MATHEMATICAL PROBLEMS IN COMPUTATION THEORY
BANACH CENTER PUBLICATIONS, VOLUME 21
PWN – POLISH SCIENTIFIC PUBLISHERS
WARSAW 1988

ON THE USE OF HOMOMORPHISMS FOR PROVING THE EQUIVALENCE OF SOME PROGRAMS

DIMITER SKORDEV

Sofia University, Sofia, Bulgaria

In [5], we tried to show by an example the usefulness of a certain notion of homomorphism for proving the equivalence of some structured programs (in that example the equivalence of two concrete Pascal programs was proved). The homomorphisms which appear in the example in question were in fact partial-multivalued homomorphisms between many-sorted partial-multivalued algebras with only unary operations, although the notion of partial-multivalued homomorphism was introduced in the same paper for the general case of many-sorted partial-multivalued algebras with arbitrary signatures (and even for a more general class of systems). Unfortunately, the description of the signatures of the many-sorted algebras used overburdens heavily the exposition even in the case of only unary operations. Therefore we now consider it desirable to avoid the use of many sorts of objects in the treatment of such examples. This turns out to be possible due to the fact that for each given program of the kind in question we have a corresponding set of memory states (namely the set of all admissible valuations of its variables) and each part of the program can be considered as describing some partial mapping of this set into itself (although a small part of the program usually concerns only a small number of its variables and therefore the above set of memory states could seem not to be the most suitable for the study of such a part).

First of all, let us give the necessary definitions without superfluous generality and in a form convenient for our considerations.

DEFINITION 1. Let A and B be sets, F a partial mapping of A into A, and G a partial mapping of B into B. A *homomorphism* from F to G is any pair (H_0, H_1), where H_0 and H_1 are subsets of the Cartesian product $A \times B$ and the following condition is satisfied: whenever $(a, b) \in H_0$ and $a \in \operatorname{dom} F$,

then $b \in \operatorname{dom} G$ and $(F(a), G(b)) \in H_1$. A *bihomomorphism* from F to G is any homomorphism (H_0, H_1) from F to G such that (H_0^{-1}, H_1^{-1}) is a homomorphism from G to F.

DEFINITION 2. Let A and B be sets, P a partial predicate on A, and Q a partial predicate on B. A *homomorphism* from P to Q is any subset H of the Cartesian product $A \times B$ satisfying the following condition: whenever $(a, b) \in H$ and $a \in \operatorname{dom} P$, then $b \in \operatorname{dom} Q$ and $P(a) = Q(b)$. A *bihomomorphism* from P to Q is any homomorphism H from P to Q such that H^{-1} is a homomorphism from Q to P.

Remark 1. It is clear that, in Definition 1, the pair (H_0, H_1) is a bihomomorphism from F to G iff H_0 and H_1 are subsets of $A \times B$ and whenever $(a, b) \in H_0$, then the conditions $a \in \operatorname{dom} F$ and $b \in \operatorname{dom} G$ are equivalent and imply that $(F(a), G(b)) \in H_1$. In Definition 2, H is a bihomomorphism from P to Q iff H is a subset of $A \times B$ and whenever $(a, b) \in H$, then the conditions $a \in \operatorname{dom} P$ and $b \in \operatorname{dom} Q$ are equivalent and imply that $P(a) = Q(b)$. In the case where F and G, respectively P and Q, are total, the corresponding notions of homomorphism and of bihomomorphism obviously coincide.

Remark 2. The parts of Definitions 1 and 2 concerning homomorphisms can be considered as special cases of the definition given in [5]. In the case of Definition 1, let us consider the signature $(\{0, 1\}, \{\omega\}, \tau)$, where $\tau(\omega) = (0 \to 1)$, and the two-sorted algebras

$$\mathscr{A} = ((A_0, A_1), (F_\omega)), \qquad \mathscr{B} = ((B_0, B_1), (G_\omega))$$

with this signature, where $A_0 = A_1 = A$, $F_\omega = F$, $B_0 = B_1 = B$, $G_\omega = G$. Then the homomorphisms from F to G in the sense of Definition 1 are exactly the homomorphisms from $\mathscr{A}$ to $\mathscr{B}$ in the sense of [5]. In the case of Definition 2, we have to take $A_0 = A$, $B_0 = B$, $A_1 = B_1 = \{\text{true, false}\}$, $F_\omega = P$, $G_\omega = Q$. Let

$$I = \{(\text{true, true}), (\text{false, false})\}.$$

Then H is a homomorphism from P to Q in the sense of Definition 2 iff (H, I) is a homomorphism from $\mathscr{A}$ to $\mathscr{B}$ in the sense of [5].

Remark 3. Usually, one considers total single-valued homomorphisms. In the case of Definition 1, this can be achieved by means of the additional assumption that H_0 and H_1 are (the graphs of) some single-valued mappings of A into B. Then the condition of (H_0, H_1) being a homomorphism (a bihomomorphism) from F to G is equivalent to the following one: whenever $a \in \operatorname{dom} F$, then $H_0(a) \in \operatorname{dom} G$ and $G(H_0(a)) = H_1(F(a))$ (for all a in A, the conditions $a \in \operatorname{dom} F$ and $H_0(a) \in \operatorname{dom} G$ are equivalent and imply the equality $G(H_0(a)) = H_1(F(a))$). In the case of Definition 2, we have to assume that H is (the graph of) a single-valued mapping of A into B. Then the condition

of H being a homomorphism (a bihomomorphism) from P to Q is equivalent to the following one: whenever $a \in \operatorname{dom} P$, then $H(a) \in \operatorname{dom} Q$ and $Q(H(a)) = P(a)$ (for all a in A, the conditions $a \in \operatorname{dom} P$ and $H(a) \in \operatorname{dom} Q$ are equivalent and imply the equality $Q(H(a)) = P(a)$). The notions obtained in this way are special cases of the notions of homomorphism and closed homomorphism which have been studied in the theory of many-sorted partial algebras (cf. [1], [3]; for the case of Definition 1 with coinciding H_0 and H_1 which are single-valued and total, cf. also [2]).

Remark 4. If we combine the assumptions made in Remark 3 with the assumptions that F and G, respectively P and Q, are total then we have the following: (H_0, H_1) is a homomorphism from F to G iff $G(H_0(a)) = H_1(F(a))$ for all a in A, and H is a homomorphism from P to Q iff $Q(H(a)) = P(a)$ for all a in A. Let us note that the usual definition of homomorphism for the case of predicates requires only that $Q(H(a)) = \text{true}$ whenever $P(a) = \text{true}$. We do not adopt this definition here since it is not convenient for our purposes.

We should like to use homomorphisms for the study of Pascal statements built from simple ones by means of forming compound statements, branching and the **while** – **do** – construction. This can be done on the basis of the following propositions, where the meaning of **begin** – **end**, **if** – **then** – **else** and **while** – **do** with mappings instead of statements is obvious (for example, **begin** $F_1; \ldots; F_n$ **end** denotes the composition of the mappings $F_1, \ldots, F_n$):

PROPOSITION 1. *Let $F_1, \ldots, F_n$ be partial mappings of a set A into itself and let $G_1, \ldots, G_n$ be partial mappings of a set B into itself. Let (H_{i-1}, H_i) be a homomorphism (a bihomomorphism) from F_i to G_i, $i = 1, \ldots, n$. Then (H_0, H_n) is a homomorphism (a bihomomorphism) from* **begin** $F_1; \ldots; F_n$ **end** *to* **begin** $G_1; \ldots; G_n$ **end**.

PROPOSITION 2. *Let F_1 and F_2 be partial mappings of a set A into itself, let G_1 and G_2 be partial mappings of a set B into itself, let P be a partial predicate on A and Q a partial predicate on B. Let (H_0, H_1) be a homomorphism (a bihomomorphism) from F_i to G_i, $i = 1, 2$, such that H_0 is a homomorphism (a bihomomorphism) from P to Q. Then (H_0, H_1) is a homomorphism (a bihomomorphism) from* **if** P **then** F_1 **else** F_2 *to* **if** Q **then** G_1 **else** G_2.

PROPOSITION 3. *Let F be a partial mapping of a set A into itself, G a partial mapping of a set B into itself, P a partial predicate on A and Q a partial predicate on B. Let H be a homomorphism (a bihomomorphism) from P to Q such that (H, H) is a homomorphism (a bihomomorphism) from F to G. Then (H, H) is a homomorphism (a bihomomorphism) from* **while** P **do** F *to* **while** Q **do** G.

The proofs of these propositions are straightforward and will not be given here. Let us note that Propositions 1 and 2 follow immediately from

Propositions 2 and 5 of [4], on taking into account that the notions of homomorphism and bihomomorphism of the present paper correspond to the notions of semi-homomorphism and homomorphism of [4], respectively; as to. Proposition 3 above, its part concerning bihomomorphisms can be obtained by an application of Proposition 6 of [4].

In order to show the use of Definitions 1, 2 and Propositions 1, 2, 3, we shall consider again the example from [5], where the equivalence of the following two Pascal programs was proved (for the time being, let us pay no attention to the symbols in brackets on the left whose meaning will be explained later):

```
        program f (input, output);
          var n, x, y, z: integer;
(F)       begin
(F1)          x := 1;
(F2)          y := 1;
(F3)          read(n);
(F4)          while
(P0)              n > 1
              do
(F0)              begin
                    z := x+y; x := y; y := z; n := n-1
                  end;
(F5)          write(y)
          end.
        program g (input, output);
          var n, x, y: integer;
(G)       begin
(G1)          x := 1;
(G2)          y := 1;
(G3)          read(n);
(G4)          while
(Q0)              n > 1
              do
(G0)              begin
                    if x > y then y := x+y else x := x+y;
                    n := n-1
                  end;
(G5)          if x > y then write(x) else write(y)
          end.
```

Let A be the set of all admissible valuations of the variables of **program** f, namely the variables n, x, y, z, and the file variables input and output, where, for simplicity, we assume the finite sequences of integers to be the

admissible values of input and output (intuitively, such a sequence, taken as a value of input, represents the integers waiting to be read and, taken as a value of output, represents the integers which have already been written). In a similar way, let B be the set of all admissible valuations of the variables of **program** g (the admissible values of input and output are taken to be the same as above).

An arbitrary element of A will be represented by a table of the form

$$\begin{pmatrix} n & x & y & z & \text{input} & \text{output} \\ n_1 & x_1 & y_1 & z_1 & \text{input}_1 & \text{output}_1 \end{pmatrix}. \tag{1}$$

where n_1, x_1, y_1, z_1 are integers and input_1, output_1 are finite sequences of integers (in order to avoid some inessential complications, just as in [5] we will not bother about maxint). The analogous representation of an arbitrary element of B is

$$\begin{pmatrix} n & x & y & \text{input} & \text{output} \\ n_2 & x_2 & y_2 & \text{input}_2 & \text{output}_2 \end{pmatrix}. \tag{2}$$

Let each of the letters F, F_0–F_5 which occur on the left of **program** f denote the partial mapping of A into itself which is described by the statement beginning on the same line (hence F can be considered as the action of **program** f). Let P_0 be the predicate on A described by the Boolean expression on the corresponding line (hence P_0 has value true for the elements (1) of A satisfying the condition $n_1 > 1$, and has value false for all other elements of A). Let the meaning of G, G_0–G_5 and Q_0 be defined in a similar way. We have the following equalities:

$$F = \textbf{begin } F_1;\ F_2;\ F_3;\ F_4;\ F_5 \textbf{ end}, \tag{3}$$

$$G = \textbf{begin } G_1;\ G_2;\ G_3;\ G_4;\ G_5 \textbf{ end}, \tag{4}$$

$$F_4 = \textbf{while } P_0 \textbf{ do } F_0, \tag{5}$$

$$G_4 = \textbf{while } Q_0 \textbf{ do } G_0. \tag{6}$$

We shall introduce now some subsets H_{start}, H_{final}, H_1, H_2, H_3 of the Cartesian product $A \times B$. Each of them will be defined as the set of all pairs of elements (1), (2) which satisfy a corresponding condition. These conditions are chosen as follows:

for H_{start}: $\text{input}_1 = \text{input}_2$, $\text{output}_1 = \text{output}_2 = \Lambda$, where Λ is the empty sequence;

for H_{final}: $\text{output}_1 = \text{output}_2$;

for H_1: $x_1 = x_2 = 1$, $\text{input}_1 = \text{input}_2$, $\text{output}_1 = \text{output}_2 = \Lambda$;

for H_2: $x_1 = x_2 = y_1 = y_2 = 1$, $\text{input}_1 = \text{input}_2$, $\text{output}_1 = \text{output}_2 = \Lambda$;

for H_3: $n_1 = n_2$, $x_1 = \min\{x_2, y_2\}$, $y_1 = \max\{x_2, y_2\}$, $x_2 \geqslant 0$, $y_2 \geqslant 0$, $\text{input}_1 = \text{input}_2$, $\text{output}_1 = \text{output}_2 = \Lambda$.

The equivalence of **program** f and **program** g will be established by proving that $(H_{\text{start}}, H_{\text{final}})$ is a bihomomorphism from F to G. In view of equalities (3), (4) and Proposition 1, it is sufficient to prove the following assertions:

1. (H_{start}, H_1) is a bihomomorphism from F_1 to G_1.
2. (H_1, H_2) is a bihomomorphism from F_2 to G_2.
3. (H_2, H_3) is a bihomomorphism from F_3 to G_3.
4. (H_3, H_3) is a bihomomorphism from F_4 to G_4.
5. (H_3, H_{final}) is a bihomomorphism from F_5 to G_5.

Assertions 1, 2, 3 are obviously true (note only that the mappings F_3 and G_3 are not total). We shall give the proofs of the remaining two assertions.

Proof of assertion 4. In view of equalities (5), (6) and Proposition 3, it is sufficient to prove that H_3 is a bihomomorphism from P_0 to Q_0 (this is obvious) and that (H_3, H_3) is a bihomomorphism from F_0 to G_0. Suppose an element (1) of A and an element (2) of B are given. The application of F_0 and G_0 to (1) and (2), respectively, gives the elements

$$(1') \qquad \begin{pmatrix} n & x & y & z & \text{input} & \text{output} \\ n_1-1 & y_1 & x_1+y_1 & x_1+y_1 & \text{input}_1 & \text{output}_1 \end{pmatrix},$$

$$(2') \qquad \begin{pmatrix} n & x & y & \text{input} & \text{output} \\ n_2-1 & x_2' & y_2' & \text{input}_2 & \text{output}_2 \end{pmatrix},$$

where

$$\{x_2', y_2'\} = \begin{cases} \{x_2, x_2+y_2\} & \text{if } x_2 > y_2, \\ \{x_2+y_2, y_2\} & \text{otherwise.} \end{cases}$$

Suppose now the pair (1), (2) belongs to H_3. Then

$$\min\{x_2', y_2'\} = \max\{x_2, y_2\} = y_1, \qquad \max\{x_2', y_2'\} = x_2+y_2 = x_1+y_1,$$

no matter whether $x_2 > y_2$ or not. Since the equality $n_1 = n_2$ implies $n_1 - 1 = n_2 - 1$, it is now clear that the pair (1'), (2') also belongs to H_3.

Proof of assertion 5. Let the pair (1), (2) belong to H_3. Then the application of F_5 and G_5 to (1) and (2), respectively, gives the elements

$$(1'') \qquad \begin{pmatrix} n & x & y & z & \text{input} & \text{output} \\ n_1 & x_1 & y_1 & z_1 & \text{input}_1 & y_1 \end{pmatrix},$$

$$(2'') \qquad \begin{pmatrix} n & x & y & \text{input} & \text{output} \\ n_2 & x_2 & y_2 & \text{input}_2 & \max\{x_2, y_2\} \end{pmatrix}.$$

Obviously, the pair (1''), (2'') belongs to H_{final}.

Thus the equivalence of **program** f and **program** g is proved (without

making use of Proposition 2 and using only that part of Propositions 1 and 3 which concerns bihomomorphisms).

It is easy to give examples where all Propositions 1, 2, 3 have to be used. Clearly, only their part concerning bihomomorphisms will be essential when one proves the equivalence of programs. However, the case of homomorphisms can be useful for some other kinds of transference of properties of a given program to another one, for example for proving interrelations of the following form: whenever the first program gives a result, the second one gives the same result.

In the example which we have considered, there is a sophisticated step, namely the choice of H_3. This is connected with the fact that **program** g cannot be obtained from **program** f by an absolutely simple transformation. Other examples can be given concerning pairs of programs where the second one can be obtained from the first in some much simpler way. In such cases it would be much easier to choose the necessary homomorphisms or bihomomorphisms. For example, consider the above **program** f and another program which is obtained by replacing the line

$$z := x+y; \quad x := y; \quad y := z; \quad n := n-1$$

in **program** f by the following one:

$$z := y; \quad y := x+y; \quad x := z; \quad n := n-1.$$

Then the equivalence of the two programs can be proved using quite simple bihomomorphisms. In particular, it is convenient to use instead of H_3 the set of all pairs of valuations (1) which coincide on all their variables with the possible exception of z.

We should like to note that it is also possible to study in a similar way some pairs of programs written in different programming languages which admit the constructions considered in Propositions 1, 2, 3. Of course, analogs of these propositions are true for many other programming constructions.

So far, we have considered only deterministic programs. In order to be able to consider nondeterministic ones, we have to generalize Definitions 1 and 2 by allowing F, G, P and Q to be multivalued. Then the condition in Definition 1 of (H_0, H_1) being a homomorphism from F to G must be replaced by the following on: whenever $(a, b) \in H_0$ and a' is a value of $F(a)$, then there is a value b' of $G(b)$ such that $(a', b') \in H_1$. The condition in Definition 2 of H being a homomorphism from P to Q must be replaced by the following one: whenever $(a, b) \in H$, then each value of $P(a)$ is also a value of $Q(b)$. Of course, H_0, H_1 and H are still subsets of the Cartesian product $A \times B$. After defining the notion of homomorphism in this way, we can reduce the notion of bihomomorphism to it in the same way as in Definitions 1 and 2.

References

[1] P. Burmeister, *A model theoretic approach to partial algebras*, Akademie-Verlag, Berlin 1985.

[2] G. Grätzer, *Universal algebra*, Springer-Verlag, New York–Heidelberg–Berlin 1979.

[3] H. Reichel, *Structural induction on partial algebras*, Akademie-Verlag, Berlin 1984.

[4] D. Skordev, *On multivalued homomorphisms in the theory of computability*, In: 7th International Congress of Logic, Methodology and Philosophy of Science (Salzburg, Austria, July 11–16, 1983), vol. 1, Abstracts of Sections 1, 2, 3, 4 and 7, 152–155.

[5] –, *On multi-valued homomorphisms*, In: Computation Theory, Lecture Notes in Computer Science, Springer-Verlag, vol. 208, 1985, 326–331.

Presented to the semester
Mathematical Problems in Computation Theory
September 16–December 14, 1985