JOURNAL OF FORMALIZED MATHEMATICS
Volume 8, Released 1996, Published 2003
Inst. of Computer Science, Univ. of Białystok

The $\mathbf{SCM}_{\mathrm{FSA}}$ Computer

Andrzej Trybulec
Warsaw University
Białystok

Yatsuka Nakamura
Shinshu University
Nagano

Piotr Rudnicki
University of Alberta
Edmonton

MML Identifier: SCMFSA_2.

WWW: http://mizar.org/JFM/Vol8/scmfsa_2.html

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

1. Preliminaries

One can prove the following propositions:

(3)[1] Let N be a set with non empty elements, S be a non void AMI over N, and s be a state of S. Then the instruction locations of $S \subseteq \mathrm{dom}\, s$.

(4) Let N be a set with non empty elements, S be an IC-Ins-separated non void non empty AMI over N, and s be a state of S. Then $\mathbf{IC}_s \in \mathrm{dom}\, s$.

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

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

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

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

Let us observe that $\mathbf{SCM}_{\mathrm{FSA}}$ is non empty and non void.

Next we state two propositions:

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

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

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

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

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

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

[1] The propositions (1) and (2) have been removed.

3. The Memory Structure

In the sequel k is a natural number.

The subset Int-Locations of $\mathbf{SCM}_{\mathrm{FSA}}$ is defined as follows:

(Def. 2) Int-Locations $=$ Data-Loc$_{\mathbf{SCM}_{\mathrm{FSA}}}$.

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

(Def. 3) FinSeq-Locations $=$ Data*-Loc$_{\mathbf{SCM}_{\mathrm{FSA}}}$.

We now state the proposition

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

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

(Def. 4) It $\in$ Data-Loc$_{\mathbf{SCM}_{\mathrm{FSA}}}$.

An object of $\mathbf{SCM}_{\mathrm{FSA}}$ is called a finite sequence location if:

(Def. 5) It $\in$ Data*-Loc$_{\mathbf{SCM}_{\mathrm{FSA}}}$.

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

The following propositions are true:

(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 $\mathbf{SCM}_{\mathrm{FSA}}$.

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

(15) Int-Locations misses FinSeq-Locations.

Let k be a natural number. The functor intloc(k) yields an integer location and is defined by:

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

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

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

The functor fsloc(k) yielding a finite sequence location is defined by:

(Def. 8) fsloc$(k) = -(k+1)$.

We now state a number of propositions:

(16) For all natural numbers k_1, k_2 such that $k_1 \neq k_2$ holds intloc$(k_1) \neq$ intloc(k_2).

(17) For all natural numbers k_1, k_2 such that $k_1 \neq k_2$ holds fsloc$(k_1) \neq$ fsloc(k_2).

(18) For all natural numbers k_1, k_2 such that $k_1 \neq k_2$ holds insloc$(k_1) \neq$ insloc(k_2).

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

(20) For every finite sequence location f_2 there exists a natural number i such that $f_2 =$ 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 $\mathbf{SCM}_{\rm FSA}$ are 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 every set x such that $x \in \text{Data-Loc}_{\mathrm{SCM}_{\rm FSA}}$ holds x is an integer location.

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

(30) For every set x such that $x \in \text{Instr-Loc}_{\mathrm{SCM}_{\rm FSA}}$ 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 as follows:

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

We now state two propositions:

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

(32) For every natural number k holds $\mathrm{Next}(\mathrm{insloc}(k)) = \mathrm{insloc}(k+1)$.

For simplicity, we follow the rules: l_2, l_3 are instruction-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, g are finite sequence locations, A, B are data-locations, and a, b, c, d_3 are integer locations.

The following proposition is true

(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 $\mathbf{SCM}_{\rm FSA}$. One can check that $\mathrm{InsCode}(I)$ is natural.

Next we state four propositions:

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

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

(37)[2] For every instruction i of $\mathbf{SCM}_{\rm FSA}$ 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 $\mathbf{SCM}_{\rm FSA}$.

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

(Def. 11)[3] There exist A, B such that $a = A$ and $b = B$ and $a{:=}b = A{:=}B$.

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

[2] The proposition (36) has been removed.

[3] The definition (Def. 10) has been removed.

(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 $\mathbf{SCM}_{\rm FSA}$ 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 $\mathrm{MultBy}(a,b)$ yielding an instruction of $\mathbf{SCM}_{\rm FSA}$ is defined by:

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

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

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

Next we state the proposition

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

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

(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 $\mathbf{SCM}_{\rm FSA}$ and is defined by:

(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 $\mathbf{SCM}_{\rm FSA}$ 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 $\mathbf{SCM}_{\rm FSA}$ is defined as follows:

(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 $i{:=}\mathrm{len}\,a$ yields an instruction of $\mathbf{SCM}_{\rm FSA}$ and is defined as follows:

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

The functor $a{:=}\langle \underbrace{0,\ldots,0}_{i} \rangle$ yielding an instruction of $\mathbf{SCM}_{\rm FSA}$ is defined by:

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

Next we state a number of propositions:

(42)[4] $\mathrm{InsCode}(a{:=}b) = 1$.

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

[4] The propositions (40) and (41) have been removed.

(48) $\mathrm{InsCode}(\textbf{if } a = 0 \textbf{ goto } l_3) = 7$.

(49) $\mathrm{InsCode}(\textbf{if } a > 0 \textbf{ goto } l_3) = 8$.

(50) $\mathrm{InsCode}(c{:=}{f_1}_a) = 9$.

(51) $\mathrm{InsCode}({f_1}_a{:=}c) = 10$.

(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 $\mathbf{SCM}_{\rm FSA}$ 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 $i_2 = \textbf{if } d_1 = 0 \textbf{ goto } l_3$.

(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 $i_2 = \textbf{if } d_1 > 0 \textbf{ goto } l_3$.

(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 is a state of **SCM** and s, s_1 are states of $\mathbf{SCM}_{\rm FSA}$.

One can prove the following 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 $\mathrm{Int\text{-}Locations} \subseteq \mathrm{dom}\, s$.

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

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

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

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

(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 $\mathbf{SCM}$, i_3 be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$, s be a state of $\mathbf{SCM}$, and 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 $\mathbb{Z}$.

We now 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 mention that $\mathbf{SCM}_{\mathrm{FSA}}$ is realistic, IC-Ins-separated, data-oriented, definite, and steady-programmed.

We now state several propositions:

(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 $s_1 = s_3$.

(88)[5] If $S = s \restriction \mathbb{N} {+\cdot} (\text{Instr-Loc}_{\mathrm{SCM}} \longmapsto I)$, then $\mathbf{IC}_s = \mathbf{IC}_S$.

[5] The proposition (87) has been removed.

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)(i) $(\mathrm{Exec}(\mathrm{Divide}(a,b),s))(\mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \mathrm{Next}(\mathbf{IC}_s)$,
(ii) if $a \neq b$, then $(\mathrm{Exec}(\mathrm{Divide}(a,b),s))(a) = s(a) \div s(b)$,
(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 $(\mathrm{Exec}(c{:=}g_a,s))(c) = s(g)_k$,
(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) $(\mathrm{Exec}(c{:=}\mathrm{len}\,g,s))(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}) = \mathrm{Next}(\mathbf{IC}_s)$ 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) $(\mathrm{Exec}(g{:=}\langle\underbrace{0,\ldots,0}_{c}\rangle,s))(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}) = \mathrm{Next}(\mathbf{IC}_s)$,

(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 $(\mathrm{Exec}(g{:=}\langle\underbrace{0,\ldots,0}_{c}\rangle,s))(f) = s(f)$.

7. HALT INSTRUCTION

The following propositions are true:

(102) For every $\mathbf{SCM}_{\rm FSA}$-state S such that $S = s$ holds $\mathbf{IC}_s = \mathbf{IC}_S$.

(103) For every instruction i of $\mathbf{SCM}$ and for every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that $i = I$ and i is halting holds I is halting.

(104) For every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that there exists s such that $(\mathrm{Exec}(I,s))(\mathbf{IC}_{\mathbf{SCM}_{\rm FSA}}) = \mathrm{Next}(\mathbf{IC}_s)$ holds I is non halting.

(105) $a{:=}b$ is non halting.

(106) $\mathrm{AddTo}(a,b)$ is non halting.

(107) $\mathrm{SubFrom}(a,b)$ is non halting.

(108) $\mathrm{MultBy}(a,b)$ is non halting.

(109) $\mathrm{Divide}(a,b)$ is non halting.

(110) goto l_2 is non halting.

(111) **if** $a = 0$ **goto** l_2 is non halting.

(112) **if** $a > 0$ **goto** l_2 is non halting.

(113) $c{:=}f_a$ is non halting.

(114) $f_a{:=}c$ is non halting.

(115) $c{:=}\mathrm{len}\,f$ is non halting.

(116) $f{:=}\langle\underbrace{0,\ldots,0}_{c}\rangle$ is non halting.

(117) $\langle 0, \emptyset\rangle$ is an instruction of $\mathbf{SCM}_{\rm FSA}$.

(118) For every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that $I = \langle 0, \emptyset\rangle$ holds I is halting.

(119) For every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that $\mathrm{InsCode}(I) = 0$ holds $I = \langle 0, \emptyset\rangle$.

(120) Let I be a set. Then I is an instruction of $\mathbf{SCM}_{\rm FSA}$ if and only if one of the following conditions is satisfied:

$I = \langle 0, \emptyset \rangle$ or there exist a, b such that $I = a{:=}b$ or there exist a, b such that $I = \mathrm{AddTo}(a,b)$ or there exist a, b such that $I = \mathrm{SubFrom}(a,b)$ or there exist a, b such that $I = \mathrm{MultBy}(a,b)$ or there exist a, b such that $I = \mathrm{Divide}(a,b)$ or there exists l_2 such that $I = \mathrm{goto}\ l_2$ or there exist l_3, d_1 such that $I = \mathbf{if}\ d_1 = 0\ \mathbf{goto}\ l_3$ or there exist l_3, d_1 such that $I = \mathbf{if}\ d_1 > 0\ \mathbf{goto}\ l_3$ or there exist b, a, f_1 such that $I = a{:=}{f_1}_b$ or there exist a, b, f_1 such that $I = {f_1}_a{:=}b$ or there exist a, f such that $I = a{:=}\mathrm{len} f$ or there exist a, f such that $I = f{:=}\underbrace{\langle 0, \ldots, 0 \rangle}_{a}$.

Let us mention that $\mathbf{SCM}_{\rm FSA}$ is halting.

One can prove the following propositions:

(121) For every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that I is halting holds $I = \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$.

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

(123) $\mathbf{halt}_{\mathbf{SCM}} = \mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}$.

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

(125) For every instruction i of $\mathbf{SCM}$ and for every instruction I of $\mathbf{SCM}_{\rm FSA}$ such that $i = I$ and i is non halting holds I is non halting.

REFERENCES

[1] Grzegorz Bancerek. The ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal1.html.

[2] Grzegorz Bancerek. Sequences of ordinal numbers. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/ordinal2.html.

[3] Grzegorz Bancerek. König's theorem. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/card_3.html.

[4] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finseq_1.html.

[5] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Journal of Formalized Mathematics*, 8, 1996. http://mizar.org/JFM/Vol8/funct_7.html.

[6] Czesław Byliński. Functions and their basic properties. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funct_1.html.

[7] Czesław Byliński. Functions from a set to a set. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/funct_2.html.

[8] Czesław Byliński. A classical first order language. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/cqc_lang.html.

[9] Czesław Byliński. Finite sequences and tuples of elements of a non-empty sets. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/finseq_2.html.

[10] Czesław Byliński. The modification of a function by a function and the iteration of the composition of a function. *Journal of Formalized Mathematics*, 2, 1990. http://mizar.org/JFM/Vol2/funct_4.html.

[11] Agata Darmochwał. Finite sets. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/finset_1.html.

[12] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Journal of Formalized Mathematics*, 4, 1992. http://mizar.org/JFM/Vol4/ami_1.html.

[13] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Journal of Formalized Mathematics*, 4, 1992. http://mizar.org/JFM/Vol4/ami_2.html.

[14] Jan Popiołek. Some properties of functions modul and signum. *Journal of Formalized Mathematics*, 1, 1989. http://mizar.org/JFM/Vol1/absvalue.html.

[15] Dariusz Surowik. Cyclic groups and some of their properties — part I. *Journal of Formalized Mathematics*, 3, 1991. http://mizar.org/JFM/Vol3/gr_cy_1.html.

[16] Yasushi Tanaka. On the decomposition of the states of SCM. *Journal of Formalized Mathematics*, 5, 1993. http://mizar.org/JFM/Vol5/ami_5.html.

[17] Andrzej Trybulec. Binary operations applied to functions. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funcop_1.html`.

[18] Andrzej Trybulec. Tarski Grothendieck set theory. *Journal of Formalized Mathematics*, Axiomatics, 1989. `http://mizar.org/JFM/Axiomatics/tarski.html`.

[19] Andrzej Trybulec. Tuples, projections and Cartesian products. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/mcart_1.html`.

[20] Andrzej Trybulec. Subsets of real numbers. *Journal of Formalized Mathematics*, Addenda, 2003. `http://mizar.org/JFM/Addenda/numbers.html`.

[21] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/ami_3.html`.

[22] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. An extension of **scm**. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_1.html`.

[23] Michał J. Trybulec. Integers. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/int_1.html`.

[24] Wojciech A. Trybulec. Pigeon hole principle. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/finseq_4.html`.

[25] Zinaida Trybulec. Properties of subsets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/subset_1.html`.

[26] Edmund Woronowicz. Relations and their basic properties. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/relat_1.html`.

Received February 7, 1996

Published January 2, 2004