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

Constant Assignment Macro Instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Part II

Noriko Asamoto
Ochanomizu University
Tokyo

MML Identifier: SCMFSA7B.
WWW: http://mizar.org/JFM/Vol8/scmfsa7b.html

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

In this paper m is a natural number.

We now state two propositions:

(1) For every finite sequence p of elements of the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{dom}\,\mathrm{Load}(p) = \{\mathrm{insloc}(m) : m < \mathrm{len}\, p\}$.

(2) For every finite sequence p of elements of the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{rng}\,\mathrm{Load}(p) = \mathrm{rng}\, p$.

Let p be a finite sequence of elements of the instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. Observe that $\mathrm{Load}(p)$ is initial and programmed.

Next we state several propositions:

(3) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{Load}(\langle i \rangle) = \mathrm{insloc}(0) \mathrel{\dot{\longmapsto}} i$.

(4) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{dom}\,\mathrm{Macro}(i) = \{\mathrm{insloc}(0), \mathrm{insloc}(1)\}$.

(5) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{Macro}(i) = \mathrm{Load}(\langle i, \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}} \rangle)$.

(6) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $\mathrm{card}\,\mathrm{Macro}(i) = 2$.

(7) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds if $i = \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$, then $(\mathrm{Directed}(\mathrm{Macro}(i)))(\mathrm{insloc}(0)) = \mathbf{goto}\ \mathrm{insloc}(2)$ and if $i \neq \mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$, then $(\mathrm{Directed}(\mathrm{Macro}(i)))(\mathrm{insloc}(0)) = i$.

(8) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ holds $(\mathrm{Directed}(\mathrm{Macro}(i)))(\mathrm{insloc}(1)) = \mathbf{goto}\ \mathrm{insloc}(2)$.

Let a be an integer location and let k be an integer. Observe that $a{:=}k$ is initial and programmed.

Let a be an integer location and let k be an integer. Observe that $a{:=}k$ is parahalting.

One can prove the following proposition

(9) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, a be a read-write integer location, and k be an integer. Then

(i) $(\mathrm{IExec}(a{:=}k, s))(a) = k$,

(ii) for every read-write integer location b such that $b \neq a$ holds $(\mathrm{IExec}(a{:=}k, s))(b) = s(b)$, and

(iii) for every finite sequence location f holds $(\mathrm{IExec}(a{:=}k, s))(f) = s(f)$.

Let f be a finite sequence location and let p be a finite sequence of elements of $\mathbb{Z}$. One can verify that $f{:=}p$ is initial and programmed.

Let f be a finite sequence location and let p be a finite sequence of elements of $\mathbb{Z}$. One can check that $f{:=}p$ is parahalting.

The following proposition is true

(10) Let s be a state of $\mathbf{SCM}_{\mathrm{FSA}}$, f be a finite sequence location, and p be a finite sequence of elements of $\mathbb{Z}$. Then

(i) $(\mathrm{IExec}(f{:=}p,s))(f) = p$,

(ii) for every read-write integer location a such that $a \neq \mathrm{intloc}(1)$ and $a \neq \mathrm{intloc}(2)$ holds $(\mathrm{IExec}(f{:=}p,s))(a) = s(a)$, and

(iii) for every finite sequence location g such that $g \neq f$ holds $(\mathrm{IExec}(f{:=}p,s))(g) = s(g)$.

Let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let a be an integer location. We say that i does not refer a if and only if the condition (Def. 1) is satisfied.

(Def. 1) Let b be an integer location, l be an instruction-location of $\mathbf{SCM}_{\mathrm{FSA}}$, and f be a finite sequence location. Then $b{:=}a \neq i$ and $\mathrm{AddTo}(b,a) \neq i$ and $\mathrm{SubFrom}(b,a) \neq i$ and $\mathrm{MultBy}(b,a) \neq i$ and $\mathrm{Divide}(b,a) \neq i$ and $\mathrm{Divide}(a,b) \neq i$ and **if** $a = 0$ **goto** $l \neq i$ and **if** $a > 0$ **goto** $l \neq i$ and $b{:=}f_a \neq i$ and $f_b{:=}a \neq i$ and $f_a{:=}b \neq i$ and $f{:=}\langle\underbrace{0,\ldots,0}_{a}\rangle \neq i$.

Let I be a programmed finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$ and let a be an integer location. We say that I does not refer a if and only if:

(Def. 2) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $i \in \mathrm{rng}\, I$ holds i does not refer a.

Let i be an instruction of $\mathbf{SCM}_{\mathrm{FSA}}$ and let a be an integer location. We say that i does not destroy a if and only if the condition (Def. 3) is satisfied.

(Def. 3) Let b be an integer location and f be a finite sequence location. Then $a{:=}b \neq i$ and $\mathrm{AddTo}(a,b) \neq i$ and $\mathrm{SubFrom}(a,b) \neq i$ and $\mathrm{MultBy}(a,b) \neq i$ and $\mathrm{Divide}(a,b) \neq i$ and $\mathrm{Divide}(b,a) \neq i$ and $a{:=}f_b \neq i$ and $a{:=}\mathrm{len} f \neq i$.

Let I be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$ and let a be an integer location. We say that I does not destroy a if and only if:

(Def. 4) For every instruction i of $\mathbf{SCM}_{\mathrm{FSA}}$ such that $i \in \mathrm{rng}\, I$ holds i does not destroy a.

Let I be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. We say that I is good if and only if:

(Def. 5) I does not destroy $\mathrm{intloc}(0)$.

Let I be a finite partial state of $\mathbf{SCM}_{\mathrm{FSA}}$. We say that I is halt-free if and only if:

(Def. 6) $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}} \notin \mathrm{rng}\, I$.

One can verify that there exists a macro instruction which is halt-free and good.

Next we state a number of propositions:

(11) For every integer location a holds $\mathbf{halt}_{\mathbf{SCM}_{\mathrm{FSA}}}$ does not destroy a.

(12) For all integer locations a, b, c such that $a \neq b$ holds $b{:=}c$ does not destroy a.

(13) For all integer locations a, b, c such that $a \neq b$ holds $\mathrm{AddTo}(b,c)$ does not destroy a.

(14) For all integer locations a, b, c such that $a \neq b$ holds $\mathrm{SubFrom}(b,c)$ does not destroy a.

(15) For all integer locations a, b, c such that $a \neq b$ holds $\mathrm{MultBy}(b,c)$ does not destroy a.

(16) For all integer locations a, b, c such that $a \neq b$ and $a \neq c$ holds $\mathrm{Divide}(b,c)$ does not destroy a.

(17) For every integer location a and for every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds goto l does not destroy a.

(18) For all integer locations a, b and for every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds **if** $b = 0$ **goto** l does not destroy a.

(19) For all integer locations a, b and for every instruction-location l of $\mathbf{SCM}_{\rm FSA}$ holds **if** $b > 0$ **goto** l does not destroy a.

(20) Let a, b, c be integer locations and f be a finite sequence location. If $a \neq b$, then $b{:=}f_c$ does not destroy a.

(21) For all integer locations a, b, c and for every finite sequence location f holds $f_c{:=}b$ does not destroy a.

(22) Let a, b be integer locations and f be a finite sequence location. If $a \neq b$, then $b{:=}{\rm len} f$ does not destroy a.

(23) For all integer locations a, b and for every finite sequence location f holds $f{:=}\langle\underbrace{0,\ldots,0}_{b}\rangle$ does not destroy a.

Let I be a finite partial state of $\mathbf{SCM}_{\rm FSA}$ and let s be a state of $\mathbf{SCM}_{\rm FSA}$. We say that I is closed on s if and only if:

(Def. 7) For every natural number k holds $\mathbf{IC}_{({\rm Computation}(s+\cdot(I+\cdot\,{\rm Start\text{-}At}({\rm insloc}(0)))))(k)} \in {\rm dom}\, I$.

We say that I is halting on s if and only if:

(Def. 8) $s+\cdot(I+\cdot\,{\rm Start\text{-}At}({\rm insloc}(0)))$ is halting.

The following propositions are true:

(24) For every macro instruction I holds I is paraclosed iff for every state s of $\mathbf{SCM}_{\rm FSA}$ holds I is closed on s.

(25) For every macro instruction I holds I is parahalting iff for every state s of $\mathbf{SCM}_{\rm FSA}$ holds I is halting on s.

(26) Let i be an instruction of $\mathbf{SCM}_{\rm FSA}$, a be an integer location, and s be a state of $\mathbf{SCM}_{\rm FSA}$. If i does not destroy a, then $({\rm Exec}(i,s))(a) = s(a)$.

(27) Let s be a state of $\mathbf{SCM}_{\rm FSA}$, I be a macro instruction, and a be an integer location. Suppose I does not destroy a and I is closed on s. Let k be a natural number. Then $({\rm Computation}(s+\cdot(I+\cdot\,{\rm Start\text{-}At}({\rm insloc}(0)))))(k)(a) = s(a)$.

(28) ${\rm Stop}_{\mathbf{SCM}_{\rm FSA}}$ does not destroy intloc(0).

One can verify that there exists a macro instruction which is parahalting and good.

Let us note that ${\rm Stop}_{\mathbf{SCM}_{\rm FSA}}$ is parahalting and good.

Let us mention that every macro instruction which is paraclosed and good is also keeping 0.

We now state two propositions:

(29) For every integer location a and for every integer k holds ${\rm rng}\,{\rm aSeq}(a,k) \subseteq \{a{:=}{\rm intloc}(0), {\rm AddTo}(a,{\rm intloc}(0)), {\rm SubFrom}(a,{\rm intloc}(0))\}$.

(30) For every integer location a and for every integer k holds ${\rm rng}(a{:=}k) \subseteq \{\mathbf{halt}_{\mathbf{SCM}_{\rm FSA}}, a{:=}{\rm intloc}(0), {\rm AddTo}(a,{\rm intloc}(0)), {\rm SubFrom}(a,{\rm intloc}(0))\}$.

Let a be a read-write integer location and let k be an integer. Observe that $a{:=}k$ is good.

Let a be a read-write integer location and let k be an integer. Note that $a{:=}k$ is keeping 0.

References

[1] Noriko Asamoto. Some multi instructions defined by sequence of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_7.html`.

[2] Noriko Asamoto, Yatsuka Nakamura, Piotr Rudnicki, and Andrzej Trybulec. On the composition of macro instructions. Part II. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6b.html`.

[3] Grzegorz Bancerek. Cardinal numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/card_1.html`.

[4] Grzegorz Bancerek. The fundamental properties of natural numbers. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/nat_1.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 Piotr Rudnicki. On defining functions on trees. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/dtconstr.html`.

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

[8] Czesław Byliński. Functions and their basic properties. *Journal of Formalized Mathematics*, 1, 1989. `http://mizar.org/JFM/Vol1/funct_1.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] 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] Takaya Nishiyama and Yasuho Mizuhara. Binary arithmetics. *Journal of Formalized Mathematics*, 5, 1993. `http://mizar.org/JFM/Vol5/binarith.html`.

[14] Piotr Rudnicki and Andrzej Trybulec. Memory handling for $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/sf_mastr.html`.

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

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

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

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

[19] 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`.

[20] Andrzej Trybulec and Yatsuka Nakamura. Modifying addresses of instructions of $\mathbf{SCM}_{\mathrm{FSA}}$. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa_4.html`.

[21] Andrzej Trybulec, Yatsuka Nakamura, and Noriko Asamoto. On the compositions of macro instructions. Part I. *Journal of Formalized Mathematics*, 8, 1996. `http://mizar.org/JFM/Vol8/scmfsa6a.html`.

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

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

[24] Wojciech A. Trybulec. Groups. *Journal of Formalized Mathematics*, 2, 1990. `http://mizar.org/JFM/Vol2/group_1.html`.

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

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

Received August 27, 1996

Published January 2, 2004