FORMALIZED MATHEMATICS
Volume 5, Number **4**, 1996
Warsaw University - Białystok

The $\mathbf{SCM_{FSA}}$ Computer

Andrzej Trybulec
Warsaw University
Białystok

Yatsuka Nakamura
Shinshu University
Nagano

Piotr Rudnicki
University of Alberta
Edmonton

MML Identifier: `SCMFSA_2`.

The articles [20], [26], [11], [1], [24], [27], [21], [2], [14], [3], [15], [7], [17], [8], [19], [18], [10], [5], [9], [6], [25], [4], [12], [13], [22], [16], and [23] provide the notation and terminology for this paper.

1. Preliminaries

One can prove the following propositions:

(1) Let N be a non empty set with non empty elements and let S be a von Neumann definite realistic AMI over N. Then $\mathbf{IC}_S \notin$ the instruction locations of S.

(2) Let N be a non empty set with non empty elements, and let S be a definite AMI over N, and let s be a state of S, and let i be an instruction-location of S. Then $s(i)$ is an instruction of S.

(3) Let N be a non empty set with non empty elements, and let S be an AMI over N, and let s be a state of S. Then the instruction locations of $S \subseteq \operatorname{dom} s$.

(4) Let N be a non empty set with non empty elements, and let S be a von Neumann AMI over N, and let s be a state of S. Then $\mathbf{IC}_s \in \operatorname{dom} s$.

(5) Let N be a non empty set with non empty elements, and let S be an AMI over N, and let s be a state of S, and let l be an instruction-location of S. Then $l \in \operatorname{dom} s$.

ISSN 1426–2630

2. The $\mathbf{SCM}_{\rm FSA}$ Computer

The strict AMI $\mathbf{SCM}_{\rm FSA}$ over $\{\mathbb{Z}, \mathbb{Z}^*\}$ is defined by:

(Def. 1) $\mathbf{SCM}_{\rm FSA} = \langle \mathbb{Z}, 0(\in \mathbb{Z}), \text{Instr-Loc}_{\rm SCM_{FSA}}, \mathbb{Z}_{13}, 0(\in \mathbb{Z}_{13}), \text{Instr}_{\rm SCM_{FSA}}, \text{OK}_{\rm SCM_{FSA}}, \text{Exec}_{\rm SCM_{FSA}} \rangle$.

We now state two propositions:

(6) (i) The instruction locations of $\mathbf{SCM}_{\rm FSA} \neq \mathbb{Z}$,

(ii) the instructions of $\mathbf{SCM}_{\rm FSA} \neq \mathbb{Z}$,

(iii) the instruction locations of $\mathbf{SCM}_{\rm FSA} \neq$ the instructions of $\mathbf{SCM}_{\rm FSA}$,

(iv) the instruction locations of $\mathbf{SCM}_{\rm FSA} \neq \mathbb{Z}^*$, and

(v) the instructions of $\mathbf{SCM}_{\rm FSA} \neq \mathbb{Z}^*$.

(7) $\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}} = 0$.

3. The Memory Structure

In the sequel k, k_1, k_2 denote natural numbers.

The subset Int-Locations of the objects of $\mathbf{SCM}_{\rm FSA}$ is defined by:

(Def. 2) Int-Locations $=$ Data-Loc$_{\rm SCM_{FSA}}$.

The subset FinSeq-Locations of the objects of $\mathbf{SCM}_{\rm FSA}$ is defined by:

(Def. 3) FinSeq-Locations $=$ Data*-Loc$_{\rm SCM_{FSA}}$.

The following proposition is true

(8) The objects of $\mathbf{SCM}_{\rm FSA} =$ Int-Locations $\cup$ FinSeq-Locations $\cup$ $\{\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}\} \cup$ the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

An object of $\mathbf{SCM}_{\rm FSA}$ is called an integer location if:

(Def. 4) It $\in$ Data-Loc$_{\rm SCM_{FSA}}$.

An object of $\mathbf{SCM}_{\rm FSA}$ is said to be a finite sequence location if:

(Def. 5) It $\in$ Data*-Loc$_{\rm SCM_{FSA}}$.

In the sequel d_1 denotes an integer location, f_1 denotes a finite sequence location, and x is arbitrary.

We now state several propositions:

(9) $d_1 \in$ Int-Locations.

(10) $f_1 \in$ FinSeq-Locations.

(11) If $x \in$ Int-Locations, then x is an integer location.

(12) If $x \in$ FinSeq-Locations, then x is a finite sequence location.

(13) Int-Locations misses the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(14) FinSeq-Locations misses the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(15) Int-Locations misses FinSeq-Locations.

Let us consider k. The functor intloc(k) yields an integer location and is defined as follows:

(Def. 6) $\mathrm{intloc}(k) = \mathbf{d}_k$.

The functor $\mathrm{insloc}(k)$ yields an instruction-location of $\mathbf{SCM}_{\rm FSA}$ and is defined by:

(Def. 7) $\mathrm{insloc}(k) = \mathbf{i}_k$.

The functor $\mathrm{fsloc}(k)$ yields a finite sequence location and is defined as follows:

(Def. 8) $\mathrm{fsloc}(k) = -(k + 1)$.

One can prove the following propositions:

(16) For all k_1, k_2 such that $k_1 \neq k_2$ holds $\mathrm{intloc}(k_1) \neq \mathrm{intloc}(k_2)$.

(17) For all k_1, k_2 such that $k_1 \neq k_2$ holds $\mathrm{fsloc}(k_1) \neq \mathrm{fsloc}(k_2)$.

(18) For all k_1, k_2 such that $k_1 \neq k_2$ holds $\mathrm{insloc}(k_1) \neq \mathrm{insloc}(k_2)$.

(19) For every integer location d_2 there exists a natural number i such that $d_2 = \mathrm{intloc}(i)$.

(20) For every finite sequence location f_2 there exists a natural number i such that $f_2 = \mathrm{fsloc}(i)$.

(21) For every instruction-location i_1 of $\mathbf{SCM}_{\rm FSA}$ there exists a natural number i such that $i_1 = \mathrm{insloc}(i)$.

(22) Int-Locations is infinite.

(23) FinSeq-Locations is infinite.

(24) The instruction locations of $\mathbf{SCM}_{\rm FSA}$ is infinite.

(25) Every integer location is a data-location.

(26) For every integer location l holds $\mathrm{ObjectKind}(l) = \mathbb{Z}$.

(27) For every finite sequence location l holds $\mathrm{ObjectKind}(l) = \mathbb{Z}^*$.

(28) For arbitrary x such that $x \in \text{Data-Loc}_{\rm SCM_{FSA}}$ holds x is an integer location.

(29) For arbitrary x such that $x \in \text{Data}^*\text{-Loc}_{\rm SCM_{FSA}}$ holds x is a finite sequence location.

(30) For arbitrary x such that $x \in \text{Instr-Loc}_{\rm SCM_{FSA}}$ holds x is an instruction-location of $\mathbf{SCM}_{\rm FSA}$.

Let l_1 be an instruction-location of $\mathbf{SCM}_{\rm FSA}$. The functor $\mathrm{Next}(l_1)$ yields an instruction-location of $\mathbf{SCM}_{\rm FSA}$ and is defined by:

(Def. 9) There exists an element m_1 of $\text{Instr-Loc}_{\rm SCM_{FSA}}$ such that $m_1 = l_1$ and $\mathrm{Next}(l_1) = \mathrm{Next}(m_1)$.

Next we state two propositions:

(31) For every instruction-location l_1 of $\mathbf{SCM}_{\rm FSA}$ and for every element m_1 of $\text{Instr-Loc}_{\rm SCM_{FSA}}$ such that $m_1 = l_1$ holds $\mathrm{Next}(m_1) = \mathrm{Next}(l_1)$.

(32) $\mathrm{Next}(\mathrm{insloc}(k)) = \mathrm{insloc}(k + 1)$.

For simplicity we adopt the following convention: l_2, l_3 are instructions-locations of $\mathbf{SCM}_{\rm FSA}$, L_1 is an instruction-location of $\mathbf{SCM}$, i is an instruction of $\mathbf{SCM}_{\rm FSA}$, I is an instruction of $\mathbf{SCM}$, l is an instruction-location of $\mathbf{SCM}_{\rm FSA}$, f, f_1, g are finite sequence locations, A, B are data-locations, and a, b, c, d_1, d_3 are integer locations.

We now state the proposition

(33) If $l_2 = L_1$, then $\mathrm{Next}(l_2) = \mathrm{Next}(L_1)$.

4. The Instruction Structure

Let I be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$. The functor $\mathrm{InsCode}(I)$ yielding a natural number is defined as follows:

(Def. 10) $\mathrm{InsCode}(I) = I_{\mathbf{1}}$.

The following propositions are true:

(34) For every instruction I of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $\mathrm{InsCode}(I) \leq 8$ holds I is an instruction of $\mathbf{SCM}$.

(35) For every instruction I of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{InsCode}(I) \leq 12$.

(36) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $\mathrm{InsCode}(i) = 0$ holds $i = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(37) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every instruction I of $\mathbf{SCM}$ such that $i = I$ holds $\mathrm{InsCode}(i) = \mathrm{InsCode}(I)$.

(38) Every instruction of $\mathbf{SCM}$ is an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$.

Let us consider a, b. The functor $a{:=}b$ yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 11) There exist A, B such that $a = A$ and $b = B$ and $a{:=}b = A{:=}B$.

The functor $\mathrm{AddTo}(a, b)$ yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined by:

(Def. 12) There exist A, B such that $a = A$ and $b = B$ and $\mathrm{AddTo}(a, b) = \mathrm{AddTo}(A, B)$.

The functor $\mathrm{SubFrom}(a, b)$ yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 13) There exist A, B such that $a = A$ and $b = B$ and $\mathrm{SubFrom}(a, b) = \mathrm{SubFrom}(A, B)$.

The functor $\mathrm{MultBy}(a, b)$ yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 14) There exist A, B such that $a = A$ and $b = B$ and $\mathrm{MultBy}(a, b) = \mathrm{MultBy}(A, B)$.

The functor $\mathrm{Divide}(a, b)$ yielding an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined as follows:

(Def. 15) There exist A, B such that $a = A$ and $b = B$ and $\mathrm{Divide}(a, b) = \mathrm{Divide}(A, B)$.

We now state the proposition

(39) The instruction locations of $\mathbf{SCM}$ = the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

Let us consider l_2. The functor goto l_2 yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 16) There exists L_1 such that $l_2 = L_1$ and goto l_2 = goto L_1.

Let us consider a. The functor **if** $a = 0$ **goto** l_2 yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined by:

(Def. 17) There exist A, L_1 such that $a = A$ and $l_2 = L_1$ and **if** $a = 0$ **goto** l_2 = **if** $A = 0$ **goto** L_1.

The functor **if** $a > 0$ **goto** l_2 yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 18) There exist A, L_1 such that $a = A$ and $l_2 = L_1$ and **if** $a > 0$ **goto** l_2 = **if** $A > 0$ **goto** L_1.

Let c, i be integer locations and let a be a finite sequence location. The functor $c{:=}a_i$ yielding an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined by:

(Def. 19) $c{:=}a_i = \langle 9, \langle c, a, i \rangle \rangle$.

The functor $a_i{:=}c$ yielding an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined by:

(Def. 20) $a_i{:=}c = \langle 10, \langle c, a, i \rangle \rangle$.

Let i be an integer location and let a be a finite sequence location. The functor $i{:=}\mathrm{len}\,a$ yielding an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined as follows:

(Def. 21) $i{:=}\mathrm{len}\,a = \langle 11, \langle i, a \rangle \rangle$.

The functor $a{:=}\langle \underbrace{0, \ldots, 0}_{i} \rangle$ yields an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and is defined as follows:

(Def. 22) $a{:=}\langle \underbrace{0, \ldots, 0}_{i} \rangle = \langle 12, \langle i, a \rangle \rangle$.

We now state a number of propositions:

(40) $\mathbf{halt}_{\mathbf{SCM}} = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(41) $\mathrm{InsCode}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}) = 0$.

(42) $\mathrm{InsCode}(a{:=}b) = 1$.

(43) $\mathrm{InsCode}(\mathrm{AddTo}(a, b)) = 2$.

(44) $\mathrm{InsCode}(\mathrm{SubFrom}(a, b)) = 3$.

(45) $\mathrm{InsCode}(\mathrm{MultBy}(a, b)) = 4$.

(46) $\mathrm{InsCode}(\mathrm{Divide}(a, b)) = 5$.

(47) $\mathrm{InsCode}(\mathrm{goto}\ l_3) = 6$.

(48) InsCode(**if** $a = 0$ **goto** l_3) = 7.

(49) InsCode(**if** $a > 0$ **goto** l_3) = 8.

(50) $\mathrm{InsCode}(c{:=}f_a) = 9$.

(51) $\mathrm{InsCode}(f_a{:=}c) = 10$.

(52) $\mathrm{InsCode}(a{:=}\mathrm{len}\,f_1) = 11$.

(53) $\mathrm{InsCode}(f_1{:=}\langle \underbrace{0, \ldots, 0}_{a} \rangle) = 12$.

(54) For every instruction i_2 of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $\mathrm{InsCode}(i_2) = 1$ there exist d_1, d_3 such that $i_2 = d_1{:=}d_3$.

(55) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 2$ there exist d_1, d_3 such that $i_2 = \mathrm{AddTo}(d_1, d_3)$.

(56) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 3$ there exist d_1, d_3 such that $i_2 = \mathrm{SubFrom}(d_1, d_3)$.

(57) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 4$ there exist d_1, d_3 such that $i_2 = \mathrm{MultBy}(d_1, d_3)$.

(58) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 5$ there exist d_1, d_3 such that $i_2 = \mathrm{Divide}(d_1, d_3)$.

(59) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 6$ there exists l_3 such that $i_2 = \mathrm{goto}\ l_3$.

(60) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 7$ there exist l_3, d_1 such that $i_2 = \mathbf{if}\ d_1 = 0\ \mathbf{goto}\ l_3$.

(61) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 8$ there exist l_3, d_1 such that $i_2 = \mathbf{if}\ d_1 > 0\ \mathbf{goto}\ l_3$.

(62) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 9$ there exist a, b, f_1 such that $i_2 = b{:=}{f_1}_a$.

(63) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 10$ there exist a, b, f_1 such that $i_2 = {f_1}_a{:=}b$.

(64) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 11$ there exist a, f_1 such that $i_2 = a{:=}\mathrm{len} f_1$.

(65) For every instruction i_2 of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(i_2) = 12$ there exist a, f_1 such that $i_2 = f_1{:=}\langle\underbrace{0, \ldots, 0}_{a}\rangle$.

5. Relationship to SCM

In the sequel S denotes a state of **SCM** and s, s_1 denote states of $\mathbf{SCM}_{\rm FSA}$.

We now state a number of propositions:

(66) For every state s of $\mathbf{SCM}_{\rm FSA}$ and for every integer location d holds $d \in \mathrm{dom}\, s$.

(67) $f \in \mathrm{dom}\, s$.

(68) $f \notin \mathrm{dom}\, S$.

(69) For every state s of $\mathbf{SCM}_{\rm FSA}$ holds Int-Locations $\subseteq \mathrm{dom}\, s$.

(70) For every state s of $\mathbf{SCM}_{\rm FSA}$ holds FinSeq-Locations $\subseteq \mathrm{dom}\, s$.

(71) For every state s of $\mathbf{SCM}_{\rm FSA}$ holds $\mathrm{dom}(s \restriction \text{Int-Locations}) = \text{Int-Locations}$.

(72) For every state s of $\mathbf{SCM}_{\rm FSA}$ holds $\mathrm{dom}(s \restriction \text{FinSeq-Locations}) = \text{FinSeq-Locations}$.

(73) For every state s of $\mathbf{SCM}_{\rm FSA}$ and for every instruction i of **SCM** holds $s \restriction \mathbb{N} + \cdot (\text{Instr-Loc}_{\rm SCM} \longmapsto i)$ is a state of **SCM**.

(74) For every state s of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every state s' of $\mathbf{SCM}$ holds $s{+\cdot}s'{+\cdot}s \restriction \text{Instr-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$ is a state of $\mathbf{SCM}_{\mathrm{FSA}}$.

(75) Let i be an instruction of $\mathbf{SCM}$, and let i_3 be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, and let s be a state of $\mathbf{SCM}$, and let s_2 be a state of $\mathbf{SCM}_{\mathrm{FSA}}$. If $i = i_3$ and $s = s_2 \restriction \mathbb{N}{+\cdot}(\text{Instr-Loc}_{\mathrm{SCM}} \longmapsto i)$, then $\text{Exec}(i_3, s_2) = s_2{+\cdot}\text{Exec}(i, s){+\cdot}s_2 \restriction \text{Instr-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$ and let d be an integer location. Then $s(d)$ is an integer.

Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$ and let d be a finite sequence location. Then $s(d)$ is a finite sequence of elements of $\mathbb{Z}$.

Next we state several propositions:

(76) If $S = s\restriction\mathbb{N}{+\cdot}(\text{Instr-Loc}_{\mathrm{SCM}} \longmapsto I)$, then $s = s{+\cdot}S{+\cdot}s\restriction\text{Instr-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

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

(78) If $s_1 = s{+\cdot}S{+\cdot}s \restriction \text{Instr-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$, then $s_1(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = S(\mathbf{IC}_{\mathbf{SCM}})$.

(79) If $s_1 = s{+\cdot}S{+\cdot}s \restriction \text{Instr-Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$ and $A = a$, then $S(A) = s_1(a)$.

(80) If $S = s \restriction \mathbb{N}{+\cdot}(\text{Instr-Loc}_{\mathrm{SCM}} \longmapsto I)$ and $A = a$, then $S(A) = s(a)$.

Let us note that $\mathbf{SCM}_{\mathrm{FSA}}$ is halting realistic von Neumann data-oriented definite and steady-programmed.

The following propositions are true:

(81) For every integer location d_2 holds $d_2 \neq \mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(82) For every finite sequence location d_2 holds $d_2 \neq \mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}$.

(83) For every integer location i_1 and for every finite sequence location d_2 holds $i_1 \neq d_2$.

(84) For every instruction-location i_1 of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every integer location d_2 holds $i_1 \neq d_2$.

(85) For every instruction-location i_1 of $\mathbf{SCM}_{\mathrm{FSA}}$ and for every finite sequence location d_2 holds $i_1 \neq d_2$.

(86) Let s_1, s_3 be states of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that
   (i) $\mathbf{IC}_{(s_1)} = \mathbf{IC}_{(s_3)}$,
   (ii) for every integer location a holds $s_1(a) = s_3(a)$,
   (iii) for every finite sequence location f holds $s_1(f) = s_3(f)$, and
   (iv) for every instruction-location i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $s_1(i) = s_3(i)$.

   Then $s_1 = s_3$.

(87) If $S = s$, then $\mathbf{IC}_s = \mathbf{IC}_S$.

(88) If $S = s \restriction \mathbb{N}{+\cdot}(\text{Instr-Loc}_{\mathrm{SCM}} \longmapsto I)$, then $\mathbf{IC}_s = \mathbf{IC}_S$.

6. Users Guide

One can prove the following propositions:

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

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

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

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

(93) Suppose $a \neq b$. Then
 (i) $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
 (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 $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(c) = s(c)$, and
 (v) for every f holds $(\mathrm{Exec}(\mathrm{Divide}(a, b), s))(f) = s(f)$.

(94) $(\mathrm{Exec}(\mathrm{Divide}(a, a), s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$ and $(\mathrm{Exec}(\mathrm{Divide}(a, a), s))(a) = s(a) \bmod s(a)$ and for every c such that $c \neq a$ holds $(\mathrm{Exec}(\mathrm{Divide}(a, a), s))(c) = s(c)$ and for every f holds $(\mathrm{Exec}(\mathrm{Divide}(a, a), s))(f) = s(f)$.

(95) $(\mathrm{Exec}(\mathrm{goto}\ l, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = l$ and for every c holds $(\mathrm{Exec}(\mathrm{goto}\ l, s))(c) = s(c)$ and for every f holds $(\mathrm{Exec}(\mathrm{goto}\ l, s))(f) = s(f)$.

(96) (i) If $s(a) = 0$, then $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = l$,
 (ii) if $s(a) \neq 0$, then $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
 (iii) for every c holds $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l, s))(c) = s(c)$, and
 (iv) for every f holds $(\mathrm{Exec}(\mathbf{if}\ a = 0\ \mathbf{goto}\ l, s))(f) = s(f)$.

(97) (i) If $s(a) > 0$, then $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = l$,
 (ii) if $s(a) \leq 0$, then $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
 (iii) for every c holds $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l, s))(c) = s(c)$, and
 (iv) for every f holds $(\mathrm{Exec}(\mathbf{if}\ a > 0\ \mathbf{goto}\ l, s))(f) = s(f)$.

(98) (i) $(\mathrm{Exec}(c{:=}g_a, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
 (ii) there exists k such that $k = |s(a)|$ and $(\mathrm{Exec}(c{:=}g_a, s))(c) = \pi_k s(g)$,
 (iii) for every b such that $b \neq c$ holds $(\mathrm{Exec}(c{:=}g_a, s))(b) = s(b)$, and
 (iv) for every f holds $(\mathrm{Exec}(c{:=}g_a, s))(f) = s(f)$.

(99) (i) $(\mathrm{Exec}(g_a{:=}c, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
 (ii) there exists k such that $k = |s(a)|$ and $(\mathrm{Exec}(g_a{:=}c, s))(g) = s(g) +\cdot (k, s(c))$,

(iii) for every b holds $(\mathrm{Exec}(g_a{:=}c, s))(b) = s(b)$, and

(iv) for every f such that $f \neq g$ holds $(\mathrm{Exec}(g_a{:=}c, s))(f) = s(f)$.

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

(101) (i) $(\mathrm{Exec}(g{:=}\langle \underbrace{0, \ldots, 0}_{c} \rangle, s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,

(ii) there exists k such that $k = |s(c)|$ and $(\mathrm{Exec}(g{:=}\langle \underbrace{0, \ldots, 0}_{c} \rangle, s))(g) = k \mapsto 0$,

(iii) for every b holds $(\mathrm{Exec}(g{:=}\langle \underbrace{0, \ldots, 0}_{c} \rangle, s))(b) = s(b)$, and

(iv) for every f such that $f \neq g$ holds $(\mathrm{Exec}(g{:=}\langle \underbrace{0, \ldots, 0}_{c} \rangle, s))(f) = s(f)$.

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] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.

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

[6] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Formalized Mathematics*, 1(**3**):529–536, 1990.

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

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

[9] 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.

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

[11] Krzysztof Hryniewiecki. Basic properties of real numbers. *Formalized Mathematics*, 1(**1**):35–40, 1990.

[12] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.

[13] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.

[14] Jan Popiołek. Some properties of functions modul and signum. *Formalized Mathematics*, 1(**2**):263–264, 1990.

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

[16] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.

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

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

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

[20] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[21] Andrzej Trybulec. Tuples, projections and Cartesian products. *Formalized Mathematics*, 1(**1**):97–105, 1990.
[22] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[23] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **SCM**. *Formalized Mathematics*, 5(**4**):507–512, 1996.
[24] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[25] Wojciech A. Trybulec. Pigeon hole principle. *Formalized Mathematics*, 1(**3**):575–579, 1990.
[26] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[27] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.

Received February 7, 1996