Program construction by refinements preserving correctness

G. A. Lanzarone* and M. Ornaghi

Gruppo di Elettronica e Cibernetica, Istituto di Fisica dell'Università di Milano, Via Viotti, 5-20133 Milano, Italy

This paper deals with the problem of constructing the final version P^t of a flowchart program through successive refinements $P^2, \ldots, P^{t-1}$ that preserve correctness proved on its first version P^1.

Correctness conditions are associated with P^1 in the frame of Manna's formalism. With each refinement P^i a relational (data) structure $\mathscr{S}^i$ is associated, and given representation functions τ_i relate structures $\mathscr{S}^i$, $\mathscr{S}^{i+1}$ of refinements P^i, P^{i+1}.

Construction of P^{i+1} from P^i proceeds as follows: (a) every block of P^i is considered as an elementary program over $\mathscr{S}^i$ and its correctness conditions are expressed with terms of $\mathscr{S}^i$; (b) these correctness conditions are translated by using the representation function τ_i; (c) the translated correctness conditions are transformed into expansion conditions, expressed with terms of $\mathscr{S}^{i+1}$ only, for every block of P^i; (d) by means of these expansion conditions, expansions of blocks are constructed and connected to obtain P^{i+1}.

The constructive character of the above process is emphasised with a detailed example.

In the appendix a discussion relates this paper to other works connecting constructivism and program theory.

(Received July 1973)

Writing a program to meet some specific demands, one is faced with two types of problems: (1) how to build such a program; (2) how to guarantee that the program provides the required features.

With regard to Point 1, the most recent trend (Dijkstra, 1968*a*; Mills, 1971; Wirth, 1971; Woodger, 1971; Dahl, Dijkstra and Hoare, 1972) is to consider the construction of a program as a process of successive approximations, by means of a sequence of programs $P^1, P^2, \ldots, P^t$, where P^1 is an outline of the solution of the problem in hand, $P^2, \ldots, P^{t-1}$ are progressive refinements of P^1 and P^t is the final version of P^1 in the chosen programming language.

As for Point 2, the approach to the problem has been to express the behaviour of a given program by suitable conditions (called correctness conditions), and to give some procedures which permit correctness verification by proving predicate calculus theorems (Floyd, 1967; Manna, 1969*a*; 1969*b*); for a more complete reference, see review (Elspas, *et al.*, 1972) and bibliography (London, 1970).

In this last approach, there is the difficulty of expressing the characteristics required of the program as correctness conditions, because of the discrepancy between the predicate calculus language and the programming language. Besides, the proofs are often cumbersome because the correctness conditions are imposed on the program when it is already written. A way out of these difficulties is to deal with the two types of problems mentioned above by making use of both program construction and program verification methods and by trying to unify them.

So far, only a few examples have been developed along these lines (Hull, Enright and Sedgwick, 1972; Jones, 1972).

The following situation appears when dealing with the problem of proving program correctness in parallel with the construction of the sequence of programs $P^1, P^2, \ldots, P^t$.

The correspondence between the characteristics required of the program and those designed into its first version P^1 is verified when proving the total correctness of P^1 (with respect to the input and output conditions which express the desired behaviour). P^1 must be rich enough to contain explicitly and precisely all the principal functions identified by the programmer as essential to an adequate solution of the problem. On the other hand, it mustn't contain details useless for this goal, to obtain better intelligibility and simplicity of verification, and not to put unnecessary restrictions on the successive refinements. During the construction of the final version P^t through the sequence $P^2, P^3, \ldots, P^{t-1}$, the proofs no longer aim at expressing and verifying the correctness of the successively written programs, but at verifying that each refinement maintains the original characteristics required of the program.

This is in fact the situation examined in the present paper, which treats a method of construction of the refinement P^{i+1} of a program P^i, maintaining the characteristics required and already verified in P^i.

The construction process takes into account the different features of each semantic level P^i, from the one closer to the problem and dealing with 'abstract' structures, i.e. structures which enjoy general properties, to the one expressed in a programming language, concerning itself with computer core management. The question is that of determining how to make such a refinement P^{i+1} from P^i with respect to the three aspects implied in every program: data, operations (functions and predicates), control flow.

The transition from data structures on which P^i is defined to that of P^{i+1} is made by means of a representation function that expresses the properties of such a refinement, i.e. the choices made by the programmer. On this basis, and with regard to the operations available at the successive level, each block of program P^i is expanded into a subprogram of P^{i+1}, and the conditions are given under which such expansion is made correctly, that is according to the representation function chosen.

The conditions of correct expansion therefore turn out to be useful in practice as a guide to refinement constructions, and the representation function acts as documentation of the choices made by the programmer in program construction.

It is then shown that, for each level to maintain the correctness proved at the first level with respect to assigned conditions, both conditions of correct expansion and interface conditions which express the retention of the connections between blocks existing at the first level, must be satisfied.

The present paper is a result of a joint research project sponsored by CNR under contract no. 71.02104/75, and by Honeywell Information Systems Italia (HISI) (AST project—Quality Assurance Service of Software Engineering Section—Pregnana Milanese).

*Honeywell Information Systems Italia (HISI).

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/445397 by guest on 19 April 2024

The representation function here defined for refinements is similar to the simulation relation between programs given in Milner (1971). However, the simulation relation (expressed in an algebraic formalism) serves there to break down the correctness proof of a given program A into the proof of correctness (assumed easier) of a program B also given, and the proof that B simulates A; in this paper, instead, the method is centred on showing what procedures must be followed in constructing step by step the final program so that it is automatically correct (with respect to the characteristics required and proved at the first level).

The method is presented by using Manna's formalisation of program properties in predicate-calculus. It is general in that it doesn't consider specific models of data structures (these will be treated in future work); a case of characterisation of semantic levels and their hierarchical relationships is given informally in Dijkstra, (1968*b*).

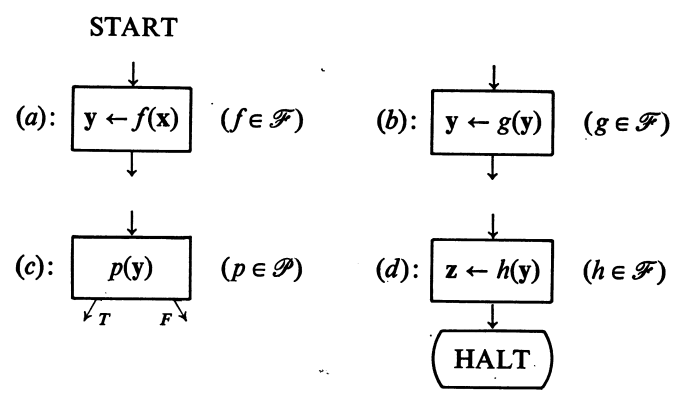
To illustrate the presented method, an example is shown and discussed at the end of this paper.

1. Introductory definitions

In the Introduction we said that a program semantic level is characterised by the structure on which it is defined; more precisely:

Definition 1:
A structure is a triple: $\mathscr{S} = \langle \mathscr{D}, \mathscr{F}, \mathscr{P} \rangle$, where:
$\mathscr{D}$ is a domain (characterising data on which the program operates);
$\mathscr{F}$ is a set of functions $f: \mathscr{D} \to \mathscr{D}$ (total over $\mathscr{D}$);
$I \in \mathscr{F}$ (I is the identity function);
$\mathscr{P}$ is a set of predicates $p: \mathscr{D} \to \mathscr{D}$ (total over $\mathscr{D}$).

Definition 2:
A program P on structure $\mathscr{S} = \langle \mathscr{D}, \mathscr{F}, \mathscr{P} \rangle$ is a flowchart with the following four types of statements (or blocks):



with the condition that in the flowchart there is only one statement of type (*a*) (initial statement), one or more statements of type (*d*) (final statements), and none, one or more statements of type (*b*) (assignment statements) and of type (*c*) (test statements).

We will call $\mathbf{x}$ input variable(s), $\mathbf{y}$ program variable(s), $\mathbf{z}$ output variable(s) ($\mathscr{D}$ can be thought of as a space of scalars or as a space of n-ples: $\mathscr{D}: \overline{\mathscr{D}} \times \overline{\mathscr{D}} \times \ldots \times \overline{\mathscr{D}}$).

Execution of program P is defined according to connections between blocks, in the usual way: given an input value $\mathbf{x} = \xi$ ($\xi \in \mathscr{D}$), the initial statement assigns value $f(\xi)$ to $\mathbf{y}$, then passes control to the next block; if this is an assignment statement, the current value of $\mathbf{y}$ is modified accordingly and control passes to the next block; if it is a test statement, the next block is selected depending on the value of the predicate $p(\xi)$, and so on.

Computation terminates only if a final block is reached; in this case, the final value $\eta = h(\bar{\xi})$ is assigned to $\mathbf{z}$ ($\bar{\xi}$ is the current value of $\mathbf{y}$ before execution of final block and h is the function it performs).

In this fashion, program P computes the function $P: \mathscr{D} \to \mathscr{D}$ defined in the following way: let $\xi \in \mathscr{D}$; $P(\xi)$ is defined if and only if the computation relative to the initial value $\mathbf{x} = \xi$ terminates with a final value $\mathbf{z} = \eta$ and $\eta = P(\xi)$ holds.

Now let $\varphi: \mathscr{D} \to \{T, F\}$ and $\psi: \mathscr{D} \times \mathscr{D} \to \{T, F\}$ be two (total) predicates such that: $\forall \mathbf{x} \exists \mathbf{z} (\varphi(\mathbf{x}) \Rightarrow \psi(\mathbf{x}, \mathbf{z}))$; they express the required input/output behaviour of the program. We use the following definition of total correctness given in Manna (1969*a*):

Definition 3:
A program P is totally correct with respect to the input predicate φ and the output predicate ψ if and only if, for every ξ such that $\varphi(\xi) = T$, $P(\xi)$ is defined and, making $\eta = P(\xi)$, $\psi(\xi, \eta) = T$ holds.

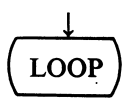
The theorem of correctness given in Manna (1969*a*) holds, which states that total correctness of a program can be reduced to the unsatisfiability (on structure $\mathscr{S}$ on which the program is defined) of a predicate-calculus formula uniquely associated with the program.

The usual techniques of both formal and informal verification (given for example in Floyd, 1967; Manna, 1969*b*; Maurer, 1972) hold too.

We assumed functions $f \in \mathscr{F}$ and predicates $p \in \mathscr{P}$ total over $\mathscr{D}$; however, the correctness theorem and the quoted verification techniques can easily be extended to functions and predicates having domains

$$D(f) \subseteq \mathscr{D},\ D(p) \subseteq \mathscr{D}$$

decidable, by introducing the special statement



(see Manna, 1970). This is important from our point of view, because in developing programs by successive refinements, functions and predicates are generally partial at each level; we are however allowed (since we consider only total correctness, not partial correctness) to assume that a sufficiently wide (possibly maximal) domain can be determined, over which functions and predicates are total.

Being that $D(f) \subseteq \mathscr{D}$, we must consider, in the block by block expansion process, some interface conditions, as will be specified in the following.

2. Relation between P^i and P^{i+1}

The aim of this section is to give a precise definition, based on the concept of representation function τ_i, of what is meant by: 'P^{i+1} is a correct refinement of P^i' (when passing from one structure to another); we will use the expression: 'P^{i+1} represents P^i with respect to (wrt) τ_i'. Let P^i be a program over $\mathscr{D}_i$ (i.e. on a structure $\mathscr{S}_i$ having domain $\mathscr{D}_i$) with input variables $\mathbf{x}^i$, program variables $\mathbf{y}^i$ and output variables $\mathbf{z}^i$; let P^i be totally correct wrt a given input predicate $\varphi^i(\mathbf{x}^i)$, and to a given output predicate $\psi^i(\mathbf{x}^i, \mathbf{z}^i)$; let P^{i+1} be a program over $\mathscr{D}_{i+1}$ with input variables $\mathbf{x}^{i+1}$, program variables $\mathbf{y}^{i+1}$ and output variables $\mathbf{z}^{i+1}$; in addition, let the following function be assigned:

$$\tau_i: \mathscr{D}_{i+1} \to \mathscr{D}_i\ .$$

We will call τ_i representation function, in the sense that each element $\eta \in \mathscr{D}_{i+1}$ represents by means of τ_i a unique element $\tau_i(\eta) \in \mathscr{D}_i$.

We have assumed that $\tau_i: \mathscr{D}_{i+1} \to \mathscr{D}_i$ is a total function over $\mathscr{D}_{i+1}$; if τ_i is a partial function over a domain $D(\tau_i) \subseteq \mathscr{D}_{i+1}$, we will extend τ_i over $\mathscr{D}_{i+1}$ by stating that:

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024

$$\forall \mathbf{x}^{i+1}(\mathbf{x}^{i+1} \notin D(\tau_i) \Leftrightarrow \omega = \tau_i(\mathbf{x}^{i+1}))\ ,$$

assuming that the domain of τ_i is decidable. We will also assume that the 'undefined' element ω is such that: $\omega \notin \mathscr{D}_i$ and that for each predicate φ^i: $\mathscr{D}_i \to \{T, F\}$;

$$\psi^i: \mathscr{D}_i \times \mathscr{D}_i \to \{T, F\};\ \text{etc.}\ ,$$

it is:

$$\varphi^i(\omega) = F;\ \forall \mathbf{x}^i \forall \mathbf{y}^i(\psi^i(\mathbf{x}^i, \omega) = \psi^i(\omega, \mathbf{y}^i) = F);\ \text{etc}\ .$$

In this way, results about total functions τ_i will also hold for partial functions τ_i extended in the above fashion.

Definition 4:

P^{i+1} represents P^i wrt the representation function τ_i if and only if there exist two predicates $\varphi^{i+1}(\mathbf{x}^{i+1})$ and $\psi^{i+1}(\mathbf{x}^{i+1}, \mathbf{z}^{i+1})$, wrt which P^{i+1} is totally correct and such that:

(*a*) $\forall \mathbf{x}^i\, \exists \mathbf{x}^{i+1}(\varphi^i(\mathbf{x}^i) \Rightarrow (\mathbf{x}^i = \tau_i(\mathbf{x}^{i+1}) \wedge \varphi^{i+1}(\mathbf{x}^{i+1})))$;

(*b*) $\forall \mathbf{x}^{i+1}\, \forall \mathbf{z}^{i+1}\, ((\varphi^{i+1}(\mathbf{x}^{i+1}) \wedge \psi^{i+1}(\mathbf{x}^{i+1}, \mathbf{z}^{i+1})) \Rightarrow (\varphi^i(\tau_i(\mathbf{x}^{i+1})) \wedge \psi^i(\tau_i(\mathbf{x}^{i+1}), \tau_i(\mathbf{z}^{i+1}))))$.

The preceding (*a*) and (*b*) formulae express the relation which must exist, by means of τ_i, between the input conditions of P^i and those of P^{i+1} (expression (*a*)) and between the input/output relations which must be satisfied by P^i and P^{i+1} respectively (expression (*b*)). Such relations correspond to that required in Milner (1971) for simulation.

In Definition 4, φ^i and ψ^i are given predicates and the existence of φ^{i+1} and of ψ^{i+1}, wrt which P^{i+1} be totally correct, is required. It can be seen that the problem of deciding whether P^{i+1} represents P^i wrt τ_i, stated in the above fashion, is a second-order problem; however, from our constructive point of view, this problem is irrelevant.

The following result can be immediately proved.

Theorem 1:

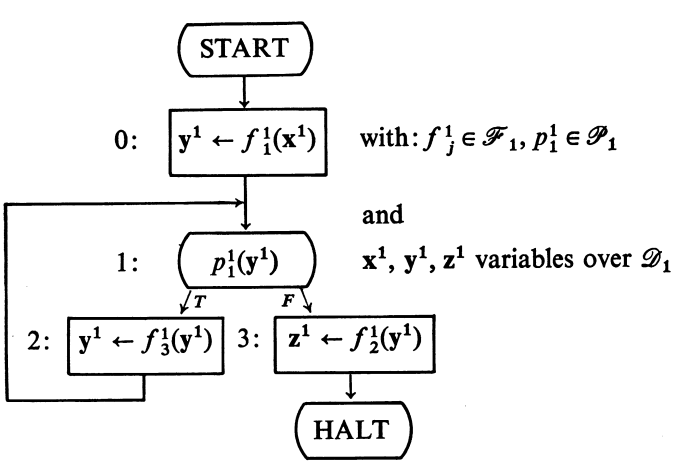
If P^2 represents P^1 wrt the representation function τ_1 and if P^3 represents P^2 wrt the representation function τ_2, then P^3 represents P^1 wrt the representation function $\tau_2.\tau_1$.

In the following section we will show how to pass from a program P^i over $\mathscr{D}_i$ to a program P^{i+1} over $\mathscr{D}_{i+1}$ in such a way that P^{i+1} represents P^i wrt a given representation function τ_i; the transitivity of the representation relation given by the above result guarantees that, passing from P^i to P^{i+1} $(1 \leqslant i \leqslant t-1)$, we are allowed to ignore what was previously done in passing from one to the other of the preceding levels.

3. Building P^{i+1} from P^i

To abbreviate notation we will examine in this section only programs P^1 and P^2 of the sequence $P^1, P^2, \ldots, P^t$; however, the following is valid for any P^i and P^{i+1}. The construction of P^2 from P^1 is done in the following way:

1. Statements of P^1 are numbered. For example, let P^1 be the following program (on first-level structure $\langle \mathscr{D}_1, \mathscr{F}_1, \mathscr{P}_1 \rangle$):



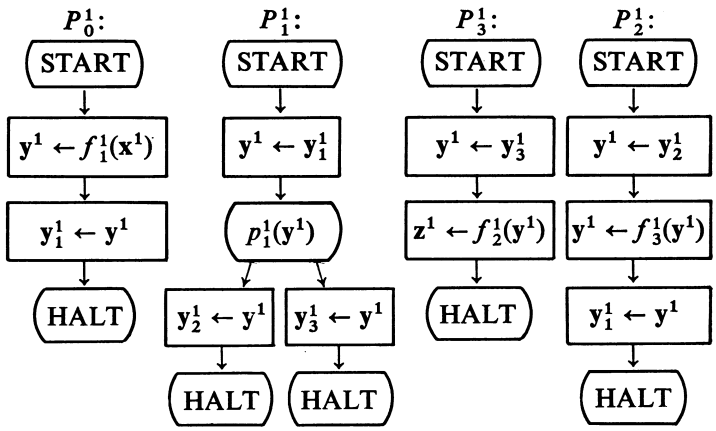
with: $f^1_j \in \mathscr{F}_1$, $p^1_1 \in \mathscr{P}_1$

and

$\mathbf{x}^1, \mathbf{y}^1, \mathbf{z}^1$ variables over $\mathscr{D}_1$

In what follows, for simplicity, we will refer to program P^1 above.

2. Statements 0, 1, 2, 3 are considered as the following elementary programs $P^1_0, P^1_1, P^1_2, P^1_3$ respectively:



The dummy variables $\mathbf{y}^1_j (1 \leqslant j \leqslant 3)$ have been introduced to express formally the input and output predicates of programs $P^1_j (0 \leqslant j \leqslant 3)$. Such predicates are clearly the following:

for P^1_0: $\mathbf{x}^1 \in D(f^1_1)$ (input predicate); $\mathbf{y}^1_1 = f^1_1(\mathbf{x}^1)$ (output predicate);

for P^1_1: $\mathbf{y}^1_1 \in D(p^1_1)$ (input); $\mathbf{y}^1_2 = \mathbf{y}^1_1 \wedge p^1_1(\mathbf{y}^1_2)$ (output of True path);

$\mathbf{y}^1_3 = \mathbf{y}^1_1 \wedge \neg p(\mathbf{y}^1_3)$ (output of False path);

for P^1_2: $\mathbf{y}^1_2 \in D(f^1_3)$ (input); $\mathbf{y}^1_1 = f^1_3(\mathbf{y}^1_2)$ (output);

for P^1_3: $\mathbf{y}^1_3 \in D(f^1_2)$ (input); $\mathbf{z}^1 = f^1_2(\mathbf{y}^1_3)$ (output);

3. Each elementary program P^1_j is expanded into a corresponding program P^2_j on $\langle \mathscr{D}_2, \mathscr{F}_2, \mathscr{P}_2. \rangle$ The structure $\langle \mathscr{D}_2, \mathscr{F}_2, \mathscr{P}_2 \rangle$ is defined by the programmer's choices and the relation between $\mathscr{D}_2$ and $\mathscr{D}_1$ is stated by the choice and construction of the representation function $\tau_1: \mathscr{D}_2 \to \mathscr{D}_1$.

This relation constitutes a guide to the construction of expansions (in a sense that will be made more precise in the appendix) by means of the following predicates which assure correctness of step 3. (we postpone the treatment of this subject to the next section, so as not to interrupt the present exposition).

For expansion P^2_0:

$$\hat{\varphi}^2_0(\mathbf{x}^2) = \tau_1(\mathbf{x}^2) \in D(f^1_1);$$

$$\hat{\psi}^2_0(\mathbf{x}^2, \mathbf{y}^2_1) \overset{\text{def.}}{=} \tau_1(\mathbf{y}^2_1) = f^1_1(\tau_1(\mathbf{x}^2)).$$

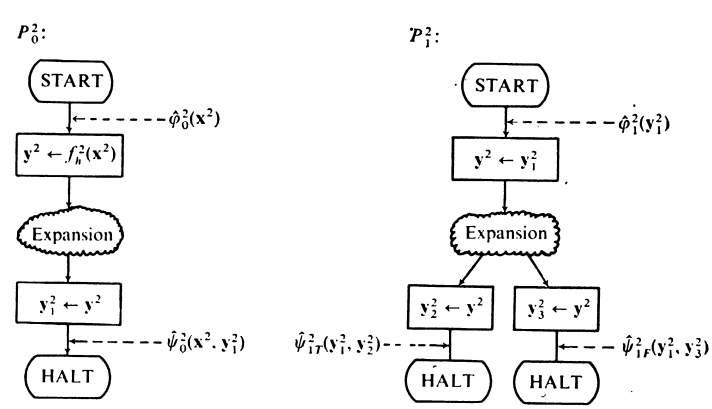
For expansion P^2_1:

$$\hat{\varphi}^2_1(\mathbf{y}^2_1) = \tau_1(\mathbf{y}^2_1) \in D(p^1_1);$$

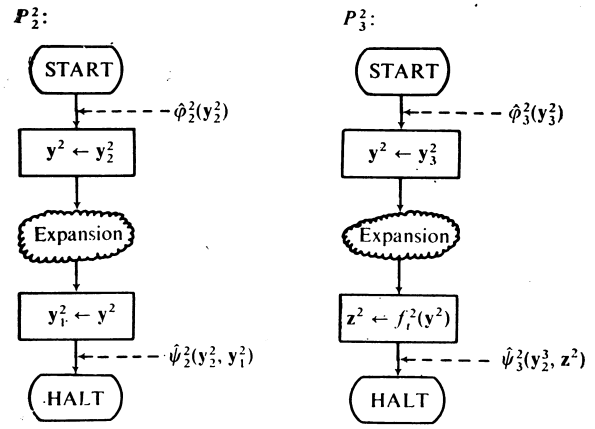
$$\hat{\psi}^2_{1T}(\mathbf{y}^2_1, \mathbf{y}^2_2) \overset{\text{def.}}{=} \tau_1(\mathbf{y}^2_2) = \tau_1(\mathbf{y}^2_1) \wedge p^1_1(\tau_1(\mathbf{y}^2_1))\ \text{(output True)};$$

$$\hat{\psi}^2_{1F}(\mathbf{y}^2_1, \mathbf{y}^2_3) \overset{\text{def.}}{=} \tau_1(\mathbf{y}^2_3) = \tau_1(\mathbf{y}^2_2) \wedge \neg p^1_1(\tau_1(y^2_1))\ \text{(output False)}.$$

Analogously for predicates $\hat{\varphi}^1_2$, $\hat{\psi}^1_2$, $\hat{\varphi}^1_3$, $\hat{\psi}^1_3$; the dummy variables $\mathbf{y}^2_1$, $\mathbf{y}^2_2$ etc. have been introduced in order to formally express the input and output predicates of the expansions, which are therefore programs of the following type:

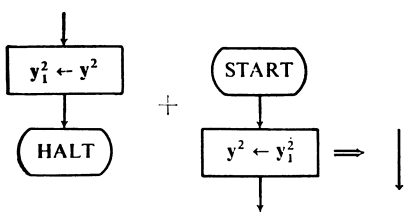


Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024



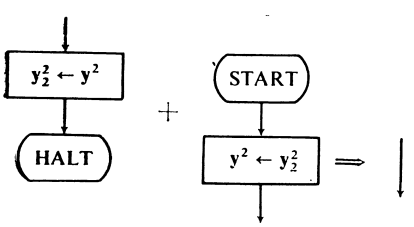
4. The various expansions of the single statements of P^1 are merged into one program P^2 as follows:

   (*a*) the exit of P_0^2 is connected to the entrance of P_1^2, eliminating the dummy assignments $y_1^2 \leftarrow y^2$ and $y^2 \leftarrow y_1^2$; such an elimination is symbolically indicated by:

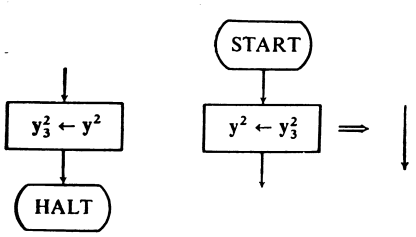


   (*b*) the exit of P_2^2 is connected to the entrance of P_1^2 in the same way;

   (*c*) the exit of P_1^2 is connected to the entrance of P_2^2 by the elimination:



   (*d*) the exit of P_1^2 is connected to the entrance of P_3^2 by the elimination:



A program P^2 with input variables $\mathbf{x}^2$, program variables $\mathbf{y}^2$ and output variables $\mathbf{z}^2$ is thus obtained.

4. Conditions of correct expansion

Note that:

(*a*) If block expansions are such as to satisfy the correctness conditions $\hat{\varphi}_j^2$, $\hat{\psi}_j^2$, the obtained program P^2 represents P^1, and therefore one is sure not to have overlooked any connection between the original blocks (in the opposite case, errors can arise, since the domains of the various blocks are different);

(*b*) On the other hand, in practice a strict requirement like this can lead in particular cases (especially in programs of great size) to a superfluous increase in the number of variables and/or instructions;

(*c*) To proceed in a more flexible way, it is preferable to adopt the condition that each block expansion be a representation of the block itself, according to Definition 4. However, this will also imply verification of interface conditions (i.e. that each exit from one expansion is accepted as input by the next one).

We will then show, first, what are the interface conditions mentioned in point (*c*) above; as a consequence we will demonstrate the validity of statement (*a*). Here also we refer to expansion of program P^1 given in the previous section.

Definition 5:
Program P_j^2 is a correct expansion, wrt τ_1, of statement j of P^1 if and only if P_j^2 represents the elementary program P_j^1 wrt τ_1, according to Definition 4.

Definition 6:
Given that φ_j^2, $\psi_j^2 (0 \leqslant j \leqslant 3)$ are the correctness predicates assigned for the expansions P_j^2 (they represent the elementary programs P_j^1 wrt τ_1), the interface conditions between expansions are the following:

(*a*) interface between P_0^2 and P_1^2:

$$C_1(\varphi_0^2, \psi_0^2, \varphi_1^2) \underset{\text{def.}}{=} \forall \mathbf{x}^1 \forall \mathbf{x}^2 \forall \mathbf{y}_1^2 \{(\mathbf{x}^1 \in D(f_1^1) \wedge f_1^1(\mathbf{x}^1) \in D(P_1^1)) \Rightarrow [(\mathbf{x}^1 = \tau_1(\mathbf{x}^2) \wedge \varphi_0^2(\mathbf{x}^2) \wedge \psi_0^2(\mathbf{x}^2, \mathbf{y}_1^2)) \Rightarrow \varphi_1^2(\mathbf{y}_1^2)]\}.$$

(*b*) interface between P_1^2 and P_2^2:

$$C_1(\varphi_1^2, \psi_{1T}^2, \varphi_2^2) \underset{\text{def.}}{=} \forall \mathbf{y}_1^1 \forall \mathbf{y}_1^2 \forall \mathbf{y}_2^2 \{(\mathbf{y}_1^1 \in D(p_1^1) \wedge p_1^1(\mathbf{y}_1^1) \wedge \mathbf{y}_1^1 \in D(f_3^1)) \Rightarrow [(\mathbf{y}_1^1 = \tau_1(\mathbf{y}_2^2) \wedge \varphi_1^2(\mathbf{y}_1^2) \wedge \varphi_{1T}^2(\mathbf{y}_1^2, \mathbf{y}_2^2)) \Rightarrow \varphi_2^2(\mathbf{y}_2^2)]\}.$$

The interface conditions between P_1^2 and P_3^2 and between P_2^2 and P_1^2 are obtained in a similar way.

The following theorem can be proved, by induction on the execution sequences of P_1 (for the definition of execution sequence see for instance Manna, 1969b.)

Theorem 2:
If program P^1 is totally correct wrt the input predicate $\varphi^1(\mathbf{x}^1)$ and the output predicate $\psi^1(\mathbf{x}^1, \mathbf{z}^1)$; if all the expansions P_j^2 of the statements j of program P^1 are 'correct expansions' in the sense of Definition 5; if all the interface conditions $C_1(\varphi_0^2, \psi_0^2, \varphi_1^2)$, $C_1(\varphi_1^2, \psi_1^2, \varphi_2^2)$ etc. are true; and if P^2 is constructed connecting to each other the different expansions P_j^2 as stated in Point 4 of previous section; then P^2 is totally correct wrt the input predicate $\varphi^1(\tau_1(\mathbf{x}^2)) \wedge \varphi_0^2(\mathbf{x}^2)$ and the output predicate $\psi^1(\tau_1(\mathbf{x}^2), \tau_1(\mathbf{z}^2))$.

Corollary 1:
P^2 represents P^1 wrt τ_1.

The proof of corollary 1 is immediate, when considering that:

(*a*) φ^1 is the input predicate of P^1, for which the condition: $\forall \mathbf{x}^1(\varphi^1(\mathbf{x}^1) \Rightarrow \mathbf{x}^1 \in D(f_1^1))$ must hold;

(*b*) P_0^2 is a correct expansion of P_0^1, for which the condition: $\forall \mathbf{x}^1 \exists \mathbf{x}^2(\mathbf{x}^1 \in D(f_1^1) \Rightarrow (\mathbf{x}^1 = \tau_1(\mathbf{x}^2) \wedge \varphi_0^2(\mathbf{x}^2)))$ must hold (because of Definitions 4 and 5).

Considering that the program P^1, to which we have referred till now, contains every type of statement, the expansion procedure, Theorem 2 and Corollary 1 given above hold in general. Since the representation relation is transitive (Theorem 1), the following corollary holds.

Corollary 2:
If in the sequence $P^1, P^2, \ldots, P^t$ of programs, P^{i+1} is obtained from P^i $(1 \leqslant i \leqslant t-1)$ according to Steps 1, 2, 3, 4 given

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024

above, then, τ_i being the representation function $\mathscr{D}_{i+1} \to \mathscr{D}_i$, P^t represents P^1 wrt the representation function:

$$\tau_1 \cdot \tau_2 \cdot \ldots \cdot \tau_{t-1} : \mathscr{D}_t \to \mathscr{D}_1 \ .$$

The correctness predicates of P^t are of the form:
input predicate: $\varphi^1(\tau_1 \cdot \tau_2 \cdot \ldots \cdot \tau_{t-1}(\mathbf{x}^t)) \wedge \gamma^t(\mathbf{x}^t)$
output predicate: $\psi^1(\tau_1 \cdot \tau_2 \ldots \cdot \tau_{t-1}(\mathbf{x}^t), \tau_1 \cdot \tau_2 \ldots \cdot \tau_{t-1}(\mathbf{z}^t))$
where $\gamma^t(\mathbf{x}^t)$ is a predicate which depends on the $t-1$ refinements made; for example, for $t = 3$, $\gamma^3(\mathbf{x}^3) =$
$(\varphi_0^2(\tau_2(\mathbf{x}^3)) \wedge \varphi_0^3(\mathbf{x}^3))$.

Some remarks are at this point opportune with regard to the validity of Theorem 2:
(*a*) we have now to prove that if expansions are totally correct wrt the predicates $\hat{\varphi}_j^2$, $\hat{\psi}_j^2$, then verification of the interface conditions is not necessary. In fact, let

$$C_3(\varphi_1^2) \underset{\text{def.}}{=} \forall \mathbf{y}_1^2(\tau_1(\mathbf{y}_1^2) \in D(p_1^1) \Rightarrow \varphi_1^2(\mathbf{y}_1^2))$$

$$C_2(\varphi_1^2) \underset{\text{def.}}{=} \forall \mathbf{x}^1 \, \forall \mathbf{y}_1^2((\mathbf{x}^1 \in D(f_1^1) \wedge f_1^1(\mathbf{x}^1) \in D(p_1^1)) \Rightarrow (f_1^1(\mathbf{x}^1) = \tau_1(\mathbf{y}_1^2) \Rightarrow \varphi_1^2(\mathbf{y}_1^2)))$$

then

$$C_3(\varphi_1^2) \Rightarrow C_2(\varphi_1^2) \ ,$$

and if P_0^2 is totally correct wrt φ_0^2 and ψ_0^2, then

$$C_2(\varphi_1^2) \Rightarrow C_1(\varphi_0^2, \psi_0^2, \varphi_1^2) \ .$$

(and analogously for expansions of the other blocks). On the other hand, conditions C_3 are automatically satisfied when the expansions are correct wrt the predicates $\hat{\varphi}_j^2$, $\hat{\psi}_j^2$; therefore, in this case, all the conditions of Theorem 2 are verified. Note that C_3 and C_2 are sufficient conditions weaker than C_1 and it is often easier to verify them.
(*b*) C_1 also are conditions sufficient but not necessary for the validity of the theorem. Necessary conditions can be determined on the basis of φ_j^2, ψ_j^2 only when these express a functional connection, that is, are such that: $\forall \mathbf{x} \exists ! \mathbf{z} \ (\varphi_j^2(\mathbf{x}) \Rightarrow \psi_j^2(\mathbf{x}, \mathbf{z}))$; in general the connection is a relation because of the constructive character of the procedure, i.e. the fact that the correctness conditions φ_j^2, ψ_j^2 of the expansions are determined before making the block expansions themselves.

5. Example

In this example we will drop some of the assumptions about formalism that were used in the previous exposition. Specifically, instead of considering domains $\mathscr{D}_1 = \overline{\mathscr{D}}_1 \times \ldots \times \overline{\mathscr{D}}_1$ $\mathscr{D}_2 = \overline{\mathscr{D}}_2 \times \ldots \times \overline{\mathscr{D}}_2$, we will consider several variables, of different types, over $\overline{\mathscr{D}}_1$ and $\overline{\mathscr{D}}_2$ and the representation function will therefore also express the relationship between those variables. The necessary modifications will be specified as they appear.

1. *First-level program* P^1

The problem is the following: 'A symmetric matrix with positive or null elements has to be multiplied by itself until the maximum of its elements is greater than or equal to an assigned number $\alpha \in R^+$ (R^+ is the set of positive reals)'.

The minimum requirement to do this is that in the space S of symmetric matrices $N \times N$ an internal product is defined and that with each matrix $s \in S$ a number $\|s\| \in R^+$ (max of elements) is uniquely associated, to be compared with $\alpha \in R^+$.

The first-level structure is therefore:

$$\mathscr{S}_1 = \langle S \cup R^+, \{\| \ \|, \cdot\}, \{\geqslant\}\rangle$$

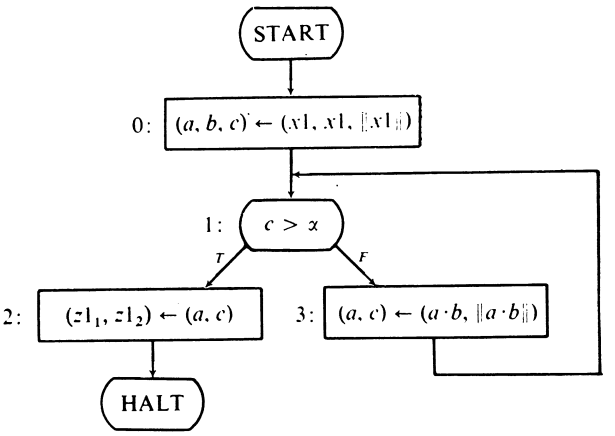
with the following properties:

1. $\langle R^+, \geqslant\rangle$ is the structure of positive or null reals with the ordering $\geqslant$;
2. $\forall s_1, s_2 \in S: s_1 \cdot s_2 \in S$;
3. $\forall s \in S: \|s\| \in R^+$;
4. a non-empty subset $S_\alpha \subseteq S$ exists, such that:

$$\forall s \in S_\alpha: \lim_{n \to \infty} \|s^n\| > \alpha \ .$$

Point 4 corresponds to the requirement of a non-empty solution of the problem.

Let's now consider the following program P^1 on $\mathscr{S}_1$:



where $(x1)$, (a, b, c), $(z1_1, z1_2)$ are input, program and output variables, respectively.

The properties of structure $\mathscr{S}_1$ given above are sufficient to demonstrate that P^1 is totally correct with respect to the following predicates:

$\varphi^1(x1) \equiv x1 \in S_\alpha$;
$\psi^1(x1, z1_1, z1_2) \equiv$
$\exists n(z1_1 = x1^n \wedge z1_2 = \|x1^n\| \wedge \forall m(1 \leqslant m < n \Rightarrow \|x1^m\| \leqslant \alpha) \wedge z1_2 > \alpha)$
$(n, m \in \eta \subseteq R^+, \eta$ set of natural numbers).

Space S of symmetric matrices over R^+ satisfies the properties of structure $\mathscr{S}_1$, and the predicates φ^1 and ψ^1 formally express the requirements of the problem definition; in addition, P^1 is easily understandable and can therefore be considered a good assessment of the problem. Nothing is said, at the first level, about how functions $\cdot$ and $\| \ \|$ will be realised; this will depend on how matrices will be represented in core memory. However, we are now sure, since the properties required for $\mathscr{S}_1$ hold, that the problem has a solution, and an outline of it is P^1. The implied variables are of different types, i.e. $x1, a, b, z1_1$ are individual variables over S and $c, z1_2$ are individual variables over R^+. The domains of the various blocks are therefore different, and we have to take this into account not only in the verification, but also in the expansion process; to achieve this, we can make use of the following input/output conditions for the elementary programs P_j^1 $(0 \leqslant j \leqslant 3)$:

$\varphi_0^1(x1) \equiv (x1 \in S)$; $\psi_0^1(x1, a_1, b_1, c_1) \equiv (a_1 = b_1 = x1 \wedge c_1 = \|x1\|)$;
$\varphi_1^1(a_1, b_1, c_1) \equiv (c_1 \in R^+)$;
$\psi_{1T}^1(a_1, b_1, c_1, a_2, b_2, c_2) \equiv ((a_1, b_1, c_1) = (a_2, b_2, c_2) \wedge c_2 > \alpha)$;
$\psi_{1F}^1(a_1, b_1, c_1, a_3, b_3, c_3) \equiv ((a_1, b_1, c_1) = (a_3, b_3, c_3) \wedge c_3 \leqslant \alpha)$;
$\varphi_2^1(a_2, b_2, c_2) \equiv (a_2 \in S \wedge c_2 \in R^+)$;
$\psi_2^1(a_2, b_2, c_2, z1_1, z1_2) \equiv (z1_1 = a_2 \wedge z1_2 = c_2)$;
$\varphi_3^1(a_3, b_3, c_3) \equiv (a_3, b_3 \in S \wedge c_3 \in R^+)$;
$\psi_3^1(a_3, b_3, c_3, a_1, b_1, c_1) \equiv (a_1 = a_3 \cdot b_3 \wedge b_1 = b_3 \wedge c_1 = \|a_1\|)$.

We can now deal with the representation of data in the expansions of P^1.

2. *The representation function*

To save memory space, only elements $s(i, j)$ with $i \leqslant j$ are stored in a vector $\mathbf{v}$. This can be done by stating the following correspondence between the matrix indices and the vector indices:

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024

$$\begin{array}{cccccccc}(1,1), & (1,2), & (2,2), & (1,3), & (2,3), & (3,3), & \ldots\\ 1, & 2, & 3, & 4, & 5, & 6, & \ldots\end{array}$$

Such a correspondence is assigned by the function:

(*a*) $f(l) = (i,j)$, with $j = \max\{k | k(k-1)/2 < l\}$ and $i = l - j(j-1)/2$

(*b*) $f^{-1}(i,j) = l$, with $l = j(j-1)/2 + i$ for $i \leqslant j$
$f^{-1}(i,j) = f^{-1}(j,i)$ for $i > j$.

Then there exists a one-to-one correspondence between the space V of vectors of dimensions $N(N+1)/2$ and the space S of symmetric matrices $N \times N$:

$\bar{\tau}: V \longleftrightarrow S$, which is assigned as follows:

$\forall \mathbf{v} \in V(s = \bar{\tau}(\mathbf{v}) \Leftrightarrow (s(f(l)) = \mathbf{v}(l)$ for $1 \leqslant l \leqslant N(N+1)/2))$;
$\forall s \in S(\mathbf{v} = \bar{\tau}^{-1}(s) \Leftrightarrow (\mathbf{v}(f^{-1}(i,j)) = s(i,j)$ for $1 \leqslant i \leqslant j$ and $1 \leqslant j \leqslant N))$.

$\bar{\tau}$ has the following properties:

5. $\forall \mathbf{v}(\mathbf{v} \in V \Leftrightarrow \bar{\tau}(\mathbf{v}) \in S)$;
6. $\forall \mathbf{v}_1, \mathbf{v}_2 \in V(\bar{\tau}(\mathbf{v}_1) = \bar{\tau}(\mathbf{v}_2) \Leftrightarrow \mathbf{v}_1 = \mathbf{v}_2)$.

The second-level domain is therefore: $\mathscr{D}_2 = V \cup R^+$. We will consider for P^2 the following variables:

$(\mathbf{x}^2, i, j, \ldots)$ input variables;
$(\mathbf{e}, \mathbf{f}, d, i, j, \ldots)$ program variables;
$(\mathbf{z}2_1, z2_2, i, j, \ldots)$ output variables.

$i, j, \ldots$ are variables over η and have been introduced in order to construct the operations acting on single elements of vectors $\mathbf{v} \in V$. Their final number will be known only at the end of the expansion construction and is unimportant at this point since only $\mathbf{x}2, \mathbf{e}, \mathbf{f}, d, \mathbf{z}2_1, z2_2$ represent the first-level variables $x1, a, b, c, z1$ in the way specified by the following functions:

$\tau_I(\mathbf{x}2, i, j, \ldots) = x1$ if and only if $x1 = \bar{\tau}(\mathbf{x}2)$;
$\tau_P(\mathbf{e}, \mathbf{f}, d, i, j, \ldots) = (a, b, c)$ if and only if $a = \bar{\tau}(\mathbf{e}), b = \bar{\tau}(\mathbf{f})$ and $d = c$;
$\tau_U(\mathbf{z}2_1, z2_2, i, \ldots) = (z1_1, z1_2)$ if and only if $z1_1 = \bar{\tau}(\mathbf{z}2_1) \wedge z1_2 = z2_2$.

τ_I connects the input variables, τ_P the program variables, τ_U the output variables. It is necessary to distinguish between τ_I, τ_P, τ_U since the input, program and output variables are not the same; however, both for P^1 and P^2, the program variables are the same for all the blocks and therefore the formulae given in the previous sections still hold.

3. *The correctness predicates*

Predicates $\hat{\varphi}_j^2, \hat{\psi}_j^2$ $(0 \leqslant j \leqslant 3)$ are obtained from φ_j^1 and ψ_j^1 by means of τ_I, τ_P, τ_U in the following way:

$$\hat{\varphi}_0^2(\mathbf{x}2, i, \ldots) = \varphi_0^1(\tau_I(\mathbf{x}2, i, \ldots)) \equiv (\mathbf{x}2 \in V)$$

(on the basis of property 5 of $\bar{\tau}$);

$$\hat{\psi}_0^2(\mathbf{x}2, i_0, \mathbf{e}_1, \mathbf{f}_1, d_1, i_1, \ldots) = \psi_0^1(\tau_I(\mathbf{x}2, i_0, \ldots), \tau_P(\mathbf{e}_1, \mathbf{f}_1, d_1, \ldots)) \equiv (\mathbf{e}_1 = \mathbf{f}_1 = \mathbf{x}2 \wedge d_1 = \|\bar{\tau}(\mathbf{x}2)\|)$$

(on the basis of properties 5 and 6 of $\bar{\tau}$).

Analogously:

$$\hat{\varphi}_1^2(\mathbf{e}_1, \mathbf{f}_1, d_1, i_1, \ldots) \equiv d_1 \in R^+;$$

$$\hat{\psi}_{1T}^2(\mathbf{e}_1, \mathbf{f}_1, d_1, i_1, \ldots, \mathbf{e}_2, \mathbf{f}_2, d_2, i_2, \ldots) \equiv \mathbf{e}_1 = \mathbf{e}_2 \wedge \mathbf{f}_1 = \mathbf{f}_2 \wedge d_1 = d_2 \wedge d_2 > \alpha;$$

$$\hat{\psi}_{1F}^2(\mathbf{e}_1, \mathbf{f}_1, d_1, i_1, \ldots, \mathbf{e}_3, \mathbf{f}_3, d_3, i_3, \ldots) \equiv \mathbf{e}_1 = \mathbf{e}_3 \wedge \mathbf{f}_1 = \mathbf{f}_3 \wedge d_1 = d_3 \wedge d_3 \leqslant \alpha;$$

$$\hat{\varphi}_2^2(\mathbf{e}_2, \mathbf{f}_2, d_2, i_2, \ldots) \equiv \mathbf{e}_2 \in V \wedge d_2 \in R^+;$$

$$\hat{\psi}_2^2(\mathbf{e}_2, \mathbf{f}_2, d_2, i_2, \ldots, \mathbf{z}2_1, z2_2) \equiv \mathbf{z}2_1 = \mathbf{e}_2 \wedge d_2 = z2_2;$$

$$\hat{\varphi}_3^2(\mathbf{e}_3, \mathbf{f}_3, d_3, i_3, \ldots) \equiv \mathbf{e}_3, \mathbf{f}_3 \in V \wedge d_3 \in R^+;$$

$$\hat{\psi}_3^2(\mathbf{e}_3, \mathbf{f}_3, d_3, i_3, \ldots, \mathbf{e}_1, \mathbf{f}_1, d_1, i_1, \ldots) \equiv (\bar{\tau}(\mathbf{e}_1) = \bar{\tau}(\mathbf{e}_3) \cdot \bar{\tau}(\mathbf{f}_3) \wedge \mathbf{f}_1 = \mathbf{f}_3 \wedge d_1 = \|\bar{\tau}(\mathbf{e}_1)\|)$$.

4. *Construction of expansions*

Expansions P_j^2 are made on the structure:

$$\mathscr{S}_2 = \langle V \cup R^+, \{+, *, \text{Max}, /, \mathbf{v}(l)\}, \{\geqslant\}\rangle$$

where $+$, $*$, Max, / and $\geqslant$ are defined in the usual fashion in R^+. $\mathbf{v}(l)$ is the extraction of the lth component of vector $\mathbf{v}$.

We have to express the predicates $\hat{\varphi}_j^2$ and $\hat{\psi}_j^2$ on the structure $\mathscr{S}_2$; this will constitute a guide to construction of expansions P_j^2. The only terms that appear in the above predicates $\hat{\varphi}_j^2$ and $\hat{\psi}_j^2$ not already expressed on $\mathscr{S}_2$ are:

$$\text{(I) } \|\bar{\tau}(\mathbf{x}2)\|; \text{ (II) } \bar{\tau}(\mathbf{e}_1) = \bar{\tau}(\mathbf{e}_3) \cdot \bar{\tau}(\mathbf{f}_3); \text{ (III) } \|\bar{\tau}(\mathbf{e}_1)\| .$$

Besides, having defined $\|s\| = \text{Max}\{s(i,j) | 1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N\}$, the expression

$$\|\bar{\tau}(\mathbf{v})\| = \text{Max}\{\mathbf{v}(l) | 1 \leqslant l \leqslant N(N+1)/2\}$$

holds for every vector $\mathbf{v}$ of dimension $N(N+1)/2$.

As for the relation II between $\mathbf{e}_1, \mathbf{e}_3, \mathbf{f}_3$, we have to find a vector product $\circledast$, expressed in terms of the components and such that:

7. $\forall \mathbf{v}_1, \mathbf{v}_2 \in V(\bar{\tau}(\mathbf{v}_1 \circledast \mathbf{v}_2) = \bar{\tau}(\mathbf{v}_1) \cdot \bar{\tau}(\mathbf{v}_2))$.

In fact, in such a case $\bar{\tau}(\mathbf{e}_1) = \bar{\tau}(\mathbf{e}_3) \cdot \bar{\tau}(\mathbf{f}_3)$ can be expressed as $\mathbf{e}_1 = \mathbf{e}_3 \circledast \mathbf{f}_3$.

To construct the product $\circledast$, we use the following procedure:

(*a*) The matrix product is defined as usual by means of sums and products between the components, i.e.:

8. $s_{12}(i,j) = \sum_k s_1(i,k) * s_2(k,j) \quad (s_{12} = s_1 \cdot s_2)$.

(*b*) On the other hand, we must have:

9. $\bar{\tau}^{-1}(s_1 \cdot s_2) = \bar{\tau}^{-1}(s_1) \circledast \bar{\tau}^{-1}(s_2)$.

(*c*) Remembering that $\bar{\tau}^{-1}(s) = \mathbf{v}$ if and only if $\mathbf{v}(f^{-1}(i,j)) = s(i,j)$ for $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N$ (see para. 2), making $\mathbf{v}_1 = \bar{\tau}^{-1}(s_1)$, $\mathbf{v}_2 = \bar{\tau}^{-1}(s_2)$, $\mathbf{v}_{12} = \bar{\tau}^{-1}(s_{12})$, the following must hold:

$$\mathbf{v}_{12}(f^{-1}(i,j)) = \sum_k \mathbf{v}_1(f^{-1}(i,k)) * \mathbf{v}_2(f^{-1}(k,j))$$

for $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N$. Therefore:

$$\begin{aligned}10.\ \mathbf{v}_{12}(j(j-1)/2+i) = &\sum_{k=1}^{i} \mathbf{v}_1(i(i-1)/2-k) * \mathbf{v}_2(j(j-1)/2+k) + \\ &\sum_{k=i+1}^{j} \mathbf{v}_1(k(k-1)/2+i) * \mathbf{v}_2(j(j-1)/2+k) + \\ &\sum_{k=j+1}^{N} \mathbf{v}_1(k(k-1)/2+i) * \mathbf{v}_2(k(k-1)/2+j)\end{aligned}$$

for $1 \leqslant j \leqslant N$ and $i \leqslant j$.

We are now able to express each component $\mathbf{v}_{12}(l)$ $(l = j(j-1)/2 + i)$ by means of sums and products on the components of $\mathbf{v}_1$ and $\mathbf{v}_2$, that is, we have constructed the product $\circledast$ required. Expression (10) will serve as a guide to construction of the corresponding expansion.

This procedure is more general, i.e. can be applied each time some given operations (and their properties) on a data structure are to be represented on a different structure, whose relationship with the first one is known.

Coming back to the present example, we have now:

11. $\|\bar{\tau}(\mathbf{x}2)\| = \text{Max}\{\mathbf{x}2(l) | 1 \leqslant l \leqslant N(N+1)/2\}$

(analogously for $\|\bar{\tau}(\mathbf{e}_1)\|$) and we can express $\bar{\tau}(\mathbf{e}_1) = \bar{\tau}(\mathbf{e}_3) \cdot \bar{\tau}(\mathbf{f}_3)$ as follows:

$$12.\ \mathbf{e}_1(j(j-1)/2+i) = \sum_{k=1}^{i} \mathbf{e}_3(i(i-1)/2+k) \circledast \mathbf{f}_3(j(j-1)/2+k) + \ldots$$

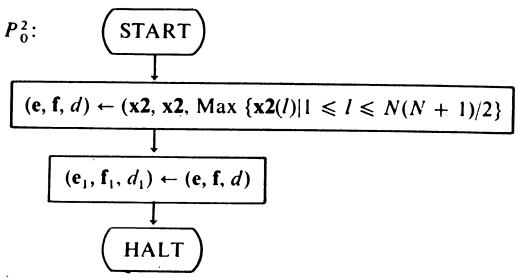
(continuing as in (10)) for $1 \leqslant j \leqslant N$ and $1 \leqslant i \leqslant j$.

The expansions P_j^2 correct wrt $\hat{\varphi}_j^2$ and $\hat{\psi}_j^2$ are therefore the following:

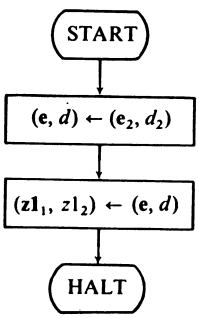
Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024

(P_0^2 on the basis of (11) and of $\hat{\varphi}_0^2$, $\hat{\psi}_0^2$;
P_2^2 on the basis of $\hat{\varphi}_2^2$, $\hat{\psi}_2^2$;
P_1^2 on the basis of $\hat{\varphi}_1^2$, $\hat{\psi}_1^2$;
P_3^2 on the basis of (11), (12) and of $\hat{\varphi}_3^2$, $\hat{\psi}_3^2$.)

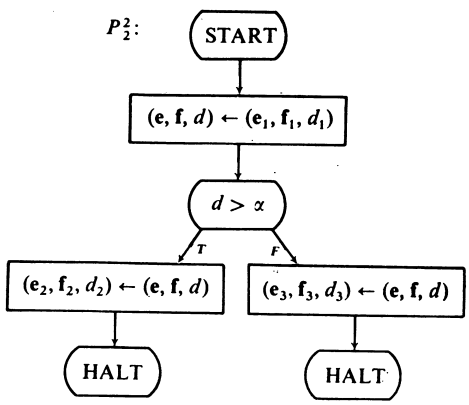
P_0^2:



P_1^2:



P_2^2:



The variables $i, j, \ldots$ don't appear in P_1^2 and P_2^2 because no property is required of them, and they are not modified (the same for i, j, **f** in P_2^2). These variables could appear in P_1^2, P_2^2 with arbitrary assignments, but this is unnecessary.

Finally, merging the expansions P_j^2 (according to step 4 of Section 3) the second-level program P^2 is obtained, correct wrt the following predicates (because of what is stated in Section 4):

φ^2: $\bar{\tau}(\mathbf{x2}) \in S_\alpha$

ψ^2: $\exists n[\bar{\tau}(\mathbf{z2}_1) = \bar{\tau}(\mathbf{x2})^n \wedge z2_2 = \|\bar{\tau}(\mathbf{x2})^n\| \wedge$
$\wedge \forall m(1 \leqslant m \leqslant n \Rightarrow \|\bar{\tau}(\mathbf{x2})^m\| < \alpha) \wedge z2_2 \geqslant \alpha]$.

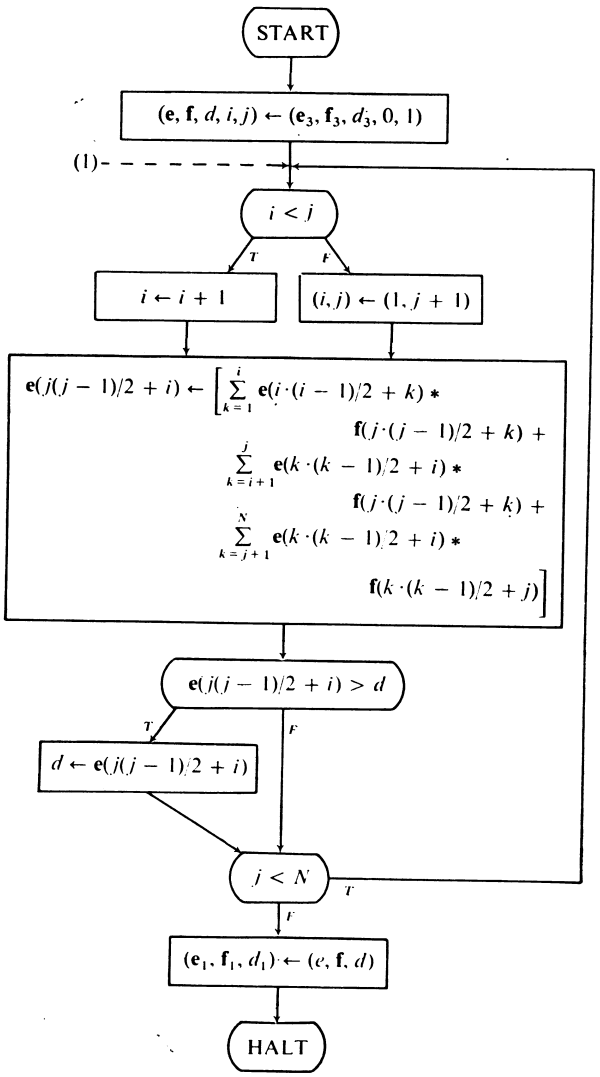
The predicates φ^2 and ψ^2, the representation functions τ_I, τ_P, τ_U and the first-level program P^1 constitute a documentation of the program P^2 and of the choices made for its construction, i.e.:

(*a*) the input/output behaviour of P^2 is expressed, by means of $\bar{\tau}$, on the basis of that of P^1, e.g. the input vector **x2** corresponds to the matrix $x1$ in the way specified by $\bar{\tau}(\mathbf{x2}) = x1$ and the output vector $\mathbf{z2}_1$ corresponds to the matrix $z1_1 = x1^n$ in the way specified by $\bar{\tau}(\mathbf{z2}_1) = \bar{\tau}(\mathbf{x2})^n$;

(*b*) τ_I, τ_U indicate how to supply the input and output data to obtain the initial requirement made in P^1.

In this example the interface conditions didn't appear since the expansions were correct wrt the predicates φ_j^2, ψ_j^2. If, for instance, the expansion of block 2 were called (as a subroutine) by several points of a more complex program with different matrix dimensions, instead of constructing different pairs of predicates φ_j^2, ψ_j^2 for each call, it would be necessary to assign to the subroutine a unique pair of correctness predicates which contain the matrix dimension as a parameter. In this and other practical cases, therefore, verification of interface conditions becomes necessary.

P_3^2:



Appendix

With the following remarks, we want here to make more precise what we intended in the previous sections for 'constructive character' of the described method:

(*a*) P^1 is a program on a structure $\mathscr{S}_1 = \langle \mathscr{D}_1, \mathscr{F}_1, \mathscr{P}_1 \rangle$, correctness predicates and other assertions about program P^1 must contain terms referring to structure $\mathscr{S}_1$. Formally, they must be expressed in a theory $\mathscr{C}_1$ of which $\mathscr{S}_1$ is a model;

(*b*) In the same way, P^2 is a program on a structure $\mathscr{S}_2 = \langle \mathscr{D}_2, \mathscr{F}_2, \mathscr{P}_2 \rangle$, thus containing terms referring to $\mathscr{S}_2$. The properties of the representation function $\tau_1 : \mathscr{D}_2 \to \mathscr{D}_1$ and the predicates φ_j^2, ψ_j^2 containing τ_1 must be given in terms that appear both in $\mathscr{S}_1$ and $\mathscr{S}_2$. Formally, φ_j^2, ψ_j^2 must be expressed in a theory $\mathscr{C}_{12}$ of which a suitable structure $\mathscr{S}_{12}$ (union of $\mathscr{S}_1$ and $\mathscr{S}_2$ and containing τ_1) is a model;

(*c*) If a pair of predicates $\tilde{\varphi}_j^2$, $\tilde{\psi}_j^2$ containing only terms of $\mathscr{S}_2$ and equivalent to $\hat{\varphi}_j^2$, $\hat{\psi}_j^2$ in $\mathscr{C}_{12}$ can be found, then this constitutes a guide to construction of expansions P_j^2 (as in the example). Theoretically, this is related to the program synthesis problem, stated as follows (Constable,

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 16 April 2021

1971; Marini and Miglioli, 1973, Degli Antoni, Miglioli and Onaghi, 1974);

Let φ and ψ be two predicates expressed in a theory $\mathscr{C}$ having a model $\mathscr{S}$; is it possible, by proving proposition

$$\mathscr{C} \vdash \forall x(\varphi(x) \Rightarrow \exists z(\psi(x, z)))$$

($\vdash$ means 'provable in $\mathscr{C}$'), to build a partial recursive function f such that:

1. $D(f) = \{\xi | \mathscr{S} \models \varphi(\xi)\}$ ($\models$ means 'valid in $\mathscr{S}$')
2. if $\xi \in D(f)$ and $\eta = f(\xi)$, then $\mathscr{S} \models \psi(\xi, \eta)$?

Obviously, the answer depends on theory $\mathscr{C}$; it is positive for Kleene's intuitionistic theory of natural numbers (see the quoted references). We only observe here that, since the equivalence between predicates $\tilde{\varphi}_j^2$, ψ_j^2 and predicates $\hat{\varphi}_j^2$, $\hat{\psi}_j^2$ is verified, it is also verified that:

3. $\mathscr{C}_{12} \vdash \forall x^2(\tilde{\varphi}_j^2(x^2) \Rightarrow \exists z^2(\tilde{\psi}_j^2(x^2, z^2)))$. In some cases the proof of 3. can be reduced into a proof of:
4. $\mathscr{C}_2 \vdash \forall x^2(\tilde{\varphi}_j^2(x^2) \Rightarrow \exists z^2(\tilde{\psi}_j^2(x^2, z^2)))$; if in $\mathscr{C}_2$ the synthesis is possible, the related techniques can be applied. In any case, the construction of $\tilde{\varphi}_j^2$, $\tilde{\psi}_j^2$ can be a useful guide both for verification of formula 4. and for construction of expansions P_j^2.

References

CONSTABLE R. L. (1971). Constructive mathematics and automatic program writers, *IFIP congress 71*, Ljubljana, p. 8.

DAHL, O.-J., DIJKSTRA, E. W., and HOARE, C. A. R. (1972). *Structured Programming*, Academic Press.

DEGLI ANTONI, G., MIGLIOLI, P. A., and ORNAGHI, M. (1974). Top-down approach to the synthesis of programs. *Colloques sur la programmation*, Paris, 9-11 April 1974. Lecture notes on Compt. Sc., Springer-Verlag.

DIJKSTRA, E. W. (1968*a*). A constructive approach to the problem of program correctness, *BIT*, Vol. 8, pp. 174-186.

DIJKSTRA, E. W. (1968*b*). The structure of 'THE' multiprogramming system, *CACM*, Vol. 11, No. 5, p. 341.

DIJKSTRA, E. W. (1969). Structured programming, *Conference on Software Engineering Techniques*, Rome, Oct. 1969.

ELSPAS, B. *et al.* (1972). An assessment of techniques for proving program correctness, *ACM Computing Surveys*, Vol. 4, No. 2, June 1972.

FLOYD, R. W. (1967). Assigning meanings to programs, *Proc. Symp. Appl. Math.* 19, American Math. Soc., pp. 19-32.

HULL, T. E., ENRIGHT, W. H., and SEDGWICK, A. E. (1972). The correctness of numerical algorithms, *ACM SIGPLAN Notices*, Vol. 7, No. 1, pp. 66-73.

JONES, C. B. (1972). Formal development of correct algorithms: an example based on Earley's recogniser, *ACM SIGPLAN Notices*, Vol. 7, No. 1, pp. 150-169.

LONDON, R. L. (1970). Bibliography on proving the correctness of computer programs, *Machine Intelligence*, Vol. 5, Eds. Meltzer, Michie, American Elsevier, pp. 569-580.

MANNA, Z. (1969*a*). The correctness of programs, *J. of Comp. and Sys. Scie.*, Vol. 3.

MANNA, Z. (1969*b*). Properties of programs and the first-order predicate calculus, *JACM*, Vol. 16, No. 2, Ap. 1969.

MANNA, Z. (1970). *Introduction to mathematical theory of computation*, C.S. 382A (Spring 1970), Computer Sc. Dept, Stanford University, (to appear by McGraw-Hill).

MARINI, D., and MIGLIOLI, P. A. (1973). Characterization of programs and their synthesis from a formalized theory, *MFCS Symposium*, High Tatras, Sept. 1973.

MAURER, W. D. (1972). *Theory and practice of algorithm verification.* Memo No. ERL-M 315, Berkeley, Calif., Jan. 1972.

MILNER, R. (1971). An algebraic definition of simulation between programs, *Second Int. Joint Conf. on Artificial Intelligence*, 1-3 Sept. 1971, p. 481.

MILLS, H. (1971). Top-down programming in large systems, in: Courant Comp. Sc. Symp. on *Debugging Techniques in Large Systems*, Randall Rustin ed. pp. 41-55.

WIRTH, N. (1971). Program development by stepwise refinement, *CACM*, April 1971.

WOODGER, M. (1971). On semantic levels in programming, *IFIP Congress 71*, Ljubljana, Aug. 1971.

Downloaded from https://academic.oup.com/comjnl/article/18/1/55/455207 by guest on 19 April 2024