FORMALIZED MATHEMATICS
Volume 3, Number 2, 1992
Université Catholique de Louvain

A Mathematical Model of CPU

Yatsuka Nakamura
Shinshu University
Nagano

Andrzej Trybulec[1]
Warsaw University
Białystok

Summary. This paper is based on a previous work of the first author [12] in which a mathematical model of the computer has been presented. The model deals with random access memory, such as RASP of C. C. Elgot and A. Robinson [11], however, it allows for a more realistic modeling of real computers. This new model of computers has been named by the author (Y. Nakamura, [12]) Architecture Model for Instructions (AMI). It is more developed than previous models, both in the description of hardware (e.g., the concept of the program counter, the structure of memory) as well as in the description of instructions (instruction codes, addresses). The structure of AMI over an arbitrary collection of mathematical domains N consists of:

- a non-empty set of objects,
- the instruction counter,
- a non-empty set of objects called instruction locations,
- a non-empty set of instruction codes,
- an instruction code for halting,
- a set of instructions that are ordered pairs with the first element being an instruction code and the second a finite sequence in which members are either objects of the AMI or elements of one of the domains included in N,
- a function that assigns to every object of AMI its kind that is either *an instruction* or *an instruction location* or an element of N,
- a function that assigns to every instruction its execution that is again a function mapping states of AMI into the set of states.

By a state of AMI we mean a function that assigns to every object of AMI an element of the same kind. In this paper we develop the theory of AMI. Some properties of AMI are introduced ensuring it to have some properties of real computers:

- a von Neumann AMI, in which only addresses to instruction locations are stored in the program counter,
- data oriented, those in which instructions cannot be stored in data locations,
- halting, in which the execution of the halt instruction is the identity mapping of the states of an AMI,
- steady programmed, the condition in which the contents of the instruction locations do not change during execution,

[1]The work has been done while the second author was visiting Nagano in autumn 1992.

ISSN 0777–4028

- definite, a property in which only instructions may be stored in instruction locations.

We present an example of AMI called a Small Concrete Model which has been constructed in [12]. The Small Concrete Model has only one kind of data: integers and a set of instructions, small but sufficient to cope with integers.

MML Identifier: AMI_1.

The terminology and notation used here have been introduced in the following articles: [19], [5], [6], [15], [2], [20], [14], [3], [17], [16], [10], [1], [4], [18], [13], [7], [9], [21], and [8].

1. Preliminaries

In the sequel x is arbitrary. Next we state several propositions:

(1) $\mathbb{N} \neq \mathbb{Z}$.

(2) For arbitrary a, b holds $1 \neq \langle a, b \rangle$.

(3) For arbitrary a, b holds $2 \neq \langle a, b \rangle$.

(4) For arbitrary a, b, c, d and for every function g such that $\operatorname{dom} g = \{a, b\}$ and $g(a) = c$ and $g(b) = d$ holds $g = [a \longmapsto c, b \longmapsto d]$.

(5) For arbitrary a, b, c, d such that $a \neq b$ holds $\prod[a \longmapsto \{c\}, b \longmapsto \{d\}] = \{[a \longmapsto c, b \longmapsto d]\}$.

Let A be a set, and let B be a non-empty set. Then $A \cup B$ is a non-empty set. Let A be a non-empty set, and let B be a set. Then $A \cup B$ is a non-empty set. A set has non-empty elements if:

(Def.1) $\emptyset \notin$ it.

One can verify that there exists a set which is non-empty with and non-empty elements.

Let A be a non-empty set. Then $\{A\}$ is a non-empty set with non-empty elements. Let B be a non-empty set. Then $\{A, B\}$ is a non-empty set with non-empty elements. Let A, B be non-empty sets with non-empty elements. Then $A \cup B$ is a non-empty set with non-empty elements.

2. General concepts

In the sequel N will be a non-empty set with non-empty elements.

We now define several new constructions. Let us consider N. We consider AMI's over N which are systems

⟨objects, a instruction counter, instruction locations, instruction codes, a halt instruction, instructions, a object kind, a execution⟩,
where the objects constitute a non-empty set, the instruction counter is an element of the objects, the instruction locations constitute a non-empty subset of the objects, the instruction codes constitute a non-empty set, the halt instruction is an element of the instruction codes, the instructions constitute a non-empty subset of $[\!:$ the instruction codes, $(\bigcup N \cup \text{the objects})^{*}\,:\!]$, the object kind is a function from the objects into $N \cup \{$the instructions,the instruction locations$\}$, and the execution is a function from the instructions into $(\prod(\text{the object kind}))^{\prod(\text{the object kind})}$. Let us consider N, and let S be an AMI over N. An object of S is an element of the objects of S.

An instruction of S is an element of the instructions of S.

An instruction-location of S is an element of the instruction locations of S.

Let us consider N, and let S be an AMI over N. The functor $\mathbf{IC}_S$ yields an object of S and is defined by:

(Def.2) $\mathbf{IC}_S =$ the instruction counter of S.

Let us consider N, and let S be an AMI over N, and let o be an object of S. The functor ObjectKind(o) yielding an element of $N \cup \{$the instructions of S, the instruction locations of $S\}$ is defined by:

(Def.3) ObjectKind(o) = (the object kind of S)(o).

Let A be a set, and let B be a non-empty set with non-empty elements, and let f be a function from A into B. Then $\prod f$ is a non-empty set of functions. Let P be a non-empty set of functions. We see that the element of P is a function. Let us consider N, and let S be an AMI over N. A state of S is an element of $\prod$(the object kind of S).

Let us consider N, and let S be an AMI over N, and let I be an instruction of S, and let s be a state of S. The functor $\mathrm{Exec}(I, s)$ yielding a state of S is defined by:

(Def.4) $\mathrm{Exec}(I, s) =$ (the execution of S **qua** a function from the instructions of S into $(\prod(\text{the object kind of } S))^{\prod(\text{the object kind of } S)})(I)(s)$.

Let us consider N, and let S be an AMI over N satisfying the condition: ⟨the halt instruction of S, ε⟩ $\in$ the instructions of S. The functor $\mathbf{halt}_S$ yields an instruction of S and is defined as follows:

(Def.5) $\mathbf{halt}_S =$ ⟨the halt instruction of S, ε⟩.

Let us consider N. An AMI over N is von Neumann if:

(Def.6) ObjectKind($\mathbf{IC}_{\mathrm{it}}$) = the instruction locations of it.

An AMI over N is data-oriented if:

(Def.7) (the object kind of it) $^{-1}$ {the instructions of it} $\subseteq$ the instruction locations of it.

An AMI over N is halting if:

(Def.8) for every state s of it holds $\mathrm{Exec}(\mathbf{halt}_{\mathrm{it}}, s) = s$.

An AMI over N is steady-programmed if:

(Def.9) for every state s of it and for every instruction i of it and for every instruction-location l of it holds $(\mathrm{Exec}(i,s))(l) = s(l)$.

An AMI over N is definite if:

(Def.10) for every instruction-location l of it holds $\mathrm{ObjectKind}(l) =$ the instructions of it.

Let us consider N. Note that there exists a von Neumann data-oriented halting steady-programmed definite strict AMI over N.

Let us consider N, and let S be a von Neumann AMI over N, and let s be a state of S. The functor $\mathbf{IC}_s$ yields an instruction-location of S and is defined as follows:

(Def.11) $\mathbf{IC}_s = s(\mathbf{IC}_S)$.

3. A small concrete model

In the sequel i, k will be natural numbers. We now define four new functors. The non-empty subset $\mathrm{Loc}_{\mathrm{SCM}}$ of $\mathbb{N}$ is defined by:

(Def.12) $\mathrm{Loc}_{\mathrm{SCM}} = \mathbb{N} \setminus \{0\}$.

The element $\mathrm{Halt}_{\mathrm{SCM}}$ of $\mathbb{Z}_9$ is defined as follows:

(Def.13) $\mathrm{Halt}_{\mathrm{SCM}} = 0$.

The non-empty subset $\mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$ of $\mathrm{Loc}_{\mathrm{SCM}}$ is defined as follows:

(Def.14) $\mathrm{Data\text{-}Loc}_{\mathrm{SCM}} = \{2 \cdot k + 1\}$.

The non-empty subset $\mathrm{Instr\text{-}Loc}_{\mathrm{SCM}}$ of $\mathbb{N}$ is defined by:

(Def.15) $\mathrm{Instr\text{-}Loc}_{\mathrm{SCM}} = \{2 \cdot k : k > 0\}$.

We adopt the following convention: I, J, K are elements of $\mathbb{Z}_9$, a, a_1, a_2 are elements of $\mathrm{Instr\text{-}Loc}_{\mathrm{SCM}}$, and b, b_1, b_2, c, c_1 are elements of $\mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$. The non-empty subset $\mathrm{Instr}_{\mathrm{SCM}}$ of $[\!:\mathbb{Z}_9, \bigcup\{\mathbb{Z}\} \cup \mathbb{N}^*:\!]$ is defined as follows:

(Def.16) $\mathrm{Instr}_{\mathrm{SCM}} = \{\langle\mathrm{Halt}_{\mathrm{SCM}}, \varepsilon\rangle\} \cup \{\langle J, \langle a\rangle\rangle : J = 6\} \cup \{\langle K, \langle a_1, b_1\rangle\rangle : K \in \{7,8\}\} \cup \{\langle I, \langle b, c\rangle\rangle : I \in \{1,2,3,4,5\}\}$.

The following propositions are true:

(6) $\mathrm{Instr}_{\mathrm{SCM}} = \{\langle\mathrm{Halt}_{\mathrm{SCM}}, \varepsilon\rangle\} \cup \{\langle J, \langle a\rangle\rangle : J = 6\} \cup \{\langle K, \langle a_1, b_1\rangle\rangle : K \in \{7,8\}\} \cup \{\langle I, \langle b, c\rangle\rangle : I \in \{1,2,3,4,5\}\}$.

(7) $\langle 0, \varepsilon\rangle \in \mathrm{Instr}_{\mathrm{SCM}}$.

(8) $\langle 6, \langle a_2\rangle\rangle \in \mathrm{Instr}_{\mathrm{SCM}}$.

(9) If $x \in \{7,8\}$, then $\langle x, \langle a_2, b_2\rangle\rangle \in \mathrm{Instr}_{\mathrm{SCM}}$.

(10) If $x \in \{1,2,3,4,5\}$, then $\langle x, \langle b_1, c_1\rangle\rangle \in \mathrm{Instr}_{\mathrm{SCM}}$.

The function $\mathrm{OK}_{\mathrm{SCM}}$ from $\mathbb{N}$ into $\{\mathbb{Z}\} \cup \{\mathrm{Instr}_{\mathrm{SCM}}, \mathrm{Instr\text{-}Loc}_{\mathrm{SCM}}\}$ is defined by:

(Def.17) $\mathrm{OK}_{\mathrm{SCM}}(0) = \mathrm{Instr\text{-}Loc}_{\mathrm{SCM}}$ and for every natural number k holds $\mathrm{OK}_{\mathrm{SCM}}(2 \cdot k + 1) = \mathbb{Z}$ and $\mathrm{OK}_{\mathrm{SCM}}(2 \cdot k + 2) = \mathrm{Instr}_{\mathrm{SCM}}$.

The following four propositions are true:

(11) $\text{Instr-Loc}_{\text{SCM}} \neq \mathbb{Z}$ and $\text{Instr}_{\text{SCM}} \neq \mathbb{Z}$ and $\text{Instr-Loc}_{\text{SCM}} \neq \text{Instr}_{\text{SCM}}$.

(12) For every i holds $\text{OK}_{\text{SCM}}(i) = \text{Instr-Loc}_{\text{SCM}}$ if and only if $i = 0$.

(13) For every i holds $\text{OK}_{\text{SCM}}(i) = \mathbb{Z}$ if and only if there exists k such that $i = 2 \cdot k + 1$.

(14) For every i holds $\text{OK}_{\text{SCM}}(i) = \text{Instr}_{\text{SCM}}$ if and only if there exists k such that $i = 2 \cdot k + 2$.

A $\text{state}_{\text{SCM}}$ is an element of $\prod(\text{OK}_{\text{SCM}})$.

In the sequel s is a $\text{state}_{\text{SCM}}$. We now state several propositions:

(15) For every element a of $\text{Data-Loc}_{\text{SCM}}$ holds $\text{OK}_{\text{SCM}}(a) = \mathbb{Z}$.

(16) For every element a of $\text{Instr-Loc}_{\text{SCM}}$ holds $\text{OK}_{\text{SCM}}(a) = \text{Instr}_{\text{SCM}}$.

(17) For every element a of $\text{Instr-Loc}_{\text{SCM}}$
and for every element t of $\text{Data-Loc}_{\text{SCM}}$ holds $a \neq t$.

(18) $\pi_0 \prod(\text{OK}_{\text{SCM}}) = \text{Instr-Loc}_{\text{SCM}}$.

(19) For every element a of $\text{Data-Loc}_{\text{SCM}}$ holds $\pi_a \prod(\text{OK}_{\text{SCM}}) = \mathbb{Z}$.

(20) For every element a of $\text{Instr-Loc}_{\text{SCM}}$ holds $\pi_a \prod(\text{OK}_{\text{SCM}}) = \text{Instr}_{\text{SCM}}$.

We now define two new functors. Let s be a $\text{state}_{\text{SCM}}$. The functor $\mathbf{IC}_s$ yielding an element of $\text{Instr-Loc}_{\text{SCM}}$ is defined by:

(Def.18) $\mathbf{IC}_s = s(0)$.

Let s be a $\text{state}_{\text{SCM}}$, and let u be an element of $\text{Instr-Loc}_{\text{SCM}}$. The functor $\text{Chg}_{\text{SCM}}(s, u)$ yields a $\text{state}_{\text{SCM}}$ and is defined as follows:

(Def.19) $\text{Chg}_{\text{SCM}}(s, u) = s +\cdot (0 \dot{\longmapsto} u)$.

The following three propositions are true:

(21) For every $\text{state}_{\text{SCM}}$ s and for every element u of $\text{Instr-Loc}_{\text{SCM}}$ holds $(\text{Chg}_{\text{SCM}}(s, u))(0) = u$.

(22) For every $\text{state}_{\text{SCM}}$ s and for every element u of $\text{Instr-Loc}_{\text{SCM}}$ and for every element m_1 of $\text{Data-Loc}_{\text{SCM}}$ holds $(\text{Chg}_{\text{SCM}}(s, u))(m_1) = s(m_1)$.

(23) For every $\text{state}_{\text{SCM}}$ s and for all elements u, v of $\text{Instr-Loc}_{\text{SCM}}$ holds $(\text{Chg}_{\text{SCM}}(s, u))(v) = s(v)$.

Let s be a $\text{state}_{\text{SCM}}$, and let t be an element of $\text{Data-Loc}_{\text{SCM}}$, and let u be an integer. The functor $\text{Chg}_{\text{SCM}}(s, t, u)$ yielding a $\text{state}_{\text{SCM}}$ is defined by:

(Def.20) $\text{Chg}_{\text{SCM}}(s, t, u) = s +\cdot (t \dot{\longmapsto} u)$.

The following four propositions are true:

(24) For every $\text{state}_{\text{SCM}}$ s and for every element t of $\text{Data-Loc}_{\text{SCM}}$ and for every integer u holds $(\text{Chg}_{\text{SCM}}(s, t, u))(0) = s(0)$.

(25) For every $\text{state}_{\text{SCM}}$ s and for every element t of $\text{Data-Loc}_{\text{SCM}}$ and for every integer u holds $(\text{Chg}_{\text{SCM}}(s, t, u))(t) = u$.

(26) For every $\text{state}_{\text{SCM}}$ s and for every element t of $\text{Data-Loc}_{\text{SCM}}$ and for every integer u and for every element m_1 of $\text{Data-Loc}_{\text{SCM}}$ such that $m_1 \neq t$ holds $(\text{Chg}_{\text{SCM}}(s, t, u))(m_1) = s(m_1)$.

(27) For every $\text{state}_{\text{SCM}}$ s and for every element t of $\text{Data-Loc}_{\text{SCM}}$ and for every integer u and for every element v of $\text{Instr-Loc}_{\text{SCM}}$ holds $(\text{Chg}_{\text{SCM}}(s,t,u))(v) = s(v)$.

We now define two new functors. Let x be an element of $\text{Instr}_{\text{SCM}}$. Let us assume that there exist m_1, m_2 of the type elements of $\text{Data-Loc}_{\text{SCM}}$; I such that $x = \langle\!\langle I, \langle m_1, m_2\rangle\rangle\!\rangle$. The functor $x\text{address}_1$ yields an element of $\text{Data-Loc}_{\text{SCM}}$ and is defined by:

(Def.21) there exists a finite sequence f of elements of $\text{Data-Loc}_{\text{SCM}}$ such that $f = x_{\mathbf{2}}$ and $x\text{address}_1 = \pi_1 f$.

The functor $x\text{address}_2$ yields an element of $\text{Data-Loc}_{\text{SCM}}$ and is defined by:

(Def.22) there exists a finite sequence f of elements of $\text{Data-Loc}_{\text{SCM}}$ such that $f = x_{\mathbf{2}}$ and $x\text{address}_2 = \pi_2 f$.

One can prove the following proposition

(28) For every element x of $\text{Instr}_{\text{SCM}}$ and for all elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ and for every I such that $x = \langle\!\langle I, \langle m_1, m_2\rangle\rangle\!\rangle$ holds $x\text{address}_1 = m_1$ and $x\text{address}_2 = m_2$.

Let x be an element of $\text{Instr}_{\text{SCM}}$. Let us assume that there exist m_1 of the type an element of $\text{Instr-Loc}_{\text{SCM}}$; I such that $x = \langle\!\langle I, \langle m_1\rangle\rangle\!\rangle$. The functor $x\text{address}_{\text{j}}$ yielding an element of $\text{Instr-Loc}_{\text{SCM}}$ is defined as follows:

(Def.23) there exists a finite sequence f of elements of $\text{Instr-Loc}_{\text{SCM}}$ such that $f = x_{\mathbf{2}}$ and $x\text{address}_{\text{j}} = \pi_1 f$.

We now state the proposition

(29) For every element x of $\text{Instr}_{\text{SCM}}$ and for every element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and for every I such that $x = \langle\!\langle I, \langle m_1\rangle\rangle\!\rangle$ holds $x\text{address}_{\text{j}} = m_1$.

We now define two new functors. Let x be an element of $\text{Instr}_{\text{SCM}}$. Let us assume that there exist m_1 of the type an element of $\text{Instr-Loc}_{\text{SCM}}$; m_2 of the type an element of $\text{Data-Loc}_{\text{SCM}}$; I such that $x = \langle\!\langle I, \langle m_1, m_2\rangle\rangle\!\rangle$. The functor $x\text{address}_{\text{j}}$ yields an element of $\text{Instr-Loc}_{\text{SCM}}$ and is defined as follows:

(Def.24) there exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and there exists an element m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $\langle m_1, m_2\rangle = x_{\mathbf{2}}$ and $x\text{address}_{\text{j}} = \pi_1\langle m_1, m_2\rangle$.

The functor $x\text{address}_{\text{c}}$ yielding an element of $\text{Data-Loc}_{\text{SCM}}$ is defined by:

(Def.25) there exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and there exists an element m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $\langle m_1, m_2\rangle = x_{\mathbf{2}}$ and $x\text{address}_{\text{c}} = \pi_2\langle m_1, m_2\rangle$.

The following proposition is true

(30) For every element x of $\text{Instr}_{\text{SCM}}$ and for every element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and for every element m_2 of $\text{Data-Loc}_{\text{SCM}}$ and for every I such that $x = \langle\!\langle I, \langle m_1, m_2\rangle\rangle\!\rangle$ holds $x\text{address}_{\text{j}} = m_1$ and $x\text{address}_{\text{c}} = m_2$.

We now define five new functors. Let s be a $\text{state}_{\text{SCM}}$, and let a be an element of $\text{Data-Loc}_{\text{SCM}}$. Then $s(a)$ is an integer. Let D be a non-empty set, and let x, y be arbitrary, and let a, b be elements of D. Then $(x = y \rightarrow a, b)$ is an element of D. Let D be a non-empty set, and let x, y be real numbers, and let a, b be elements of D. The functor $(x > y \to a, b)$ yields an element of D and is defined as follows:

(Def.26)

$$(x > y \to a, b) = \begin{cases} a, & \text{if } x > y, \\ b, & \text{otherwise.} \end{cases}$$

Let d be an element of $\text{Instr-Loc}_{\text{SCM}}$. The functor $\text{Next}(d)$ yields an element of $\text{Instr-Loc}_{\text{SCM}}$ and is defined as follows:

(Def.27) $\text{Next}(d) = d + 2$.

Let x be an element of $\text{Instr}_{\text{SCM}}$, and let s be a $\text{state}_{\text{SCM}}$. The functor $\text{Exec-Res}_{\text{SCM}}(x, s)$ yielding a $\text{state}_{\text{SCM}}$ is defined as follows:

(Def.28) (i) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(s, x\text{address}_1, s(x\text{address}_2)), \text{Next}(\mathbf{IC}_s))$ if there exist elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 1, \langle m_1, m_2 \rangle \rangle$,

(ii) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(s, x\text{address}_1, s(x\text{address}_1) + s(x\text{address}_2)), \text{Next}(\mathbf{IC}_s))$ if there exist elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 2, \langle m_1, m_2 \rangle \rangle$,

(iii) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(s, x\text{address}_1, s(x\text{address}_1) - s(x\text{address}_2)), \text{Next}(\mathbf{IC}_s))$ if there exist elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 3, \langle m_1, m_2 \rangle \rangle$,

(iv) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(s, x\text{address}_1, s(x\text{address}_1) \cdot s(x\text{address}_2)), \text{Next}(\mathbf{IC}_s))$ if there exist elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 4, \langle m_1, m_2 \rangle \rangle$,

(v) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(\text{Chg}_{\text{SCM}}(s, x\text{address}_1, s(x\text{address}_1) \div s(x\text{address}_2)), x\text{address}_2, s(x\text{address}_1) \bmod s(x\text{address}_2)), \text{Next}(\mathbf{IC}_s))$ if there exist elements m_1, m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 5, \langle m_1, m_2 \rangle \rangle$,

(vi) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(s, x\text{address}_{\text{j}})$ if there exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ such that $x = \langle 6, \langle m_1 \rangle \rangle$,

(vii) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(s, (s(x\text{address}_{\text{c}}) = 0 \to x\text{address}_{\text{j}}, \text{Next}(\mathbf{IC}_s)))$ if there exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and there exists an element m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 7, \langle m_1, m_2 \rangle \rangle$,

(viii) $\text{Exec-Res}_{\text{SCM}}(x, s) = \text{Chg}_{\text{SCM}}(s, (s(x\text{address}_{\text{c}}) > 0 \to x\text{address}_{\text{j}}, \text{Next}(\mathbf{IC}_s)))$ if there exists an element m_1 of $\text{Instr-Loc}_{\text{SCM}}$ and there exists an element m_2 of $\text{Data-Loc}_{\text{SCM}}$ such that $x = \langle 8, \langle m_1, m_2 \rangle \rangle$,

(ix) $\text{Exec-Res}_{\text{SCM}}(x, s) = s$, otherwise.

The function Exec_{SCM} from $\text{Instr}_{\text{SCM}}$ into $\prod \text{OK}_{\text{SCM}}^{\prod \text{OK}_{\text{SCM}}}$ is defined by:

(Def.29) for every element x of $\text{Instr}_{\text{SCM}}$ and for every $\text{state}_{\text{SCM}}$ y holds $(\text{Exec}_{\text{SCM}}(x)\ \mathbf{qua}$ an element of $(\prod(\text{OK}_{\text{SCM}}))^{\prod(\text{OK}_{\text{SCM}})})(y) = \text{Exec-Res}_{\text{SCM}}(x, y)$.

The von Neumann strict AMI **SCM** is defined by:

(Def.30) $\mathbf{SCM} = \langle \mathbb{N}, 0, \text{Instr-Loc}_{\text{SCM}}, \mathbb{Z}_9, \text{Halt}_{\text{SCM}}, \text{Instr}_{\text{SCM}}, \text{OK}_{\text{SCM}}, \text{Exec}_{\text{SCM}} \rangle$.

Next we state three propositions:

(31) **SCM** is data-oriented.

(32) **SCM** is definite.

(33) The objects of $\mathbf{SCM} = \mathbb{N}$ and the instruction counter of $\mathbf{SCM} = 0$ and the instruction locations of $\mathbf{SCM} = \text{Instr-Loc}_{\text{SCM}}$ and the instruction codes of $\mathbf{SCM} = \mathbb{Z}_9$ and the halt instruction of $\mathbf{SCM} = \text{Halt}_{\text{SCM}}$ and the instructions of $\mathbf{SCM} = \text{Instr}_{\text{SCM}}$ and the object kind of $\mathbf{SCM} = \text{OK}_{\text{SCM}}$ and the execution of $\mathbf{SCM} = \text{Exec}_{\text{SCM}}$.

An object of **SCM** is said to be a data-location if:

(Def.31) it $\in \text{Data-Loc}_{\text{SCM}}$.

Let s be a state of **SCM**, and let d be a data-location. Then $s(d)$ is an integer.

We adopt the following convention: a, b, c denote data-locations, l_1 denotes an instruction-location of **SCM**, and I denotes an instruction of **SCM**. We now define several new functors. Let us consider a, b. The functor $a{:=}b$ yielding an instruction of **SCM** is defined by:

(Def.32) $a{:=}b = \langle 1, \langle a, b \rangle \rangle$.

The functor $\text{AddTo}(a, b)$ yielding an instruction of **SCM** is defined by:

(Def.33) $\text{AddTo}(a, b) = \langle 2, \langle a, b \rangle \rangle$.

The functor $\text{SubFrom}(a, b)$ yielding an instruction of **SCM** is defined by:

(Def.34) $\text{SubFrom}(a, b) = \langle 3, \langle a, b \rangle \rangle$.

The functor $\text{MultBy}(a, b)$ yields an instruction of **SCM** and is defined by:

(Def.35) $\text{MultBy}(a, b) = \langle 4, \langle a, b \rangle \rangle$.

The functor $\text{Divide}(a, b)$ yields an instruction of **SCM** and is defined as follows:

(Def.36) $\text{Divide}(a, b) = \langle 5, \langle a, b \rangle \rangle$.

Let us consider l_1. The functor goto l_1 yields an instruction of **SCM** and is defined by:

(Def.37) goto $l_1 = \langle 6, \langle l_1 \rangle \rangle$.

Let us consider a. The functor **if** $a = 0$ **goto** l_1 yielding an instruction of **SCM** is defined as follows:

(Def.38) **if** $a = 0$ **goto** $l_1 = \langle 7, \langle l_1, a \rangle \rangle$.

The functor **if** $a > 0$ **goto** l_1 yields an instruction of **SCM** and is defined as follows:

(Def.39) **if** $a > 0$ **goto** $l_1 = \langle 8, \langle l_1, a \rangle \rangle$.

In the sequel s will denote a state of **SCM**. Next we state two propositions:

(34) $\mathbf{IC}_{\mathbf{SCM}} = 0$.

(35) For every $\text{state}_{\text{SCM}}$ S such that $S = s$ holds $\mathbf{IC}_s = \mathbf{IC}_S$.

Let l_1 be an instruction-location of **SCM**. The functor $\text{Next}(l_1)$ yielding an instruction-location of **SCM** is defined by:

(Def.40) there exists an element m_3 of Instr-Loc$_{\mathrm{SCM}}$ such that $m_3 = l_1$ and $\mathrm{Next}(l_1) = \mathrm{Next}(m_3)$.

Next we state two propositions:

(36) For every instruction-location l_1 of **SCM** and for every element m_3 of Instr-Loc$_{\mathrm{SCM}}$ such that $m_3 = l_1$ holds $\mathrm{Next}(m_3) = \mathrm{Next}(l_1)$.

(37) For every element i of Instr$_{\mathrm{SCM}}$ such that $i = I$ and for every state$_{\mathrm{SCM}}$ S such that $S = s$ holds $\mathrm{Exec}(I, s) = \text{Exec-Res}_{\mathrm{SCM}}(i, S)$.

4. Users guide

One can prove the following propositions:

(38) $(\mathrm{Exec}(a{:=}b, s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(a{:=}b, s))(a) = s(b)$ and for every c such that $c \neq a$ holds $(\mathrm{Exec}(a{:=}b, s))(c) = s(c)$.

(39) $(\mathrm{Exec}(\mathrm{AddTo}(a, b), s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{AddTo}(a, b), s))(a) = s(a) + s(b)$ and for every c such that $c \neq a$ holds $(\mathrm{Exec}(\mathrm{AddTo}(a, b), s))(c) = s(c)$.

(40) $(\mathrm{Exec}(\mathrm{SubFrom}(a, b), s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{SubFrom}(a, b), s))(a) = s(a) - s(b)$ and for every c such that $c \neq a$ holds $(\mathrm{Exec}(\mathrm{SubFrom}(a, b), s))(c) = s(c)$.

(41) $(\mathrm{Exec}(\mathrm{MultBy}(a, b), s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{MultBy}(a, b), s))(a) = s(a) \cdot s(b)$ and for every c such that $c \neq a$ holds $(\mathrm{Exec}(\mathrm{MultBy}(a, b), s))(c) = s(c)$.

(42) Suppose $a \neq b$. Then

(i) $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$,

(ii) $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(a) = s(a) \div s(b)$,

(iii) $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(b) = s(a) \bmod s(b)$,

(iv) for every c such that $c \neq a$ and $c \neq b$ holds $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(c) = s(c)$.

(43) $(\mathrm{Exec}(\mathrm{goto}\ l_1, s))(\mathbf{IC}_{\mathbf{SCM}}) = l_1$ and $(\mathrm{Exec}(\mathrm{goto}\ l_1, s))(c) = s(c)$.

(44) If $s(a) = 0$, then $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1, s))(\mathbf{IC}_{\mathbf{SCM}}) = l_1$ and also if $s(a) \neq 0$, then $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1, s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l_1, s))(c) = s(c)$.

(45) If $s(a) > 0$, then $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1, s))(\mathbf{IC}_{\mathbf{SCM}}) = l_1$ and also if $s(a) \leq 0$, then $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1, s))(\mathbf{IC}_{\mathbf{SCM}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l_1, s))(c) = s(c)$.

(46) $\mathrm{Exec}(\mathbf{halt}_{\mathbf{SCM}}, s) = s$.

(47) For every state s of **SCM** and for every instruction-location i of **SCM** holds $s(i)$ is an instruction of **SCM**.

References

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

[2] Grzegorz Bancerek. König's theorem. *Formalized Mathematics*, 1(**3**):589–593, 1990.

[3] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.

[4] Czesław Byliński. A classical first order language. *Formalized Mathematics*, 1(**4**):669–676, 1990.

[5] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.

[6] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.

[7] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Formalized Mathematics*, 1(**3**):521–527, 1990.

[8] Czesław Byliński. Products and coproducts in categories. *Formalized Mathematics*, 2(**5**):701–709, 1991.

[9] Czesław Byliński. Subcategories and products of categories. *Formalized Mathematics*, 1(**4**):725–732, 1990.

[10] Agata Darmochwał. Finite sets. *Formalized Mathematics*, 1(**1**):165–167, 1990.

[11] C.C. Elgot and A. Robinson. Random access stored-program machines, an approach to programming languages. *J.A.C.M.*, 11(4):365–399, Oct 1964.

[12] Yatsuka Nakamura. *On a Mathematical Model of CPU and Algorithm*. Technical Report, Shinshu University, Aug 1991.

[13] Henryk Oryszczyszyn and Krzysztof Prażmowski. Real functions spaces. *Formalized Mathematics*, 1(**3**):555–561, 1990.

[14] Dariusz Surowik. Cyclic groups and some of their properties - part I. *Formalized Mathematics*, 2(**5**):623–627, 1991.

[15] Andrzej Trybulec. Binary operations applied to functions. *Formalized Mathematics*, 1(**2**):329–334, 1990.

[16] Andrzej Trybulec. Domains and their Cartesian products. *Formalized Mathematics*, 1(**1**):115–122, 1990.

[17] Andrzej Trybulec. Enumerated sets. *Formalized Mathematics*, 1(**1**):25–34, 1990.

[18] Andrzej Trybulec. Function domains and Frænkel operator. *Formalized Mathematics*, 1(**3**):495–500, 1990.

[19] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.

[20] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.

[21] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.

Received October 14, 1992