FORMALIZED MATHEMATICS
Vol. 34, No. 1, pp. 59–72, 2026
DOI: 10.2478/forma-2026-0005
e-ISSN: 1898–9934

Formal Verification of a Generic Iterative Algorithm

Artur Korniłowicz
Faculty of Computer Science
University of Białystok
Poland

Summary. In this paper we formally verify in the Mizar system the correctness of a parameterized algorithm using SCM (Simple Concrete Model). Generic properties of the algorithm are formulated and proved as Mizar schemes. In the last section two classical cases: the computation of the factorial of a natural number and the computation of the n-th triangular number are verified using the established framework.

MSC: 68Q25 68W40 68V15 68V20

Keywords: formal verification; generic algorithm; factorial; triangle number

MML identifier: `AMI_8`, version: `8.1.15 5.103.1512`

Introduction

Formal verification is one of the fundamental areas of formal methods, providing mathematical guarantees that a program satisfies its specification. Unlike testing, which can only increase confidence in the correctness of an implementation, formal verification establishes correctness for all admissible inputs. Interactive theorem provers, such as Mizar [2, 3], play an important role in this process by enabling the formalization of mathematical concepts, computational models, and correctness proofs of algorithms.

The aim of this paper is to formally verify two classical iterative algorithms: the computation of the factorial of a natural number and the computation of the n-th triangular number. Rather than verifying these algorithms independently, we consider their common control structure, represented by the following generic program:

```
0) if b > 0 then goto 2
1) halt
2) OPER
3) b := b-a
4) if b = 0 then goto 1
5) goto 2
```

where a, b, and c represent data-locations and a stores 1 as a loop step, b stores an input value, and c stores the result value.

The program is parameterized by the instruction OPER, which specifies the operation performed during each iteration of the loop. Consequently, the same control-flow structure can be used to implement different algorithms by instantiating OPER with an appropriate arithmetic instruction. The algorithm is formalized in the Mizar system using the SCM (Simple Concrete Model) [4]. The program is represented as a finite sequence of SCM instructions by the definition of the function IterativeAlgorithm(OPER), whose only parameter is the instruction executed in the loop body. This formulation cleanly separates the properties that depend solely on the control structure from those determined by the particular arithmetic operation. The correctness proof is carried out within the framework of operational semantics [5], where the execution of a program is modeled as a sequence of state transitions.

The analysis focuses on the evolution of the instruction counter (IC) and the contents of the registers representing the program variables (dl.0, dl.1, and dl.2) throughout the execution. Properties that are independent of the concrete definition of OPER are formulated and proved as a collection of generic Mizar schemes. These reusable results capture the behavior of the program's control flow and provide a common basis for verifying different instances of the algorithm.

In the final section of the paper, the generic schemes are instantiated to verify two concrete algorithms. In the first case, OPER is defined as the multiplication instruction MultBy(dl.2,dl.1), yielding a formal proof of correctness for the computation of $n!$. In the second case, OPER is instantiated by the addition instruction AddTo(dl.2,dl.1), leading to a proof of correctness for the computation of the n-th triangular number.

These case studies demonstrate that the developed framework provides a reusable methodology for the formal verification of a family of iterative algorithms sharing the same control-flow structure while differing only in the operation performed within the loop.

1. Preliminaries

From now on x_0, x_1, x_2, x_3, x_4, x_5, x_6 denote objects, f denotes a function, j, k, m, n denote natural numbers, and i, i_1, i_2, i_3 denote integers.

Let a, b, c, x, y, z be objects. Observe that $(a, b, c) \mapsto (x, y, z)$ is finite.

Now we state the propositions:

(1) $\mathrm{dom}\langle x_0, x_1, x_2, x_3\rangle = \{0, 1, 2, 3\}$.

(2) $\mathrm{len}\langle x_0, x_1, x_2, x_3\rangle = 4$.

(3) Let us consider functions f, g, and objects w, x, y, z. Suppose $\mathrm{dom}\, f = \mathrm{dom}\, g$ and $f(w) = g(w)$ and $f(x) = g(x)$ and $f(y) = g(y)$ and $f(z) = g(z)$. Then $f{\restriction}\{w, x, y, z\} = g{\restriction}\{w, x, y, z\}$.

2. SEQUENCES

Let us consider x_0, x_1, x_2, x_3, and x_4. The functor $\langle x_0, x_1, x_2, x_3, x_4\rangle$ yielding a set is defined by the term

(Def. 1) $\langle x_0, x_1, x_2, x_3\rangle \frown \langle x_4\rangle$.

Note that $\langle x_0, x_1, x_2, x_3, x_4\rangle$ is function-like and relation-like and $\langle x_0, x_1, x_2, x_3, x_4\rangle$ is finite and transfinite sequence-like.

Now we state the propositions:

(4) $\mathrm{dom}\langle x_0, x_1, x_2, x_3, x_4\rangle = \{0, 1, 2, 3, 4\}$. The theorem is a consequence of (1).

(5) $\mathrm{len}\langle x_0, x_1, x_2, x_3, x_4\rangle = 5$.

Let us consider x_0, x_1, x_2, x_3, and x_4. One can check that $\langle x_0, x_1, x_2, x_3, x_4\rangle(0)$ reduces to x_0 and $\langle x_0, x_1, x_2, x_3, x_4\rangle(1)$ reduces to x_1 and $\langle x_0, x_1, x_2, x_3, x_4\rangle(2)$ reduces to x_2 and $\langle x_0, x_1, x_2, x_3, x_4\rangle(3)$ reduces to x_3 and $\langle x_0, x_1, x_2, x_3, x_4\rangle(4)$ reduces to x_4.

Let us consider x_5. The functor $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle$ yielding a set is defined by the term

(Def. 2) $\langle x_0, x_1, x_2, x_3, x_4\rangle \frown \langle x_5\rangle$.

One can check that $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle$ is function-like and relation-like and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle$ is finite and transfinite sequence-like.

Now we state the propositions:

(6) $\mathrm{dom}\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle = \{0, 1, 2, 3, 4, 5\}$. The theorem is a consequence of (4).

(7) $\mathrm{len}\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle = 6$.

Let us consider x_0, x_1, x_2, x_3, x_4, and x_5. Let us note that $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(0)$ reduces to x_0 and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(1)$ reduces to x_1 and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(2)$ reduces to x_2 and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(3)$ reduces to x_3 and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(4)$ reduces to x_4 and $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle(5)$ reduces to x_5.

Now we state the proposition:

(8) If $\langle x_0, x_1, x_2, x_3, x_4, x_5\rangle \subseteq f$, then $f(0) = x_0$ and $f(1) = x_1$ and $f(2) = x_2$ and $f(3) = x_3$ and $f(4) = x_4$ and $f(5) = x_5$. The theorem is a consequence of (6).

3. General Theory of SCM

From now on s denotes a state of **SCM**, I denotes an instruction of **SCM**, P denotes an instruction sequence of **SCM**, and X denotes a finite 0-sequence of the instructions of **SCM**.

Now we state the proposition:

(9) Let us consider a data-location d. Then $i \in$ (the values of **SCM**)(d).

Let l_1 be a data-location and a be an integer. Observe that $l_1 \dot{\longmapsto} a$ is (the values of **SCM**)-compatible.

Let l_1, l_2 be data-locations and a, b be integers. Let us observe that $[l_1 \longmapsto a, l_2 \longmapsto b]$ is (the values of **SCM**)-compatible.

Let l_1, l_2, l_3 be data-locations and a, b, c be integers. One can verify that $(l_1, l_2, l_3) \mapsto (a, b, c)$ is (the values of **SCM**)-compatible.

Let d be a data-location and i be an integer. One can check that $d \dot{\longmapsto} i$ is data-only as a part state of **SCM**.

Let p, q be data-only finite partial states of **SCM**. Let us note that $p +\cdot q$ is data-only.

Let l_1, l_2 be data-locations and a, b be integers. Note that $[l_1 \longmapsto a, l_2 \longmapsto b]$ is data-only as a finite partial state of **SCM**.

Let l_1, l_2, l_3 be data-locations and a, b, c be integers. One can verify that $(l_1, l_2, l_3) \mapsto (a, b, c)$ is data-only as a finite partial state of **SCM**.

Let us consider n and I. We say that I is n-dl.constant if and only if

(Def. 3) for every instruction sequence P of **SCM** and for every state s of **SCM** and for every natural number k such that $\mathrm{CurInstr}(P, \mathrm{Comput}(P, s, k)) = I$ holds $(\mathrm{Comput}(P, s, k+1))(\mathbf{d}_n) = (\mathrm{Comput}(P, s, k))(\mathbf{d}_n)$.

Now we state the propositions:

(10) If $m \neq j$, then $\mathrm{AddTo}(\mathbf{d}_m, \mathbf{d}_n)$ is j-dl.constant.

(11) If $m \neq j$, then $\mathrm{SubFrom}(\mathbf{d}_m, \mathbf{d}_n)$ is j-dl.constant.

(12) If $m \neq j$, then $\mathrm{MultBy}(\mathbf{d}_m, \mathbf{d}_n)$ is j-dl.constant.

Let us consider m. Let n be a non zero natural number. Let us note that $\mathrm{AddTo}(\mathbf{d}_{m+n}, \mathbf{d}_m)$ is m-dl.constant and $\mathrm{SubFrom}(\mathbf{d}_{m+n}, \mathbf{d}_m)$ is m-dl.constant and $\mathrm{MultBy}(\mathbf{d}_{m+n}, \mathbf{d}_m)$ is m-dl.constant and $\mathrm{AddTo}(\mathbf{d}_2, \mathbf{d}_1)$ is 0-dl.constant and $\mathrm{SubFrom}(\mathbf{d}_2, \mathbf{d}_1)$ is 0-dl.constant and $\mathrm{MultBy}(\mathbf{d}_2, \mathbf{d}_1)$ is 0-dl.constant.

Let us consider n. Note that there exists an instruction of **SCM** which is sequential and n-dl.constant.

Now we state the propositions:

(13) Let us consider a sequential instruction I of **SCM**. Suppose CurInstr(P, Comput(P, I and $\mathbf{IC}_{\mathrm{Comput}(P,s,n)} \in \mathrm{dom}\, P$. Then $\mathbf{IC}_{\mathrm{Comput}(P,s,n+1)} = \mathbf{IC}_{\mathrm{Comput}(P,s,n)} + 1$.

(14) Let us consider an n-started state s consisting of $\langle i_1, i_2, i_3 \rangle$. Then

(i) $s(\mathbf{d}_0) = i_1$, and

(ii) $s(\mathbf{d}_1) = i_2$, and

(iii) $s(\mathbf{d}_2) = i_3$.

Let us consider X. One can check that there exists an instruction sequence of **SCM** which is X-extending.

Now we state the proposition:

(15) Let us consider a state s of **SCM**. Suppose Initialize($(\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (i_1, i_2, i_3)) \subseteq s$. Then s is a state consisting of $\langle i_1, i_2, i_3 \rangle$.

4. Correctness of Algorithms Computing Operations on One Previous Element

Let OPER be an instruction of **SCM**. The functor IterativeAlgorithm(OPER) yielding a finite 0-sequence of the instructions of **SCM** is defined by the term

(Def. 4) $\langle$**if** $\mathbf{d}_1 > 0$ **goto** 2, $\mathbf{halt}_{\mathbf{SCM}}$, OPER, SubFrom($\mathbf{d}_1, \mathbf{d}_0$), **if** $\mathbf{d}_1 = 0$ **goto** 1, SCM-goto 2

Now we state the propositions:

(16) Suppose IterativeAlgorithm(I) $\subseteq P$. Suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 0$. Then

(i) if (Comput(P, s, k))($\mathbf{d}_1$) > 0, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 2$, and

(ii) if (Comput(P, s, k))($\mathbf{d}_1$) $\leqslant 0$, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 1$, and

(iii) for every j, (Comput($P, s, k+1$))($\mathbf{d}_j$) = (Comput(P, s, k))($\mathbf{d}_j$).

PROOF: If (Comput(P, s, k))($\mathbf{d}_1$) $\leqslant 0$, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 1$ by [4, (9)]. □

(17) If IterativeAlgorithm(I) $\subseteq P$, then P halts at 1. The theorem is a consequence of (8).

(18) Suppose IterativeAlgorithm(I) $\subseteq P$. If $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 1$, then Comput($P, s, k+j$) = Comput($P, s, k$). The theorem is a consequence of (17).

Let us consider k. Now we state the propositions:

(19) Suppose IterativeAlgorithm(AddTo($\mathbf{d}_m, \mathbf{d}_n$)) $\subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 2$. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 3$, and

(ii) $(\mathrm{Comput}(P,s,k{+}1))(\mathbf{d}_m) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_m){+}(\mathrm{Comput}(P,s,k))(\mathbf{d}_n)$, and

(iii) for every j such that $j \neq m$ holds $(\mathrm{Comput}(P,s,k+1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_j)$.

The theorem is a consequence of (8).

(20) Suppose $\mathrm{IterativeAlgorithm}(\mathrm{SubFrom}(\mathbf{d}_m,\mathbf{d}_n)) \subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)}$ 2. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 3$, and

(ii) $(\mathrm{Comput}(P,s,k{+}1))(\mathbf{d}_m) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_m){-}(\mathrm{Comput}(P,s,k))(\mathbf{d}_n)$, and

(iii) for every j such that $j \neq m$ holds $(\mathrm{Comput}(P,s,k+1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_j)$.

The theorem is a consequence of (8).

(21) Suppose $\mathrm{IterativeAlgorithm}(\mathrm{MultBy}(\mathbf{d}_m,\mathbf{d}_n)) \subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)}$ = 2. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 3$, and

(ii) $(\mathrm{Comput}(P,s,k{+}1))(\mathbf{d}_m) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_m){\cdot}(\mathrm{Comput}(P,s,k))(\mathbf{d}_n)$, and

(iii) for every j such that $j \neq m$ holds $(\mathrm{Comput}(P,s,k+1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_j)$.

The theorem is a consequence of (8).

(22) Suppose $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 3$. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 4$, and

(ii) $(\mathrm{Comput}(P,s,k{+}1))(\mathbf{d}_1) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_1){-}(\mathrm{Comput}(P,s,k))(\mathbf{d}_0)$, and

(iii) for every j such that $j \neq 1$ holds $(\mathrm{Comput}(P,s,k{+}1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))($

The theorem is a consequence of (8).

(23) Suppose $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 4$. Then

(i) if $(\mathrm{Comput}(P,s,k))(\mathbf{d}_1) = 0$, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 1$, and

(ii) if $(\mathrm{Comput}(P,s,k))(\mathbf{d}_1) \neq 0$, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 5$, and

(iii) for every j, $(\mathrm{Comput}(P,s,k+1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_j)$.

PROOF: If $(\mathrm{Comput}(P,s,k))(\mathbf{d}_1) \neq 0$, then $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 5$ by [4, (8)]. □

(24) Suppose $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Then suppose $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 5$. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,k+1)} = 2$, and

(ii) for every j, $(\mathrm{Comput}(P,s,k+1))(\mathbf{d}_j) = (\mathrm{Comput}(P,s,k))(\mathbf{d}_j)$.

The theorem is a consequence of (8).

Let us consider I. One can verify that $\mathrm{IterativeAlgorithm}(I)$ is (the instructions of **SCM**)-valued and $\mathrm{IterativeAlgorithm}(I)$ is non halt-free.

Now we state the propositions:

(25) Suppose I is sequential and $\mathrm{IterativeAlgorithm}(I) \subseteq P$ and $\mathbf{IC}_{\mathrm{Comput}(P,s,0)} = 0$. Then $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} \in \{0,1,2,3,4,5\}$.

PROOF: Define $\mathcal{P}[\text{natural number}] \equiv \mathbf{IC}_{\mathrm{Comput}(P,s,\$_1)} \in \{0,1,2,3,4,5\}$. For every j such that $\mathcal{P}[j]$ holds $\mathcal{P}[j+1]$ by (16), (18), (8), [**?** , (3), (28)]. For every j, $\mathcal{P}[j]$ from [1, Sch. 2]. □

(26) Suppose I is sequential and 0-dl.constant and $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Let us consider a 0-started state s consisting of $\langle i, i_1, i_2\rangle$. Then $(\mathrm{Comput}(P,s,k))(\mathbf{d}$ i.

PROOF: Set $a = \mathbf{d}_0$. Define $\mathcal{P}[\text{natural number}] \equiv (\mathrm{Comput}(P,s,\$_1))(a) = i$. $\mathcal{P}[0]$. For every k such that $\mathcal{P}[k]$ holds $\mathcal{P}[k+1]$ by (25), (16), (18), (22). For every k, $\mathcal{P}[k]$ from [1, Sch. 2]. □

(27) Suppose I is sequential and 0-dl.constant. Let us consider an instruction sequence P of **SCM**. Suppose $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Let us consider a 0-started state s consisting of $\langle i_1, 0, i_2\rangle$. Then

(i) $\mathbf{IC}_{\mathrm{Comput}(P,s,0)} = 0$, and

(ii) for every natural number k such that $k \geqslant 1$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,k)} = 1$, and

(iii) for every natural number k, $(\mathrm{Comput}(P,s,k))(\mathbf{d}_0) = i_1$ and $(\mathrm{Comput}(P,s,k))(\mathbf{d}$ 0 and $(\mathrm{Comput}(P,s,k))(\mathbf{d}_2) = i_2$, and

(iv) $\mathrm{LifeSpan}(P,s) = 1$.

The theorem is a consequence of (14), (16), (18), (26), and (8).

Let us consider an integer A, a natural number N, an instruction sequence P of **SCM**, and a 0-started state s consisting of $\langle 1, N, A\rangle$. Now we state the propositions:

(28) Suppose $\mathrm{IterativeAlgorithm}(I) \subseteq P$ and I is sequential and 0-dl.constant. Then suppose $(\mathrm{Comput}(P,s,2))(\mathbf{d}_1) = (\mathrm{Comput}(P,s,1))(\mathbf{d}_1)$ and for every natural number k, $(\mathrm{Comput}(P,s,4\cdot k+6))(\mathbf{d}_1) = (\mathrm{Comput}(P,s,4\cdot k+5))(\mathbf{d}_1)$. Then

(i) P is halting on s, and

(ii) $\mathbf{IC}_{\mathrm{Comput}(P,s,0)} = 0$, and

(iii) $(\mathrm{Comput}(P,s,1))(\mathbf{d}_2) = A$, and

(iv) for every natural number k such that $0 < k < N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k)} = 5$, and

(v) for every natural number k such that $k < N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+1)} = 2$ and $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+2)} = 3$ and $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+3)} = 4$ and $(\mathrm{Comput}(P,s,4\cdot k+1))(\mathbf{d}_1) = N - k$ and $(\mathrm{Comput}(P,s,4\cdot k+2))(\mathbf{d}_1) = N - k$ and $(\mathrm{Comput}(P,s,4\cdot k+3))(\mathbf{d}_1) = N - k - 1$ and $(\mathrm{Comput}(P,s,4\cdot k+4))(\mathbf{d}_1) = N - k - 1$ and $(\mathrm{Comput}(P,s,4\cdot k))(\mathbf{d}_1) > 0$ and $(\mathrm{Comput}(P,s,4\cdot k+1))(\mathbf{d}_1) > 0$ and $(\mathrm{Comput}(P,s,4\cdot k+2))(\mathbf{d}_1) > 0$, and

(vi) if $N \neq 0$, then $\mathrm{LifeSpan}(P,s) = 4\cdot N$ and for every natural numbers k, j such that $k \geqslant N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+j)} = 1$ and $(\mathrm{Comput}(P,s,4\cdot k+1+j))(\mathbf{d}_1) = 0$.

The theorem is a consequence of (8), (14), (16), (6), (13), (26), (22), (23), (24), and (18).

(29) Suppose $(I = \mathrm{AddTo}(\mathbf{d}_2,\mathbf{d}_1)$ or $I = \mathrm{SubFrom}(\mathbf{d}_2,\mathbf{d}_1)$ or $I = \mathrm{MultBy}(\mathbf{d}_2,\mathbf{d}_1))$ and $\mathrm{IterativeAlgorithm}(I) \subseteq P$. Then

(i) P is halting on s, and

(ii) $(\mathrm{Comput}(P,s,1))(\mathbf{d}_2) = A$, and

(iii) for every natural number k such that $0 < k < N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k)} = 5$, and

(iv) for every natural number k such that $k < N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+1)} = 2$ and $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+2)} = 3$ and $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+3)} = 4$ and $(\mathrm{Comput}(P,s,4\cdot k+1))(\mathbf{d}_1) = N - k$ and $(\mathrm{Comput}(P,s,4\cdot k+2))(\mathbf{d}_1) = N - k$ and $(\mathrm{Comput}(P,s,4\cdot k+3))(\mathbf{d}_1) = N - k - 1$ and $(\mathrm{Comput}(P,s,4\cdot k+4))(\mathbf{d}_1) = N - k - 1$ and $(\mathrm{Comput}(P,s,4\cdot k))(\mathbf{d}_1) > 0$ and $(\mathrm{Comput}(P,s,4\cdot k+1))(\mathbf{d}_1) > 0$ and $(\mathrm{Comput}(P,s,4\cdot k+2))(\mathbf{d}_1) > 0$, and

(v) if $N \neq 0$, then $\mathrm{LifeSpan}(P,s) = 4\cdot N$ and for every natural numbers k, j such that $k \geqslant N$ holds $\mathbf{IC}_{\mathrm{Comput}(P,s,4\cdot k+j)} = 1$ and $(\mathrm{Comput}(P,s,4\cdot k+1+j))(\mathbf{d}_1) = 0$.

The theorem is a consequence of (8), (14), (16), (19), (20), (21), (22), (23), (24), and (18).

Let us consider i_1, i_2, k, and n. Let s be a 0-started state consisting of $\langle n, i_1, i_2\rangle$, I be a sequential, 0-dl.constant instruction of **SCM**, and P be a (IterativeAlgorit extending instruction sequence of **SCM**. Let us observe that $(\mathrm{Comput}(P, s, k))(\mathbf{d}_0)$ is natural.

Now we state the proposition:

(30) Suppose $I = \mathrm{AddTo}(\mathbf{d}_2, \mathbf{d}_1)$ or $I = \mathrm{SubFrom}(\mathbf{d}_2, \mathbf{d}_1)$ or $I = \mathrm{MultBy}(\mathbf{d}_2, \mathbf{d}_1)$. Let us consider a 0-started state s consisting of $\langle 1, n, i\rangle$, and a (IterativeAlgorithm(I)) extending instruction sequence P of **SCM**. Then $(\mathrm{Comput}(P, s, k))(\mathbf{d}_1)$ is natural. The theorem is a consequence of (27), (14), and (29).

Let us consider i, k, and n. Let s be a 0-started state consisting of $\langle 1, n, i\rangle$ and P be a (IterativeAlgorithm(AddTo$(\mathbf{d}_2, \mathbf{d}_1)$))-extending instruction sequence of **SCM**. One can verify that $(\mathrm{Comput}(P, s, k))(\mathbf{d}_1)$ is natural.

Let P be a (IterativeAlgorithm(MultBy$(\mathbf{d}_2, \mathbf{d}_1)$))-extending instruction sequence of **SCM**. Let us note that $(\mathrm{Comput}(P, s, k))(\mathbf{d}_1)$ is natural.

The scheme *AddOfOneSch* deals with an integer $\mathcal{A}$ and a natural number $\mathcal{N}$ and a unary functor $\mathcal{F}$ yielding an integer and states that

(Sch. 1) For every (IterativeAlgorithm(AddTo$(\mathbf{d}_2, \mathbf{d}_1)$))-extending instruction sequence P of **SCM** for every 0-started state s consisting of $\langle 1, \mathcal{N}, \mathcal{A}\rangle$, for every natural number k such that $k < \mathcal{N}$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_2) = \mathcal{A} + (\mathcal{F}(\mathcal{N}) - \mathcal{F}((\mathrm{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_1)))$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_2) = \mathcal{A} + (\mathcal{F}(\mathcal{N}) - \mathcal{F}((\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_1) -' 1))$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 3))(\mathbf{d}_2) = (\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_2)$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 4))(\mathbf{d}_2) = (\mathrm{Comput}(P, s, 4 \cdot k + 3))(\mathbf{d}_2)$ and for every natural numbers k, j such that $k \geqslant \mathcal{N}$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1 + j))(\mathbf{d}_2) = \mathcal{F}(\mathcal{N})$ and $(\mathrm{Result}(P, s))(\mathbf{d}_2) = \mathcal{F}(\mathcal{N})$

provided

- $\mathcal{F}(0) = \mathcal{A}$ and
- for every natural number n, $\mathcal{F}(n + 1) = n + 1 + \mathcal{F}(n)$.

The scheme *MultOfOneSch* deals with an integer $\mathcal{A}$ and a natural number $\mathcal{N}$ and a unary functor $\mathcal{F}$ yielding an integer and states that

(Sch. 2) For every (IterativeAlgorithm(MultBy$(\mathbf{d}_2, \mathbf{d}_1)$))-extending instruction sequence P of **SCM** for every 0-started state s consisting of $\langle 1, \mathcal{N}, \mathcal{A}\rangle$, for every natural number k such that $k < \mathcal{N}$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_2) = \mathcal{A} \cdot \frac{\mathcal{F}(\mathcal{N})}{\mathcal{F}((\mathrm{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_1))}$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_2) = \mathcal{A} \cdot \frac{\mathcal{F}(\mathcal{N})}{\mathcal{F}((\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_1) -' 1)}$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 3))(\mathbf{d}_2) = (\mathrm{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_2)$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 4))(\mathbf{d}_2) = (\mathrm{Comput}(P, s, 4 \cdot k + 3))(\mathbf{d}_2)$ and for every natural numbers k, j such that $k \geqslant \mathcal{N}$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1 + j))(\mathbf{d}_2) = \mathcal{F}(\mathcal{N})$ and $(\mathrm{Result}(P, s))(\mathbf{d}_2) = \mathcal{F}(\mathcal{N})$

provided

- $\mathcal{F}(0) = \mathcal{A}$ and
- $\mathcal{A} \neq 0$ and
- for every natural number n, $\mathcal{F}(n+1) = (n+1) \cdot \mathcal{F}(n)$.

Now we state the propositions:

(31) Let us consider a (IterativeAlgorithm(AddTo($\mathbf{d}_2, \mathbf{d}_1$)))-extending instruction sequence P of **SCM**, and a 0-started state s consisting of $\langle 1, n, 0 \rangle$. Then

(i) for every natural number k such that $k < n$ holds (Comput($P, s, 4 \cdot k + 1$))($\mathbf{d}_2$) = Triangle n − Triangle(Comput($P, s, 4 \cdot k + 1$))($\mathbf{d}_1$) and (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) = Triangle n − Triangle((Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_1$) $-'$ 1) and (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) and (Comput($P, s, 4 \cdot k + 4$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$), and

(ii) for every natural numbers k, j such that $k \geqslant n$ holds (Comput($P, s, 4 \cdot k + 1 + j$))($\mathbf{d}_2$) = Triangle n, and

(iii) (Result(P, s))($\mathbf{d}_2$) = Triangle n.

PROOF: Define $\mathcal{T}$(natural number) = Triangle $\$_1$. For every (IterativeAlgorithm(Add extending instruction sequence P of **SCM** and for every 0-started state s consisting of $\langle 1, n, 0 \rangle$, for every natural number k such that $k < n$ holds (Comput($P, s, 4 \cdot k + 1$))($\mathbf{d}_2$) = $0 + (\mathcal{T}(n) - \mathcal{T}((\text{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_1)))$ and (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) = $0 + (\mathcal{T}(n) - \mathcal{T}((\text{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_1) -' 1))$ and (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) and (Comput($P, s, 4 \cdot k + 4$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$) and for every natural numbers k, j such that $k \geqslant n$ holds (Comput($P, s, 4 \cdot k + 1 + j$))($\mathbf{d}_2$) = $\mathcal{T}(n)$ and (Result(P, s))($\mathbf{d}_2$) = $\mathcal{T}(n)$ from *AddOfOneSch*. □

(32) Let us consider a (IterativeAlgorithm(MultBy($\mathbf{d}_2, \mathbf{d}_1$)))-extending instruction sequence P of **SCM**, and a 0-started state s consisting of $\langle 1, n, 1 \rangle$. Then

(i) for every natural number k such that $k < n$ holds (Comput($P, s, 4 \cdot k + 1$))($\mathbf{d}_2$) = $\frac{n!}{(\text{Comput}(P, s, 4 \cdot k + 1))(\mathbf{d}_1)!}$ and (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) = $\frac{n!}{((\text{Comput}(P, s, 4 \cdot k + 2))(\mathbf{d}_1) -' 1)!}$ and (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 2$))($\mathbf{d}_2$) and (Comput($P, s, 4 \cdot k + 4$))($\mathbf{d}_2$) = (Comput($P, s, 4 \cdot k + 3$))($\mathbf{d}_2$), and

(ii) for every natural numbers k, j such that $k \geqslant n$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1 + j))(\mathbf{d}_2) = n!$, and

(iii) $(\mathrm{Result}(P, s))(\mathbf{d}_2) = n!$.

PROOF: Set $b = \mathbf{d}_1$. Set $c = \mathbf{d}_2$. Define $\mathcal{F}(\text{natural number}) = \$_1!$. For every (IterativeAlgorithm(MultBy(c, b)))-extending instruction sequence P of **SCM** and for every 0-started state s consisting of $\langle 1, n, 1\rangle$, for every natural number k such that $k < n$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1))(c) = 1 \cdot \frac{\mathcal{F}(n)}{\mathcal{F}((\mathrm{Comput}(P,s,4\cdot k+1))(b))}$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 2))(c) = 1 \cdot \frac{\mathcal{F}(n)}{\mathcal{F}((\mathrm{Comput}(P,s,4\cdot k+2))(b)-'1)}$ and $(\mathrm{Comput}(P, s, 4\cdot k+3))(c) = (\mathrm{Comput}(P, s, 4 \cdot k + 2))(c)$ and $(\mathrm{Comput}(P, s, 4 \cdot k + 4))(c) = (\mathrm{Comput}(P, s, 4 \cdot k + 3))(c)$ and for every natural numbers k, j such that $k \geqslant n$ holds $(\mathrm{Comput}(P, s, 4 \cdot k + 1 + j))(\mathbf{d}_2) = \mathcal{F}(n)$ and $(\mathrm{Result}(P, s))(c) = \mathcal{F}(n)$ from *MultOfOneSch*. □

5. EXAMPLES

The scheme *PartFuncFinPartStEx* deals with integers i_1, i_2 and a unary functor $\mathcal{F}$ yielding an integer and states that

(Sch. 3) There exists a partial function f from FinPartSt(**SCM**) to FinPartSt(**SCM**) such that for every finite partial states p, q of **SCM**, $\langle p, q\rangle \in f$ iff there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (i_1, n, i_2)$ and $q = \mathbf{d}_2 \mapsto \mathcal{F}(n)$.

The scheme *PartFuncFinPartStUn* deals with integers i_1, i_2 and a unary functor $\mathcal{F}$ yielding an integer and states that

(Sch. 4) For every partial functions f_1, f_2 from FinPartSt(**SCM**) to FinPartSt(**SCM**) such that for every finite partial states p, q of **SCM**, $\langle p, q\rangle \in f_1$ iff there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (i_1, n, i_2)$ and $q = \mathbf{d}_2 \mapsto \mathcal{F}(n)$ and for every finite partial states p, q of **SCM**, $\langle p, q\rangle \in f_2$ iff there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (i_1, n, i_2)$ and $q = \mathbf{d}_2 \mapsto \mathcal{F}(n)$ holds $f_1 = f_2$.

The functor powerSCMFunction yielding a partial function from FinPartSt(**SCM**) to FinPartSt(**SCM**) is defined by

(Def. 5) for every finite partial states p, q of **SCM**, $\langle p, q\rangle \in it$ iff there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 1)$ and $q = \mathbf{d}_2 \mapsto n!$.

Now we state the propositions:

(33) Let us consider an object p. Then $p \in \mathrm{dom}(\text{powerSCMFunction})$ if and only if there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 1)$.

(34) $(\text{powerSCMFunction})((\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 1)) = \mathbf{d}_2 \mathrel{\dot{\longmapsto}} n!$.

The scheme *AutonomicSch* deals with a partial function f from FinPartSt(**SCM**) to FinPartSt(**SCM**) and an instruction $\mathcal{I}$ of **SCM** and a unary functor $\mathcal{F}$ yielding a natural number and states that

(Sch. 5) For every finite partial state x of **SCM** such that $x \in \operatorname{dom} f$ holds Initialize(x) is (IterativeAlgorithm($\mathcal{I}$))-autonomic

provided

- $\mathcal{I} = \text{AddTo}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{SubFrom}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{MultBy}(\mathbf{d}_2, \mathbf{d}_1)$ and
- for every object x such that $x \in \operatorname{dom} f$ there exists a natural number n such that $x = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, \mathcal{F}(n))$ and
- for every instruction sequences P, Q of **SCM** such that IterativeAlgorithm($\mathcal{I}$) $\subseteq P$ and IterativeAlgorithm($\mathcal{I}$) $\subseteq Q$ for every finite partial state x of **SCM** such that $x \in \operatorname{dom} f$ for every states s_1, s_2 of **SCM** such that Initialize(x) $\subseteq s_1$ and Initialize(x) $\subseteq s_2$ for every natural number k, $(\text{Comput}(P, s_1, k))(\mathbf{d}_2) = (\text{Comput}(Q, s_2, k))(\mathbf{d}_2)$.

The scheme *HaltedSch* deals with a partial function f from FinPartSt(**SCM**) to FinPartSt(**SCM**) and an instruction $\mathcal{I}$ of **SCM** and a unary functor $\mathcal{F}$ yielding a natural number and states that

(Sch. 6) For every finite partial state x of **SCM** such that $x \in \operatorname{dom} f$ holds Initialize(x) is (IterativeAlgorithm($\mathcal{I}$))-halted

provided

- $\mathcal{I} = \text{AddTo}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{SubFrom}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{MultBy}(\mathbf{d}_2, \mathbf{d}_1)$ and
- for every object x such that $x \in \operatorname{dom} f$ there exists a natural number n such that $x = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, \mathcal{F}(n))$.

The scheme *AutonomySch* deals with a partial function f from FinPartSt(**SCM**) to FinPartSt(**SCM**) and an instruction $\mathcal{I}$ of **SCM** and a unary functor $\mathcal{F}$ yielding a natural number and states that

(Sch. 7) For every finite partial state x of **SCM** such that $x \in \operatorname{dom} f$ holds Initialize(x) is an autonomy of IterativeAlgorithm($\mathcal{I}$)

provided

- $\mathcal{I} = \text{AddTo}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{SubFrom}(\mathbf{d}_2, \mathbf{d}_1)$ or $\mathcal{I} = \text{MultBy}(\mathbf{d}_2, \mathbf{d}_1)$ and
- for every object x such that $x \in \operatorname{dom} f$ there exists a natural number n such that $x = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, \mathcal{F}(n))$ and

- for every instruction sequences P, Q of **SCM** such that IterativeAlgorithm($\mathcal{I}$) $\subseteq$ P and IterativeAlgorithm($\mathcal{I}$) $\subseteq$ Q for every finite partial state x of **SCM** such that $x \in \operatorname{dom} f$ for every states s_1, s_2 of **SCM** such that Initialize(x) $\subseteq$ s_1 and Initialize(x) $\subseteq$ s_2 for every natural number k, (Comput(P, s_1, k))($\mathbf{d}_2$) = (Comput(Q, s_2, k))($\mathbf{d}_2$).

Now we state the propositions:

(35) Let us consider a finite partial state x of **SCM**. Suppose $x \in$ dom(powerSCMFuncti Then Initialize(x) is an autonomy of IterativeAlgorithm(MultBy($\mathbf{d}_2, \mathbf{d}_1$)). PROOF: Set $a = \mathbf{d}_0$. Set $b = \mathbf{d}_1$. Set $c = \mathbf{d}_2$. Set pf = powerSCMFunction. Set I = MultBy(c, b). Define $\mathcal{F}$(natural number) = 1. For every object x such that $x \in \operatorname{dom} pf$ there exists a natural number n such that $x = (a, b, c) \mapsto (1, n, \mathcal{F}(n))$. For every instruction sequences P, Q of **SCM** such that IterativeAlgorithm(I) $\subseteq$ P and IterativeAlgorithm(I) $\subseteq$ Q for every finite partial state x of **SCM** such that $x \in \operatorname{dom} pf$ for every states s_1, s_2 of **SCM** such that Initialize(x) $\subseteq$ s_1 and Initialize(x) $\subseteq$ s_2 for every natural number k, (Comput(P, s_1, k))(c) = (Comput(Q, s_2, k))(c) by (33), (15), [**?** , (17)], (14). For every finite partial state x of **SCM** such that $x \in \operatorname{dom} pf$ holds Initialize(x) is an autonomy of IterativeAlgorithm(I) from *AutonomySch*. □

(36) IterativeAlgorithm(MultBy($\mathbf{d}_2, \mathbf{d}_1$)), StartAt(0, **SCM**) computes StartAt(0, **SCM** The theorem is a consequence of (33), (35), (34), (15), and (32).

The functor sumSCMFunction yielding a partial function from FinPartSt(**SCM**) to FinPartSt(**SCM**) is defined by

(Def. 6) for every finite partial states p, q of **SCM**, $\langle p, q \rangle \in it$ iff there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 0)$ and $q = \mathbf{d}_2 \mapsto\!\!\cdot\!\!\longrightarrow$ Triangle n.

Now we state the propositions:

(37) Let us consider an object p. Then $p \in$ dom(sumSCMFunction) if and only if there exists a natural number n such that $p = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 0)$.

(38) (sumSCMFunction)($(\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \mapsto (1, n, 0)$) = $\mathbf{d}_2 \mapsto\!\!\cdot\!\!\longrightarrow$ Triangle n.

(39) Let us consider a finite partial state x of **SCM**. Suppose $x \in$ dom(sumSCMFunction Then Initialize(x) is an autonomy of IterativeAlgorithm(AddTo($\mathbf{d}_2, \mathbf{d}_1$)). PROOF: Set $a = \mathbf{d}_0$. Set $b = \mathbf{d}_1$. Set $c = \mathbf{d}_2$. Set pf = sumSCMFunction. Define $\mathcal{T}$(natural number) = Triangle \$$_1$. Set I = AddTo(c, b). Define $\mathcal{F}$(natural number) = 0. For every object x such that $x \in \operatorname{dom} pf$ there exists a natural number n such that $x = (a, b, c) \mapsto (1, n, \mathcal{F}(n))$. For every instruction sequences P, Q of **SCM** such that IterativeAlgorithm(I) $\subseteq$ P and IterativeAlgorithm(I) $\subseteq$ Q for every finite partial state x of **SCM**

such that $x \in \operatorname{dom} pf$ for every states s_1, s_2 of **SCM** such that Initialize$(x) \subseteq s_1$ and Initialize$(x) \subseteq s_2$ for every natural number k, (Comput$(P, s_1, k))(c) =$ (Comput$(Q, s_2, k))(c)$ by (37), (15), [**?** , (17)], (14). For every finite partial state x of **SCM** such that $x \in \operatorname{dom} pf$ holds Initialize(x) is an autonomy of IterativeAlgorithm(I) from *AutonomySch*. □

(40) IterativeAlgorithm(AddTo$(\mathbf{d}_2, \mathbf{d}_1)$), StartAt$(0, \mathbf{SCM})$ computes StartAt$(0, \mathbf{SCM})$ The theorem is a consequence of (37), (39), (38), (15), and (31).

References

[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Formalized Mathematics*, 1(**1**):41–46, 1990.

[2] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, Karol Pąk, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer International Publishing, 2015. ISBN 978-3-319-20614-1. doi:10.1007/978-3-319-20615-8_17.

[3] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pąk. The role of the Mizar Mathematical Library for interactive proof development in Mizar. *Journal of Automated Reasoning*, 61(1):9–32, 2018. doi:10.1007/s10817-017-9440-6.

[4] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.

[5] Glynn Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, 1993.

Received July 20, 2026