19), (8.20), and (8.21) ensure that an expression that contains an occurrence of a dummy is not moved outside (or inside) the scope of that dummy.

A More General Renaming

- Consider $\langle\, \Sigma i : 2 \leqslant i \leqslant 10 : i^2 \,\rangle$.
- We may rewrite this expression so that the range starts at 0 instead of 2: $\langle\, \Sigma k : 0 \leqslant k \leqslant 8 : (k+2)^2 \,\rangle$.
- Note the relationship: $i = k + 2$, and $k = i - 2$.
- The second expression is $\langle\, \Sigma k : (2 \leqslant i \leqslant 10)[i \backslash k + 2] : (i^2)[i \backslash k + 2] \,\rangle$.

(8.22) **Change of dummy** :
$\langle\, \star x : R : P \,\rangle = \langle\, \star y : R[x \backslash f\ y] : P[x \backslash f\ y] \,\rangle$,
provided $\neg \mathit{occurs}(y, \{R, P\})$,
and f has an inverse

- f has an inverse: $x = f\ y \equiv y = f^{-1}\ x$.

Proving (8.22)

Proof.

$$
\begin{array}{ll}
& \langle \star y : R[x\backslash f\ y] : P[x\backslash f\ y] \rangle \\
= & \quad \{ \text{ one-point rule (8.14) } \} \\
& \langle \star y : R[x\backslash f\ y] : \langle \star x : x = f\ y : P \rangle \rangle \\
= & \quad \{ \text{ nesting (8.20), } \neg occurs(x, R[x\backslash f\ y]) \ \} \\
& \langle \star x, y : R[x\backslash f\ y] \wedge (x = f\ y) : P \rangle \\
= & \quad \{ \text{ (3.84a) } \} \\
& \langle \star x, y : R[x\backslash x] \wedge (x = f\ y) : P \rangle \\
= & \quad \{ \text{ since } R[x\backslash x] = R \ \} \\
& \langle \star x, y : R \wedge (x = f\ y) : P \rangle \\
= & \quad \{ \text{ nesting (8.20), } \neg occurs(y, R) \ \} \\
& \langle \star x : R : \langle \star y : x = f\ y : P \rangle \rangle \\
= & \quad \{ \text{ assumption: } x = f\ y \equiv y = f^{-1}\ x \ \} \\
& \langle \star x : R : \langle \star y : y = f^{-1}\ x : P \rangle \rangle \\
= & \quad \{ \text{ one-point rule (8.14) } \} \\
& \langle \star x : R : P[y\backslash f^{-1}\ x] \rangle \\
= & \quad \{ \text{ since } \neg occurs(y, P) \ \} \\
& \langle \star x : R : P \rangle
\end{array}
$$

□

The proof is not as hard as it may appear at first, because each step is almost *forced* by the shape of the expression at that point and the shape of the final goal. In several steps there is only one choice.

1. The first step is the hardest. In retrospect, it is the *only* rule that can be applied.

2. Step 2: moving dummy x to the outside gets us closer to the final form.

3. $R[x\backslash f\ y]$ must be removed at some point. The substitution in step 3 and 4 makes it possible.

4. After step 5 we get a quantification in x alone.

5. The quantification in y can only be removed using the one-point rule. To prepare for that we need $y = f^{-1}\ x$.

3 Rules for Specific Operators

Existential Quantification

Trading :

$$\langle \exists i : R \wedge S : P \rangle = \langle \exists i : R : S \wedge P \rangle$$

Distributivity :

$$Q \wedge \langle \exists i : R : S \rangle = \langle \exists i : R : Q \wedge S \rangle \ ,$$

provided $\neg occurs(i, Q)$

$$Q \vee \langle \exists i : R : S \rangle = \langle \exists i : R : Q \vee S \rangle \ ,$$

provided $\neg occurs(i, Q)$ and R non-empty

Universal Quantification

Trading :

$$\langle \forall i : R \wedge S : P \rangle = \langle \forall i : R : S \Rightarrow P \rangle$$

Distributivity :

$$Q \vee \langle \forall i : R : S \rangle = \langle \forall i : R : Q \vee S \rangle \ ,$$

provided $\neg occurs(i, Q)$

$$Q \wedge \langle \forall i : R : S \rangle = \langle \forall i : R : Q \wedge S \rangle \ ,$$

provided $\neg occurs(i, Q)$ and R non-empty

de Morgan :

$$\neg \langle \exists i : R : S \rangle = \langle \forall i : R : \neg S \rangle$$

Minimum and Maximum

More distributivity. Provided that $\neg occurs(i, F)$:

$$F \downarrow \langle \uparrow i : R : S \rangle = \langle \uparrow i : R : F \downarrow S \rangle$$
$$F \uparrow \langle \downarrow i : R : S \rangle = \langle \downarrow i : R : F \uparrow S \rangle$$

Provided that $\neg occurs(i, F)$ and R non-empty:

$$F + \langle \uparrow i : R : S \rangle = \langle \uparrow i : R : F + S \rangle$$
$$F + \langle \downarrow i : R : S \rangle = \langle \downarrow i : R : F + S \rangle$$

For $F \geqslant 0$, $\neg occurs(i, F)$ and R non-empty:

$$F \times \langle \uparrow i : R : S \rangle = \langle \uparrow i : R : F \times S \rangle$$
$$F \times \langle \downarrow i : R : S \rangle = \langle \downarrow i : R : F \times S \rangle$$
$$-\langle \uparrow i : R : S \rangle = \langle \downarrow i : R : -S \rangle$$

Upper/Lower Bounds

Least upper bound and greatest lower bound:

$$S\ x = \langle \uparrow i : R\ i : S\ i \rangle \equiv$$
$$R\ x \wedge \langle \forall i : R\ i : S\ i \leqslant S\ x \rangle$$
$$S\ x = \langle \uparrow i : R\ i : S\ i \rangle \equiv$$
$$R\ x \wedge \langle \forall i : R\ i : S\ i \leqslant S\ x \rangle$$

Number Of ...

Let $\langle\, \#i : R\, i : S\, i \,\rangle$ denote "the number of i in range $R\, i$ such that $S\, i$ is true".

Definition

1. Function $\# : \mathit{Bool} \to \mathit{Int}$ is defined by $\#\, \mathit{False} = 0$ and $\#\, \mathit{True} = 1$.

2. $\langle\, \#i : R\, i : S\, i \,\rangle$ = $\langle\, \Sigma i : R\, i : \#(S\, i) \,\rangle$.

4 Manipulating Ranges

(8.23) **Theorem, Split off term** : for $n : \mathit{Nat}$,

(a) $\langle\, \star i : 0 \leqslant i < n+1 : P \,\rangle$
$= \langle\, \star i : 0 \leqslant i < n : P \,\rangle \star P[i \backslash n]$

(b) $\langle\, \star i : 0 \leqslant i < n+1 : P \,\rangle$
$= P[i \backslash 0] \star \langle\, \star i : 0 < i < n+1 : P \,\rangle$

Important: notice that by $n : \mathit{Nat}$ we are assuming that $0 \leqslant n$, therefore the range $0 \leqslant i < n+1$ is never empty.

There is a more general variation that is less used in this course:

(8.23)′ **Theorem, Split off term** :
for $m, n : \mathit{Nat}$ such that $m \leqslant n$,

(a) $\langle\, \star i : m \leqslant i < n+1 : P \,\rangle$
$= \langle\, \star i : m \leqslant i < n : P \,\rangle \star P[i \backslash n]$

(b) $\langle\, \star i : m \leqslant i < n+1 : P \,\rangle$
$= P[i \backslash m] \star \langle\, \star i : m < i < n+1 : P \,\rangle$

Proof of (8.23a)

Proof.

$$
\begin{array}{ll}
 & \langle\, \star i : 0 \leqslant i < n+1 : P \,\rangle \\
= & \quad \{\ 0 \leqslant i < n+1 \equiv 0 \leqslant i < n \vee i = n\ \} \\
 & \langle\, \star i : 0 \leqslant i < n \vee i = n : P \,\rangle \\
= & \quad \{\ \text{range split (8.16),} \\
 & \qquad \text{since } 0 \leqslant i < n \wedge i = n \equiv \mathit{False}\ \} \\
 & \langle\, \star i : 0 \leqslant i < n : P \,\rangle \star \langle\, \star i : i = n : P \,\rangle \\
= & \quad \{\ \text{one-point rule (8.14)}\ \} \\
 & \langle\, \star i : 0 \leqslant i < n : P \,\rangle \star P[i \backslash n]
\end{array}
$$

□

An Assumed Property about Arithmetics

In the proof of (8.23a) we used the following theorem regarding natural numbers:

(8.24)
$$b \leqslant c \leqslant d \;\Rightarrow\; (b \leqslant i < d \;\equiv\; b \leqslant i < c \vee c \leqslant i < d)$$

In a course on discrete mathematics, such properties are justified by axioms for arithmetics (see Chapter 15 of Gries and Schneider [GS93].) For this course, we just take them as granted.

Examples

Let $0 \leqslant n$.

$$
\begin{array}{l}
\langle\, \Sigma i : 0 \leqslant i < n+1 : b[i] \,\rangle = \\
\qquad \langle\, \Sigma i : 0 \leqslant i < n : b[i] \,\rangle + b[n] \\
\langle\, \Pi i : 0 \leqslant i < n+1 : b[i] \,\rangle = \\
\qquad b[0] \times \langle\, \Pi i : 0 < i < n+1 : b[i] \,\rangle \\
\langle\, \forall i : 0 \leqslant i \leqslant n : b[i] = 0 \,\rangle = \\
\qquad \langle\, \forall i : 0 \leqslant i < n : b[i] = 0 \,\rangle \wedge b[n] = 0 \\
\langle\, \Pi i : 0 \leqslant i \leqslant n : b[i] \,\rangle = \\
\qquad b[0] \times \langle\, \Pi i : 0 < i \leqslant n : b[i] \,\rangle
\end{array}
$$

Example: Sum of a Triangular Array

Let $0 \leqslant n$. Consider the following expression:

(8.25) $\langle\, \Sigma i, j : 0 \leqslant i \leqslant j < n+1 : c[i,j] \,\rangle$

It is the sum of a triangular portion of an array.

We will show that it equals

$$
\begin{array}{l}
\langle\, \Sigma i, j : 0 \leqslant i \leqslant j < n : c[i,j] \,\rangle + \\
\qquad \langle\, \Sigma i : 0 \leqslant i \leqslant n : c[i,n] \,\rangle
\end{array}
$$

That is, we can compute the sum of the last row and the sum of the rest of the triangle, before adding them.

To Split the Range ...

...we note that $\leqslant$ and $<$ is used conjunctively. That is, $a \leqslant b < c$ is an abbreviation of $a \leqslant b$ and $b < c$.

We reason:

$$
\begin{array}{ll}
 & 0 \leqslant i \leqslant j < n+1 \\
= & \{ \text{ remove abbreivation } \} \\
 & 0 \leqslant i \leqslant j \wedge j < n+1 \\
= & \{ \ j < n+1 \equiv j < n \vee j = n \ \} \\
 & 0 \leqslant i \leqslant j \wedge (j < n \vee j = n) \\
= & \{ \text{ distributivity (3.46) } \} \\
 & (0 \leqslant i \leqslant j \wedge j < n) \vee (0 \leqslant i \leqslant j \wedge j = n) \\
= & \{ \text{ use abbreivation } \} \\
 & (0 \leqslant i \leqslant j < n) \vee (0 \leqslant i \leqslant j \wedge j = n)
\end{array}
$$

The Calculation

We can now manipulate (8.25):

$$
\begin{array}{ll}
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n+1 : c[i,j] \, \rangle \\
= & \{ \text{ the proof above } \} \\
 & \langle \, \Sigma i, j : (0 \leqslant i \leqslant j < n) \vee (0 \leqslant i \leqslant j \wedge j = n) : c[i,j] \, \rangle \\
= & \{ \text{ range split (8.16) } \} \\
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n : c[i,j] \, \rangle + \\
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j \wedge j = n : c[i,j] \, \rangle \\
= & \{ \text{ nesting (8.20) } \} \\
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n : c[i,j] \, \rangle + \\
 & \langle \, \Sigma j : j = n : \langle \, \Sigma i : 0 \leqslant i \leqslant j : c[i,j] \, \rangle \, \rangle \\
= & \{ \text{ one-point rule (8.14) } \} \\
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n : c[i,j] \, \rangle + \\
 & \langle \, \Sigma i : 0 \leqslant i \leqslant n : c[i,n] \, \rangle
\end{array}
$$

The calculation looks tedious, but is familiar to people in this field, and can be considered trivial. In practice, such manipulation is usually quickly condensed in one step:

$$
\begin{array}{ll}
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n+1 : c[i,j] \, \rangle \\
= & \{ \text{ range split (8.16); one-point rule (8.14) } \} \\
 & \langle \, \Sigma i, j : 0 \leqslant i \leqslant j < n : c[i,j] \, \rangle + \\
 & \langle \, \Sigma i : 0 \leqslant i \leqslant n : c[i,n] \, \rangle
\end{array}
$$

References

[Dij00] E. W. Dijkstra. The notational conventions I adopted, and why. EWD 1300, 2000.

[GS93] D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Springer, October 22, 1993.

[Kal90] A. Kaldewaij. *Programming: the Derivation of Algorithms*. Prentice Hall, 1990.