FORMALIZED MATHEMATICS
Volume 9, Number 4, 2001
University of Białystok

On the Instructions of SCM[1]

Artur Korniłowicz
University of Białystok

MML Identifier: AMI_6.

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

For simplicity, we adopt the following rules: a, b are data-locations, i_1, i_2, i_3 are instruction-locations of **SCM**, s_1, s_2 are states of **SCM**, T is an instruction type of **SCM**, and k is a natural number.

We now state a number of propositions:

(1) $a \notin$ the instruction locations of **SCM**.

(2) $\text{Data-Loc}_{\text{SCM}} \neq$ the instruction locations of **SCM**.

(3) For every object o of **SCM** holds $o = \mathbf{IC}_{\mathbf{SCM}}$ or $o \in$ the instruction locations of **SCM** or o is a data-location.

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

(5) If s_1 and s_2 are equal outside the instruction locations of **SCM**, then $s_1(a) = s_2(a)$.

(6) Let N be a set with non empty elements, S be a realistic IC-Ins-separated definite non empty non void AMI over N, t, u be states of S, i_1 be an instruction-location of S, e be an element of $\text{ObjectKind}(\mathbf{IC}_S)$, and I be an element of $\text{ObjectKind}(i_1)$. If $e = i_1$ and $u = t{+}{\cdot}[\mathbf{IC}_S \longmapsto e, i_1 \longmapsto I]$, then $u(i_1) = I$ and $\mathbf{IC}_u = i_1$ and $\mathbf{IC}_{\text{Following}(u)} = (\text{Exec}(u(\mathbf{IC}_u), u))(\mathbf{IC}_S)$.

(7) $\text{AddressPart}(\mathbf{halt}_{\mathbf{SCM}}) = \emptyset$.

(8) $\text{AddressPart}(a{:=}b) = \langle a, b\rangle$.

(9) $\text{AddressPart}(\text{AddTo}(a, b)) = \langle a, b\rangle$.

(10) $\text{AddressPart}(\text{SubFrom}(a, b)) = \langle a, b\rangle$.

(11) $\text{AddressPart}(\text{MultBy}(a, b)) = \langle a, b\rangle$.

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

ISSN 1426–2630

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

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

(14) $\mathrm{AddressPart}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2) = \langle i_2, a\rangle$.

(15) $\mathrm{AddressPart}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2) = \langle i_2, a\rangle$.

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

Let us consider T. One can check that $\mathrm{AddressParts}\, T$ is non empty.

The following propositions are true:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(35) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(\mathrm{goto}\ i_2)}(1) =$ the instruction locations of **SCM**.

(36) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(\mathbf{if}\ a=0\ \mathbf{goto}\ i_2)}(1) =$ the instruction locations of **SCM**.

(37) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(\mathbf{if}\ a=0\ \mathbf{goto}\ i_2)}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$.

(38) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(\mathbf{if}\ a>0\ \mathbf{goto}\ i_2)}(1) =$ the instruction locations of **SCM**.

(39) $\prod_{\mathrm{AddressParts}\, \mathrm{InsCode}(\mathbf{if}\ a>0\ \mathbf{goto}\ i_2)}(2) = \mathrm{Data\text{-}Loc}_{\mathrm{SCM}}$.

(40) $\mathrm{NIC}(\mathbf{halt}_{\mathbf{SCM}}, i_1) = \{i_1\}$.

Let us note that $\mathrm{JUMP}(\mathbf{halt}_{\mathbf{SCM}})$ is empty.

One can prove the following proposition

(41) $\mathrm{NIC}(a:=b, i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. One can verify that JUMP(a:=b) is empty.

Next we state the proposition

(42) $\mathrm{NIC}(\mathrm{AddTo}(a,b), i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. Note that JUMP(AddTo(a, b)) is empty.

The following proposition is true

(43) $\mathrm{NIC}(\mathrm{SubFrom}(a,b), i_1) = \{\mathrm{Next}(i_1)\}$.

Let us consider a, b. One can check that JUMP(SubFrom(a, b)) is empty.

Next we state the proposition

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

Let us consider a, b. Observe that JUMP(MultBy(a, b)) is empty.

The following proposition is true

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

Let us consider a, b. Note that JUMP(Divide(a, b)) is empty.

We now state two propositions:

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

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

Let us consider i_2. One can check that JUMP(goto i_2) is non empty and trivial.

The following two propositions are true:

(48) $i_2 \in \mathrm{NIC}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2, i_1)$ and $\mathrm{NIC}(\mathbf{if}\ a = 0\ \mathbf{goto}\ i_2, i_1) \subseteq \{i_2, \mathrm{Next}(i_1)\}$.

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

Let us consider a, i_2. Note that JUMP(**if** $a = 0$ **goto** i_2) is non empty and trivial.

One can prove the following propositions:

(50) $i_2 \in \mathrm{NIC}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2, i_1)$ and $\mathrm{NIC}(\mathbf{if}\ a > 0\ \mathbf{goto}\ i_2, i_1) \subseteq \{i_2, \mathrm{Next}(i_1)\}$.

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

Let us consider a, i_2. One can check that JUMP(**if** $a > 0$ **goto** i_2) is non empty and trivial.

Next we state two propositions:

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

(53) Let f be a function from $\mathbb{N}$ into the instruction locations of **SCM**. Suppose that for every natural number k holds $f(k) = \mathbf{i}_k$. 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 $k \leqslant j$.

Let us note that **SCM** is standard.

One can prove the following three propositions:

(54) $\mathrm{il}_{\mathbf{SCM}}(k) = \mathbf{i}_k$.

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

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

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

Let us observe that $\mathbf{halt}_{\mathbf{SCM}}$ is jump-only.

Let us consider i_2. Observe that InsCode(goto i_2) is jump-only.

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

Let us consider a, i_2. One can verify that InsCode(**if** $a = 0$ **goto** i_2) is jump-only and InsCode(**if** $a > 0$ **goto** i_2) is jump-only.

Let us consider a, i_2. One can verify that **if** $a = 0$ **goto** i_2 is jump-only non sequential and non instruction location free and **if** $a > 0$ **goto** i_2 is jump-only non sequential and non instruction location free.

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

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

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

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

Let us note that **SCM** is homogeneous and has explicit jumps and no implicit jumps.

Let us observe that **SCM** is regular.

We now state three propositions:

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

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

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

Let us note that **SCM** is IC-good and Exec-preserving.

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. Sequences of ordinal numbers. *Formalized Mathematics*, 1(**2**):281–290, 1990.

[4] Grzegorz Bancerek and Krzysztof Hryniewiecki. Segments of natural numbers and finite sequences. *Formalized Mathematics*, 1(**1**):107–114, 1990.
[5] Grzegorz Bancerek and Andrzej Trybulec. Miscellaneous facts about functions. *Formalized Mathematics*, 5(**4**):485–492, 1996.
[6] Czesław Byliński. Functions and their basic properties. *Formalized Mathematics*, 1(**1**):55–65, 1990.
[7] Czesław Byliński. Functions from a set to a set. *Formalized Mathematics*, 1(**1**):153–164, 1990.
[8] Artur Korniłowicz. On the composition of macro instructions of standard computers. *Formalized Mathematics*, 9(**2**):303–316, 2001.
[9] Yatsuka Nakamura and Andrzej Trybulec. A mathematical model of CPU. *Formalized Mathematics*, 3(**2**):151–160, 1992.
[10] Yatsuka Nakamura and Andrzej Trybulec. On a mathematical model of programs. *Formalized Mathematics*, 3(**2**):241–250, 1992.
[11] Yasushi Tanaka. On the decomposition of the states of SCM. *Formalized Mathematics*, 5(**1**):1–8, 1996.
[12] Yozo Toda. The formalization of simple graphs. *Formalized Mathematics*, 5(**1**):137–144, 1996.
[13] Andrzej Trybulec. Tarski Grothendieck set theory. *Formalized Mathematics*, 1(**1**):9–11, 1990.
[14] Andrzej Trybulec and Yatsuka Nakamura. Some remarks on the simple concrete model of computer. *Formalized Mathematics*, 4(**1**):51–56, 1993.
[15] Andrzej Trybulec, Piotr Rudnicki, and Artur Korniłowicz. Standard ordering of instruction locations. *Formalized Mathematics*, 9(**2**):291–301, 2001.
[16] Michał J. Trybulec. Integers. *Formalized Mathematics*, 1(**3**):501–505, 1990.
[17] Zinaida Trybulec. Properties of subsets. *Formalized Mathematics*, 1(**1**):67–71, 1990.
[18] Zinaida Trybulec and Halina Święczkowska. Boolean properties of sets. *Formalized Mathematics*, 1(**1**):17–23, 1990.
[19] Edmund Woronowicz. Relations and their basic properties. *Formalized Mathematics*, 1(**1**):73–83, 1990.
[20] Edmund Woronowicz. Relations defined on sets. *Formalized Mathematics*, 1(**1**):181–186, 1990.

Received May 8, 2001