JOURNAL OF FORMALIZED MATHEMATICS
Volume 13, Released 2001, Published 2003
Inst. of Computer Science, Univ. of Białystok

# On the Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$[1]

Artur Korniłowicz
University of Białystok

MML Identifier: `SCMFSA10`.

WWW: `http://mizar.org/JFM/Vol13/scmfsa10.html`

The articles [20], [19], [29], [3], [16], [21], [15], [30], [7], [8], [2], [22], [27], [1], [9], [5], [10], [11], [17], [4], [6], [28], [13], [14], [23], [25], [24], [18], [26], and [12] provide the notation and terminology for this paper.

For simplicity, we adopt the following convention: $a$, $b$ denote integer locations, $f$ denotes a finite sequence location, $i_1$, $i_2$, $i_3$ denote instruction-locations of $\mathbf{SCM}_{\mathrm{FSA}}$, $T$ denotes an instruction type of $\mathbf{SCM}_{\mathrm{FSA}}$, and $k$ denotes a natural number.

The following two propositions are true:

(1) For every function $f$ and for all sets $a$, $A$, $b$, $B$, $c$, $C$ such that $a \neq b$ and $a \neq c$ holds $(f+\cdot(a\dot{\longmapsto}A)+\cdot(b\dot{\longmapsto}B)+\cdot(c\dot{\longmapsto}C))(a) = A$.

(2) For all sets $a$, $b$ holds $\langle a \rangle +\cdot (1,b) = \langle b \rangle$.

Let $l_1$, $l_2$ be integer locations and let $a$, $b$ be integers. Then $[l_1 \longmapsto a, l_2 \longmapsto b]$ is a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$.

One can prove the following propositions:

(3) $a \notin$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(4) $f \notin$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(5) $\text{Data-Loc}_{\mathbf{SCM}_{\mathrm{FSA}}} \neq$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(6) $\text{Data}^*\text{-Loc}_{\mathbf{SCM}_{\mathrm{FSA}}} \neq$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$.

(7) Let $o$ be an object of $\mathbf{SCM}_{\mathrm{FSA}}$. Then
(i) $o = \mathbf{IC}_{\mathbf{SCM}_{\mathrm{FSA}}}$, or
(ii) $o \in$ the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$, or
(iii) $o$ is an integer location and a finite sequence location.

(8) If $i_2 \neq i_3$, then $\text{Next}(i_2) \neq \text{Next}(i_3)$.

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

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

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

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

[1]This work has been partially supported by TYPES grant IST-1999-29001.

(13) $\mathrm{Divide}(a,b) = \langle 5, \langle a,b \rangle \rangle$.

(14) goto $i_1 = \langle 6, \langle i_1 \rangle \rangle$.

(15) **if** $a = 0$ **goto** $i_1 = \langle 7, \langle i_1, a \rangle \rangle$.

(16) **if** $a > 0$ **goto** $i_1 = \langle 8, \langle i_1, a \rangle \rangle$.

(17) $\mathrm{AddressPart}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}) = \emptyset$.

(18) $\mathrm{AddressPart}(a{:=}b) = \langle a,b \rangle$.

(19) $\mathrm{AddressPart}(\mathrm{AddTo}(a,b)) = \langle a,b \rangle$.

(20) $\mathrm{AddressPart}(\mathrm{SubFrom}(a,b)) = \langle a,b \rangle$.

(21) $\mathrm{AddressPart}(\mathrm{MultBy}(a,b)) = \langle a,b \rangle$.

(22) $\mathrm{AddressPart}(\mathrm{Divide}(a,b)) = \langle a,b \rangle$.

(23) $\mathrm{AddressPart}(\mathrm{goto}\ i_2) = \langle i_2 \rangle$.

(24) AddressPart(**if** $a = 0$ **goto** $i_2$) $= \langle i_2, a \rangle$.

(25) AddressPart(**if** $a > 0$ **goto** $i_2$) $= \langle i_2, a \rangle$.

(26) $\mathrm{AddressPart}(b{:=}f_a) = \langle b, f, a \rangle$.

(27) $\mathrm{AddressPart}(f_a{:=}b) = \langle b, f, a \rangle$.

(28) $\mathrm{AddressPart}(a{:=}\mathrm{len} f) = \langle a, f \rangle$.

(29) $\mathrm{AddressPart}(f{:=}\langle \underbrace{0,\ldots,0}_{a} \rangle) = \langle a, f \rangle$.

(30) If $T = 0$, then $\mathrm{AddressParts}\, T = \{0\}$.

Let us consider $T$. Observe that $\mathrm{AddressParts}\, T$ is non empty.
Next we state a number of propositions:

(31) If $T = 1$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(32) If $T = 2$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(33) If $T = 3$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(34) If $T = 4$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(35) If $T = 5$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(36) If $T = 6$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1\}$.

(37) If $T = 7$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(38) If $T = 8$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(39) If $T = 9$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2,3\}$.

(40) If $T = 10$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2,3\}$.

(41) If $T = 11$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(42) If $T = 12$, then $\mathrm{dom} \prod_{\mathrm{AddressParts}\, T} = \{1,2\}$.

(43) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(a:=b)}(1) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(44) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(a:=b)}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}_{\mathrm{FSA}}}$.

(45) $\prod_{\rm AddressParts\,InsCode(AddTo(\mathit{a},\mathit{b}))}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(46) $\prod_{\rm AddressParts\,InsCode(AddTo(\mathit{a},\mathit{b}))}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(47) $\prod_{\rm AddressParts\,InsCode(SubFrom(\mathit{a},\mathit{b}))}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(48) $\prod_{\rm AddressParts\,InsCode(SubFrom(\mathit{a},\mathit{b}))}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(49) $\prod_{\rm AddressParts\,InsCode(MultBy(\mathit{a},\mathit{b}))}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(50) $\prod_{\rm AddressParts\,InsCode(MultBy(\mathit{a},\mathit{b}))}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(51) $\prod_{\rm AddressParts\,InsCode(Divide(\mathit{a},\mathit{b}))}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(52) $\prod_{\rm AddressParts\,InsCode(Divide(\mathit{a},\mathit{b}))}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(53) $\prod_{\rm AddressParts\,InsCode(goto\ \mathit{i}_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(54) $\prod_{\rm AddressParts\,InsCode(\mathbf{if}\ \mathit{a}=0\ \mathbf{goto}\ \mathit{i}_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(55) $\prod_{\rm AddressParts\,InsCode(\mathbf{if}\ \mathit{a}=0\ \mathbf{goto}\ \mathit{i}_2)}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(56) $\prod_{\rm AddressParts\,InsCode(\mathbf{if}\ \mathit{a}>0\ \mathbf{goto}\ \mathit{i}_2)}(1) =$ the instruction locations of $\mathbf{SCM}_{\rm FSA}$.

(57) $\prod_{\rm AddressParts\,InsCode(\mathbf{if}\ \mathit{a}>0\ \mathbf{goto}\ \mathit{i}_2)}(2) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(58) $\prod_{\rm AddressParts\,InsCode(\mathit{b}:=\mathit{f}_\mathit{a})}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(59) $\prod_{\rm AddressParts\,InsCode(\mathit{b}:=\mathit{f}_\mathit{a})}(2) = \text{Data}^*\text{-Loc}_{\rm SCM_{FSA}}$.

(60) $\prod_{\rm AddressParts\,InsCode(\mathit{b}:=\mathit{f}_\mathit{a})}(3) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(61) $\prod_{\rm AddressParts\,InsCode(\mathit{f}_\mathit{a}:=\mathit{b})}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(62) $\prod_{\rm AddressParts\,InsCode(\mathit{f}_\mathit{a}:=\mathit{b})}(2) = \text{Data}^*\text{-Loc}_{\rm SCM_{FSA}}$.

(63) $\prod_{\rm AddressParts\,InsCode(\mathit{f}_\mathit{a}:=\mathit{b})}(3) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(64) $\prod_{\rm AddressParts\,InsCode(\mathit{a}:=len\,\mathit{f})}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(65) $\prod_{\rm AddressParts\,InsCode(\mathit{a}:=len\,\mathit{f})}(2) = \text{Data}^*\text{-Loc}_{\rm SCM_{FSA}}$.

(66) $\prod_{\rm AddressParts\,InsCode(\mathit{f}:=\langle\underbrace{0,\ldots,0}_{a}\rangle)}(1) = \text{Data-Loc}_{\rm SCM_{FSA}}$.

(67) $\prod_{\rm AddressParts\,InsCode(\mathit{f}:=\langle\underbrace{0,\ldots,0}_{a}\rangle)}(2) = \text{Data}^*\text{-Loc}_{\rm SCM_{FSA}}$.

(68) $\text{NIC}(\mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}, i_1) = \{i_1\}$.

Let us note that $\text{JUMP}(\mathbf{halt}_{\mathbf{SCM}_{\rm FSA}})$ is empty.
One can prove the following proposition

(69) $\text{NIC}(a:=b, i_1) = \{\text{Next}(i_1)\}$.

Let us consider $a$, $b$. Observe that $\text{JUMP}(a:=b)$ is empty.
The following proposition is true

(70) $\text{NIC}(\text{AddTo}(a,b), i_1) = \{\text{Next}(i_1)\}$.

Let us consider $a$, $b$. Observe that $\text{JUMP}(\text{AddTo}(a,b))$ is empty.
Next we state the proposition

(71) $\text{NIC}(\text{SubFrom}(a,b), i_1) = \{\text{Next}(i_1)\}$.

Let us consider $a$, $b$. Note that $\mathrm{JUMP}(\mathrm{SubFrom}(a,b))$ is empty.
Next we state the proposition

(72) $\mathrm{NIC}(\mathrm{MultBy}(a,b),i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $b$. One can check that $\mathrm{JUMP}(\mathrm{MultBy}(a,b))$ is empty.
Next we state the proposition

(73) $\mathrm{NIC}(\mathrm{Divide}(a,b),i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $b$. One can check that $\mathrm{JUMP}(\mathrm{Divide}(a,b))$ is empty.
We now state two propositions:

(74) $\mathrm{NIC}(\mathrm{goto}\ i_2,i_1) = \{i_2\}$.

(75) $\mathrm{JUMP}(\mathrm{goto}\ i_2) = \{i_2\}$.

Let us consider $i_2$. Note that $\mathrm{JUMP}(\mathrm{goto}\ i_2)$ is non empty and trivial.
The following two propositions are true:

(76) $\mathrm{NIC}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2,i_1) = \{i_2,\mathrm{Next}(i_1)\}$.

(77) $\mathrm{JUMP}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2) = \{i_2\}$.

Let us consider $a$, $i_2$. One can check that $\mathrm{JUMP}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2)$ is non empty and trivial.
Next we state two propositions:

(78) $\mathrm{NIC}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2,i_1) = \{i_2,\mathrm{Next}(i_1)\}$.

(79) $\mathrm{JUMP}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2) = \{i_2\}$.

Let us consider $a$, $i_2$. Observe that $\mathrm{JUMP}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2)$ is non empty and trivial.
One can prove the following proposition

(80) $\mathrm{NIC}(a{:=}f_b,i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $b$, $f$. Observe that $\mathrm{JUMP}(a{:=}f_b)$ is empty.
The following proposition is true

(81) $\mathrm{NIC}(f_b{:=}a,i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $b$, $f$. One can check that $\mathrm{JUMP}(f_b{:=}a)$ is empty.
Next we state the proposition

(82) $\mathrm{NIC}(a{:=}\mathrm{len} f,i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $f$. One can verify that $\mathrm{JUMP}(a{:=}\mathrm{len} f)$ is empty.
One can prove the following proposition

(83) $\mathrm{NIC}(f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle,i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider $a$, $f$. Observe that $\mathrm{JUMP}(f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle)$ is empty.

One can prove the following two propositions:

(84) $\mathrm{SUCC}(i_1) = \{i_1,\mathrm{Next}(i_1)\}$.

(85) Let $f$ be a function from $\mathbb{N}$ into the instruction locations of $\mathbf{SCM}_{\mathrm{FSA}}$. Suppose that for every natural number $k$ holds $f(k) = \mathrm{insloc}(k)$. Then

(i) $f$ is bijective, and

(ii) for every natural number $k$ holds $f(k+1) \in \mathrm{SUCC}(f(k))$ and for every natural number $j$ such that $f(j) \in \mathrm{SUCC}(f(k))$ holds $k \leq j$.

Let us mention that $\mathbf{SCM}_{\mathrm{FSA}}$ is standard.

The following three propositions are true:

(86) $\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k) = \mathrm{insloc}(k)$.

(87) $\mathrm{Next}(\mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k)) = \mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(k+1)$.

(88) $\mathrm{Next}(i_1) = \mathrm{NextLoc}\, i_1$.

Let us note that $\mathrm{InsCode}(\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}})$ is jump-only.

One can check that $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ is jump-only.

Let us consider $i_2$. Observe that $\mathrm{InsCode}(\mathrm{goto}\ i_2)$ is jump-only.

Let us consider $i_2$. Observe that goto $i_2$ is jump-only, non sequential, and non instruction location free.

Let us consider $a$, $i_2$. Observe that $\mathrm{InsCode}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2)$ is jump-only and $\mathrm{InsCode}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2)$ is jump-only.

Let us consider $a$, $i_2$. Observe that $\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2$ is jump-only, non sequential, and non instruction location free and $\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2$ is jump-only, non sequential, and non instruction location free.

Let us consider $a$, $b$. One can check the following observations:

* $\mathrm{InsCode}(a{:=}b)$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{AddTo}(a,b))$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{SubFrom}(a,b))$ is non jump-only,
* $\mathrm{InsCode}(\mathrm{MultBy}(a,b))$ is non jump-only, and
* $\mathrm{InsCode}(\mathrm{Divide}(a,b))$ is non jump-only.

Let us consider $a$, $b$. One can verify the following observations:

* $a{:=}b$ is non jump-only and sequential,
* $\mathrm{AddTo}(a,b)$ is non jump-only and sequential,
* $\mathrm{SubFrom}(a,b)$ is non jump-only and sequential,
* $\mathrm{MultBy}(a,b)$ is non jump-only and sequential, and
* $\mathrm{Divide}(a,b)$ is non jump-only and sequential.

Let us consider $a$, $b$, $f$. Observe that $\mathrm{InsCode}(b{:=}f_a)$ is non jump-only and $\mathrm{InsCode}(f_a{:=}b)$ is non jump-only.

Let us consider $a$, $b$, $f$. One can check that $b{:=}f_a$ is non jump-only and sequential and $f_a{:=}b$ is non jump-only and sequential.

Let us consider $a$, $f$. Observe that $\mathrm{InsCode}(a{:=}\mathrm{len} f)$ is non jump-only and $\mathrm{InsCode}(f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle)$ is non jump-only.

Let us consider $a$, $f$. Note that $a{:=}\mathrm{len} f$ is non jump-only and sequential and $f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle$ is non jump-only and sequential.

One can check that $\mathbf{SCM}_{\mathrm{FSA}}$ is homogeneous and has explicit jumps and no implicit jumps.

One can verify that $\mathbf{SCM}_{\mathrm{FSA}}$ is regular.

The following propositions are true:

(89) $\mathrm{IncAddr}(\mathrm{goto}\ i_2, k) = \mathrm{goto}\ \mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2)+k)$.

(90) $\mathrm{IncAddr}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2, k) = \mathbf{if}\ a = 0\ \mathbf{goto}\ \mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2)+k)$.

(91) $\mathrm{IncAddr}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2, k) = \mathbf{if}\ a > 0\ \mathbf{goto}\ \mathrm{il}_{\mathbf{SCM}_{\mathrm{FSA}}}(\mathrm{locnum}(i_2)+k)$.

One can verify that $\mathbf{SCM}_{\mathrm{FSA}}$ is IC-good and Exec-preserving.

## REFERENCES


[1] Grzegorz Bancerek. The fundamental properties of natural numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/nat_1.html`.

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

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

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

[5] 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`.

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

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

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

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

[10] 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`.

[11] 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`.

[12] Artur Korniłowicz. On the composition of macro instructions of standard computers. *Journal of Formalized Mathematics*, 12, 2000. `http://mizar.org/JFM/Vol12/amistd_2.html`.

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

[14] 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`.

[15] Beata Padlewska. Families of sets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/setfam_1.html`.

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

[17] 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`.

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

[19] Andrzej Trybulec. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/enumset1.html`.

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

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

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

[23] 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`.

[24] Andrzej Trybulec, Yatsuka Nakamura, and Piotr Rudnicki. The **SCM**$_{\mathrm{FSA}}$ computer. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_2.html`.

[25] 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`.

[26] Andrzej Trybulec, Piotr Rudnicki, and Artur Korniłowicz. Standard ordering of instruction locations. *Journal of Formalized Mathematics*, 12, 2000. `http://mizar.org/JFM/Vol12/amistd_1.html`.

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

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

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

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

*Received May 8, 2001*

*Published January 2, 2004*